Thomas Magedanz
Anastasius Gavras
Nguyen Huu Thanh
Jeffry S. Chase (Eds.)

46

Testbeds and Research Infrastructures

Development of Networks and Communities

LNICST

6th International ICST Conference, TridentCom 2010
Berlin, Germany, May 2010
Revised Selected Papers

Lecture Notes of the Institute
for Computer Sciences, Social-Informatics
and Telecommunications Engineering 46

Thomas Magedanz Anastasius Gavras
Huu Thanh Nguyen Jeffry S. Chase (Eds.)

Testbeds and Research Infrastructures

Development of Networks and Communities

6th International ICST Conference, TridentCom 2010
Berlin, Germany, May 18-20, 2010
Revised Selected Papers

Springer

Volume Editors

Thomas Magedanz
Technische Universität Berlin, Institute for Telecommunications Systems
Franklinstr. 28/29, 10587 Berlin, Germany
E-mail: thomas.magedanz@tu-berlin.de

Anastasius Gavras
Eurescom GmbH
Wieblinger Weg 19/4, 69123 Heidelberg, Germany
E-mail: gavras@eurescom.eu

Huu Thanh Nguyen
Department of Communication Engineering
Hanoi University of Technology, C9-412
Hanoi, Vietnam
E-mail: thanhnh@mail.hu.vn

Jeffry S. Chase
Duke University, Department of Computer Science
Durham, NC 27708-0129, United States
E-mail: chase@cs.duke.edu

Library of Congress Control Number: 2010940997

CR Subject Classification (1998): D.2, I.6, D.2.5, C.2, C.4, D.2.1

ISSN 1867-8211
ISBN-10 3-642-17850-2 Springer Berlin Heidelberg New York
ISBN-13 978-3-642-17850-4 Springer Berlin Heidelberg New York

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India
Printed on acid-free paper
06/3180 5 4 3 2 1 0

Preface

From 18 to 20 May 2010, Berlin was the focal point of the community working on testbeds and experimental infrastructures. Berlin hosted and welcomed the 6th International Conference on Testbeds and Research Infrastructures for the Development of Networks and Communities, better known as TridentCom 2010.

More than 100 experts attended the conference that provided a forum to explore existing and planned testbed concepts, infrastructures, and tools for addressing the research and business challenges of ICT convergence. The technical program represented a snapshot of the best-of-breed international research on testbeds and research infrastructures conducted in Europe, the Americas, Asia, and Australia. The strong technical program during the three days brought together researchers from across the world that presented and discussed their latest results. Out of more than 100 submitted contributions, the Program Committee finally selected after a peer-review process 15 full papers, 26 practices papers, and 22 posters. Overall the presented contributions originate from 22 nations underlining the world-wide scope and significance of the conference.

Thomas Magedanz from Fraunhofer FOKUS and Technical University Berlin, who acted as the General Chair, opened the conference by pointing out that the research and development in the areas of converging networks, unified communications, as well as emerging cross-sector smart applications is getting increasingly complex and expensive. For this reason open testbeds and research infrastructures are becoming the enabling infrastructure for achieving innovations in various domains, ranging from networking and services up to various application domains.

In the first keynote session, Max Lemke from the European Commission presented the European view on the role of experimentation in future Internet research. He also gave an outlook on the future activities in this area that are subsumed under the Future Internet Research and Experimentation (FIRE) Initiative. Chip Elliott from the GENI project office presented the approach taken in the USA towards exploring networks of the future. He presented the current status and plans of GENI (Global Environment for Network Innovations) as well as the program activities of the GENI project office. Phuoc Tran-Gia from the University of Würzburg presented in his keynote the concept and federation issues of the G-Lab project, a large German initiative to deploy testbeds and experimental platforms in Germany.

In the second keynote session, Akihiro Nakao from the University of Tokyo presented the relevant activities in Japan towards the design and development of testbeds for the future Internet. He devoted particular attention to infrastructures that support visualization as one of the fundamental concepts in the area. Finally, Bernard Barani from the European Commission presented the European Public Private Partnership on the Future Internet (PPP-FI). The PPP-FI

implicitly had a strong influence on the conference, as it represents a significant effort to demonstrate future Internet services and applications.

With the emergence of the future Internet, including the network of the future, the Internet of things, and the Internet of services, the traditional borders of network and service layers are vanishing. Cross-layer experimental platforms are being established around the globe to enable rapid prototyping and validation of innovative ideas but also taking into account migration and interworking with existing network and service platforms. Thus, this year's TridentCom emphasized Testbeds and Experimental Facilities for the Future Internet and also featured additional testbed highlights from other domains. The accepted contributions resulted in 11 technical sessions, addressing:

- Federated and large-scale testbeds
- Future Internet testbeds
- Future wireless testbeds
- Monitoring in large scale testbeds
- Network and resource visualization for future Internet research
- Future Internet testbeds for wireless sensors, media, and mobility
- Wireless and mobile networking testbeds
- Monitoring, QoS, and application instrumentation in large-scale testbeds
- Management, provisioning, and tools for future network testbeds
- Experimentally driven research and user experience testbeds

The conference program also featured an interactive panel and three tutorials. TridentCom 2010 was also the site of the second focused workshop on Open NGN and IMS Testbeds (ONIT).

The TridentCom conference "brand" is now established as the main yearly conference of the research and development community for testbeds and experimental infrastructures. The sixth conference this year impressively demonstrated that the community is very active and is taking up the challenge to deploy the necessary infrastructure for supporting future Internet and future network research.

The conference concluded with the announcement that the next conference, TridentCom 2011, will take place in Shanghai, the flourishing center of commerce, finance, and business of China, organized by the Chinese National Engineering Research Center for Broadband Networks and Applications.

Further information is available at http://www.tridentcom.org

May 2010

Thomas Magedanz
Jeff Chase
Anastasius Gavras
Nguyen Huu Thanh

Organization

Steering Committee

Imrich Chlamtac	Create-Net, University of Trento, Italy
Csaba A. Szabo	BUTE, Hungary

General Chair

Thomas Magedanz	TU Berlin, Fraunhofer Fokus, Germany

Program Committee Co-chairs

Anastasius Gavras	Eurescom, Germany
Nguyen Huu Thanh	Hanoi University of Technology, Vietnam
Jeff Chase	Duke University, USA

Conference Coordinator

Barbara Torok	ICST

Workshops Chair

Paul Müller	University of Kaiserslautern, Germany

Demonstrations Chair

Sandor Szabo	BUTE, Hungary

Tutorials Chair

Thomas Magedanz	TU Berlin, Fraunhofer Fokus, Germany

Publicity Chair

Carlos Becker Westphall	Universidade Federal de Santa Catarina, Brazil

Publications Chair

Milon Gupta	Eurescom, Germany

Local Arrangements Chair

Brigitte Henckel	Fraunhofer Fokus, Germany

Web Chair

Sibylle Kolb	Eurescom Germany

Technical Program Committee

Sudhir Aggarwal	Florida State University, USA
Khalid Al-Begain	University of Glamorgan, UK
Anish Arora	Ohio State University, USA
Paolo Bellavista	University of Bologna, Italy
Carlos Jesús Bernardos Cano	Carlos III University of Madrid, Spain
Antoine Boutet	IRISA, France
Mauro Campanella	GARR, Italy
Zhongren Cao	University of California at San Diego, USA
Dipanjan Chakraborty	IBM, India
Jyh-Cheng Chen	Telcordia Technologies, USA, and National Tsing Hua University, Taiwan
Maoke Chen	Tshinghua University, China
Michael Chen	Institute for Information Industry, Taiwan
Yingying Chen	Stevens Institute of Technology, USA
Landon Cox	Duke University, USA
Piet Demeester	IBBT, Belgium
Spyros Denazis	University of Patras, Greece
Lee Dongman	Korean Advanced Institute of Science and Technology, S. Korea
Christian Esteve Rothenberg	CPqD, Brazil
Serge Fdida	University Pierre and Marie Curie, France
Sergi Figuerola	i2CAT, Spain
Stefan Fischer	University of Lübeck, Germany
Alex Galis	University College London, UK
Aun Haider	NICT, Japan
Jason O. Hallstrom	Clemson University, USA
Henry Jerez Morales	Microsoft, USA
Hongbo Jiang	Huazhong University, China
Sun Moo Kang	National Information Society Agency of Korea, S. Korea
Sastry Kompella	Naval Research Laboratory, USA
Thanasis Korakis	Polytechnic Institute of NYU, USA

Georgios Kormentzas	University of the Aegean, Greece
Akira Kurokawa	NTT, Japan
Young-Hee Lee	Information and Communication University, S. Korea
Jian Liang	New York University, USA
Yow-Jian Lin	Telcordia, USA
Jason Liu	Florida International University, USA
Pei Liu	New York University, USA
Edmundo Madeira	State University of Campinas, Brazil
Shiwen Mao	Auburn University, USA
Josep Martrat	ATOS Research, Spain
Fermín Galán Márquez	Telefonica I+D, Spain
Dean Mathoorasing	Orange - France Télécom Group, France
Michael Menth	University of Würzburg, Germany
Scott Midkiff	NSF, USA
Eugen Mikoczy	Slovak Telekom, Slovak Republic
Takai Mineo	University of California, Los Angeles, USA
Roberto Minerva	Telecom Italia, Italy
Chen Minghua	The Chinese University of Hong Kong, China
Paul Müller	University of Kaiserslautern, Germany
Toon Norp	TNO, The Netherlands
Max Ott	NICTA, Australia
Santosh Pandey	Cisco, USA
Miguel R. Ponce de Leon	Waterford Institute of Technology, Ireland
Robert P. Ricci	University of Utah, USA
Michael Rumsewicz	University of Adelaide, Australia
Sambit Sahu	IBM, USA
Jochen Schiller	Free University Berlin, Germany
Bruno Schulze	National Laboratory for Scientific Computing, Brazil
Pablo Serrano	Carlos III University of Madrid, Spain
Lei Shu	National University of Ireland, Galway, Ireland
Dirk Trossen	BT Research, UK
Damla Turgut	University of Central Florida, USA
Scott Valcourt	University of New Hampshire, USA
Pablo Vidales	Deutsche Telekom Laboratories, Germany
Neco Ventura	University of Cape Town, South Africa
Sebastian Wahle	Fraunhofer Fokus, Germany
Sherry Wang	Johns Hopkins Applied Physics Laboratory, USA
Xiang Weidong	University of Michigan, USA
Kenneth Yocum	University of California, San Diego, USA
Youping Zhao	Shared Spectrum Company, USA
Harold Zheng	Johns Hopkins Applied Physics Laboratory, USA
Martina Zitterbart	University of Karlsruhe, Germany

ONIT Workshop

Organizing Committee

Dragos Vingarzan	Fraunhofer FOKUS
Hassnaa Moustafa	France Telecom
Eugen Mikoczy	Slovak Telekom

Technical Program Committee Members

Thomas Magedanz	Fraunhofer FOKUS, Germany
Miguel Ponce de Leon	Waterford Institute of Technology, Ireland
Spyros Denazis	Patras University, Greece
Ana Sobrino	Telefonica I&D, Spain
Wolfgang Brandstätter	Telekom Austria, Austria
Mazlan Abbas	MIMOS, Malaysia
Peter Weik	Fraunhofer FOKUS, Germany
Dorgham Sisalem	Tekelec, Germany
Sebastian Wahle	Fraunhofer FOKUS, Germany
Franz Edler	University of Applied Sciences, Technikum Wien, Austria
Joachim Fabini	Vienna University of Technology, Austria
Daniel-Constantin Mierla	Kamailio, Romania
Sherali Zeadally	University of District of Columbia, USA
Pascal Lorenz	University of Haute-Alsace, France
Andreas Kassler	Karlstad University, Sweden
Artur Krukowski	Intracom, Greece
Oscar Martinez	University of Miguel Hernandez, Spain
Paolo Bellavista	University of Bologna, Italy
Daniel Díaz Sanchez	Universidad Carlos III de Madrid, Spain

Table of Contents

TridentCom 2010 – Practices Papers Session 2: Future Internet Testbeds for Wireless Sensors, Media and Mobility

TridentCom 2010 – Practices Papers Session 3: Wireless and Mobile Networking Testbeds

TridentCom 2010 – Practices Papers Session 4: Monitoring, QoS and Application Instrumentation in Large Scale Testbeds

TridentCom 2010 – Full Papers Session 3: Future Wireless Testbeds

TridentCom 2010 – Full Papers Session 4: Management, Provisioning and Tools for Future Network Testbeds

TridentCom 2010 – Practices Papers Session 5: Federated and Large Scale Testbeds

TridentCom 2010 – Practices Papers Session 6: Experimentally Driven Research and User Experience Testbeds

TridentCom 2010 – Full Papers Session 5: Monitoring in Large Scale Testbeds

TridentCom 2010 – Posters

ONIT Workshop 2010 – Session 1: Services and Architectures

ONIT Workshop 2010 – Session 2: Architectures and Services Convergence

ONIT Workshop 2010 – Session 3: Prototyping and Interoperability

TridentCom 2010

Full Paper Session 1: Federated and Large Scale Testbeds

How to Build Complex, Large-Scale Emulated Networks

Hung Nguyen[1], Matthew Roughan[1], Simon Knight[1], Nick Falkner[1], Olaf Maennel[2], and Randy Bush[3]

[1] University of Adelaide, Australia
[2] University of Loughborough, United Kingdom
[3] IIJ, Japan

Abstract. This paper describes AutoNetkit, an auto-configuration tool for complex network emulations using Netkit, allowing large-scale networks to be tested on commodity hardware. AutoNetkit uses an object orientated approach for router configuration management, significantly reducing the complexities in large-scale network configuration. Using AutoNetkit, a user can generate large and complex emulations quickly without errors. We have used AutoNetkit to successfully generate a number of different large networks with complex routing/security policies. In our test case, AutoNetkit can generate 100,000 lines of device configuration code from only 50 lines of high-level network specification code.

1 Introduction

Emulation is a key enabling technology in network research. It allows experiments that are more realistic than simulations, which would otherwise be expensive to construct in hardware. Hardware networks are also difficult to reconfigure if multiple different test networks are needed for a large-scale experiment.

However, it is almost as hard to build large-scale, complex networks in emulation as it is in hardware. Emulation removes issues such as the need to physically place interconnecting wires, but still requires configuration of many devices, including routers and switches. Router configuration is particularly difficult in complex networks [2, 3, 8]. Manual configuration is the root of the problem, because it introduces the possibility of human error, and lacks transparency as it is not self-documenting.

We will be examining large-scale networks, which may contain thousands of routers. Although this could involve hundreds or thousands of configuration files, the amount of data which differs between these files is often relatively small, and typically limited to areas such as IP configuration, community attributes, and small changes to routing weights and policy. The majority of complex configuration changes are reserved for BGP policy implementations on the network's edge. For instance, in a small Netkit network of only 14 routers, configuration files for these routers can be compressed by a factor of 40 (using gzip), showing a large amount of redundancy and repetition in these files. We wish to focus on only those items of data which are crucial in differentiating the network configs, not to the vast bulk of configuration information required to meet the syntactic requirements of a vendor's configuration language.

T. Magedanz et al. (Eds.): TridentCom 2010, LNICST 46, pp. 3–18, 2011.

A solution to this problem is the use of fixed templates. A typical configuration template has relatively small amounts of crucial varying information inserted at the correct point. This provides user with several benefits. From an operational viewpoint, we now change a much smaller amount of data to describe a functioning system. Additionally, an automatic generation mechanism can be made self documenting, so that the changes made are much easier to track, and if necessary, reverse. But fixed templates can only go so far. Most complex tasks are still configured manually. For example, network resources such as IP address blocks and BGP community attributes are still manually allocated. These tasks can quickly become complex for large networks.

This paper is one of the first steps towards fully automated configuration, generated from a description of network capabilities. We describe AutoNetkit, which provides a high-level approach to specifying network functionality. We have implemented an automated mechanism to generate and deploy configurations for networks emulated using the Netkit framework. The task is non-trivial, particularly for BGP (Border Gateway Protocol) configuration, which is highly technical and non-transparent [8]. We plan to add support for other platforms in the future, such as Cisco IOS and Juniper Junos, described using the same high-level approach.

AutoNetkit enables Netkit users to create larger and more complex networks, easily and quickly. It is written in Python, making it portable and easily extensible. It also allows scripted creation of networks so that a series of networks can be created, and tests run on each.

The results are not just useful for Netkit, they provide insights into the general problem of automating network configuration for real networks. Furthermore, emulations powered by AutoNetkit have important applications in operational networks. By being able to construct a fundamental model of the key aspects of a network, we are in a position to carry out tests on this network within an emulated environment. We can also test proposed changes to our network, such as maintenance or upgrades, on an emulated network which reflects our real network. We refer to this mirrored network model as the *shadow model*. The shadow model of the network allows us to reserve infrastructure for future development, test future growth options, and to determine the outcome of failure scenarios. This is also of great benefit to operational staff; they can have a much better idea of the performance of their network, under a wide variety of scenarios, without needing to physically realise that scenario.

AutoNetkit is based on an emulation approach to network research. This differs to simulation approaches; in our emulations we run virtual instances of real routers, communicating through real routing protocols, whereas simulations instead approximate router behaviour [4]. At the other end of the spectrum, testbeds [13] provide real hardware-based networks, and so are more realistic, but also more expensive and less flexible. AutoNetkit aims to address the middle ground, allowing the user to quickly and cheaply carry out realistic network research.

2 Background

2.1 Netkit

Netkit is an open source software package which simplifies the process of creating and connecting virtual network devices using User Mode Linux (UML) [15]. UML allows

multiple virtual Linux systems to be run on the same host Linux system, with each virtual systems retaining standard Linux features. In particular, networking is configured using standard tools. Additional software packages can be installed for extra features, such as BIND for DNS services, or the Quagga routing suite. Quagga provides an implementation of common routing protocols, allowing a Linux system to function as an IP router.

Netkit provides a set of tools to manage the process of setting up and launching a UML virtual system. Once an emulated network has been specified in a configuration file, Netkit takes care of creating and launching the UML virtual systems. Typically, Netkit creates one virtual host for each router and launches routing services on each of these routers. Each router has one or more virtual network interfaces that are connected using virtual switches. These switches and the routing services running on the routers allow emulations of large networks.

Netkit simplifies the process of launching the emulated network, including services and virtual switches. However it does not provide tools to automate the configuration of each network device. Netkit emulations can be extended beyond one physical host machine, which we will describe in this paper.

Examples of Netkit networks. Figure 1 shows an example, drawn from the test example given in [1], of a Netkit network which emulates a small Internet. To emulate this network Netkit requires a description of each network device and the links between them. Routing requires a set of configuration files for each network device. These describe interface configuration, such as IP addressing, and interior routing protocols, such as IS-IS or OSPF. Border routers also require the BGP exterior routing protocols to be configured. The network in Figure 1 with only 14 routing devices requires more than 500 lines of configuration code, most of which is described in an arcane low-level router configuration language.

One of the strengths of emulation is to build networks larger than would be affordable to construct in hardware. It is easy to conceive of networks with thousands of devices and tens of thousands of lines of configuration code [17], but, at present, emulating these networks is constrained by the configuration process. What is more, many research projects require evaluations on multiple networks to test robustness and flexibility. The complexity of device configuration means that creating a large-scale network is a time consuming and error-prone process. This is true for both physical networks and emulated networks. Our tool simplifies this configuration process: our large-scale example network consists of 527 routers connected by 1634 links, a size which would be infeasible to manually generate.

Other Emulation Tools. VNUML [10] is a medium scale software emulator, similar to Netkit, that uses a User-Mode Linux kernel. There are also other emulation tools such as Einar [7]. As AutoNetkit has been designed to be a high-level auto-configuration tool, independent of a specific emulation technique, it could be used in these emulation environments with only minor modifications.

2.2 Router Configuration

Each equipment vendor has a specific configuration language used to configure their routers. These languages differ between vendors; to configure the same feature on two

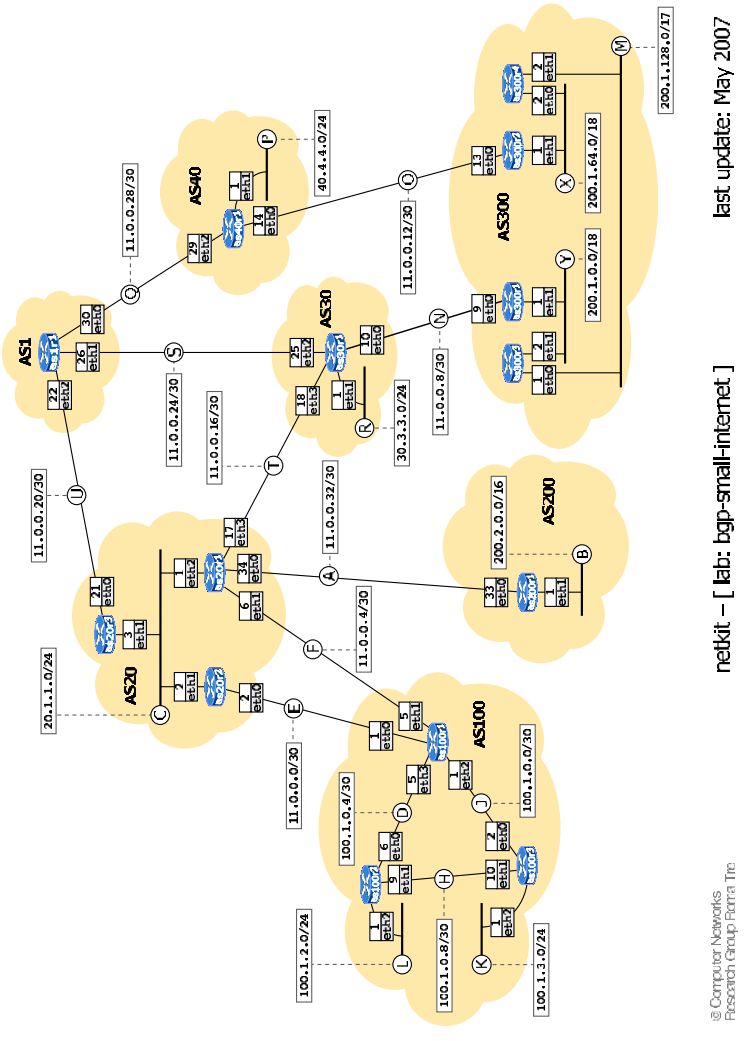


Fig. 1. Small Internet Netkit Lab [1]

different routers may involve very different syntax. This requires an operator to learn a new configuration language for each vendor, limits code portability, and makes it difficult to manage a network containing routers from different vendors.

An example of a Quagga BGP configuration file is below, showing low level configuration requirements. Quagga is an open source routing protocol suite [14], used by Netkit. Quagga configuration syntax is similar to that used in Cisco routers, but very different to Juniper router syntax. All network related numbers in these configurations, such as IP addresses and AS numbers, must be consistent across all network devices.

```
router bgp 300
network 200.1.0.0/16

network 200.1.0.0/17
!
neighbor 11.0.0.10 remote-as 30
neighbor 11.0.0.10 description Router as30r1
neighbor 11.0.0.10 prefix-list mineOutOnly out
neighbor 11.0.0.10 prefix-list defaultIn in
!
ip prefix-list mineOutOnly permit 200.1.0.0/16
ip prefix-list mineOutOnly permit 200.1.0.0/17
ip prefix-list defaultIn permit 0.0.0.0/0
```

Common router configuration tasks include setting up each interface and configuring the routing protocols used to exchange routing information. A correctly operating network requires each router's configuration to be syntactically and semantically correct with configurations consistent across the network. If these conditions are not met, the network will not operate correctly. For example, the IP address at each end of a point to point link must belong to the same subnet.

These configuration files are usually generated by hand — a slow process with the time taken being roughly proportional to the number of devices in the network. Each router must have its own configuration file, and manually generating each configuration file is impractical for large networks. Template based configuration methods [2,8] are an improvement, but still require network resources to be allocated. Efficiently allocating network resources such as IP address blocks, BGP community attributes can quickly become complex for large networks.

Our goal is to automate this configuration process. This is a complex problem for a hardware device based network: hardware faults, device dependent configuration languages, physical device connections, and multiple users accessing the system all must be considered. Auto-configuration of a software based network is a more constrained problem. When using Netkit we are able to dictate the target platform, and ensure that the underlying network connections meet the desired structure. Configuration of emulated networks still present a number of configuration problems such as routing and security policy implementation, automatic IP address allocation, which will be discussed in this paper. Existing configuration tools for Netkit such as Netkit Interface Utility for Boring Basic Operations (NIUBBO) [19], do not provide these features and only allow small networks with very basic routing options. Even though languages such as RPSL [22] and its associated tool RtConfig [23] can be used for complex BGP policy specification and configuration generation, they still work at the device level. AutoNetkit aims at a higher level, being able to configure networks from high-level concepts.

3 AutoNetkit

AutoNetkit automates the process of creating a large and complex network. The aim of AutoNetkit is to allow a user to say what they want to achieve with a network, such as the business relationship to be expressed or the logical structure of the network without requiring details of specific device configuration. Instead of assigning specific values to configuration parameters of the devices, we want to be able to express high-level concepts for the network such as: "There must be at least one DNS server in the network"; "Business relationship with neighboring ASs should be enforced"; and "OSPF link weights should be assigned using algorithm ABC".

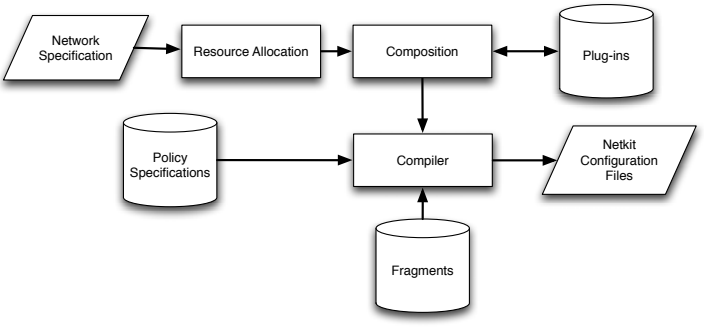


Fig. 2. AutoNetkit System Overview

We adopt an approach inspired by [3]. The system is illustrated in Figure 2. The user specifies a network model which describes the logical structure of the network and the resources in the network such as devices, IP address blocks. In addition, the user needs to specify the rules/policies for the network such as routing policies. The rules/policies pull in fragments (small templates of configuration code) to implement the network. These components are described below. The system design allows the use of plugins to interact with the network model. This may involve reading and modifying network object attributes.

The AutoNetkit language is implemented as an object oriented language using Python [21]. Object orientated languages are well suited to configuration specification as they allow natural expression of network devices as objects [6, 12]. To aid in describing the components of our approach we will use the simple, but non-trivial network as in Figure 1. The AS level topology and BGP policies applied to these networks are shown in Figure 3.

3.1 The Network Specification

The network model is specified by the network designer using the AutoNetkit language. This model describes the resources, devices, and logical structure of the network. The details of these objects are described below.

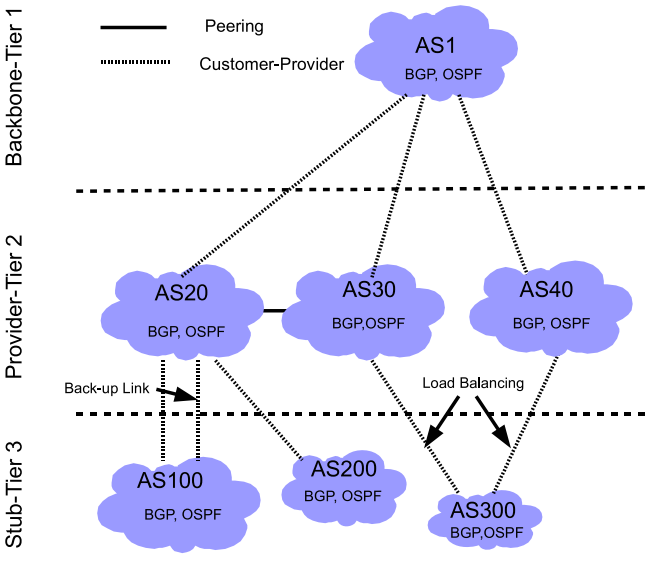


Fig. 3. AS level topology showing high-level specification of desired inter-AS policies for the network in Figure 1. Two types of business relationships are shown. Customer-provider relationships are shown as the dashed line between two vertically adjacent nodes — the AS on the lower layer is the customer and the AS on the upper layer is the provider. The peering relationship is shown as the horizontal solid line between AS20 and AS30. The figure also shows two other BGP policies: load balancing over multiple links, and a back-up link.

Resources in the network. Each network resource is represented by an object in the AutoNetkit language, with attributes managed or modified by the network policies. The two main resources are IP address blocks and devices. Examples of these are provided below:

– IP address blocks;

```
### Networks with IP address resource
AS1=Network(1,['1.0.0.0/8','100.0.0.0/8'])
AS20=Network(20, ['20.0.0.0/8'] )
AS100=Network(100, ['100.0.0.0/8'] )
AS200=Network(200, ['200.1.0.0/16'] )
```

– Devices (routers, switches, etc.).

```
AS1.add_router('AS1R1','Quagga','Router 1')
```

In the above example, each AS is given a set of one or more address blocks. These are used to assign IP addresses to the interfaces in that AS. Each router is represented by an object inside the AS object. In this example, a router object, AS1R1, is added to an autonomous system, AS1, with the specified initial values assigned to the router object attributes. During the configuration process, objects inside the AS are

modified to satisfy user specified connectivity and policy requirements. For example, interface objects will be added to the AS1R1 router object for connectivity configuration and BGP objects will be added to this router object to implement business relationships with other ASs.

Network Logical Structure. The user is also required to specify the logical structure of the network, describing how the devices will be interconnected. This specification may include details such as link capacity, and link weight. Link weights can be assigned to links or interfaces, and are used to control the path decisions made by routing protocols.

A link between routers in the same AS can be easily setup using the *add link* command in default mode, which takes 2 routers as parameters and creates a link between them. It will automatically add an interface to each router and assign an appropriate IP address to each interface.

```
### add intra AS links
AS100.add_link(AS100R1, AS100R2)
AS100.add_link(AS100R1, AS100R3)
```

When creating a link that spans two autonomous systems, we use the *add link* command with specific options. If the remote autonomous system is managed by another entity (such as another ISP), its configuration is outside of our control. In this case, we cannot automatically assign the remote router an IP address, so we provide the option for a user to choose to manually specify link IP address details. This configuration flexibility is shown in the following example:

```
### add inter AS links
AS30.add_link(AS30R1,AS300,AS300R1, constraints =
{"subnet":'11.0.0.8/30','int1ip":'11.0.0.10',
                          "int2ip": '11.0.0.9'})
```

We have described what is specified in the network model, but it is also important to consider what is not specified. Everything in the network model is specified by the user. For instance, the user indicates which routers are interconnected (although this may be the output of a network generation program such as BRITE [16]). Hence, it is important to avoid specifying pieces of no interest to users, even though they may be required in the actual network configuration.

It is common to implement a point-to-point link between two routers as a /30 subnet, which provides a usable IP address for each end of the link. Each interface in a link must be within the same subnet, but the choice of the subnet itself is often unimportant, provided that the allocation is not used again elsewhere in the network. It is a simple task for an auto-configuration tool to choose such addresses from allocated blocks, saving the user from needless work in making specific allocations that they are not concerned with. Automating allocation tasks also reduces the chance of bugs due to human error.

Similarly, creating a link requires the interfaces on each end of the link to be configured, but the specific settings are often unimportant, providing they are consistent at both ends of the link. An example is routing policies, which are applied to an interface. Automating allocation tasks is analogous to using a software compiler to handle

low level resource allocation, freeing the programmer to write high-level code describing only the functions they are concerned with.

3.2 Resource Allocation

As discussed, the compiler must realise the high-level network model, converting it into a detailed model based on the implementation details. IP addresses are allocated by taking the pool of IP addresses specified when the AS object was created, dividing them into the relevant size subnet, and then allocating an IP from this subnet to the relevant interface. This automated process avoids conflicting IP addresses and ensures each interface has an IP address belonging to the same subnet.

There are some issues that need to be considered carefully in this step. For instance, although not needed, it can make it easier for a user if this process is deterministic. Determinism is the property where instantiating the same network twice will result in the same resource allocations being made. However, changing a subset of the inputs should not necessarily lead to a widespread change in the final allocation and we may wish to limit the effect of change on the allocations in an instantiated network. We refer to the property of an allocation scheme to limit unnecessary change as *insensitivity*. To implement this property we use a *sticky allocation* mechanism.

Sticky allocation allows the allocation to a subset of nodes in the network to remain constant in the face of change, unless the change will force a change in allocation, either through address space exhaustion or the addition of links or hardware that directly connect to that subset. The major advantage of sticky allocation is that it limits the number of configuration changes that are required on the target devices, and this allows more efficient incremental improvements to be carried out in the network. Neither of these features are required but they are desirable, as they improve efficiency and make debugging easier. Our current tool makes deterministic and sticky allocations.

We show part of the resource allocation for router AS20R1 in AS20 of the example in Figure 1

```
eth0 20.255.255.253/30
eth1 11.0.0.6/30
lo 127.0.0.1
lo:1 1.1.1.1/30
```

The interfaces have been automatically configured, with their IP addresses either assigned from the pool of available addresses given for that AS, or from the user's manually specified settings in the case of an inter AS link. The loop back interface *lo:1* is also configured on the router for use by the BGP routing protocol.

3.3 Rule/Policy Specifications

Rules are used to describe high-level requirements placed on a network, and range from routing policies to security and resiliency requirements. To define high-level requirements, the user needs to specify which rules are going to apply to which objects inside the network as part of the input. Each rule is broken down further to a set of smaller objects called fragments. A fragment is the smallest element that can be

easily translated into device specific configuration code. Fragments are described in more detail in the next section. A rule/policy is a precise statement of which fragments will be applied to which device objects and the exact values of the attributes that the fragment is going to give to the object. Rules are implemented in AutoNetkit as objects. Typical rules are routing policies. For example, to specify the interior gateway protocol (IGP) to be OSPF with configurable area information

```
## Add IGP logic
scope={'type':'router','select':'all'}
parameters=['Area','new',1] # OSPF parameters
AS100.add_rule('OSPF',scope,para)
```

and to enforce business relationship with neighbour ASs, AS100 adds the *peering()* policy to all of its sessions.

```
## BGP policy for enforcing peering relationship
scope={'type':'session','select':'all'}
parameters={}
Rule = peering(scope,parameters)
AS100.add_BGP_policy('Enforce business relationship', Rule)
```

AutoNetkit has a library of rules (i.e., network services/ policies) implemented. These include rules to set up DNS server, a large set of different BGP policies to maintain business relationship, contract obligations, security and back-up requirements. The user needs to specifies in the rule specification which of these rules are going to be used in each network. Each rule requires a "scope" and a "parameters" input. The "scope" defines the BGP sessions that the policy applies to, and the "parameters" field is used to provide special parameters to the policies.

3.4 Fragments

Many router configurations have a high degree of similarity, which allows for the script based configuration methods discussed previously. It also simplifies the configuration process, allowing most device specific configuration to be performed with simple templates. These templates are filled in with the relevant values from a resource database, created based on the network model.

Some components of a router configuration are only needed on certain routers, e.g., we only require eBGP on edge routers. Simple templates are less useful in these cases. Instead we use the concept of fragments [3]: small pieces of configuration code, each typically controlling a single configuration aspect. Each fragment is defined by the object attributes that it will creates or modifies.

Complex tasks require several fragments. AutoNetkit also provides an extensive library of fragments that can be used to construct the policies. These fragments can be used to implement almost all realistic BGP policies including black hole, Martian filters, and peering. For example, a peering policy can be realised by using one fragment to mark all routes on ingress with a community that encodes the peering type of the BGP session. On egress the routes are filtered based on the community tags and the peering type of the session, using another fragment. The peering type, a parameter of the session, determines which fragment (BGP statement) to use. Additional fragments can be

used if complex traffic policies are implemented using the peering type. As discussed, we have used these fragments to implement a library of BGP policies, including load balancing and a back-up link, as per the example of Figure 1.

3.5 Plugins

AutoNetkit has been designed to be extensible, allowing the user to interact with the network structure using plugins. We have implemented a plugin which exports the network as a graph, where routers are represented by nodes and links by edges. Operations can be carried out on this graph, and the results applied back to the AutoNetkit network objects. The NetworkX [18] Python package is used to represent and analyse networks. This package includes common graph analysis functions such as shortest path or resiliency algorithms, which can be used to study the network model in AutoNetkit.

The graph structure allows existing research to be implemented in a Netkit network. As an example we have implemented a standard traffic engineering algorithm. The algorithm optimises link utilization (minimises congestion), by adapting the link weights used by the network's routing protocol to choose shortest paths [9]. The result is that traffic is balanced across network paths (it may be surprising that this simple form of traffic engineering is effective, but previous results [9] have shown it can be almost as good as MPLS). Our implementation uses the network model described above via the plugin architecture, as well as a user provided traffic matrix. This algorithm is used to analyse and optimise a network created using AutoNetkit, and apply the optimised weights to the network, where they are used to generate the appropriate configuration file entries.

We have also used simple mathematical functions to deliver powerful network results. The NetworkX function to find the central node in a graph is used for optimal server placement: the DNS server in each network can be automatically set to be the central node in the network. AutoNetkit's plugin framework allows users to easily apply mathematical research to networking, and then analyse the results in an emulated environment. AutoNetkit includes tools to verify the correct application of these weights, which we will describe later.

3.6 Compiler

The compiler produces configuration files for the Netkit devices, based on the network description, the rule/policy specification, and the library of available rules and fragments.

The compilation process starts by creating an object for each device declared in the network specification. It then examines the rules, creating an object for each rule, and attaching relevant device objects. The template implementation of each rule is then read, and the fragment objects for that rule created. These fragment objects are then attached to the appropriate device objects, as specified by the rule.

After the fragment objects have been attached to the device objects, the individual device configurations take place. The compiler first configures interface objects, assginging the IP address and network mask to each interface object, as per the resource allocation process described earlier. The router objects are then configured,

with the internal routing protocols configured first using the IGP fragment objects, and BGP using BGP fragments, if required.

Fragments within each device and across different devices may have a dependency relationship: some fragments need to be applied before the others, and multiple fragments can modify the same attribute in the device objects. This is especially the case for BGP fragments. One of the most important tasks of the compiler is to resolve these dependencies. AutoNetkit provides two simple methods to solve these dependency problems. First, all fragments are given a unique sequence number, used to capture the order dependency between fragments. A fragment with small sequence number is always applied before a segment with larger sequence number. Second, after sequencing if two fragments still attempt to modify the same attribute, AutoNetkit issues a warning and does not configure the device where conflict occurs. In this case the user must manually resolve the conflict. While these two simple methods are successful in resolving all test networks, more advanced methods are needed to resolve complex dependencies. These are the topics of our future research.

Once the conflicts are resolved, the device objects are configured and written in Quagga syntax to the configuration files, ready for deployment.

3.7 Deployment and Verification

AutoNetkit simplifies the process of automatically deploying the generated configuration files to a Linux machine running Netkit. The deployment module copies across the configuration files, stops any previous Netkit labs, starts the new lab, and verifies that all hosts have been successfully started.

The deployment module can verify that the output of the optimisation plugin, detailed previously, was successfully applied to the running network. It compares the output of the NetworkX shortest paths algorithm for each source-destination pair in the network, against the *traceroute* command output for the Netkit network each pair in the network.

3.8 Emulation Scalability

The use of software to emulate a network simplifies some aspects of hardware networks, but also introduces new considerations. The most important is the resources the virtual systems requires from the host system, including memory and processor usage, which increase with emulated network size.

A typical Netkit virtual system requires a minimum of 6MB RAM from the physical host for the basic services required to run Linux, such as the kernel. This increases to approximately 16 MB of RAM if the virtual system is to act as a router. More memory are required to provide network monitoring tools, such as traceroute, ping, and tcpdump. Packet inspection can be performed using tcpdump, but is more suitable for debugging than large-scale traffic analysis: due to resource constraints, emulated networks are better suited to testing protocols than large traffic flows.

Resource constraints limit the number of virtual systems that can be run on a single Linux machine. To emulate large networks we run emulations on multiple Linux machines, which are connected using vde_switch [5]. The size of the emulated network is then limited only by number of physical Linux hosts available, rather than

the resources of a single machine. This allows large-scale simulations to be deployed using a number of inexpensive Linux machines. We have successfully used vde_switch to scale Netkit emulated networks to several hundred virtual routers, across multiple physical machines.

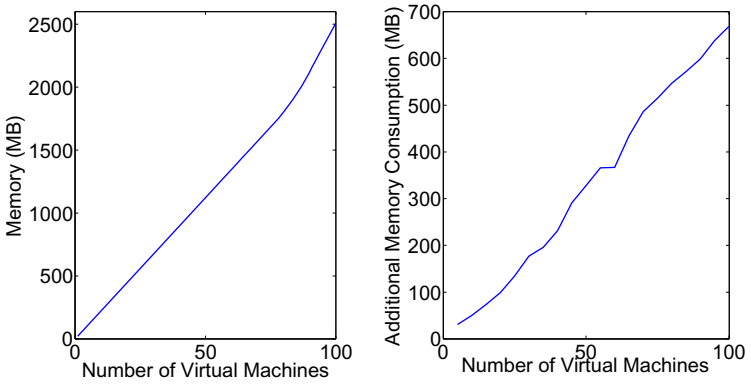


Fig. 4. Basic memory consumption with BGP and OSPF (left) and the additional memory consumption with *ping* and *tcpdump* running inside the virtual machines (right)

Memory consumption on these virtual routers grows linearly with the size of the network, for both case of with and without running applications. This is shown in Figure 4. Note that the memory consumption also depends on the size of the data inside the applications. For example, large BGP tables can easily consume more than 16MB.

3.9 Visualization

AutoNetkit allows the user to plot their networks, providing visual confirmation of designs and aiding in troubleshooting. The NetworkX graph representations discussed previously are used with pydot [20], a library to plot NetworkX graphs using Graphviz [11] graph visualisation software. We have made formatting customisations to better suit the display of computer networks, which can be be seen in Figure 5. This figure shows a section of the visualisation generated from AutoNetkit, based on the lab described in Figure 1. Different link types can be seen; internal links are shown as solid lines and external links are shown as dashed lines. Interface and subnet details are also visible. Future work will add additional visualisation features.

4 AutoNetkit Performance: A Case Study

We have evaluated AutoNetkit performance in two areas: scalability, by generating a large-scale test network, and ease of use, by comparing AutoNetkit to manually configuring the demonstration network shown in Figure 1.

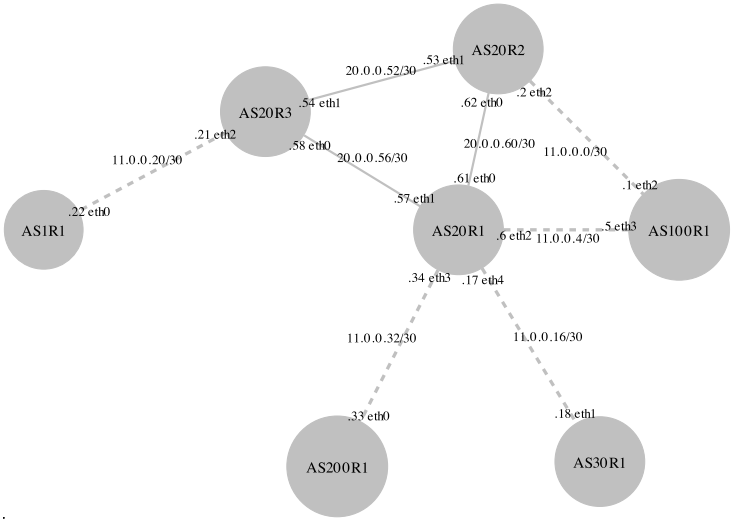


Fig. 5. Visualisation output showing the topology for AS20 in Figure 1. IGP links are shown as solid lines and eBGP links are shown as dashed lines. Resources such as interface numbers and IP addresses have been automatically allocated by AutoNetkit.

A large-scale network can be quickly and easily. For instance, to configure a randomly generated network of 100 ASs, with 527 routers connected by 1634 links, over 100,000 lines of device configuration code are needed. AutoNetkit only requires 50 lines of high-level code, consisting of loops to generate each AS, add routers to the AS, and then interconnect these routers. Generating this network, including configuration of OSPF, BGP, and DNS, is fast: AutoNetkit takes only 15 seconds on standard desktop computer, with a 3 GHz Intel Core2 Duo CPU processor.

We also configured the Netkit demonstration network, shown in Figure 1. This network may appear simple compared to large-scale networks, but still requires extensive configuration, including OSPF, BGP, DNS, and appropriate resource allocations. This adds a significant overhead to testing a simple network. Using AutoNetkit, the network model and policies for the this network can be described in 100 lines of AutoNetkit code, compared to 500 lines of device-specific configuration code. The AutoNetkit code is high-level and descriptive, and allows the user to deal with their network, not device configuration. It is also easy to alter the network: adding a link or router is simple in AutoNetkit, a task which is tedious and error-prone when manually creating configuration files.

5 Discussion

AutoNetkit achieves the goal of automating network configuration for Netkit, and provides a number of benefits:

- *Scale at lower cost:* the cost (in time) for configuring a large network is reduced, and is sublinear (rather than the linear costs of generating the whole network effectively by hand).
- *Reliability:* the reliability of emulations is improved, in the sense that we can be more confident that the emulated network is exactly what we intended, i.e., there are no misconfigurations that might stall routing, and hence change the performance of the network.
- *Consistency:* consistency is part of reliability (consistency across routers is needed), but it also involves consistency between the network, and the operators view of the network, which is critical for ongoing design, debugging, and transparency.
- *Flexibility:* our approach maintains the flexibility of Netkit to emulate complex networks and protocols.
- *Scripting:* AutoNetkit is written in Python, and so can be easily scripted into larger sets of experiments, for instance creating multiple instances of networks to compare performance of different configuration.

Another way to view the activity is by analogy to programming. In the grim old days, when programs were written in machine code, only a few gurus could program, and they were highly specialized to particular machines. Programs were typically very limited in size, and complexity. The advent of high-level programming languages made programming a commodity skill, and separated the meaning of programs from the particular hardware. Larger and more complex programs have resulted. More recently, software-engineering and related programming tools including integrated programming environments, standard portable APIs, and specification languages have helped enable very large software projects, with what could be described as a production line for code.

One view of AutoNetkit is as a high-level language and compiler for Netkit. Similar to the benefit that high-level languages bring to programming, AutoNetkit can make the network configuration process much easier, and enable emulations of large and complex networks.

6 Conclusions and Future Work

We have developed AutoNetkit, a tool that allows a user to easily generate large-scale emulated networks. AutoNetkit has been successfully used to generate a number of test networks, including one of the principal Netkit test labs described in Figure 1. AutoNetkit will be made available at `http://bandicoot.maths.adelaide.edu.au/AutoNetkit/`

There are many additional features we intend to implement in the future. We plan to extend AutoNetkit to other emulators and to real networks, including deployment to hardware networks consisting of Cisco and Juniper devices. We will also implement additional features in AutoNetkit for other routing protocols, such as RIP and IS-IS, support for MPLS, and filtering using Access Control Lists. It is important for an auto-configuration tool to test generated configurations. We currently perform path checking using *traceroute*, and will expand this verification in future AutoNetkit development.

References

1. Di Battista, G., Patrignani, M., Pizzonia, M., Ricci, F., Rimondini, M.: netkit-lab-bgp: small-internet (May 2007), http://www.netkit.org/
2. Bellovin, S.M., Bush, R.: Configuration management and security. IEEE Journal on Selected Areas in Communications 27(3), 268–274 (2009)
3. Bohm, H., Feldmann, A., Maennel, O., Reiser, C., Volk, R.: Design and realization of an AS-wide inter-domain routing policy. Deustch Telekom Technical Report (2008)
4. Chen, J., Gupta, D., Vishwanath, K.V., Snoeren, A.C., Vahdat, A.: Routing in an Internet-scale network emulator. In: Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS), Citeseer (2004)
5. Davoli, R.: Vde: Virtual distributed ethernet. Technical Report UBLCS-2004-12, University of Bologna (June 2007)
6. Delaet, T., Joosen, W.: Podim: a language for high-level configuration management. In: Proceedings of the 21st conference on Large Installation System Administration Conference, Berkeley, CA, pp. 261–273 (2007)
7. EINAR. Einar router simulator, http://www.isk.kth.se/proj/einar
8. Enck, W., McDaniel, P., Sen, S., Sebos, P., Spoerel, S., Greenberg, A., Rao, S., Aiello, W.: Configuration management at massive scale: system design and experience. In: Proceedings of the 2007 USENIX Annual Technical Conference, Santa Clara, CA, pp. 1–14 (June 2007)
9. Fortz, B., Thorup, M.: Internet traffic engineering by optimizing OSPF weights. In: IEEE INFOCOM, vol. 2, pp. 519–528. Citeseer (2000)
10. Galan, F., Fernadez, D., Rui, a., Walid, O., de Miguel, T.: Use of virtualization tools in computer network laboratories. In: Proc. International Conference on Infor- mation technology Based Higher Education and Training (2004)
11. Graphviz. Graph visualization software, http://www.graphviz.org/
12. Griffin, T.G., Bush, R.: Toward networks as formal objects. Position paper, private communication (2003)
13. Huang, M.: VNET: PlanetLab virtualized network access. In: PlanetLab Design Note, PDN-05- 029 (2005), https://www.planet-lab.org/doc/pdn
14. Ishiguro, K.: Quagga routing software, http://www.quagga.net
15. User Mode Linux. Uml, http://user-mode-linux.sourceforge.net/
16. Medina, A., Lakhina, A., Matta, I., Byers, J.: Brite: An approach to universal topology generation. In: Proceedings of the International Workshop on Modeling, Analysis and Simulation of Computer and Telecommunications Systems- MASCOTS, Cincinnati, Ohio (August 2001)
17. Muhlbauer, W., Feldmann, A., Maennel, O., Roughan, M., Uhlig, S.: Building an AStopology model that captures route diversity. In: Proceedings of the ACM SIGCOMM 2006, Pisa, Italy (2006)
18. NetworkX. High productivity software for complex networks, http://networkx.lanl.gov
19. NIUBBO. Netkit interface utility for boring basic operations, http://wiki.netkit.org/download/niubbo/niubbo-2.1.2.tar.gz
20. pydot. a python interface to graphviz's dot language, http://code.google.com/p/pydot/
21. Python. Python programming language – official website, http://www.python.org
22. RPSL. Routing policy specification language. RFC-2622
23. RtConfig. Rtconfig, http://irrtoolset.isc.org/wiki/RtConfig

Testbed and Experiments for High-Performance Networking

Nageswara S.V. Rao, Susan E. Hicks, Stephen W. Poole, and Paul Newman

Oak Ridge National Laboratory
raons@ornl.gov

Abstract. UltraScience Net is a network research testbed for supporting the development and testing of wide-area networking technologies for high-performance computing and storage systems. It provides dynamic, dedicated, high-bandwidth channels to support large data transfers, and also provides stable high-precision channels to support fine command and control operations. Its data-plane consists of 8,600 miles of cross-country dual OC192 backbone, which can be dynamically provisioned at different bandwidths. Its out-of-band control-plane is implemented using hardware Virtual Private Network (VPN) devices. In terms of the testbed infrastructure, it demonstrated the following capabilities: (i) ability to build and operate national-scale switched network testbeds, (ii) provisioning of suites of 1 and 10 Gbps connections of various lengths up to 70,000 and 8,600 miles, respectively, through automated scripts, (iii) secure control-plane for signaling and management operations, and (iv) bandwidth scheduler for in-advance connection reservation and provisioning. A number of structured and systematic experiments were conducted on this facility for the following tasks: (i) performance analysis and peering of layer 1-3 connections and their hybrid concatenations, (ii) scalability analysis of 8Gbps InfiniBand (IB) transport over wide-area connections of thousands of miles, (iii) diagnosis of TCP performance problems in using dedicated connections to supercomputers, (iv) detailed TCP performance analysis of wide-area application acceleration devices, and (v) TCP throughput improvements due to 10Gbps High Assurance Internet Protocol Encryptor (HAIPE) devices.

Keywords: Network testbed, dedicated channels, SONET, 10GigE WAN-PHY, control-plane, data-plane, bandwidth scheduler, WAN accelerators, HAIPE, InfiniBand.

1 Introduction

Large-scale computing and storage applications require high-performance networking capabilities of two broad classes: (a) high bandwidth connections, typically with multiples of 10Gbps, to support bulk data transfers, and (b) stable bandwidth connections, typically at much lower bandwidths such as 100s of Mbps, to support operations such as computational steering, remote visualization and remote control of instrumentation. These networking capabilities

T. Magedanz et al. (Eds.): TridentCom 2010, LNICST 46, pp. 19–34, 2011.

may be achieved by providing dedicated connections of the required bandwidths directly between the end users or applications. Such connections are needed only for certain durations between select sites, for example, for archiving at a remote storage facility a terabyte dataset produced on a supercomputer, or actively monitoring and steering a computation on a supercomputer from a remote user workstation. Current Internet technologies, however, are severely limited in meeting these demands because such bulk bandwidths are available only in the backbone, and stable control channels are hard to realize over shared network infrastructures. The design, building and operation of the needed network infrastructure with such capabilities require a number of technologies that are not readily available in Internet environments, which are typically based on shared, packet-switched frameworks. Furthermore, there have been very few network experimental facilities where such component technologies can be developed and robustly tested at the scale needed for large-scale computing and storage systems distributed across the country or around the globe.

The UltraScience Net (USN) was commissioned in 2004 by the U. S. Department of Energy (DOE) to facilitate the development of high-performance networking technologies needed for large-scale science applications, and has been supported by U. S. Department of Defense (DOD) since 2007. USN's footprint consists of dual dedicated OC192 connections, from Oak Ridge to Chicago to Seattle to Sunnyvale. It supports dynamic provisioning of dedicated 10Gbps channels as well as dedicated connections at 150Mbps resolution. There have been a number of testbeds such as UCLP [30], CHEETAH [6], DRAGON [15], HOPI [13] and others that provide dedicated dynamic channels, and in comparison, USN has larger backbone bandwidth and footprint. Compared to research initiatives such as GENI [12] in the U.S., FIRE [11] and FEDERICA [10] in Europe, AKARI [3] in Japan, and CNGI [7] in China, USN has a more focused goal of high-performance applications as in CARRIOCAS project [4] and ARRA ANI [28]. However, we note that USN has been operational for the past five years compared to CARRIOSCAS and ARRA ANI, which are currently being deployed.

The contributions of USN project are in two categories:

(a) **Infrastructure Technologies for Network Experimental Facility:** USN developed and/or demonstrated a number of infrastructure technologies needed for a national-scale network experimental facility. In terms of backbone connectivity at DWDM, USN's design and deployment is similar to the Internet. However, its data-plane is different in that it can be partitioned into isolated layer-1 or layer-2 connections. Its control-plane is quite different mainly due to the ability of users and applications to setup and tear down channels as needed as in [31,4,28]. In 2004, USN design required several new components including a Virtual Private Network (VPN) infrastructure, a bandwidth and channel scheduler, and a dynamic signaling daemon. The control-plane employs a centralized scheduler to compute the channel allocations and a signaling daemon to generate configuration signals to switches.

(b) **Structured Network Research Experiments:** A number of network research experiments have been conducted on USN. It settled an open matter by demonstrating that the bandwidth of switched connections and Multiple Protocol Label Switching (MPLS) tunnels over routed networks are comparable [22]. Furthermore, such connections can be easily peered, and the bandwidth stability of the resultant hybrid connections is still comparable to the constituent pure connections. USN experiments demonstrated that InfiniBand transport can be effectively extended to wide-area connections of thousands of miles, which opens up new opportunities for efficient bulk data transport [24,18]. USN provided dedicated connections to Cray X1 supercomputer and helped diagnose TCP performance problems which might have been otherwise incorrectly attributed to traffic on shared connections [21]. Also, experiments were conducted to assess the performance of application acceleration devices that employ flow optimization and data compression methods to improve TCP performance [19]. USN demonstrated file transfer rates exceeding 1Gbps over 1GigE connections of thousands of miles. Recently, experiments were conducted to assess the effect of 10Gbps High Assurance Internet Protocol Encryptor (HAIPE) devices on TCP throughput over wide-area connections. Somewhat surprisingly, these devices lead to improvements in TCP throughput over connections of several thousands of miles [17].

This paper is organized as follows. In Section 2, we describe USN technologies for data- and control-planes. The results from network experiments are described in Section 3. This paper provides an overview of these topics and details can be found in the references [23,20,22,19,21,25,24,17,18,16].

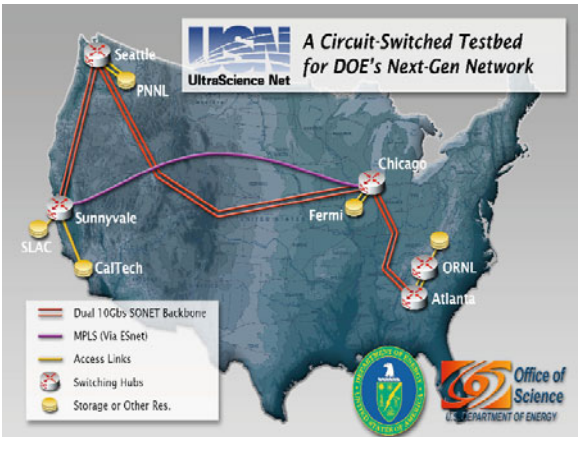


Fig. 1. UltraScience Net backbone consists of dual 10 Gbps lambdas from Oak Ridge to Chicago to Seattle to Sunnyvale

2 USN Infrastructure and Technologies

USN infrastructure is supported by co-location sites at Oak Ridge, Chicago, Seattle and Sunnyvale as shown in Figure 1. USN backbone utilizes ORNL network infrastructure to provide two OC192 SONET connections from Oak Ridge to Chicago, and two OC192 SONET connections from National Lambda Rail (NLR) between Chicago, Seattle and Sunnyvale. USN peered with ESnet [9] at both Chicago and Sunnyvale, and with Internet2 [14] in Chicago. USN architecture is based on out-of-band control-plane as shown in Figure 2, since lack of data-plane continuity makes in-band signaling infeasible.

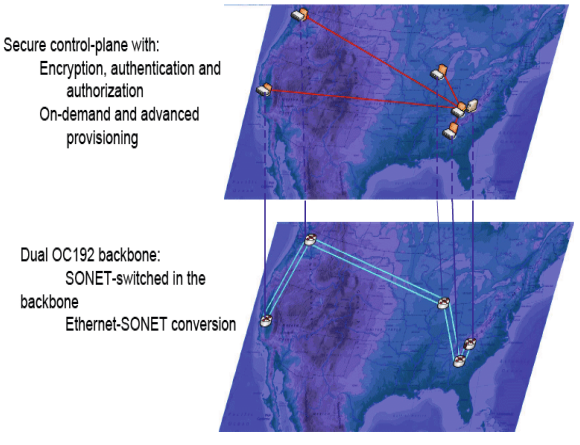


Fig. 2. USN architecture is based on separate data- and control-planes

2.1 Data-Plane

The data-plane of USN consists of two dedicated OC192 SONET (9.6 Gbps) connections as shown in Figure 3, which are terminated on Ciena CDCI core switches. At Oak Ridge, Chicago and Sunnyvale nodes, Force10 E300 Ethernet switches are connected to the corresponding core switches. At Oak Ridge and Chicago core switches also provide 1GigE ports. A variety of data-plane connections can be provisioned using combinations of core and edge switches. SONET connections with bandwidth in the range, 150Mbps - 9.6Gbps, at OC3 (150Mbps) resolution can be provisioned using the core switches. 1GigE connections can be provisioned using OC21 connections between the core switches and cross-connecting them to their 1GigE ports using General Framing Protocol (GFP). Wide-area OC192 connections are provisioned by switching entire lambdas exclusively at core switches. 10GigE WAN-PHY connections are provisioned by terminating OC192 connections on edge switches at the ends; also, intermediate WAN-PHY connections may be provisioned by utilizing E300 switches at

those nodes. Connections switched at 10GigE LAN-PHY are realized by utilizing E300 switches to terminate the WAN-PHY connections and then suitably cross-connecting them to LAN-PHY ports. Thus, USN provides dedicated channels of various resolutions at distances ranging from few hundred miles to thousands of miles, which may be terminated on third party routers or switches or hosts. USN also provides Linux hosts connected to edge switches as shown in Figure 3 to support the development and testing of protocols, middleware, and applications.

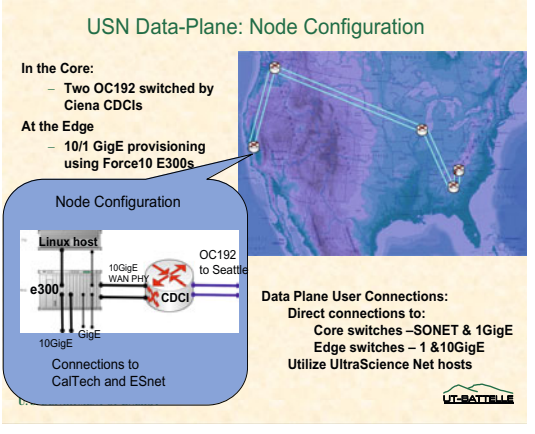
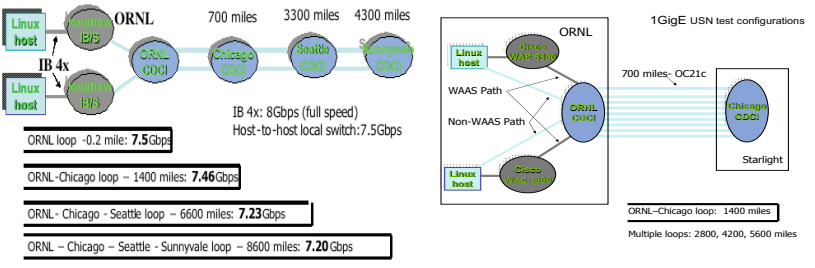


Fig. 3. USN data-plane

In conducting experiments to test the distance scalability of devices and protocols, suites of connections with varying lengths are provisioned between two fixed ports. A suite of 10Gbps connections of lengths 0, 1400, 6600 and 8600 miles are provisioned as shown in Figure 4(a) for InfiniBand experiments. By utilizing OC21 multiplexing we provision 1GigE non-interfering connections on a single OC192 connection, and by switching them at the ends realize several lengths. By using 700 mile dual OC192 connections between Oak Ridge and Chicago we create 1GigE connections with lengths from 0 to 12600 miles in increments of 1400 miles as shown in Figure 4(b). We developed automated scripts that dynamically cycle through all connections of a test suite by invoking a single script.

2.2 Control-Plane

USN control plane consists of the following components [23]: (a) client interface, (b) server front-end, (c) user management, (d) token management, (e) database management, (f) bandwidth scheduler, and (g) signaling daemon. USN control-plane software is implemented in C++ using CGI and PHP scripts for the server and JavaScript and HTML for user interface. It is deployed on a central management node on a Linux workstation at ORNL. The control-plane is implemented



(a) 10Gbps OC192 connections

(b) 1GigE GFP-mapped OC21 connections

Fig. 4. Suites of 1 and 10 Gbps connections provisioned on USN

using hardware-based VPN devices as shown in Fig. 5. Secure VPN tunnels are implemented using a main unit (Netscreen NS-50) at ORNL and secondary units (Netscreen NS-5) at each of the remote sites so that only authenticated and authorized traffic is allowed, and the traffic is encrypted. Each VPN tunnel carries three types of encrypted traffic flows: (i) user access to hosts, (ii) management access to hosts and switches, and (iii) the signaling messages to switches.

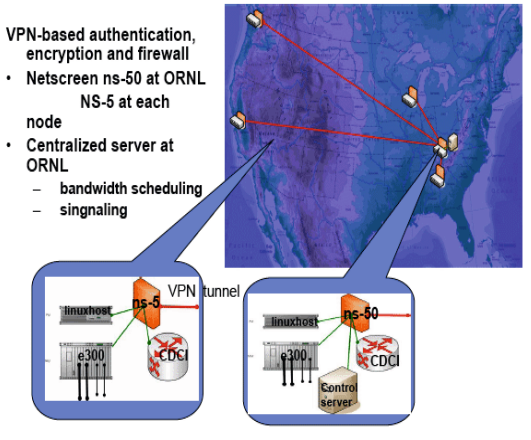


Fig. 5. USN control-plane

Based on network topology and bandwidth allocations, the scheduler computes a path as per user request. The signaling daemon executes scripts to set up or tear down the needed connections. The CDCI core switches are signaled using TL1 commands, and E300 edge switches are signaled using CLI commands. In both cases, EXPECT scripts are utilized by the signaling daemon to login via encrypted VPN tunnels to issue the commands. The bandwidth scheduling

algorithms are developed for: (i) a specified bandwidth in a specified time slot, (ii) earliest available time with a specified bandwidth and duration, (iii) highest available bandwidth in a specified time slot, and (iv) all available time slots with a specified bandwidth and duration. The first three algorithms are extensions of the classical Dijkstra's algorithm [8], and the last one is an extension of Bellman-Ford algorithm, which is an improvement over previous transitive closure algorithm [20]. USN control-plane has the first (2005) implementation of mathematically-validated advance scheduling capability, and more recently such capabilities have been developed in [4,31,2].

3 USN Network Experiments

In this section, we present a summary of experiments conducted on USN facility; their detailed accounts can be found in the references.

3.1 Hybrid Network Connections

Dedicated bandwidth connections may be provisioned at layers 1 through 3 or as combinations. For example, they can be MPLS tunnels over routed network as in ESnet [2], or Ethernet over SONET as in CHEETAH [33], or InfiniBand over SONET as in USN [5], or pure Ethernet paths [1]. An objective comparison of the characteristics of the connections using these technologies is important in making deployment decisions. Once deployed, the costs of replacing them could be very high, for example, replacing MPLS tunnels with SONET circuits entails replacing routers with switches. We collected measurements and compared the throughput and message delays over OC21C SONET connections, 1Gbps MPLS tunnels, and their concatenations over USN and ESnet.

For these experiments we utilized (a) OC21C connections of lengths 700, 1400, ..., 6300 miles on USN as described in the previous section, and (b) 1Gbps 3600 mile VLAN-tagged MPLS tunnel on ESnet between Chicago and Sunnyvale via Cisco and Juniper routers. USN peered with ESnet in Chicago as shown in Figure 6, and 1GigE USN and ESnet connections are cross-connected using E300 switch. This configuration provided hybrid dedicated channels of varying lengths, 4300, 5700, ... , 9900 miles, composed of Ethernet-mapped layer 1 and layer 3 connections. We collected throughput measurements using iperf and Peak Link Utilization Protocol (PLUT) over these connections.

For TCP, we varied the number of streams n from 1 and 10, and for UDP we varied the target rate as 100, 200, ..., 1000, 1100 Mbps; each set of measurements is repeated 100 times. First, we consider USN and ESnet connections of lengths 3500 and 3600 miles respectively and their concatenation. TCP throughput is maximized when n is around 7 or 8 and remained constant around 900, 840 and 840 Mbps for SONET, MPLS and hybrid connections, respectively. For UDP, the peak throughput is 957, 953 and 953 Mbps for SONET, MPLS and hybrid connections, respectively. Hence, there is difference of 60Mbps and 4Mbps between the TCP and UDP peak throughput, respectively, over SONET and

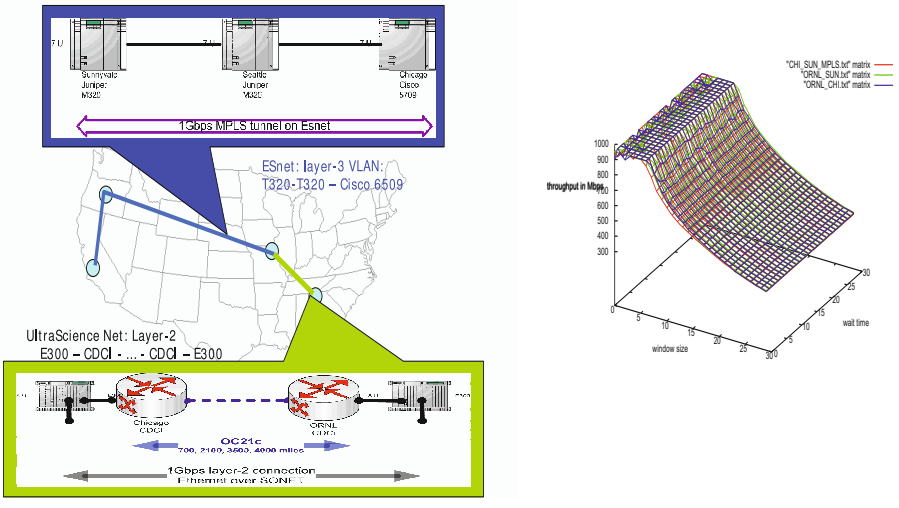


Fig. 6. USN switched OC21C connections and MPLS tunnels implemented using ESnet routers are peered using an Ethernet switch

MPLS connections. Also, there is a difference in peak throughput achieved by TCP and UDP in all cases, namely, 57 and 93 Mbps for SONET and MPLS connections, respectively. We measured file transfer rates over these connections using UDP-based PLUT, which achieved 955, 952 and 952 over SONET, MPLS and hybrid connections, respectively.

USN thus demonstrated that connections provisioned at layers 1-3 can be peered and carried across networks using VLAN technologies, and throughput and message delay measurements indicate comparable performance of layer 1, layer 3 and hybrid connections (detailed results can be found in [22]).

3.2 InfiniBand over Wide-Area

The data transport across wide-area networks has traditionally been based on 1/10GigE technologies combined with SONET or WAN-PHY technologies in the wide-area. InfiniBand was originally developed for data transport over enterprise-level interconnections for clusters, supercomputers and storage systems. It is quite common to achieve data transfer rates of 7.6 Gbps using commodity IB Host Channel Adapters (HCA) (SDR 4X, 8 Gbps peak) by simply connecting them to IB switches. However, geographically separated IB deployments still rely on transition to TCP/IP and its ability to sustain 7.0-8.0 Gbps rates for wide-area data transfers, which by itself requires significant per-connection optimization. Recently, there have been hardware implementations of InfiniBand over Wide-Area (IBoWA) devices, in particular Longbow XR and NX5010ae.

USN was among the first to conduct experiments that showed that these technologies could provide throughput far superior to TCP for dedicated high

bandwidth data transfers [24]. We utilize 0.2, 1,400, 6,600 and 8,600 mile connections and Longbow IBoWA devices in configurations shown in Figure 4 (a). Our results indicate that IB throughput of 7.6Gbps (4x) scales well (with 5% degradation) with no customization to 8600 mile 10GigE and OC192 connections as shown below:

connection length (miles)	0.2	1400	6600	8600	average
average throughput (Gbps)	7.48	7.47	7.37	7.34	7.47
std, dev (Mbps)	45.27	0.07	0.09	0.07	11.40
decrease per mile (Mbps/mile)	0	0.012	0.017	0.016	0.015

In contrast, various TCP (BIC, HTCP, HSTCP and CUBIC) achieved only a few Gbps on this connection. An additional benefit of IBoWA solution is that one could utilize native IB solutions to access remote files systems. However, this solution performed poorly under cross-traffic and dynamic bandwidth conditions, wherein cross-traffic levels of above 2Gbps degraded IB throughputs to about 1Gbps over 8600 mile connection. Thus this approach is mainly suited for dedicated high-bandwidth connections. This work illustrates that transport solutions for high-performance applications could be radically different from current TCP/IP based solutions. More details on this work can be found in [5,24,18].

3.3 Dedicated Connections to Supercomputers

The shared Internet connection from Cray X1 supercomputer at ORNL to North Carolina State University (NCSU) is provisioned at a peak rate of 1Gbps in 2006. But, the data path is complicated. Data from a Cray node traverses System Port Channel (SPC) channel and then transits to FiberChannel (FC) connection to CNS (Cray Network Subsystem) as shown in Figure 7. Then CNS converts FC frames to Ethernet LAN segments and sends them onto GigE NIC. These Ethernet frames are then mapped at ORNL router onto SONET long-haul connection to NCSU; then they transit to Ethernet LAN and arrive at the cluster node via GigE NIC. Thus the data path consists of a sequence of different segments: SPC, FC, Ethernet LAN, SONET long-haul, and Ethernet LAN. The default TCP over this connection achieved throughputs of the order 50 Mbps. Then bbpc protocol adapted for Cray X1 achieved throughputs in the range 200-300Mbps using multiple TCP streams. This low throughput is thought to have been the result of traffic congestion on the shared connection.

Since the capacity of Cray X1's NIC is limited to shared 1Gbps, we developed a dedicated interconnection configuration with 1Gbps capacity by using USN host and direct FC connections from Cray. Hurricane protocol that achieved 90% utilization on other 1Gbps connections was tuned for this configuration. But it only achieved throughputs of the order 400Mps when no jobs are running on Cray X1. Furthermore, its throughput degraded to 200Mbps as jobs are brought online. The throughput problem was diagnosed to the inadequate CPU cycles being allocated to TCP stack, and the network connection had not been the main bottleneck from the start.

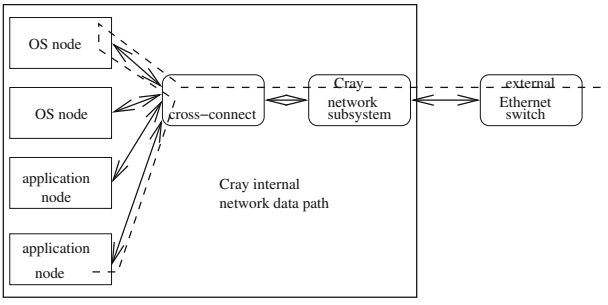
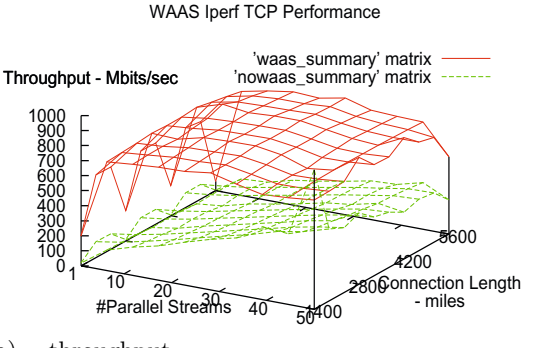


Fig. 7. Dedicated channel to Cray X1 supercomputer

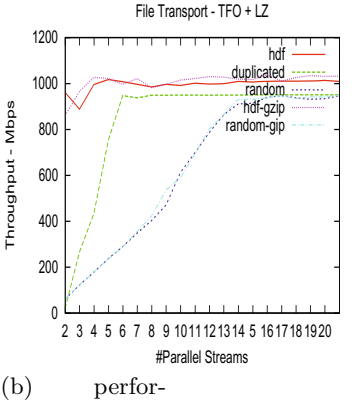
3.4 Wide-Area Application Accelerators

To address the challenges of optimizing the wide-area transport performance, new generation devices are being developed that can simply be "dropped-in" at the network edges. These devices transparently optimize the network flow performance using a combination of Data Redundancy Elimination (DRE) [27], TCP Flow Optimization (TFO) [26] and network data compression [29]. Cisco Wide Area Application Services (WAAS) products are examples of these new technologies. We conducted experiments to quantify the performance of Cisco WAE-500 and WAE-7300 series devices over 1Gbps connections.

We measured iperf TCP throughputs on paths with and without WAAS devices at the ends, called WAAS and non-WAAS paths respectively. We used various connection lengths shown in Figure 4(a) and varied the number of parallel streams from 1 to 50. The relative performance of WAAS and non-WAAS



(a) throughput profiles of WAAS (top) and non-WAAS (bottom)



(b) performance of TFO and LZ on 1400 mile connection

Fig. 8. Performance of WAAS devices

paths for default TCP are summarized in Figure 8. For single streams, WAAS throughput is higher by at least factor of 10 and was as high as 40 in some instances as shown in 8(a). For most multiple streams, WAAS paths throughputs are at least three times higher than non-WAAS paths. The two distinguishing features of WAAS paths are: (a) throughputs reached their maximum with about 5-10 streams, where as non-WAAS paths needed 40 or more streams to reach same levels, (b) WAAS path throughputs were not monotonic in the number of streams unlike the non-WAAS paths.

To study the effects of file contents, we measured file transfer throughputs using iperf with -F option for three different types of files over 1400 mile 1Gbps connection: (a) file with repeated bytes, (b) file with uniformly randomly generated bytes, and (c) supernova simulation files in hdf format. We also gziped these files and utilized them in iperf measurements; gzip implements Limpel-Ziv (LZ) compression on the entire file unlike the incremental implementation on WAE devices. In case (a), gziped file is highly compressed to about 1030 times smaller than the original size, and in case (c) compressed file size is about 0.6831 times the original size. In case (b), however, the gziped file is larger by 0.01% since the file contents were not compressible and the header added to the size.

TFO achieved the best performance for hdf files with throughputs exceeding 1Gbps with 3-6 streams. Least performance improvements are observed for files with repeated contents and random contents. TFO combined with LZ achieved the best performance for hdf files with throughputs exceeding 1Gbps (1017Mbps) with 4-5 streams as shown in Figure 8(b). This performance is about the same order as TFO alone. Least performance improvements are observed for files with random contents, but the performance is much higher than using TFO alone but somewhat lower than using TFO and DRE. In particular, throughputs of 900Mbps were achieved with more than 13 streams, whereas TFO-DRE achieved 950 Mbps with 7 streams. But TFO-LZ performed better than using TFO alone which could sustain 852Mbps with 13 or more streams. USN experimental results can be summarized as follows: (i) highest and lowest throughputs are achieved for hdf and random data files, respectively; (ii) most throughputs were maximized by utilizing 5-10 parallel TCP streams; and (iii) pre-compression of files using gzip did not have a significant effect. In all cases, throughput measurements varied when experiments were repeated.

3.5 IP Encryption Devices

Recent High Assurance Internet Protocol Encryptor (HAIPE) devices are designed to provide 10 Gbps encrypted IP traffic flows. HAIPE is Type 1 encryption device that utilizes cryptography Suites A and B for encrypting IP packet flows. HAIPE's interoperability specification is based on IPsec with additional restrictions and enhancements. They act as gateways between two enclaves to exchange data over an untrusted or lower-classification network. HAIPE device looks up the destination IP address of a packet in its internal Security Association Database (SAD) and picks up the encrypted tunnel based on the appropriate entry. They use internal Security Policy Database (SPD) to set up

tunnels with appropriate algorithms and settings. We generate the performance profiles for TCP and UDP with and without encryption devices. We enabled jumbograms consistently at all devices on the encrypted path, which achieved better performance compared to unencrypted path: (a) for connections 1400 miles and shorter, same throughput levels as unencrypted case were achieved with less number of parallel streams, and (b) throughput improved by more than 50% for longer connections. Thus for TCP, the encryption devices have an effect equivalent to reducing the end-to-end latency which in turn increases the throughput.

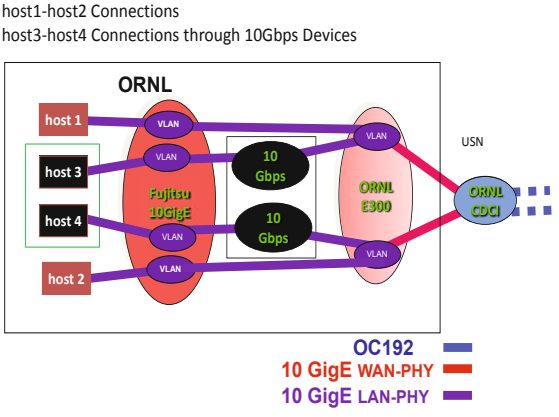


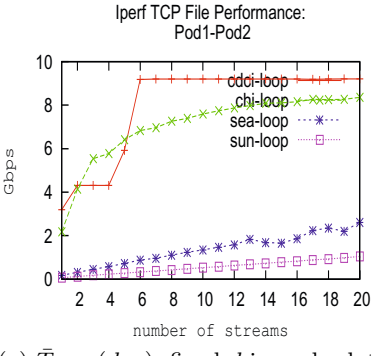
Fig. 9. Wide-area connections with and without HAIPE devices

The hosts are connected to Fujitsu 10 GigE switch which provides multiple connections to E300 switch. First pair of connections are via HAIPE devices as shown in Figure 4, and the second pair are direct connections. By appropriately utilizing VLANS as shown in Figure 9 we realize wide-area connections between pairs of hosts with and without HAIPE devices to realize encrypted and plain connections, respectively. Thus this setup enables side-by-side comparison of the performance of encrypted and plain connections. The same measurement software is used in both cases, and IP address scheme is used to direct the flows onto the encrypted and plain connections as needed. The same wide-area connection is used for both types of traffic, and the experiments are mutually exclusively scheduled so that only one type of traffic is allowed during each test.

We generate throughput profiles for TCP and UDP between hosts connected over connections of different lengths to characterize the achievable throughputs and the corresponding configuration parameters. For TCP, let $T_{TCP}(d, n)$, denote the throughput measurement for files transfers using iperf with -F option over connection of length d using n TCP streams. Let $\bar{T}_{TCP}(d, n)$ denote the average TCP throughput over 10 repeated measurements. TCP distance-profiles are generated by measuring throughputs $T_{TCP}(d, n)$ for the number of parallel

streams n from 1 to 20, for connection length $d = 0.2, 1400, 6600, 8600$ miles. To collect the *stability-profile* for fixed connection length d, we repeat throughput measurements 10 times, for $n = 1, 2, \ldots 20$ for TCP, and $r = 4, 5, 6, 10$, 11 Gbps.

We show the performance profiles without encryptors in Figure 10. For local connections, TCP throughput of 9.17 Gbps was achieved with 6 parallel streams, and it was 8.10 Gbps over 1400 mile connection with 15 streams. However, the throughput was about 1Gbps for 8600 mile connection even with 20 streams. These results were produced with BIC congestion control [32], and similar results with other congestion control modules are described in [24].



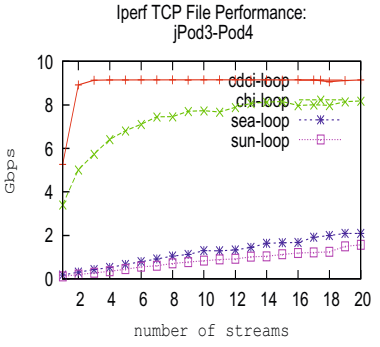
(a) $\bar{T}_{TCP}(d, n)$, fixed d in each plot

miles	0.2	1400	6600	8600
throughput (Gbps)	9.17	8.10	2.61	1.04
# flows	6	15	20	20

(b) TCP file transfer throughputs

Fig. 10. TCP performance over unencrypted connections

We collected iperf measurements over when encryptors are included, and the results are shown in Figure 11. There are several interesting observations. (i) for local connection 9Gbps throughputs were achieved with 2-3 parallel streams compared to 6 streams needed for unencrypted connection; (ii) for 1400 mile connection, 8Gbps throughput was achieved with about 12 parallel streams compared to about 20 for unencrypted case; and (ii) for 8600 mile connection, 1.57 Gbps throughput was achieved with 20 streams compared to about 1Gbps for the unencrypted connection. In summary, the TCP file transfer throughputs were higher when encryptors were employed for the same number of parallel streams as shown in Figure 12. Such throughput improvement is consistently present in all performance profiles at all tested connection lengths. This performance improvement is attributed to the availability of buffers on HAIPE devices which smoothens TCP dynamics and has an effect similar to shortened RTT.



(a) $\bar{T}_{TCP}(d, n)$, fixed d in each plot

miles	0.2	1400	6600	8600
throughput (Gbps)	9.12	8.06	3.11	1.57
# flows	3	13	29	20

(b) TCP file transfer throughputs

Fig. 11. TCP performance over unencrypted connections

files	0.2	1400	6600	8600
encrypted (Gbps)	9.12	8.06	3.11	1.57
plain (Gbps)	9.17	8.10	2.61	1.04

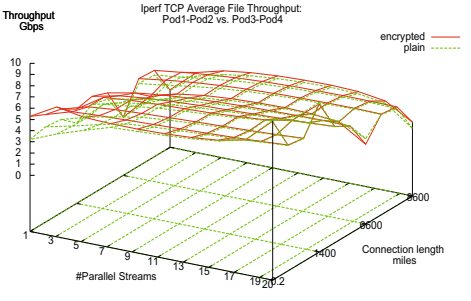


Fig. 12. Comparison of TCP over encrypted and plain connections

4 Conclusions

UltraScience Net is a national-scale testbed conceived and built in support of the development of high-performance networking technologies. It developed and/or tested infrastructure technologies needed for national-scale network experimental facilities, and also supported a number of high-performance research network experiments.

It would of interest to complement USN's experimental facilities with capabilities to (a) realize user-specified network footprints, and (b) analyze, interpolate and extrapolate the collected measurements. It would be beneficial to develop methods to map and realize a user-specified target network on USN infrastructure as closely as possible; such capability enables the testing of national-scale networks which are more general than the connection suites described here.

Despite the availability of connection suites, there are limits on the lengths of connections that can be physically realized on USN. It would be interesting to develop the theory and tools for the design of network experiments to: (i) identify a set of probe connections to optimize the cost and information from the measurements collected, and (ii) suitably interpolate or extrapolate the measurements to predict the performance at desired connection lengths and also derive qualitative performance profiles.

Acknowledgments

This work is currently sponsored by U. S. Department of Defense and is carried out at Extreme Scale Systems Center, Oak Ridge National Laboratory managed by UT-Battelle, LLC for U. S. Department of Energy under contract No. DE-AC05-00OR22725.

References

1. Dynamic resource allocation via GMPLS optical networks, `http://dragon.maxgigapop.net`
2. On-demand secure circuits and advance reservation system, `http://www.es.net/oscars`
3. Akari architecture design project for new generation network, `http://akari-project.nict.go.jp`
4. Audouin, O., Cavalli, A., Chiosi, A., Leclere, O., Mouton, C., Oksman, J., Pasin, M., Rodrigues, D., Thual, L.: CARRIOCAS Project: an experimental high bit rate optical network tailored for computing and data intensive distributed applications. In: Proc. Tridentcom (2009)
5. Carter, S.M., Minich, M., Rao, N.S.V.: Experimental evaluation of infiniband transport over local and wide-area networks. In: Proceedings of High Performance Computing Conference (2007)
6. End-To-End Provisioned Optical Network Testbed for Large-Scale eScience Application, `http://www.ece.virginia.edu/mv/html-files/ein-home.html`
7. China next generation network, `http://www.cernet2.edu.cn/en/bg.htm`
8. Cormen, T.H., Leiserson, C.E., Rivest, R.L.: Introduction to Algorithms. Hill Book Co. McGraw-Hill, New York (1990)
9. Energy Sciences Network, `http://www.es.net`
10. Federated e-infrastructure dedicated to european researchers innovating in computing network architecture, `http://www.fp7-federica.eu`
11. Future internet research and experimentation, `http://cordis.europa.eu/fp7/ict/fire`
12. GENI: Global environment for network innivations, `http://www.geni.net`
13. Hybrid Optical and Packet Infrastructure, `http://networks.internet2.edu/hopi`
14. Internet2, `http://www.internet2.edu`
15. NSF Shared Cyberinfrastructure Division PI Meeting, February 18-20 (2004), `http://hpn.east.isi.edu/nsf-sci`

16. Rao, N.S.V., Wing, W.R., Hicks, S.E., Poole, S.W., Denap, F.A., Carter, S.M., Wu, Q.: Ultrascience net: High-performance network research test-bed. In: International Symposium on on Computer and Sensor Network Systems (2008)
17. Rao, N.S.V., Poole, S.W., Hicks, S.E., Kemper, C., Hodson, S., Hinkel, G., Lothian, J.: Experimental study of wide-area 10gbps ip transport technologies. In: Proc. of Milcom (2009)
18. Rao, N.S.V., Poole, S.W., Newman, P., Hicks, S.E.: Wide-area Infiniband RDMA: Experimenal evaluation. In: Workshop on High-Speed Interconnects for Distributed Computing (2009)
19. Rao, N.S.V., Poole, S.W., Wing, W.R., Carter, S.M.: Experimental analysis of flow optimization and data compression for tcp enhancement. In: INFOCOM 2009 Workshop on Terabits Networks (2009)
20. 20. N. S. V. Rao, W. R. Wing,, S. M. Carter, and Q. Wu. Ultrascience net: Network testbed for large-scale science applications. IEEE Communications Magazine, expanded version (2005) (in press), http://www.csm.ornl.gov/ultranet
21. Rao, N.S.V., Wing, W.R., Carter, S.M., Wu, Q.: High-speed dedicated channels and experimental results with hurricane protocol. Annals of Telecommunications 61(1-2), 21–45 (2006)
22. Rao, N.S.V., Wing, W.R., Wu, Q., Ghani, N., Lehman, T., Dart, E., Guok, C.P.: Measurements on hybrid dedicated bandwidth connections. In: INFOCOM 2007 Workshop on Terabits Networks (2007)
23. Rao, N.S.V., Wu, Q., Carter, S.M., Wing, W.R., Ghosal, D., Banerjee, A., Mukherjee, B.: Control plane for advance bandwidth scheduling in ultra high-speed networks. In: INFOCOM 2006 Workshop on Terabits Networks (2006)
24. Rao, N.S.V., Yu, W., Wing, W.R., Poole, S.W., Vetter, J.S.: Wide-area performance profiling of 10gige and infiniband technologies. In: Pautasso, C., Tanter, É. (eds.) SC 2008. LNCS, vol. 4954. Springer, Heidelberg (2008)
25. Sahni, S., Rao, N.S.V., Li, Y., Jung, K., Ranka, S., Kamath, N.: Bandwidth scheduling and path computation algorithms for connection-oriented networks. In: Proceedings of International Conference on Networking (2007)
26. Semke, J., Madhavi, J., Mathis, M.: Automatic TCP buffer tuning. In: Proc. ACM SIGCOMM 1998 (1998)
27. Spring, N.T., Wetherall, D.: A protocol independent technique for eliminating redundant network traffic. In: Proceedings of the 2000 ACM SIGCOMM Conference (2000)
28. Tierney, B.L.: The ARRA ANI Network Testbed Project. In: JointTechs Meeting (2010)
29. Tye, C., Fairhurt, G.: A review of ip packet compression techniques. In: Proc.GNet (2003)
30. User Controlled LightPath Provisioning, http://phi.badlab.crc.ca/uclp
31. Varvarigos, E., Sourlas, V., Christodoulopoulos, K.: Routing and scheduling connections in networks that support advanced reservations. Computer Networks 52, 2988–3006 (2008)
32. Xu, L., Harfoush, K., Rhee, I.: Binary increase congestion control (bic) for fast long-distance networks. In: INFOCOM (2004)
33. Zheng, X., Veeraraghavan, M., Rao, N.S.V., Wu, Q., Zhu, M.: CHEETAH: Circuit-switched high-speed end-to-end transport architecture testbed. IEEE Communications Magazine (2005)

Interoperability in Heterogeneous Resource Federations

Sebastian Wahle[1], Thomas Magedanz[2], and Konrad Campowsky[2]

[1] Fraunhofer FOKUS
Kaiserin-Augusta-Allee 31, 10589 Berlin, Germany
sebastian.wahle@fokus.fraunhofer.de
[2] Technische Universität Berlin, Lehrstuhl Architekturen der Vermittlungsknoten (AV)
Franklinstr. 28-29, 10587 Berlin, Germany
frollen@cs.tu-berlin.de, tm@cs.tu-berlin.de

Abstract. Currently, a number of research activities worldwide focus on resource federations to enable remote resource access and the setup of large scale experiments. With the global efforts in sharing experimental facility resources across organizational boundaries, interoperability issues are becoming increasingly important. To this end, this paper describes a model that defines the necessary entities for inter resource federation scenarios. The model is then discussed by means of a concrete use case federating Panlab and PlanetLab resources. In this respect Panlab and PlanetLab are considered to be federations themselves. Our findings show that the two approaches can be matched following the general model. This allows for federating resources not only across the boundaries of administrative (organizational) domains but even across the boundaries of federations. Those federations in themselves provide collections of resources offered by several administrative domains and use different control frameworks to enable resource access and management.

Keywords: Resource Federation, Model, Panlab, PlanetLab, SFA, Teagle, Interoperability.

1 Introduction

The current trend of federation is followed by several research projects and programs worldwide. While the concept of federation can be applied to a number of fields such as identity management, networking, trust, and security, in the context of this paper we concentrate on the federation of testbeds and experimental facilities. Generally, a federation is understood to be an organization within which smaller divisions have some internal autonomy (Oxford definition). Merriam-Webster defines federal as: (1) formed by a compact between political units that surrender their individual sovereignty to a central authority but retain limited residuary powers of government; (2) of or constituting a form of government in which power is distributed between a central authority and a number of constituent territorial units.

This concept, clearly stemming from a political background, is also applied in the context of our research in sharing hardware and software resources across the borders

T. Magedanz et al. (Eds.): TridentCom 2010, LNICST 46, pp. 35–50, 2011.

of individual administrative domains. This is important because federation enables combining infrastructural network resources and services of more than one independently controlled domain which enhances the utility of testbeds significantly. This is true for the following reasons: access can be given to more resources, increasing the scale of experiments. Furthermore, individual testbeds may include unique infrastructural resources or configuration properties that allow experimenters to execute new kinds of experiments. Finally, because testbeds act as gathering points for experimenters in a given field, combining testbed resources can promote collaboration between very different communities (e.g. Internet community and Telco community) and research groups [1].

Furthermore, with the speed of network convergence and technology evolution, today's ICT systems require sophisticated heterogeneous experimental network infrastructures to design and test new solutions. The problem is that infrastructure lifetime – the time an infrastructure remains at technology's cutting edge – has decreased dramatically, making investments in expensive isolated research infrastructure more risky than they were already. This is especially true for complex cross-layer and cross-technology testbeds.

In this regard, federation can play a major role in the worldwide research activities that are currently under way to investigate alternative solutions and design the architecture of a so-called Future Internet (FI). The FI architecture discussion has been triggered by the fact the original Internet design and its protocols were devised in the Seventies and numerous fixes and extensions have been introduced since then to address a variety of problems such as scalability and security. This led to a highly heterogeneous landscape of different protocols and technologies combined with a significant increase in the overall system complexity. At the same time the Internet is still a best effort network and Quality of Service (QoS) guarantees are hard to realize. In addition to the development of innovative foundational Internet architectures, the setup and provisioning of large scale testbeds and experimental facilities is considered to be of major importance in international research in order to develop, test, and validate FI research results. Therefore, large research programs have been launched to address both the development of new Internet architectures as well as suitable experimental facilities and test environments. Examples in this context are the United States NSF programs GENI (Global Environment for Network Innovations) [2] and FIND (Future Internet Design) [3] as well as the European FIRE (Future Internet Research & Experimentation) [4],[5] initiative. The focus of GENI is on the design of experimental platforms whereas FIND is mainly addressing foundational concepts and methods for the Future Internet. In the context of GENI, there are currently five competing testbed control frameworks (TIED [6],[1], PlanetLab [7], ProtoGENI [8], ORCA [9], ORBIT [10]) under development that are organized in clusters. In the FIRE context, several projects (e.g. Onelab2 [11], Federica [14], PII [12],[13]) are contributing to the experimental facility. In Asia similar programs have been launched such as AKARI [15] in Japan. Joint Asian activities are carried out under the APAN (Asia-Pacific Advanced Network) [16] initiative, the Asia Future Internet Forum (AsiaFI) [18] as well as PlanetLab CJK (China, Japan, Korea), a joint PlanetLab cooperation by China, Japan, and Korea. An in-depth discussion and comparison between the different control framework approaches for experimental facilities has been published earlier by the authors [19].

This article proposes a generic model for federation and demonstrates an approach for federating resources between two of the above mentioned control frameworks, PlanetLab and Panlab. The implementation is partly based on the Slice Based Facility Architecture (SFA) outlined in [20]. SFA has been chosen, on the one hand because of its aim to support generic resource types and on the other hand because an implementation of it already exists for the PlanetLab federation.

The paper is organized as follows: First, an introduction to the federation model is given as well as a short comparison of the scope and some architectural aspects of the PlanetLab and Panlab control frameworks, along with an introduction to SFA. Then, a use case scenario and an implementation that satisfies the requirements of this scenario are described. Finally, we discuss the implementation along with the challenges faced and decisions taken and evaluate our findings.

It is important to note that this article focuses on issues regarding the technical design and infrastructural layout of the respective control frameworks and a possible federation between them, leaving aside organizational, business, and legal aspects.

2 Federation Model

In the introduction we gave a definition of federation which has been motivated by a political background. Such definitions are based on the concept of surrendering individual sovereignty to a central authority. This general understanding is extended in our problem field as resource federations “at eye level” are possible.

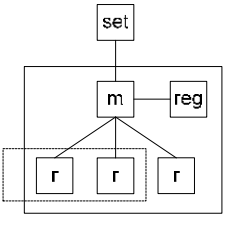


Fig. 1. Federation model entities

However, independent of the level of surrender there are similar functional entities that need to be provided to enable cross-domain and cross-technology federation concepts. The entities are shown in figure 1 and are listed below:

- Resources (r)
- Virtual grouping of resources (dotted rectangle)
- Domain managers (m)
- Registries (reg)
- Creation / setup tools (set)
- Administrative authorities (solid rectangle)

In the following we will apply different levels of "surrender" to depict several federation scenarios. The first scenario shown in figure 2 on the left hand side is what we call the "full surrender" scenario where the resources committed from domain B can be controlled via domain A. An example of the full surrender scenario is the Panlab Federation [12] where all Panlab domains allow TEAGLE, the central composition engine, to control resources inside their domain using a central registry wher resource from all participating domains are registered. Also, earlier approaches such as [21] can be modeled in this way.

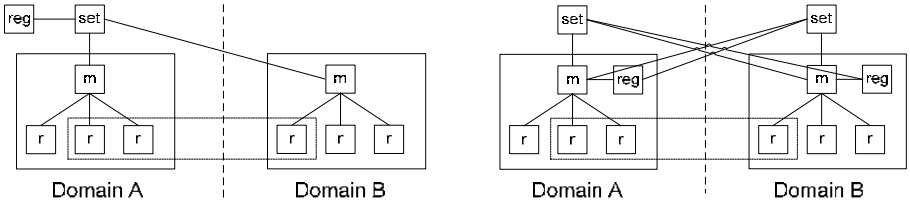


Fig. 2. Left: full surrender scenario, right: federation at eye-level scenario

The scenario shown in figure 2 on the right hand side is what we call the "federation at eye level" scenario as the participating domains allow the mutual control of resources across the borders of the individual domains. This scenario has been implemented to federate Panlab resources and resources from a private PlanetLab installation and will be explained in detail in the following sections.

Several other scenarios that are somewhere in between the both extreme scenarios explained above are possible and might be applied to satisfy various constraints and requirements in specific federation contexts. An example might be that only the registries of domains are shared allowing users to "roam" between domains and use resources from various domains without combining them across the borders of those domains.

The model has been influenced by existing federation approaches such as PlanetLab, the SFA, and Panlab.

3 PlanetLab, Panlab and SFA

This section introduces the previously mentioned frameworks and highlights the architectural concepts that are important for our further analysis and implementation of a federation scenario with mutual resource sharing between the top-level authorities.

3.1 PlanetLab Control Framework

PlanetLab is one of the platforms working under the umbrella of the GENI initiative and is being extended in this context to meet FI research infrastructure requirements. It is a global research network in the form of a distributed computing platform,

designed to support the development of new network services. The actual control framework software used by PlanetLab is called PlanetLab Central (PLC). Its primary focus lies on multiplexing the computing resources of nodes (servers) through distributed virtualization. Nodes run a minimal version of a UNIX operating system, currently the Linux flavor Fedora Core 8 (FC8) [22] and are divided into virtual containers. At present, PlanetLab is employing the Vserver [23] technology to perform this virtualization though other solutions could be used as well. The resources (reserved CPU percentage, memory and bandwidth allocations, etc.) bundled by such a virtual container are collectively called a sliver. PLC then groups these slivers into slices which are owned and subsequently administered by the party requesting the instantiation of a slice. Note that a sliver always belongs to exactly one distinct slice. Figure 3 shows this concept:

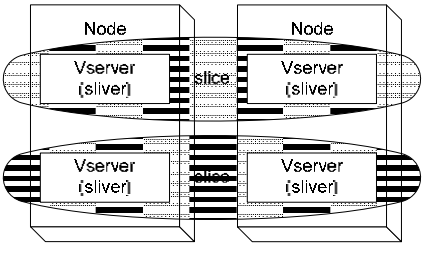


Fig. 3. Logical schematic of the PlanetLab architecture

Slices could also be described as virtual testbeds spanning parts or even the whole of the PlanetLab network of nodes. From the perspective of researchers, a slice is a *substrate-wide network of computing and communication resources capable of running an experiment or a wide-area network service* [24]. Slices also serve as the primary entity for accounting and accountability. The consuming of resources on individual nodes can be traced to a distinct slice as well as the possibly malicious behavior of experiments or other programs running within the network.

Furthermore, PLC allows the party owning a slice to grant individual researchers (users) access to it. A user associated with a slice always has access to all the slivers it comprises. Researchers can use their slices to run experiments, either by directly administering the associated slivers through SSH or by using one of several overlay technologies developed by PlanetLab or third parties [25]. Additionally, PlanetLab offers a number of operations on slices to its users, for example centralized monitoring facilities or the ability to reboot or reset all slivers in a slice.

3.2 Panlab Control Framework

One important conceptual difference between PlanetLab and the Panlab approach is that while PlanetLab's focus is centered on computing resources (nodes), Panlab aims at being a truly generic design which is able to deploy, manage, and subsequently offer arbitrary resources and services. These range from infrastructure devices like routers or VPN gateways to logical resources, for example user accounts or relations in a database. Panlab also includes computing resources as offered by PlanetLab.

The fundamental technical management authority of a Panlab domain is a Panlab Testbed Manager (PTM) which has been described in more detail in [26]. A PTM's main functionality is the execution of generic provisioning operations on resources under its control. These resources are distinctly typed, uniquely named, and can expose further specific operations in addition to PTM's generic operations. Resources follow a dynamic lifecycle; resource types can be instantiated, these instances can be worked with, (re-)configured, and finally de-provisioned and removed from the system. A high level overview of a Panlab domain and its PTM is given in figure 4.

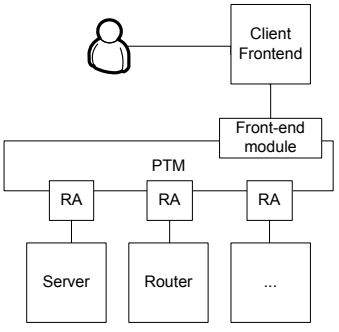


Fig. 4. Structural overview of a PTM domain

Given the high heterogeneity of resource types a PTM has to support, it cannot be directly aware of all resource specific semantics. Thus, an abstraction layer is necessary to allow for common management operations. This abstraction layer is made up of so-called resource adapters (RA) to which the PTM delegates the task of addressing resource specific types of communication. These resource adapters can be viewed as device drivers in the sense that they possess detailed knowledge about semantics of the resources instances they are responsible for. At its core level, a PTM itself is completely unaware of this nature and acknowledges resource instances merely as fundamental entities.

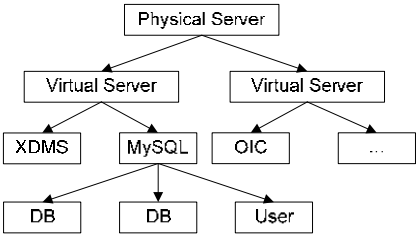


Fig. 5. Logical view of PTM's containment hierarchy

The entities are organized in a strictly hierarchical manner; a resource instance that is not at the root of the resource hierarchy has exactly one distinct parent instance and is thought of as being contained within its parent. An example of this structure is visualized in figure 5 where the arrows represent containment. The physical server at

the root of the hierarchy contains two virtual servers (Xen machines [27]) which in turn contain deployments of different software packages. In this example, these are the Fraunhofer FOKUS XML Document Management Server (XDMS) [28], an Open IMS Core (OIC) [29] and an instance of a MySQL database system [30]. The administrative issues of the MySQL instance are represented as further resources, namely individual database and database user (access credentials) instances. It must be noted, that a PTM does currently not support mechanisms for user management or access control. It assumes a trusted relationship with a client front-end to which it offers the entirety of its services through one or more front-end modules. Currently, this is the TEAGLE portal [31] which in turn provides a Virtual Customer Testbed (VCT) design tool to allow the design and provisioning of arbitrary testbed layouts. However, it is expected that additional mechanisms are needed to allow different domains to define specific access policies. This is currently under development.

3.3 Slice Based Facility Architecture

The Slice Based Facility Architecture is one of the key parts in the high level architecture of the GENI initiative. SFA defines a minimal set of interfaces and data-types to allow a federation of slice-based network substrates to interoperate. This specification was designed to allow federation among facilities like PlanetLab, Emulab [32], and other GENI frameworks. However, it is intended to support a much broader range of heterogeneous resources and services than those systems currently embody. SFA's general layout and abstractions closely resemble those of PLC. In fact, the interfaces the SFA defines represent a subset of the operations PLC exposes. Thus, PlanetLab entities can usually be directly mapped to SFA entities and vice versa, as illustrated by figure 6:

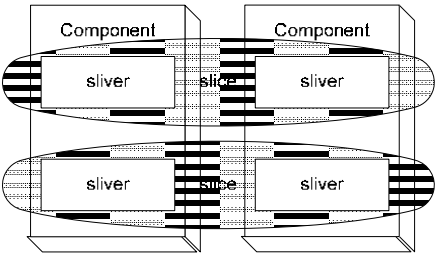


Fig. 6. Logical schematic of the SFA

The fundamental building block of the SFA is a component (CM). Components could for example correspond to physical servers, customizable routers, or access points. Their direct counterparts in PLC are nodes, which would be viewed as components in SFA. Components encapsulate a collection of resources which might include physical (for example CPU and memory allocation) as well as logical ones such as port numbers or access rights. The resources a component possesses are described in a resource specification (RSpec), the exact format of which has yet to be finalized. It is to be noted that resources don't necessarily have to be contained in a single physical device but might also be distributed across a set of devices which would

however be viewed as a single component by the SFA. A given resource always belongs to exactly one component.

The resources a component comprises can be multiplexed for use by multiple users. This can either happen via a form of virtualization, where a user is granted a virtual copy of the component's resources or by partitioning the component into several distinct resource sets. Regardless of how it is acquired, such a set of resources is called a sliver. The slivers (Vservers) created by PLC on PlanetLab nodes directly correspond to slivers in the SFA. Just like PLC does, SFA then bundles these slivers into slices. Again, a sliver always belongs to exactly one slice. Slices in SFA perform the same functions as they do in PLC, which is being the basis for accounting and accountability. Researchers who are granted access to a slice likewise gain access to all the resources represented by the slivers the slice is composed of.

Collections of components can be represented by a single aggregate. Such an aggregate can be accessed via an aggregate manager (AM) which exposes the same interface as the respective components themselves and delegates requests towards them. In the current PlanetLab implementation, central functions like the creation of slices or user management are exposed to administrators and users via a slice manager (SM), though this is not strictly defined as part of the standardized SFA interfaces. The slice manager will use its local registry to perform look-ups and is furthermore configured to know all aggregate managers of foreign parties which it will contact in order to issue provisioning requests.

4 Resource Management and Federation

In this section we discuss the proposed federation scenario ("at eye-level") and give insights into general decisions made.

4.1 Federation Scenario

The SFA describes a number of different usage scenarios, regarding different levels of federation between partners; ranging from simply providing a different user interface over shared registry services to full federation between two or more parties on equal terms. This relates to the general federation scenario shown in figure 2 on the right hand side. The scenario we are examining within the scope of this paper is that of a full federation between a private PlanetLab and a Panlab domain. Here, both parties maintain their own registry services and both parties also provide a domain manager to allow the other party to provision resources in their respective domain.

Users and administrators interact with their local domain managers (PTM and slice manager) to create and manage slices which may span both federations. The slice managers delegate look-up requests to their local registry which will in turn query the foreign registry if necessary. Provisioning requests are issued directly to the respective domain managers which forward these requests to their components. The Panlab federation mechanisms have been slightly adapted for this scenario (implementation of an aggregate manager module and lightweight SFA registry) to support the PLC and SFA mechanisms. Eventually, if the proposed federation model will be accepted and standardized APIs and interfaces will be present, this scenario will work without one side adapting to the mechanisms of the other side. However, even today the

implementation of an adaptor module making the Panlab side "PlanetLab-compatible" was feasible in terms of time investments given the modular structure of both approaches and a certain overlap in the architecture design.

4.2 Design Considerations

Our solution for realizing the above scenario takes into consideration the envisioned federation benefit and utility for both sides. Please note, that for the full federation scenario to become true, such a federation must not simply provide any level of integration and interaction, but must rather enable both parties to fully incorporate the committed resources and services into their own schema without further adjustments.

From the PlanetLab perspective, this means that PLC must be able to facilitate the same services and access rights for Panlab nodes as for PlanetLab nodes. Researchers accessing the federation via PlanetLab's slice manager must be able to request slivers from Panlab components, add them to their slices and access them in just the same way they would interact with nodes provided solely by PlanetLab itself. That implies that a PTM, or rather the resource adapter providing virtual nodes for that matter, must have a certain awareness of the requirement to deploy a PlanetLab specific flavor of a virtual node. As not all resources from the Panlab framework and its partners will be available to be shared under such circumstances, specific operational processes and agreements will be needed to manage resource access and allow for different levels of commitment for individual domains and resources. The Panlab operational framework, which is currently under development, is intended to deal with such issues by defining reference point governance processes and contracts.

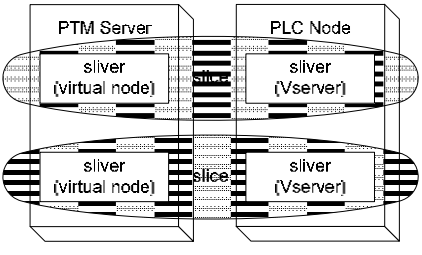


Fig. 7. PLC view on federated resources

As shown in figure 7, the SFA and through it PLC acknowledges resources provided by PTM as mere slivers and has no regard for any PTM internal layout. In the example above, these slivers represent either virtual nodes on one of PTM's physical servers or Vservers on one of PLC's nodes, both of which can equally act as slivers in a PLC slice. The federation partner that has requested these resources as well as their users can utilize them transparently, as if they were their own while being completely agnostic to the fact that they are actually provided by a different organization.

On the other hand, the PTM must also be enabled to make use of slivers provided by PlanetLab or other parties in the same way it would utilize resources under its direct control. Although SFA and PLC do not directly support PTM's notion of a

resource hierarchy, this requires little efforts. Since PLC already provides sufficient facilities (in the form of unrestricted administrative access) to allow arbitrary operations on slivers, no adjustment is needed for the PlanetLab side of the federation. PTM merely has to deploy its own management back-ends on PLC slivers, in the form of appropriate resource adapters. This already happens for PTM's own virtual nodes and the scheme can be easily adjusted to work on PLC's Vservers.

Figure 8 shows a PLC provided Vserver embedded into PTM's resource hierarchy. Note, that the example shown introduces a purely virtual *SFA* resource instance at the root of the hierarchy that bundles all slivers acquired from the SFA as children under its own hierarchy. However, this is not a technical necessity and serves only an organizational purpose.

Afterwards, the resources provided through SFA, in this example a PLC Vserver, integrate seamlessly into PTM's resource hierarchy and can be further utilized just like any other PTM resource including the possibility of deploying further subresources. Likewise, resources provisioned this way could be included into a VCT booked via the TEAGLE portal website.

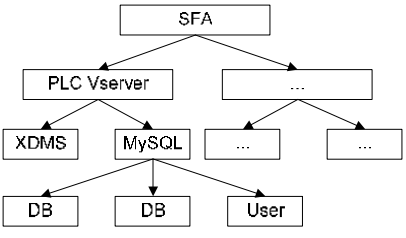


Fig. 8. PTM view on federated resources

In a further iteration, PTM's mentioned capability of creating further child resources inside a sliver acquired from the SFA could also be exposed towards other federation partners. To do so, PTM would have to advertise the sliver it was given by the SFA as another component through its aggregate manager. This virtual component would contain resources that represent the resource types PTM can deploy into the sliver. These resources could again be grouped into a (sub-)sliver and added to a slice.

However, this is not without side effects. For example one would have to define semantics for what is to happen with sub-slivers when the slivers they are contained in are being deleted. Apart from these considerations, this endeavor would currently be bound to fail for the profane reason that the current SFA implementation caches for a certain time the resources offered by an aggregate. Thus, it would be unpredictable when dynamically added resources would be picked up and made available.

5 Use Case and Prototype Implementation

This section defines a use case to be fulfilled by the implementation and gives a deeper insight into its internal architecture and layout. We will name the technologies

used and highlight key parts of the design to explain their role in the concept presented here.

5.1 Use Case Description

As a proof of concept, we have chosen to realize a simple two-way scenario that shows the implementations capabilities to provide services towards both sides of the federation:

- A researcher affiliated with PlanetLab uses his PLC credentials to access the PlanetLab slice manager. He creates (or already owns, this detail is irrelevant for this purpose) a slice and subsequently requests slivers from a PlanetLab and a Panlab node to be part of his slice.
- A researcher uses her Panlab credentials to log into the TEAGLE portal and creates a new VCT (Virtual Customer Testbed, equivalent of a slice). Using the VCT tool she adds a PlanetLab node (sliver) to her testbed and additionally chooses to deploy another software package (MySQL) onto it.

5.2 Engineering Decisions and Implementation Details

We have implemented a module, called SFAAdapter, which acts as a bridge between the internal semantics and protocols of a PTM and the SFA. In the context of PTM, this is a hybrid module in the sense that it acts as both a front-end module to PTM, serving provisioning requests issued by the SFA, and as a PTM resource adapter, making resources which federation partners offer through SFA available to other PTM internals. From the viewpoint of the SFA, this module appears just as any other federation partner, exposing the interfaces of a registry and aggregate manager.

The module itself has been implemented in the python programming language. One reason for choosing Python was that PlanetLab's implementation of the SFAexists in python. This made it possible to reuse many utility classes and functions from the existing module.

Internally, SFAAdapter comprises three main parts, shown in figure 9 as squares inside the box labeled SFAWrapper. In detail, these elements and their duties are:

- Aggregate manager (AM): As the name already implies, this entity acts as an aggregate manager as specified by the SFA. It will advertise the resources the PTM can provide when queried by the slice managers of federation partners. Furthermore, it receives provisioning requests which it will translate towards the PTM core to acquire resources.
- Resource Adapter (RA): The resource adapter part of SFAAdapter acts as the counterpart to the aggregate manager. Towards the PTM core, it poses as a native resource adapter module. It queries foreign aggregate managers about the resources they have to offer and relays the information to PTM. Subsequently, it issues provisioning requests received from the PTM side towards said aggregate managers to book resources from other parties. This module is also responsible for deploying further PTM resource adapters for the acquired resources so they can be managed by the PTM.

- Registry (Reg): This module provides a registry service as specified by the SFA. It can be queried by the PTM core as well as by remote registries and provides information about the SFA objects which this domain is responsible for. The information is obtained from a database shared between the different parts of SFAAdapter.

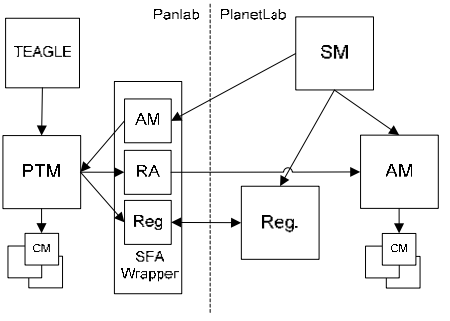


Fig. 9. Implementation layout; the arrows indicate the directions in which requests are issued

It has to be noted, that the modules providing SFA interfaces (aggregate manager and SFA registry) are both based on the original Princeton SFA implementation. This in turn means that existing SFA front-end clients can interact with them without further adjustments.

6 Proof of Concept

This section shows a glimpse of how a researcher could access federated services from each side of the federation by walking through a number of steps to set up a rudimentary testing environment. Note that in the example shown, the output is occasionally truncated for brevity and readability.

6.1 PlanetLab Perspective

A PlanetLab researcher wishes to use his PLC credentials to create a testbed environment via SFA. He has already been added to the SFA registry by a PlanetLab administrator and owns a slice (*plc.fokus.s1*) which, however, does not have any slivers associated yet:

```
#sfi.py resources plc.fokus.s1
  <RSpec ...>
    <networks>
      <NetSpec name="plc" .../>
    </networks>
  </RSpec>
```

Therefore, he first procures an overview over all available resources while at the same time saving the output for later use:

```
#sfi.py resources –o all.rspec
  <Rspec ...>
    <networks>
      <NetSpec name="ptm" ...>
        <nodes>
          <NodeSpec name=”pnode-0.ptm”>
            <net_if>
              <IfSpec addr="10.0.0.10" .../>
            </net_if>
          </NodeSpec>
        </nodes>
      </NetSpec>
    </networks>
    <networks>
      <NetSpec name="plc" ...>
        <nodes>
          <NodeSpec name=”pln0.plc”>
            <net_if>
              <IfSpec addr="10.0.0.20" .../>
          </net_if>
          </NodeSpec>
        </nodes>
      </NetSpec>
    </networks>
  </Rspec>
```

From this information, he learns that he has two nodes at his disposal, *pln0.plc* from the domain *plc* and *pnode-0.ptm* from the domain *ptm*. He adds them to his slice:

```
#sfi.py create plc.fokus.s1 all.rspec
```

The PlanetLab slice manager will now contact the PTM's aggregate manager and request it to instantiate a sliver on pnode-0.ptm. The PTM in turn contacts the appropriate resource adapter, orders it to set up a virtual node and to configure it to be PLC compatible. To give rudimentary access to researches, this especially means installing respective user credentials and configuring the sliver's SSH server. The researcher can now access the sliver and gain privileges in the same way he would access a sliver acquired from PLC:

```
#ssh –i .ssh/id_rsa focus_s1@pnode-0.ptm sudo su –
```

6.2 Panlab Perspective

The Panlab researcher mentioned in our use case opens the VCT tool via the TEAGLE portal. After entering her Panlab credentials, she starts assembling a new testbed comprising an installation of the MySQL software package on a node committed by PlanetLab (see figure 10). After carefully reviewing her setup, she issues the provisioning requests. Upon receiving these, the PTM contacts the SFA resource adapter which in turn will relay the request towards the aggregate manager of the PlanetLab side.

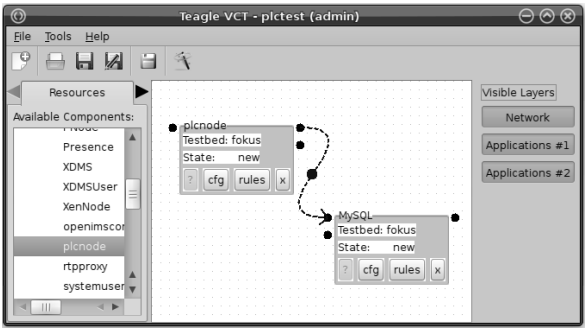


Fig. 10. VCT tool view

Since a sliver must always be part of a slice, the latter must obviously exist for the provisioning to take place. However, PTM dynamically creates those internally and hides this detail from the user. After the sliver is created by PLC, PTM accesses it and installs a number of resource adapters; among them the so called SoftwareAdapter which it will subsequently be used to deploy the requested MySQL package.

The researcher can now bring up the configuration page of the newly provisioned resource to learn some of its details. These are currently read-only values:

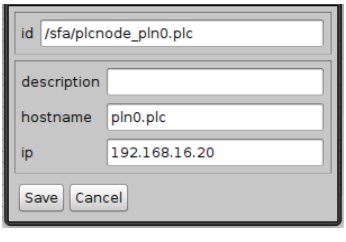


Fig. 11. Configuration page for a PLC node in TEAGLE

Note that in this example the researcher had to explicitly choose to deploy a plcnode in order to acquire a node from the PlanetLab network. In a real-life scenario a user would only choose to deploy any kind of virtual node and leave the details to the PTM internals.

7 Conclusion

It is technically possible and feasible to share resources and services between the PlanetLab and Panlab networks following the general model proposed by this paper. We had to apply an adaptor module to the Panlab Domain Manager and followed the basic SFA principles to achieve this. Both sides could benefit from the demonstrated scenario by offering a broader range of resources to their users. However, a number of things remain to be done for future efforts. As noted, the process of selecting resources is not yet completely transparent for a TEAGLE user. In the future, the

placement of virtual nodes would have to be decided fully by the PTM itself. Also, slice management could be improved. As of now, slices are not yet associated with individual VCTs as would be desirable.

Conceptually, our next steps will be to expand the implementation on the PTM side to not only commit virtual nodes but resources of arbitrary type towards the federation. This is already possible for isolated domains, however, when federating additional constraints arise. Also, the possibility of deploying further sub-resources, as already used in the TEAGLE demonstration above, could be exposed to other federation partners by dynamically advertising acquired slivers as new SFA components. Any non-technical aspects are yet completely untouched. Obviously, extensive operational agreements and procedures would have to be established before resources could be federated in the envisioned manner across several administrative domains. This includes the support for different resource access policies as well as federated identity management systems.

With this paper we showed that is possible to federated resources not only across the boundaries of administrative (organizational) domains but even across the boundaries of federations. Those federations in themselves provide collections of resources offered by several administrative domains and use different control frameworks to enable resource access and management. Although this is possible, many initiatives that are currently active in the field of resource federation would benefit from international standardization and agreed interfaces instead of bridging and adapting to each other via specific adaptor modules.

References

[1] Faber, T., Wroclawski, J.: A Federated Experiment Environment for Emulab-basedTestbeds. In: ICST (2009)
[2] National Science Foundation, GENI website, http://www.geni.net
[3] National Science Foundation, FIND website, http://www.nets-find.net
[4] European Commission, FIRE website, http://cordis.europa.eu/fp7/ict/fire
[5] Gavras, A., Karila, A., Fdida, S., May, M., Potts, M.: Future internet research and experimentation: the FIRE initiative. SIGCOMM Comput. Commun. Rev. 37(3), 89–92 (2007)
[6] Faber, T., Wroclawski, J., Lahey, K.: A DETER Federation Architecture. In: DETER Community Workshop on Cyber-Security and Test (2007)
[7] Peterson, L., Roscoe, T.: The Design Principles of PlanetLab. SIGOPS Oper. Syst. Rev. 40(1), 11–16 (2006)
[8] GENI Project Office, ProtoGENI Control Framework Overview, GENI-SE-CF-PGO-01.4 (2009)
[9] Chase, J., et al.: Beyond Virtual Data Centers: Toward an Open Resource Control Architecture. I Selected Papers from the International Conference on the Virtual Computing Initiative (ACM Digital Library) (2007)
[10] Ott, M., et al.: ORBIT Testbed Software Architecture: Supporting Experiments as a Service. In: Proceedings of IEEE Tridentcom 2005 (2005)
[11] OneLab project website, http://www.onelab.eu/

[12] Gavras, A., Hrasnica, H., Wahle, S., Lozano, D., Mischler, D., Denazis, S.: Control of Resources in Pan-European Testbed Federation. In: Towards the Future Internet - A European Research Perspective, pp. 67–78. IOS Press, Amsterdam (2009) ISBN 978-1-60750-007-0
[13] Website of Panlab and PII European projects, supported by the European Commission in its both framework programmes FP6 (2001-2006) and FP7 (2007-2013), http://www.panlab.net
[14] Campanella, M.: The FEDERICA Project - A federated infrastructure for Future Internet research. EURESCOM mess@ge, Issue 2/2008 (2008)
[15] AKARI project website, http://akari-project.nict.go.jp/eng/index2.htm
[16] Asia-Pacific Advanced Network initiative website, http://www.apan.net/
[17] Chen, M., Moon, S., Nakao, A.: Goals and Blueprint for PlanetLab CJK. Presentation at Conference for Future Internet 2008 PlanetLab BoF, Seoul, Korea, June 19 (2008)
[18] Asia Future Internet Forum website, http://www.asiafi.net/
[19] Magedanz, T., Wahle, S.: Control Framework Design for Future Internet Testbeds. e & i Elektrotechnik und Informationstechnik 126(07/08), 274–279 (2009)
[20] Peterson, L., et al.: Slice Based Facility Architecture, Princeton (2007)
[21] Fu, Y., Chase, J., Chun, B., Schwab, S., Vahdat, A.: SHARP: an architecture for secure resource peering. SIGOPS Oper. Syst. Rev. 37(5), 133–148 (2003)
[22] Fedora project website, http://www.fedoraproject.org
[23] Linux VServer project website, http://linux-vserver.org
[24] Peterson, L., Sevinc, S., Baker, S., Mack, T., Moran, R., Ahmed, F.: PlanetLab Implementation of the Slice Based Facility Architecture, Princeton (2009)
[25] PlanetLab bibliography index, http://www.planet-lab.org/biblio
[26] Wahle, S., Campowsky, K., Harjoc, B., Magedanz, T., Gavras, A.: Pan-European Testbed and Experimental Facility Federation – Architecture Refinement and Implementation. Inderscience International Journal of Communication Networks and Distributed Systems (IJCNDS), Special Issue: Recent Advances in Test-bed Driven Networking Research (forthcoming)
[27] Xen project website, http://www.xen.org
[28] XDMS project website, http://www.open-ims.org/xdms
[29] OpenIMSCore project website, http://www.openimscore.org
[30] MySQL project website, http://www.mysql.com
[31] TEAGLE portal website, http://www.fire-teagle.org
[32] Emulab website, http://www.emulab.net

TridentCom 2010

Full Paper Session 2: Future Internet Testbeds

Feather-Weight Network Namespace Isolation Based on User-Specific Addressing and Routing in Commodity OS*

Maoke Chen[1] and Akihiro Nakao[2]

[1] National Institute of Information and Communications Technology (NICT), Tokyo, Japan
[2] The University of Tokyo, Japan

Abstract. Container-based virtualization is the most popular solution for isolating resources among users in a shared testbed. Container achieves good performance but makes the code quite complicated and hard to maintain, to debug and to deploy. We explore an alternative philosophy to enable the isolation based on commodity OS, i.e., utilizing existing features in commodity OS as much as possible rather than introducing complicated containers. Merely granting each user-id in the OS a dedicated and isolated network address as well as specific routing table, we enhance the commodity OS with the functionality of network namespace isolation. We posit that an OS's built-in features plus our feather-weight enhancement meet basic requirements for separating activities among different users of a shared testbed. Applying our prototype which has been implemented, we demonstrate the functionality of our solution can support a VINI-like environment with marginal cost of engineering and tiny overhead.

Keywords: slice computing, name space isolation, socket, networking.

1 Introduction

It is evident that large-scale testbeds in wide-area network such as PlanetLab [13, 14] are viable for bringing innovations in computer networks. The proposal for GENI (Global Environment for Network Innovations) also stresses the significance of building such a testbed. Isolation of resources, i.e., enabling an environment where individual experiments will not be affected by the other ones, is a mandatory feature of such platforms. PlanetLab calls such an isolated set of resources as a "slice".

Virtualization is an instrument to implement isolation. There is a wide spectrum of virtualization technologies. Hypervisor-based solutions, like VMware [6] and Xen [7, 8], provide flexibility for running arbitrary operating systems, but are sub-optimal in scalability due to their computational overhead. A large-scale

* This work has been partly supported by Ministry of Internal Affairs and Communications (MIC) of the Japanese Government.

T. Magedanz et al. (Eds.): TridentCom 2010, LNICST 46, pp. 53–68, 2011.

slice-based network testbed emphasizes the requirement for little overhead in virtualization. Container-based solutions, VServer [3] and FreeBSD Jail [1], satisfy this requirement and have been successfully utilized in PlanetLab and Emulab [12], respectively. OpenVZ [5] and its commercial counterpart, Virtuozzo, also fall into the category of the container-base approaches. Container-based virtualization is considered suitable for a planetary scale test-bed, since it is often "light-weight" and enables over thousands of slices to run concurrently. However, when it comes to "light-weightness", we must consider not only the low overhead in *performance*, but also for that in *deployment*, including implementation, installation, daily maintenance and continuous upgrading. The PlanetLab kernel is based on VServer that requires a large amount of patches to the stock kernel and the patches may not necessarily keep up with the kernel development. Missing the latest hardware support in the kernel sometimes means we cannot utilize the cutting edge features of processors and devices in the testbed. The desire of having an isolation approach light-weight in both performance and deployment encourages new explorations on virtualization techniques.

We observe that most commodity operating systems already implement security isolation to some extent. It applies file/directory access permissions and disk quotas towards *user id*s, and offers CPU and bandwidth scheduling regarding processes and communication sessions, respectively. In the light of this observation, we pose a question: Can we support isolation only with standard OS features? The term "standard features" means functionality built-in into the current or a near-future native OS. Employing the standard features as much as possible should save the cost for deployment and incur a marginal extra overhead in performance. We elect to add a slight enhancement to the standard feature for network namespace isolation, thus, call our solution "feather-weight" isolation.

The term "network namespace" refers to a set of named entities in the network stack, including network interfaces, IP addresses, ports, forwarding and routing tables, as well as the sockets. It is one of the most important resources to be isolated but the isolation has not yet supported by any of current commodity operating systems. Early PlanetLab once took the *user id* in the commodity OS as the handle for a slice without network namespace isolated, and later the emphasis on security, scalability and enabling dedicated root environment lead to the choice of VServer as a virtual execution environment, where network namespace is isolated within a VServer container.

In this paper, we re-visit this early idea of user-id-based isolation and try to enhance it for network namespace isolation. The key of the solution is assigning IP addresses and policy-based route tables to *user id*s and enforcing separated usage in interface, address, port space, routing and bandwidth. A caveat is, for this feather-weightness, we may have to tradeoff the rigorous and secure root environment isolation. Fortunately, however, in most cases, root environment may not be necessary.

A parallel work, the Linux network namespace (NetNS) [2], addresses the same requirement for network namespace isolation. It clones the network stack into a container that is identified by a process id of a shell, and thus the shell

has an independent set of virtual interface, address, port space and socket space. NetNS has to modify the whole network stack and also other parts of OS kernel, such as filesystem. It is based on the container concept that is advantageous in performance but not so in deployment. It makes the NetNS code hard to maintain and to keep up with the evolution of the commodity OS.

This paper makes three contributions. First, unlike NetNS and any other solutions, this paper suggests the philosophy of enhancing commodity OS with isolation without over-engineering. Second, it puts this design philosophy into practice for the network namespace isolation, realizing the mechanism for isolating IP addresses and route tables within a *user id*. Finally, it demonstrates that our prototype implementation enables a VINI-like environment where we can experiment with new network architectures and services with marginal cost of engineering.

The rest of the paper is organized as follows. Section 2 summarizes the problem in enabling network namespace isolation in commodity OS and motivates our work from the observations on the drawbacks of the container-based NetNS. Section 3 identifies what is required and what is obtained with the proposed solution, especially for the separated routing tables. Section 4 demonstrates typical routing experiments based on our solutions and Section 5 presents benchmark for our prototype. Finally, Section 6 briefly concludes the paper with future work.

2 Related Work

We first identify what the current commodity OS supports for isolation and what it falls short of.

2.1 Resource and Namespace Isolation in Native OS

Even without either full virtualization or containers, native OS, either UNIX or modern Windows, facilitates resource and namespace isolation to some extent.

- *Security Isolation:* file systems protect file and directory access among *user id*s. Currently the security isolation in native OS doesn't hide one's processes and traffics from other users.
- *Performance Isolation:* disk quota in commodity OS controls storage usage per *user id*. CPU scheduling is a basic function of any OS regarding process management. Control Groups (*cgroups*) provides the aggregation for process scheduling. *User id* can be used for the progress grouping definitely. For the network bandwidth issue, traffic reshaping is applied for flows. In Linux operating system, the tool "`tc`" coupled with the `iptables` controls the flow according to its attributes like source or destination addresses, not related to *user id* yet.
- *Network Namespace Isolation:* IP addresses and port space are currently shared among *user id*s, without being isolated. Interfaces, routing tables are the same. Policy-based Routing can support multiple routing tables, but the

separation is not per *user id*. Therefore, the network namespace is what today's commodity OS does not support.

2.2 Linux Network Namespace (NetNS)

NetNS [2] is designed in response to the need of enabling network namespace isolation in OS. It clones the network stack into private sets of network resources to be assigned to one or several processes. The resource set includes device, IP address, port space, route tables, sockets and so forth. Each set is assigned to a shell process, which plays the role of the "container". NetNS can be configured with either Layer-3 router or the Ethernet bridging mode.

NetNS focuses only on network namespace isolation and thus it is much simpler than OpenVZ or VServer. However, it is still a container-based solution and accordingly suffers from the drawback of over-engineering. First, NetNS conflicts with some existing components in the commodity OS. A well-known case is *sysfs*, which must be patched until being able to co-exist with NetNS. Second, NetNS takes a process-id of a shell as the handle for a container. It implies a limitation that a "slice" implemented with such a container cannot run multiple shells with the shared network namespace. Finally, the code complexity causes a lot of troubles and bugs that hurt the stability in operation. For example, our trial of NetNS in Fedora 8 Linux with kernel 2.6.26.8, compiled from natively released source code, ends up with a hangup.

This observation on NetNS has motivated our exploration for a better and lighter-weight solution for the network namespace isolation, which incurs less cost not only in performance overhead but also in implementation and deployment than NetNS.

3 System Design

The objective of our system design is enabling network namespace isolation per *user-id* in a commodity OS, treating the *user-id* as the handle for a slice. The term "network namespace isolation" is defined as enabling following features from the bottom of network stack to its top.

R1 A *user-id* is assigned by the system administrator to one (or more) network interface(s), either physical or virtual, and not allowed to use the other network interfaces.

R2 A *user-id* is assigned by the system administrator to one (or more) IP address(es), either physical or virtual, and not allowed to use the other IP addresses.

R3 A *user-id* is assigned by the system administrator to a forwarding table, and not allowed to use the other forwarding table.

R4 A *user-id* is allowed to update the kernel forwarding table associated with this *user-id*.

We also have two constraints for the system design.

C1 Our solution must be transparent to applications. In other words, the existing applications must work without re-code and re-implementation. Binary-compatibility must be held.
C2 Our solution must involve only the networking kernel of the OS, not to conflict with other features or components, unlike NetNS.
C3 Our solution must enable new features, but leave the other existing features as they are.

The last constraint may need clarification; for example, without our solution, only the privileged user, `root`, can modify the route table in the kernel, while, after our enhancement, regular users are allowed to change their own associated route table but what the `root` can do is totally unchanged.

We design the system with three steps to meet the requirements of our design objective. First, we enable the IP address isolation among *user-id*s. Then, we apply PBR (Policy Based Routing) and traffic shaper with our modified OS kernel to make forwarding tables and bandwidth isolated. Finally, we extend the model through associating route table-id with the *user-id*, and separate the route installation from RIB (routing information base) to FIB (forwarding information base), from the user space to the kernel.

3.1 Isolating IP Addresses among Users

[2] Isolation of IP addresses among users needs two components to be added to the kernel: the data structure describing the assignment of addresses for *user-id*s and the functions that ensure the separated address usage. The data structure and the function are referred by socket system-calls transparently to the callers.

The data structure for address assignment to *user-id* is designed in the form of an association of IP address and *user-id*. An entry (*IP_address*, *user_id*) means the *user-id* is able to use the *IP address*. Considering the scalability, the data structure should be designed accessible with a hash function.

Semantically the user-address association changes the meaning of the "unspecified" (or wild-card) IP address, which is defined with the macro IN_ADDR_ANY for IPv4 and IN6_ADDR_ANY for IPv6, respectively. In a commodity OS, the wild-card means any address configured with any interface of the host. In an OS with user-specific address, however, a wild-card means any addresses among those already assigned to the *user-id* of the process owner that raises the communication.

The interpretation of wild-card complicates the port conflict detection. When two sockets attempt to bind to the same port on the wild-card, the conflict happens when the process owners are associated with at least one common address. Therefore, the port conflict detection routine should also check which addresses

[2] The content of this part is covered by our previous work [11], which focuses on the addressing architecture for slice computing but not other isolation issues for network namespace.

are used by each binding socket, i.e., finding out the user-address association for the *user-id* of the owner of each socket.

Packet dispatch from a protocol stack to a socket is also affected by the change of the semantics of wild-card. The protocol stack searches the socket pool for a socket matching the destination (address, port)-pair of a packet. Previously, if a socket is bound to the wild-card, only port should be check for the dispatching. Now the kernel must identify what the wild-card of a candidate socket mean exactly and determine if this socket's corresponding *user-id* really owns the destination address in the packet to be dispatched.

Based on the above data structures, a variety of functions are added into the socket system-calls and protocol stack.

- *Sender address availability check:* when a process tries to bind a specific address for connection to send out packets directly with the specific source address, the kernel checks if the process owner *user-id* is assigned with the address.
- *Listener address availability check:* when a process tries to bind a specific address for a daemon, the kernel checks if the listener process owner *user-id* is assigned with the address.
- *Address selection:* when a process does not specify a source address but lets the kernel choose one for itself in either of the above cases, the kernel filters only those addresses assigned to the process owner *user-id* as the candidate for the selection. More importantly, the commodity system also tries to select the source address "closest" to the destination. Our modified kernel should apply this algorithm with the addresses available for the process owner *user-id*. The address selector also helps protocol stack to identify proper socket that a connectionless datagram or a connection request is targeting.

Per-user address assignment directly results in the elimination of port conflict among *user-id*s as in a commodity OS where users can share the use of a single IP address, since the full port range of the dedicated IP address is available to each user. We may change the minimum available port number for a regular *user-id* from 1024 to 0. This is especially beneficial when each user wants to use the well-known port number of the dedicated IP address, for example, to run BGP speakers with a well-known port number such as 179.

Per-user address assignment is suitable for the circumstances where IP addresses are abundant, either in IPv4 or in IPv6, or in private IP addresses. Note that our proposed system can assign even private IP addresses to each *user-id*. Most virtual networks are using tunnels through private IPv4 addresses. Considering this, our system is especially useful for network virtualization, as demonstrated in Section 4.

3.2 Combining Address Separation with Other Features

Network namespace includes link interface, IP address, ports, routes and bandwidth. After IP addresses have been isolated among *user-id*s, we can also isolate

some of other resources among users by configuring those resources with the user-specific addresses.

- *Interface:* interface is configured with one or more IP addresses. Therefore, the use of each interface is isolated per *user-id*. However, only the *use* of the interface is isolated, not its *visibility*—each user can see all the physical and logical network interfaces on the host, knowing their addresses with commands such as `ifconfig` and `ip link`.
- *Routing:* Policy-based Routing (PBR) can be used to define a separate route (more precisely, forwarding) table for packets from a specific source address. Therefore, we can let each *user-id* own and use a separate forwarding table for packets originating from the address the *user-id* is assigned to, rather than share the default table with others. RIB (routing information base) is managed at the application layer. Therefore, it is naturally isolated when being run by specific *user-id*. However, today's commodity OS doesn't allow unprivileged *user-id* to run routing tools and to install RIB to the system FIB.
- *Bandwidth:* Traffic control tools in commodity OS (e.g., `tc` in Linux) support shaping policies with respect to a source or a destination address. Configuring `tc` with the rule specified by user-specific source or destination addresses, we can schedule bandwidth utilization among *user-id*s.
- *Raw socket:* A commodity OS allows only processes running with the set-user-id flag `suid = 0` to open a raw socket. The purpose of this design is to prevent users from applying raw socket to generate arbitrary malicious packets. A program that needs a raw socket but to be run by a non-privileged user should set the set-user-id flag correctly on its binary file. After IP addresses being isolated among *user-id*s, we do not have to prevent a regular user from using a raw socket any more. One who attempts to abuse a raw socket to forge a packet can only stick one's own IP address in the packet, since our system prohibits the usage of the other IP addresses. Enabling a user's program to open a raw socket doesn't need any further modification in the kernel. Once a user writes such a program, the privileged user can set the set-user-id flag of the executable correctly.

3.3 Routing Isolation

Isolating routing is extremely important for network experiments on virtual networks [9]. Although we mention in the discussed above that crafting PBR rules with user-specific address can achieve routing isolation, running separate routing protocols in different slices will not achieve this goal. The problem is that in a commodity OS only the privileged user can install a route to the system FIB. For this reason, we need to enable a regular user to update the FIB of the PBR associated with his or her *user-id* in the following steps.

First, to enable the route installation from a regular *user-id*, it is necessary to ask the kernel to accept the route update request from a the (non-privileged) *user-id* through either `ioctl` commands or `RTNETLINK` socket messages.

Second, once a regular *user-id* is allowed with route updates, it must be ensured that a certain user may not interfere with the routes of the others. Note that the privileged user can manipulate any tables.

When a table-id is specified in a route update request, the kernel must ensure that the table-id is associated with the *user-id*. When the table-id is not specified, then the kernel should find out which table is available for the request sender.

Therefore, to enable the comprehensive routing isolation, we have to add the route table association into the kernel as well.

When any route table-id is not explicitly associated with a certain *user-id*, it means this user wouldn't like to change the route by itself and it doesn't need a separate route table at all. Accordingly it just shares (and only reads) the default route table of the host, which is able to be update only by `root`. If a non-privileged user, associated with the default table, sends a route update request to the kernel, the kernel should reject the request and return the error, "operation not permitted".

As a summary of the system design, in Table 1, we list the features of isolation enabled (marked with "√") and not enabled (marked with "–") by our solution, compared with the container-based NetNS. We trade off only the privacy of slices for gaining the simplicity in coding, maintenance and deployment.

Table 1. Network Namespace Isolation Supports

Isolation	NetNS	Our Solution
Interface	√	√
Address/port	√	√
Forwarding	√	√ (w/ PBR)
Routing	√	√ (w/ PBR)
Traffic Reshaping	√	√ (w/ `tc`)
Raw socket	√	√ (authorized by `root`)
Visibility/privacy	√	–

We have implemented the Linux kernel patches and tools for the user-id-based network namespace isolation, with both IPv4 and IPv6. Readers are welcome to try our code which is open-source and downloadable from online[3].

The change involves three major parts: 1) new data structures associating IP address, *user-id* and route *table-id*, and their manipulation functions, which are used in new `ioctl` command as well as `procfs` parsers for the administrator doing management over the user-address-table associations; 2) modifications in socket system call instances, augmented with the *user-id* related behaviors; and 3) modifications in RTNETLINK message processing functions, enabling regular users to update route tables and dispatching the updates into proper table corresponding to *user-id*.

[3] See http://sourceforge.net/projects/uoa/ for details.

4 Demonstration: A VINI Experiment

Our proposed user-id-based namespace isolation implemented in the Linux kernel is minimal compared to the other container approach and can be easily apply to any version of Linux kernel release. This is mainly because the modified pieces are concentrated in the networking stack implementation of the Linux kernel. In this section, we demonstrate that our proposed solution can serve as a slice computing platform with network namespace isolated through achieving a similar platform to VINI [9], which also aims to provide network namespace isolation for a slice but using Linux VServer, a resource container virtualization technique.

We arrange the demonstration with a simple configuration. As is shown in Fig. 1, three computers running Linux with the modified kernel and physically connected to the same network. In the implementation of VINI, nodes are connected over the Internet, not necessarily within the same network. We separate the virtual links via tunneling, to mimic the real environment.

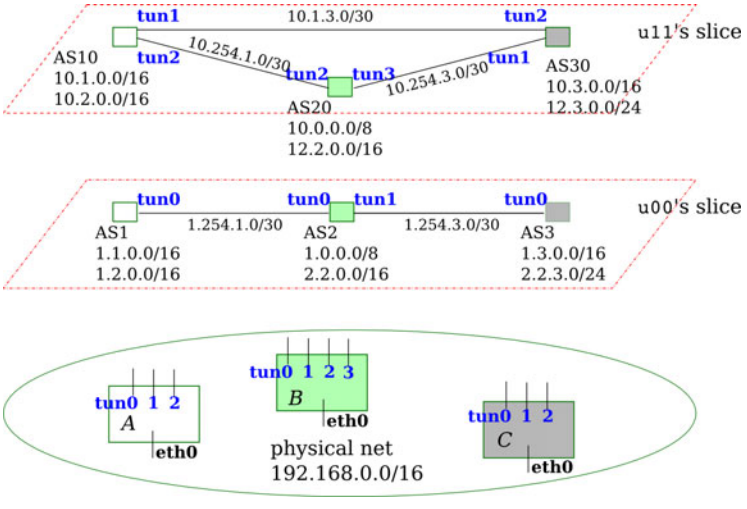


Fig. 1. Nodes, networks and slices for the VINI experiment

Two users are running their own slices with *user-id*s, `u00` and `u11`, respectively [4] . Assigning proper addresses of tunnel interfaces to the users, we have the topologies of the networks of the two slices separately defined. Fig. 1 depicts that the networks on the two slices are configured with separated address blocks.

[4] To be precise, *user-id*s of the same slice on different nodes could be vary. To simplify the discussion, we specify the *user name*s on all nodes identical to the name of the slice and don't discuss the detailed number of *user-id* in the operating system.

4.1 Configurations

We suppose that address blocks 1.0.0.0/8 and 2.0.0.0/8 are assigned to `u00` while blocks 10.0.0.0/8 and 12.0.0.0/8 to `u11`. The two users would run their own networks with their own addresses, respectively. How do they arrange the addressing and networking depends upon their needs.

Addresses and route tables: We assume that after a user has made the layout of its in-slice network, the user informs the administrator of the platform to configure the virtual interfaces, enabling virtual links. Here we apply the GRE tunneling for this purpose and the `ip tunnel/addr/link` tools are utilized.

The administrative tool, `uoamap`, is used to configure the association of IP address, *user-id* and route table-id. Table 2 presents the configurations that we make for the nodes A, B and C, respectively.

Table 2. Address and route table associations

Config.	Node	`u00`	`u11`
IP addr. (iface)	A	1.254.1.2 (tun0)	10.254.1.2 (tun2)
			10.1.3.1 (tun1)
	B	1.254.1.1 (tun0)	10.254.1.1 (tun2)
		1.254.3.1 (tun1)	10.254.3.1 (tun3)
	C	1.254.3.2 (tun0)	10.254.3.2 (tun1)
			10.1.3.2 (tun2)
table-id	A B C	32800	32811
(rules)		from 1.0.0.0/8 from 2.0.0.0/8	from 10.0.0.0/8 from 12.0.0.0/8

Routing policies: data plane Rules for routing policies are configured directly with the `ip rule` tool. According to the address block assignment, we apply the policy that allow each slice's network can transfer packets with the source addresses within the blocks assigned to the slice. For our demonstration, we have shown these rules in the Table 2.

Note that the VINI also supports interaction of real networks and in-slice overlays. In such a case, source addresses of packets running over one's slice may not be restricted in the assigned address blocks. This can be supported with applying rules regarding the incoming interface, e.g., `ip rule add dev tun0 table 32800` on node A indicates that the packets coming through the device `tun0`, which is configured with an address in `u00`'s slice, can traverse the node A according to the route table with id 32800.

Links, addresses, tables and rules are all configured by the `root` on each node. Afterwards, users have the freedom of manipulate their own route table by either the manual configuration with `ip route` commands or the automatic update with routing protocol daemons.

Routing protocols: control plane In our demonstration, a routing protocol runs as an application of regular *user-id*. Unlike the other container approaches, in our solution, there are several minor limitations such as Quagga routing suite has to be re-compiled by each user, because the installation path must be set to the user's home directory rather than the default `/usr/local`, and the daemons must run with the context of the *user-id* itself rather than the default "`zebra`" or "`daemon`". In addition, we must disable Quagga's check on the security capability. All of these can be specified with the `configure` tool in the Quagga source.

After the Quagga suite is installed, a user has to edit the configuration files, `zebra.conf` and `bgpd.conf` to define BGP peers. Finally, the user can start both `zebra` and `bgpd` to start routing in its own slice. Since the BGP peering in our demonstration is simple so for brevity's sake, it is not shown here.

To meet the design goal of transparency to slice users, the table-id of a slice is known only to the `root` while it is hidden to a regular user. The latter only sees it as if it were the main table of the system and cannot see other users' tables. Therefore, it is not necessary to assign table-id in the `zebra` configuration.

4.2 RIB and FIB

BGP sessions are established between two application-layer peers, therefore their RIB is naturally isolated. Our demonstration shows that the sessions are established and not interfere each other. For example, for node A, we check both the BGP session advertisement as well as the corresponding route table to verify our design and implementation.

BGP tables: Each slice user applies `telnet` to the Quagga BGPd user interface of any of its own sliver. On node A, `u00`'s BGP table can be accessed through 1.254.1.2 port 2605, while `u11`'s BGP session can be completely isolated of that of `u10`.

Fig. 3 illustrates what we can achieve. The success of the BGP peer establishment verifies our PBR-based source address selection is working correctly.

Isolated FIB installation: The data plane of in-slice routing is isolated through the PBR facility. Our design further realizes the isolation of route updates from RIB to the user-specific FIB. We login as each user and run the `ip route` command to check the installation of the routes.

In displaying the routes, we don't specify the table-id and let the kernel pick up the user-specific table-id automatically. Therefore, the separated route display is also transparent to slices since it is not required that the slice user get to know the detailed number of its table-id, which is not decided by itself but decided by the local administrator (the `root`).

The successful establishment of BGP peers and the updates from BGP RIB to kernel FIB validates that our solution has the ability to support comprehensive routing isolation.

```
bgpdA> show ip bgp
BGP table version is 0, local router ID is 1.254.1.2

   Network          Next Hop    Metric LocPrf Weight Path
*> 1.0.0.0          1.254.1.1        0             0 2 i
*> 1.1.0.0/16       0.0.0.0          0         32768 i
*> 1.2.0.0/16       0.0.0.0          0         32768 i
*> 1.3.0.0/16       1.254.1.1                      0 2 3 i
*> 2.2.0.0/16       1.254.1.1        0             0 2 i
*> 2.2.3.0/24       1.254.1.1                      0 2 3 i

Total number of prefixes 6
```

(a) The RIB of `u00`'s slice at node A

```
bgpA> show ip bgp
BGP table version is 0, local router ID is 10.254.1.2

   Network          Next Hop    Metric LocPrf Weight Path
*  10.0.0.0         10.1.3.2                       0 30 20 i
*>                  10.254.1.1       0             0 20 i
*> 10.1.0.0/16      0.0.0.0          0         32768 i
*> 10.2.0.0/16      0.0.0.0          0         32768 i
*  10.3.0.0/16      10.254.1.1                     0 20 30 i
*>                  10.1.3.2         0             0 30 i
*  12.2.0.0/16      10.1.3.2                       0 30 20 i
*>                  10.254.1.1       0             0 20 i
*  12.3.0.0/24      10.254.1.1                     0 20 30 i
*>                  10.1.3.2         0             0 30 i

Total number of prefixes 6
```

(b) The RIB of `u11`'s slice at node A

Fig. 2. Success of in-slice BGP peering

5 Performance Evaluation

This section describes our qualitative and quantitative evaluations on the the proposed solution, showing it is light in both engineering and performance overhead.

5.1 Qualitative Evaluation

The development needs only a small amount of coding for the kernel patch and the management toolset. Totally, the kernel patch includes adding 10 and modifying 3 header files in `include/net`, adding 8 and modifying 20 C source files in `net/ipv4` and `net/ipv6` subdirectories. The kernel patch (`diff` result

```
u00@uor: ~$ /sbin/ip route list
1.254.1.1 via 1.254.1.2 dev tun0
2.2.3.0/24 via 1.254.1.1 dev tun0  proto zebra
2.2.0.0/16 via 1.254.1.1 dev tun0  proto zebra
1.3.0.0/16 via 1.254.1.1 dev tun0  proto zebra
1.0.0.0/8 via 1.254.1.1 dev tun0  proto zebra
```

(a) The FIB of `u00`'s slice at node A

```
u11@uor: ~$ /sbin/ip route list
10.1.3.2 via 10.1.3.1 dev tun1
10.254.1.1 via 10.254.1.2 dev tun2
12.3.0.0/24 via 10.1.3.2 dev tun1  proto zebra
10.3.0.0/16 via 10.1.3.2 dev tun1  proto zebra
12.2.0.0/16 via 10.254.1.1 dev tun2  proto zebra
10.0.0.0/8 via 10.254.1.1 dev tun2  proto zebra
```

(b) The FIB of `u11`'s slice at node A

Fig. 3. Success of in-slice route installation from RIB to FIB

from stock to our kernel code) has only 3707 lines, 99KBytes, including inline comments. The code involves only the networking kernel and is well decoupled from other parts, and therefore it is easy to maintain and upgrade. In contrast, either VServer or NetNS needs a big amount of coding, and the coding heaviness is also the major reason causing the poor compatibility, resulting in troubles that we have mentioned in Section 2.

In the deployment perspective, users of a node can directly use the proposed solution as long as the `root` has configured its available address, virtual links (if necessary) and routing table. Section 4 has demonstrated the simplicity of our user-specific addressing and routing without any containers. In VServer solution, it is necessary to establish containers and install frequently-used applications in the containers one by one. In NetNS solution, the root needs to detach the default namespace from a new shell process and then assign a virtual interface to it and establish the new namespace. It is obvious that the workload to introduce our user-specific addressing-routing approach is lower than them two.

Therefore, we are confident of the lightness of our solution in the terms of the engineering for code maintenance and deployment. We need also verify the performance scalability of the proposed solution. We conducted the following benchmark with our prototype implementation and compare it to the performance of the stock kernel. In [10], one can find benchmark results for the container-based solutions, where either VServer or NetNS is significantly poorer than the stock kernel.

5.2 Benchmarking

For the benchmark, two computers equipped with Intel Core2Quad CPU Q6700@ 2.66GHz and 3.2GB memory are connected over a 1000 Mbps cross-wire Ethernet link. One of them, (A), is installed the native Linux kernel 2.6.32 for Debian 5.0.3 and our patched kernel. It can be booted up with either. The other machine, (B), is running with the native kernel. The benchmarking tool *netperf* [4] version 2.4.4 is used. *netserver* is running at (B) and (A) starts `netperf`.

Working with the kernel enabled with user-specific address and route table, (A) is running with a certain number of IPv6 addresses, one-to-one associated to the same number of *user-id*s. For a fair comparison, when the machine (A) runs the stock kernel, i.e., users are not associated with specific addresses but share them all, we also *configure* the same number of addresses to the interface, even though processes always pick up the same one. Further, we run each test twice, one round with source address unspecified while the other round with source address specified. All the tests are conducted with source port unspecified at the sender side, i.e., the kernel will perform port selection for connection or message sending.

The *netperf* benchmarking is conducted with four types of test: TCP connection/close, UDP request/response, TCP streaming and UDP streaming. Tests are run by one user among a group, every user of which is assigned with a dedicated IPv6 address. Performance impact of the user group size is depicted in Fig. 4.

TCP streaming and UDP request/response are the most popular types of applications applying TCP and UDP, respectively. Our benchmark shows that the performance of TCP streaming and UDP request/response are not impacted by the patch for the user-specific addressing and routing. Actually, for the TCP streaming, either the address availability check or the port conflict detection is done only once for a session at the connection establishment stage. The check for the address availability must consume some CPU time but this could be ignored in the macro observation for the end-to-end throughput. For UDP request/response, the check and detection is done for every request, and the address availability check consumes also the ignorable amount of CPU time.

The TCP connection/close and UDP streaming benchmarking, however, present an unexpected behavior, where the patched kernel performs even better than the stock kernel when the user group is getting quite large. This can be explained with the principle of port selection. Because the user-specific addressing decreases the probability of the port conflict, when the number of concurrent sessions gets high, stock kernel suffers higher port conflict rate and it must take more time to find an unoccupied port number for a new socket. TCP connection/close leaves a lot of sockets in the TCP TIME_WAIT state and they hold the use of their port for a quite long period in comparison to the action of connection establishment, while UDP streaming also causes a lot of long-living sockets occupying port numbers in use. Therefore, the affect of port conflict is apparent in these two types of benchmark.

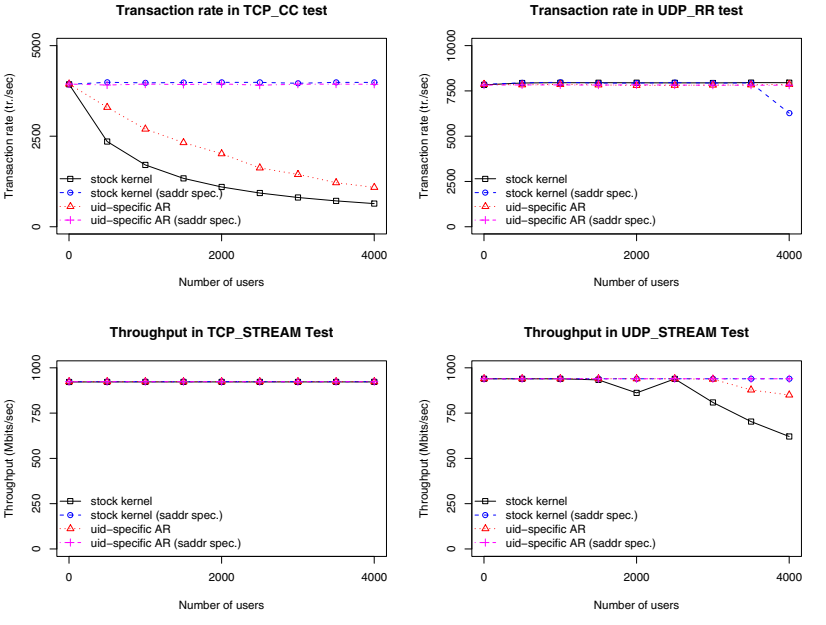


Fig. 4. Performance comparison: user-specific addressing vs. stock kernel

To summarize the benchmark, we state that even the current full-fledged prototype of our user-specific addressing and routing can achieve the end-to-end performance not worse (but sometimes better) than the commodity OS, for most cases of Internet applications.

The lightness in run-time overhead makes our user-specific addressing-routing approach competing with a container based solution like VServer. Without container, our solution can easily support up to 4K slices (actually, 4K is the maximum number of IP addresses that are able to be configured with a network interface card in Linux) over a PC and have good performance. While a 10^2 GB-capacity disk is hard to support more than hundreds of VServer guests because each guest has to consume at least tens of MB for the basic container facility.

What we trade off for the scalability is the `root` environment for a slice. On the other hand, in the privacy-sensitive circumstances, we cannot shield a slice's activity from being viewed by another.

6 Conclusions

This paper explores the feasibility of supporting isolation for slices in a commodity OS with as few modifications as possible and only within the networking kernel. For this purpose, we design and implement the idea of isolating IP addresses and routing tables among *user-id*s on top of a commodity OS. Our min-

imal modifications to the stock kernel together with the well-developed policy-based routing (PBR) facility successfully validate that the philosophy of minimal engineering can provide comprehensive isolation for network namespace. Meanwhile, the performance overhead is light enough to ensure the performance of the modified kernel not worse than the commodity OS stock kernel. The number of slices supported by the user-specific addressing and routing can reach the limit of the maximum IP addresses that a commodity OS can configure with a network interface.

Acknowledgment

We thank Taoyu Li for his early design and implementation, and Yuncheng Zhu for the discussion and help in performance evaluation.

References

1. Freebsd architecture handbook, `http://www.freebsd.org/doc/en/books/archhandbook/jail.html`
2. Linux network namespace (NetNS), `http://lxc.sourceforge.net/network.php`
3. Linux VServer, `http://www.linux-vserver.org/`
4. Netperf, `http://www.netperf.org/`
5. OpenVZ, `http://wiki.openvz.org/Main_Page`
6. VMWare, `http://www.vmware.com/`
7. Xen, `http://www.xen.org/`
8. Barham, P., Dragovic, B., Fraser, K., Hand, S., Harris, T., Ho, A., Neugebauer, R., Pratt, I., Warfield, A.: Xen and the art of virtualization. In: SOSP 2003: Proceedings of the Nineteenth ACM Symposium on Operating Systems Principles, pp. 164–177. ACM, New York (2003)
9. Bavier, A., Feamster, N., Huang, M., Peterson, L., Rexford, J.: In vini veritas: realistic and controlled network experimentation. SIGCOMM Comput. Commun. Rev. 36(4), 3–14 (2006)
10. Bhatia, S., Motiwala, M., Mühlbauer, W., Mundada, Y., Valancius, V., Bavier, A., Feamster, N., Peterson, L., Rexford, J.: Trellis: A platform for building flexible, fast virtual networks on commodity hardware. In: ACM ROADS Workshop 2008, Madrid, Spain (December 2008)
11. Chen, M., Nakao, A., Bonaventure, O., Li, T.: UOA: Useroriented addressing for slice computing. In: Proceedings of ITC Specialist Seminar on Network Virtualization, Hoi An, Vietnam (May 2009)
12. Hibler, M., Ricci, R., Stoller, L., Duerig, J., Guruprasad, S., Stack, T., Webb, K., Lepreau, J.: Large-scale virtualization in the emulab network testbed. In: ATC 2008: USENIX, Annual Technical Conference, pp. 113–128. USENIX Association, Berkeley (2008)
13. Peterson, L., Anderson, T., Culler, D., Roscoe, T.: A Blueprint for Introducing Disruptive Technology into the Internet. In: Proceedings of the 1st Workshop on Hot Topics in Networks (HotNetsI), Princeton, New Jersey (October 2002)
14. Peterson, L., Muir, S., Roscoe, T., Klingaman, A.: PlanetLab Architecture: An Overview. Technical Report PDN–06–031, PlanetLab Consortium (May 2006)

A Novel Testbed for P2P Networks

Pekka H. J. Perälä[1], Jori P. Paananen[2],
Milton Mukhopadhyay[3], and Jukka-Pekka Laulajainen[1]

VTT Technical Research Centre of Finland,
[1] Kaitoväylä 1, FI-90571 Oulu
[2] Vuorimiehentie 3, FI-02150 Espoo, Finland
firstname.lastname@vtt.fi
[3] Pioneer Digital Design Centre Ltd, Pioneer House
Hollybush Hill, Stoke Poges, Slough, SL2 4QP, United Kingdom
milton@pddresearch.com

Abstract. Peer-to-Peer (P2P) overlay networks have recently gained significant attention in the research community. P2P applications, especially the ones using BitTorrent protocol, often require communication among a large number of peers and thus are particularly challenging to test in a controlled manner, because of the volatility of overlay-network structure with peers going on and off. This paper addresses the issue by introducing a novel testbed that enables intuitive network QoS profile configuration, automated peer deployment and test case execution with keyword driven test automation, as well as wireless network testing with real networks. We evaluate the fitness of the testbed by deploying a P2P video delivery application in the network and running trials while monitoring the application behaviour throughout them. Our results demonstrate the capabilities of the testbed in three test cases with different peer access network configurations. The results verify the correct functioning of the testbed and are the first step on our analysis of the P2P video delivery application This paper provides information for P2P application developers and testers and enables them to setup up similar environments for advanced testing and research on their applications and protocols.

Keywords: peer-to-peer networks, network emulation, network testing, testbeds, wireless networks.

1 Introduction

Peer-to-Peer (P2P) paradigm has gained a significant amount of interest in research community due to the vast possibilities of the technology. With P2P technology it is possible to build scalable applications with high availability of content. BitTorrent [1] is one of the most popular file sharing protocols today and thus is on the focus of this work.

BitTorrent uses a *.torrent* metadata file to describe a file to be transferred. The metadata contains the hash of the pieces of a file, and a list of trackers that keep a registry for the file. A central tracker keeps track of the peers that have the file

T. Magedanz et al. (Eds.): TridentCom 2010, LNICST 46, pp. 69–83, 2011.

described in the metadata. A peer connects to the tracker and receives the list of peers that have the file. After getting the list, the peer connects to other peers via *peer wire protocol* to ask for pieces of the file. The downloading peer contacts the tracker again only if the current peers are not satisfactory. After downloading the whole file, the peer turns into a *seeder*, i.e. a peer with a complete copy of the file. The peers involved in data exchange around a specific file form a *swarm* [1, 2].

Due to the distributed nature of BitTorrent, it has proven difficult to reliably perform reproducible experimentation during the development process. Most often the researchers and developers have to rely on simulations, which have only limited correspondence to the real world. On the other hand live trials with instrumented clients are often laborious to set up and the results are usually not reproducible, although they correspond to real behaviour of users. Therefore our work targets the middle ground of these two by providing a fully controllable environment for early testing of new features of P2P applications.

The major contribution of this work is the novel testbed with a keyword driven automation framework that enables testing distributed applications, especially P2P applications, with ease. It allows centralized control and configuration of the network Quality of Service (QoS) parameters with GUI, centralized control, and automatic test case execution and reporting. The GUI provides interactive diagrams for visualizing the complex network topology and queuing discipline structures. A centralized keyword driven test framework allows automation of installation, execution and reporting of the test cases. We found that the keyword driven automation framework is eminently suitable for testing P2P systems as it satisfies the need to be extremely flexible with its component based modular approach, at the same time it reduces the complexity of writing test cases by allowing test cases to be created using abstract keywords and simple logic. In our trials, we have used Nextshare P2P-application, developed by FP7 project P2P-Next consortium [3] as a System Under Test (SUT). Nextshare is a novel video broadcasting system that is based on BitTorrent and is still under heavy development. This makes it an ideal test subject in our testbed development work. To the best of our knowledge, a testbed with such advanced features has not yet been presented in the literature. Especially, configurable network QoS coupled with wireless access networks is a novel contribution in controlled P2P testing.

The measurements conducted with the testbed demonstrate with three test cases the possibilities the testbed creates for P2P research. The test cases are differentiated based on access network configuration of the peers and otherwise similar scenario is executed in all three cases. This gives us opportunity to observe how tuning a single property affects the whole system. The keyword driven test execution framework takes care of the scenario configuration based on given description. We use the reporting and measurement infrastructure to collect reports on the peers and in this specific case we use one wireless node and monitor network on its perspective.

The rest of the paper is organised as follows. Section 2 discusses the special issues that P2P applications pose to test environments, Section 3 presents the developed testbed in detail. Section 4 gives the results from our experiments. Section 5 presents the related work on the area. Finally, Section 6 concludes the paper and outlines the items for future work.

2 Challenges in Testing P2P Applications

P2P applications are challenging to test and experiment with due to their distributed and volatile nature. Main difficulties in producing a reliable test environment include the following: realistic scenario configuration, measurement and reporting, and network configuration.

P2P-applications, more specifically the ones that use BitTorrent protocol are particularly challenging in the testing point of view due to their distributed nature, hence they often require distributed monitoring system as well. The measurement results gathered from multiple sources must be synchronized in order to enable comparative analysis. Furthermore, if one requires the test results to be reproducible, the nodes of the system must be controlled with permissions to e.g. start and stop the execution of the application under test.

Another issue that needs to be solved is the correspondence of the test environment with the real world. In real P2P networks, such as BitTorrent, the system consists of peers that join and leave the swarm arbitrarily. Again, this issue can be addressed by centralized control that enables launching predetermined patterns for the nodes to join and leave the swarm. Furthermore, depending on the content the users, and thus also the network, behave differently. For example, a very popular piece of content may attract almost instantly an audience of millions of users, which may exceed the whole capacity of initial seeder. However, after more than one peer has a whole copy of the content, high availability is ensured. On the other end of the rainbow is marginal content, which may or may not be available due to lack of interest from the general public. These two examples show the available spectrum of possible test scenarios for emulated P2P testbeds.

Packet switched networks are dynamic by nature, making it difficult to produce simulated or emulated environment that accurately capture the whole of the dynamics. Therefore, simplifications and generalizations are needed to reduce the complexity of the system. It is an important design decision how a particular P2P system is built up and how it changes during a particular trial. We take the aforementioned challenges into account in the design of our system and present a detailed description in the following section.

3 Our Approach

Basically three different approaches to evaluate P2P-networks exist: analytical, simulation-based, and experimental. Experimental approach can further be divided in live network experiments concerning real users and in trials in controlled environment (see the related work section and references therein). In this work we focus on the controlled experimenting, because it provides a few advantages over the other approaches. First, real network equipment may be used. This is an advantage over analytical and simulation approaches, where the network behaviour can only be approximated. Second, the results gathered are reproducible. For example, the trials with real users are hard, if not impossible to reproduce and the results may be difficult to interpret correctly. Third, the network connection characteristics of peers can be controlled. This enables us to conduct network related research on P2P-networks, i.e. how different network configurations affect the overall performance.

3.1 Testbed Architecture

The general architecture of the testbed, as it is implemented using facilities of two VTT laboratories, is shown in Fig.1. The system uses a centralized control point (admin computer in the figure), where test cases are defined and executed. The control point is used also to configure the network profile i.e. the network emulators in the router machines in the network. Finally, the Converging Networks Laboratory [4] provides multiple wireless access networks that enable research with real equipment and in this respect simplifications are not needed.

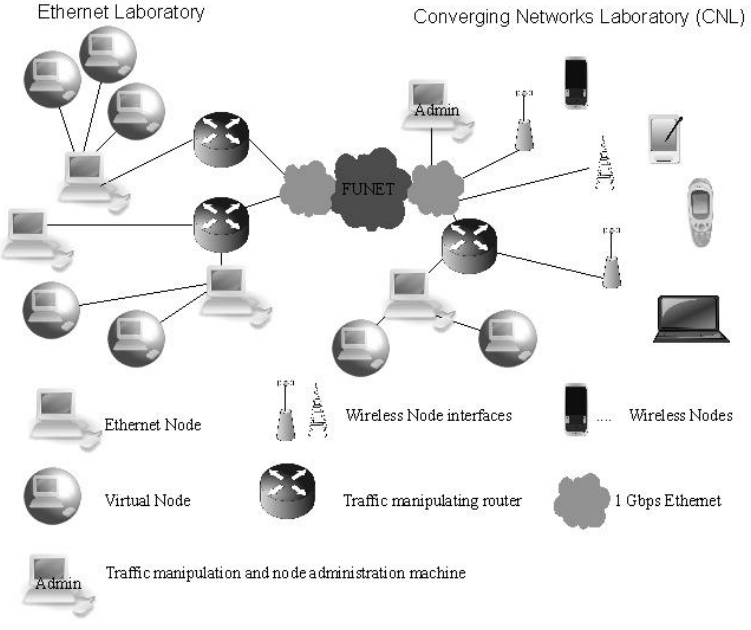


Fig. 1. The network topology of the testbed with the essential nodes

3.2 Testbed Components

The laboratory test platform consists of nodes running the applications to be tested and of traffic manipulating routers between the nodes. To restrict the number of real machines needed in the test, and to make their control easier, some nodes are run in virtual machines, while others are in real wireless devices. A centralized monitoring and control of the testbed is provided in order to enable reproducible measurements.

The traffic manipulators are routers or bridges equipped with network emulation functionality, which change the characteristics of the IP packet traffic passing through them. It is possible to emulate e.g. network connection equivalent to a typical home computer with wired connection (a broadband access via DSL or cable modem) in an Ethernet node. Several kinds of network congestion or malfunctioning cases can be created.

The testbed consists of three different types of nodes: Ethernet nodes, virtual nodes, and wireless nodes. The Ethernet nodes are machines with a wired Ethernet connection and virtual nodes are virtual machines running on the Ethernet node

machines. Following the example from the Onelab project [5], the Xen paravirtualisation system was used also in the testbed presented in this paper. The CNL laboratory provides following wireless interfaces: WLAN a/b/g/n, WiMAX d/e versions, UMTS/HSDPA cell. These are connected to the rest of the test network via 1 Gbps Ethernet. Emulation of wireless devices and networks is not needed in this case due to existing real infrastructure. These three types of nodes are used simultaneously to create test cases that emulate the heterogeneousness of the real network environment.

Central administration allows remote control of the traffic manipulating routers and nodes and is done from a central administration machine. It runs a web server, which provides a GUI for the traffic manipulator control. A tester can control the testbed with a web browser from any machine that has a network access to the web server.

3.3 Traffic Manipulation

From version 2.6.8 on, the standard Linux kernel includes a network emulation module, Netem [6], our choice for packet manipulation. Netem is one of the queuing disciplines available in Linux (A queuing discipline is a packet queue with an algorithm that decides when to send which packet [7]). Combining Netem with other disciplines, packet rate (i.e. bandwidth), delay, loss, corruption, duplication and re- ordering can be controlled [8]. In principle, these should be enough to emulate any kind of network level QoS (see e.g. [9], pp 3-4).

Possible open source alternatives for Netem would have been NIST Net [10] and Dummynet [11]. NIST Net runs in Linux but is no longer developed (Netem is partly based on NIST Net). Dummynet is part of FreeBSD and OS X firewall ipfw. It is older than Netem and has been perhaps the most popular network emulator. It is used e.g. in P2PLab (see [12] and Ch. 5). Its advantage over Netem (and NIST Net) has been, that it can manipulate both incoming and outgoing packets. By default Netem handles only egress queues. However, handling ingress queues can be achieved in Netem too by using the Intermediate Functional Block pseudo-device (IFB) or Ethernet bridging. On older Linux kernel versions, the timer accuracy of Netem was also inferior to Dummynet. In recent kernel versions, this has been corrected with High Resolution Timers subsystem. Also, the recent versions of Netem offer more features than Dummynet (e.g. packet duplication and corruption) [13].

With iproute2 [14] traffic control (tc) tool, the traffic can be classified with filters into several streams, each manipulated differently. Several types of filters exist. They may be based on routing rules, firewall marks or any value or combination of values in a packet (like source and destination IP addresses and the upper level protocol types). The two main usages of tc are to configure the queuing disciplines (qdiscs) and to configure packet classification into qdiscs [7].

A qdisc can be either classless or classful. Classful qdisc has a configurable internal subdivision, which allows packet classification, i.e. filtering into multiple streams. A qdisc tree can be built on a classful root qdisc. Since Netem is classless, it requires a separate root, when used with classification.

We use the Hierarchy Token Bucket (HTB, [15]) as a root qdisc. Besides classification, it is needed in packet rate control. It is a root qdisc choice also in [14], the other qdiscs being either classless or too complicated.

A drawback of tc is the relative complexity of its syntax of hierarchical structures, compared to e.g. Dummynet command-line interface. Our system overcomes this with a more intuitive graphical interface.

3.4 A Graphical Front End for Traffic Manipulation Control

To ease the control of traffic manipulation, a centralized graphical front end is provided for examination of the topology of the test network and for manipulation of the emulation models in the traffic manipulating routers. The GUI runs on a web server in the central administration machine.

The control and information data are exchanged via SSH-sessions between the administration machine and the manipulators and nodes (it is ensured by filtering, that the SSH-sessions are not manipulated).

3.4.1 Usage

The start page of the GUI shows a diagram of test network topology – the nodes, routers and different network types (Fig 2.). Choosing a router on the diagram displays a dialogue to launch the information and the queuing discipline model pages of the router.

The router information page provides functionality to fetch the link, routing and queuing discipline (qdiscs, classes, filters) information from the router via SSH. The queuing discipline model page enables examination of qdisc hierarchies created for the network interfaces of a router. A qdisc model is presented as a tree diagram (Fig.3.). Qdiscs, their classes and associated filters are shown as the tree leaves (the leaf parameters are shown as tooltips). The user can add or delete leaves or change their parameters. When the model is ready, the user can send it - i.e. the generated tc-commands - to the router.

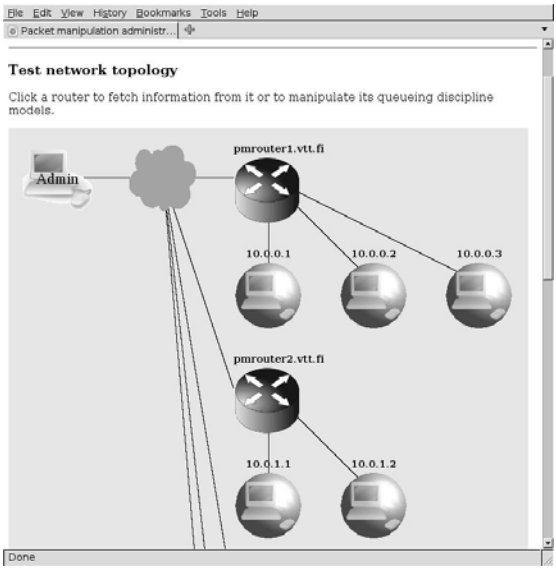


Fig. 2. The test network topology diagram on the start page of the GUI

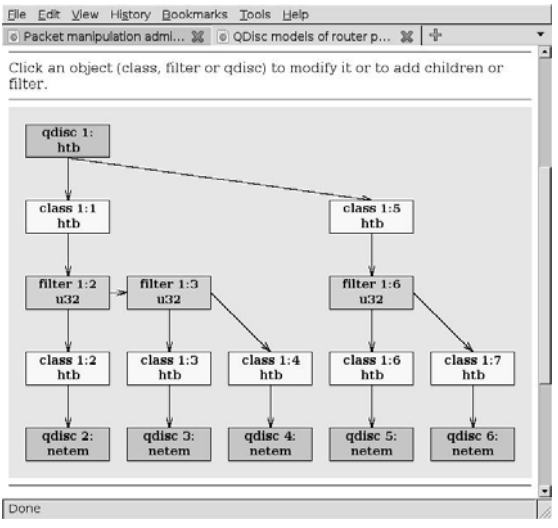


Fig. 3. The queuing discipline model diagram

3.4.2 Implementation

The HTML-content of the GUI is generated dynamically by Python scripts running on the Apache web server module mod_python [16]. Python is also used for maintaining and manipulating the model data and communicating with the traffic manipulators through SSH-connections.

Ajax ("Asynchronous JavaScript+CSS+DOM+XMLHttpRequest", [17]) enables dynamic update of content on an HTML-page. The GUI Ajax functionality of exchanging XMLHttpRequest objects is based on Prototype JavaScript framework [18].

The network topology and qdisc model diagrams are drawn with JavaScript Diagram Builder library [19]. The data of one qdisc model resides in an instance of a Python class and it contains the parameters and inheritance info of qdiscs, classes and filters. The model diagram can be unambiguously drawn from the class contents, as illustrated in Fig 3. The data is updated according to user creation, change or delete of leaves of the tree diagram. Inheritance changes, which would lead to errors when submitting model to a router, are prevented by the class and return an error message to the user. The class generates the tc-commands to be previewed or to be sent to the router. The model data chosen to be saved by the user is permanently stored to a file in Python pickle format, usable in subsequent test sessions.

3.5 Keyword Driven Test Execution Framework

The Test Execution Framework has been developed to support both functional and non-functional system testing of P2P software. In such an environment, functional testing consists of API and protocol compliance testing, while non-functional testing includes performance and security testing.

3.5.1 Conceptual Architecture

Fig. 4 presents the conceptual architecture of the test execution infrastructure. In the figure we have ignored physical nodes that do not directly take part in test execution process. It would be informative to compare this diagram with the actual physical topology as presented in Fig. 1.

Components that are important to the behaviour of the framework are: *Test Master, Test Agent, Test Peer* and a *Reporter.* The Test master is the single point of control for all the peers that are being managed by it. Test cases are managed and executed in test master. It also takes care of collecting logs and reports from agents to be stored in the file-system or database. To be able to manage peers from a remote manager, test master installs a thin agent (*Test Agent*) on each peer node. The test agent on each target peer mediates between the test master and the core peer-to-peer component on the peer node. An agent maps instruction received from the test master to APIs exposed by targets. *Test Peer*s are test script driven peers that are part of the testing ecosystem. These peer-nodes play different roles to interact with target peers depending on test scenarios. A *reporter* component is part of a P2P application that can be configured to report back logs or stats to the test master or any other monitoring node.

The test framework helps automate testing by automating individual steps in the testing process including installation, execution and report generation. Automated installation allows test master to check and install necessary test agents and other required libraries and packages on different peer nodes in an automated fashion. With the help of test framework, test cases can be organized in test suites, which can be automatically executed in a controlled manner. Also the framework helps test designers and test case writers to share test data across multiple test suites. Logs and reports generated on peer nodes are collected from each node and presented to test master automatically.

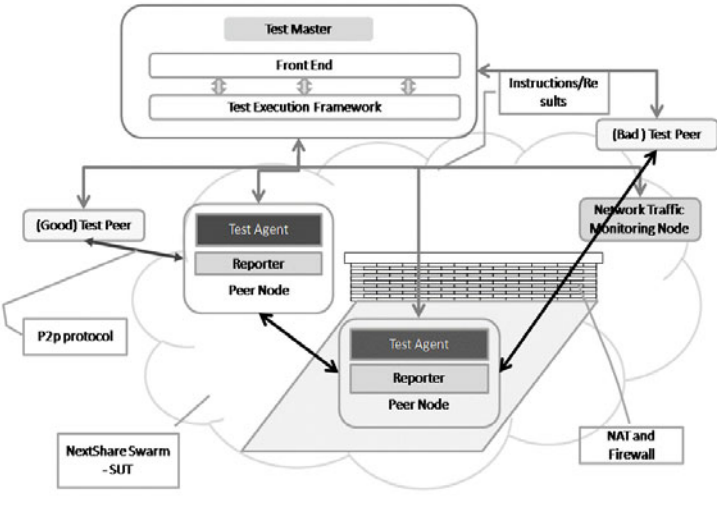


Fig. 4. Conceptual architecture of network with respect to test execution framework

3.5.2 Keyword Driven Test Automation

With a keyword driven framework, test automation is made more useful and robust by creating a test language around the system to be tested. The keyword driven framework consists of two parts, a core engine and the libraries implementing the test language. The language is comprised of a vocabulary of keywords that can be used with a number of arguments [20]. The test language is independent and therefore creates an abstraction layer between the test description and low-level vocabulary. Also, test vocabulary can have a hierarchical organization where low-level keywords can be combined to create complex high level keywords. Fig 5 shows a simplified example of such a hierarchy. The core engine interprets and executes the test cases with the help of the keyword libraries.

The keyword driven approach has several advantages over the more traditional methods. For example, writing test cases is simplified from a two step process of defining test cases and translating them to a programming language, to a one step process of writing test cases in test language pseudocode. This method decouples test case writing process from test automation [21]. Once the vocabulary is defined, it is possible to start developing testcases in parallel to test automation process. As the SUT behaviour is encapsulated within the test language, the core engine could be reused with other systems. With a well defined domain specific test language, test case developers are not required to have programming skills, while test automation engineers implementing the test language are not required to be system level subject matter experts. This system enhances reusability of test artifacts. The same set of test cases, data, reporting mechanism, comparison data and error handling can be used with different SUTs of the same type ensuring high reusability of the components as well as modularity [22]. Having a well defined vocabulary helps avoid inconsistency and enhances communication among different groups participating in system or product development.

The well defined division of ownership helps to improve the maintainability of automated test suites across configurations, versions and products. It is the responsibility of automation engineers to modify the language implementation when interfaces of SUT are changed while it is the responsibility of test case designers to add new vocabulary to the language and create corresponding test cases as new requirements are added to SUT requirements specification.

As P2P systems are being researched actively, multiple different implementations with varied interfaces are in development and require testing to compare functionalities and performance. With other types of frameworks a test case developer is obliged to understand each complex system and develop test cases for each system separately. In this case only the test language has to be ported by automation engineers.

For example some of the keywords we used in P2P media transport testing are: *start_player, stop_player, start_injector, stop_injector, start_swarm, stop_swarm, send_data_file_to_injector, collect_download_from_peer* etc. These are implemented with libraries that communicate with remote agents on peer nodes but the complexity of remote communication is completely hidden away from the testcase developers. So for instance to implement *start_swarm* test data is sent to the injector first, the injector is then instructed to start sharing content; metadata is collected from the injector which is then given to each peer, before it is instructed to join the swarm.

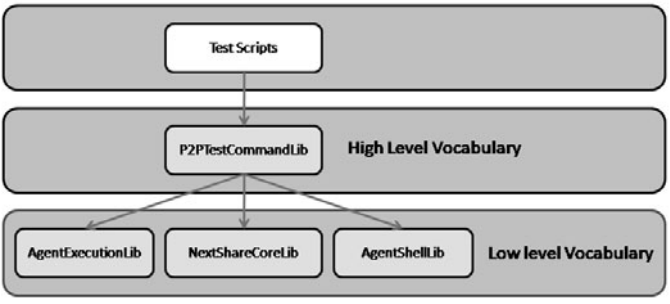


Fig. 5. Hierarchy of abstractions

4 Experiments with the Testbed

We evaluated the usefulness of our testbed by deploying a P2P video delivery application in the test network and running test cases. The application used was Nextshare, which is a video streaming application being developed in P2P-Next project [3]. The application is based on transferring the video data with essentially unchanged BitTorrent.

4.1 Trial Configuration

Our test configuration included 19 peers. One of the peers was using a wireless (IEEE 802.11g) connection to the test network and the rest of the peers were running on wired Ethernet or virtual nodes. The main BitTorrent parameters are as follows: maximum number of concurrent uploads 7, maximum number of connections not limited, maximum upload rate not limited. The traffic manipulation capabilities of our testbed were used to simulate realistic access network configurations for the wired nodes. According to [23], the average speed of an internet connection in Europe is 6.56 Mbps in the downlink and 1.43 Mbps in the uplink. We assume that these values are optimistic estimates, since the people with fast connection are more likely to use the speed tests than the ones with slow connections. Thus, we selected to use three different asymmetric access network configurations in our tests as presented in Table 1.

Table 1. Access network configurations

	Pessimistic	**Conservative**	**Optimistic**
Downlink speed (kbps)	1000	2000	6000
Uplink speed (kbps)	500	1000	1500

For all access network configuration scenarios, we run three test cases with different access network configurations. The test scenario, which is used in all test cases, is presented in Table 2. In the scenario, wireless peer is launched next after the seeder peer that has the original data source (64 MB file). The time for preparing the seeder (T_s) may vary a bit between the test runs, but is around 30 seconds in all cases.

Table 2. Test scenario description

Time (s)	Event
0	Seeder is launched
T_s	Seeder is ready and running
T_s + 10	Wireless peer is launched
T_s + 20	Wired peer #2 is launched
…	…
T_s + 180	Wired peer #18 is launched

4.2 The Experimental Results

While our testbed is able to perform detailed logging of all peers used in the test, we considered only the wireless peer in our evaluation. The statistics logged at the wireless peer were downlink data rate, uplink data rate, and the number of TCP connections to other Nextshare peers.

The results for the different access network configurations are presented in Fig. 6 for the pessimistic setup, in Fig. 7 for the conservative setup and in Fig. 8 for the optimistic setup. The effect of different access network configuration can be easily seen from the results. In the pessimistic case, the wireless peer does not get the file fully downloaded during the 1000 seconds measurement period. With the conservative settings, the download is finished after 775 seconds and with the optimistic settings after 655 seconds, as summarized in Table 3.

Table 3. Summary of numerical results

Testcase 1	Pessimistic	Conservative	Optimistic
Downlink speed (max/average kbps)	748/402	1440/800	1890/765
Uplink speed (max/average kbps)	7720/3080	12900/5930	17900/5740
Download complete time (s)	> 1000	775	655

The download speed is in all cases very limited because of the asymmetric limitations set to all the access network links of wired peers. In all cases the maximum download speed is still higher than the upload limit of the wired peers. This is possible because the download can happen from several peers simultaneously. The upload speed of the wireless node is significantly higher than download speed since that is not limited artificially and the peer is able to serve the other peers better than the wired peers. In addition, it can be noted that in the optimistic case, the capacity of the wireless link is the limiting factor for the uplink. In conservative and pessimistic cases, the wireless link is not fully used, but the bottleneck is in the wired peers' downlink limitation. The bottleneck could have been moved to the wireless link by increasing the maximum number of concurrent uploads of each peer (currently 7). This limitation also affects the rate of service that peers will receive: the one that arrives first can benefit from the upload slots of the ones that arrive later. On the other hand, the one arriving last will find that the upload slots of peers are already occupied, and it receives significantly lower data rates.

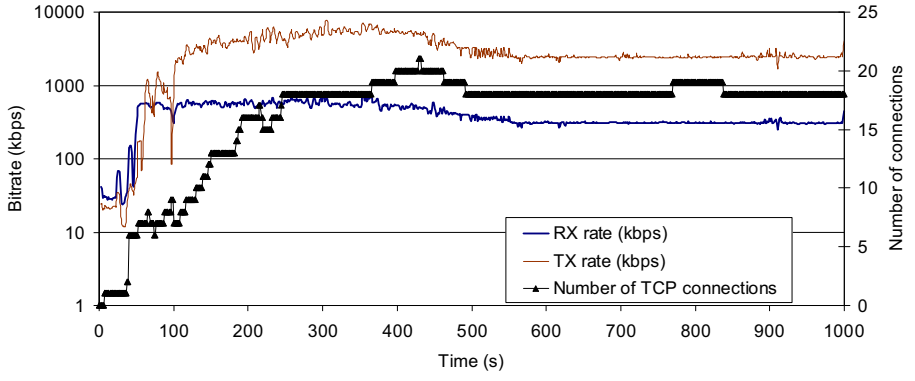


Fig. 6. Pessimistic setting

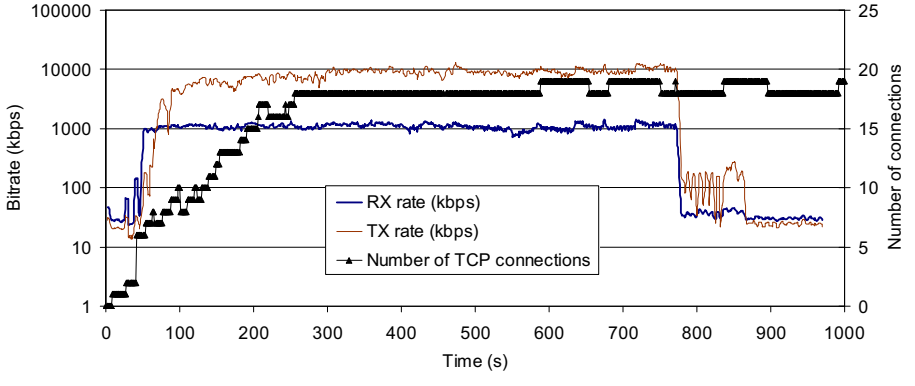


Fig. 7. Conservative setting

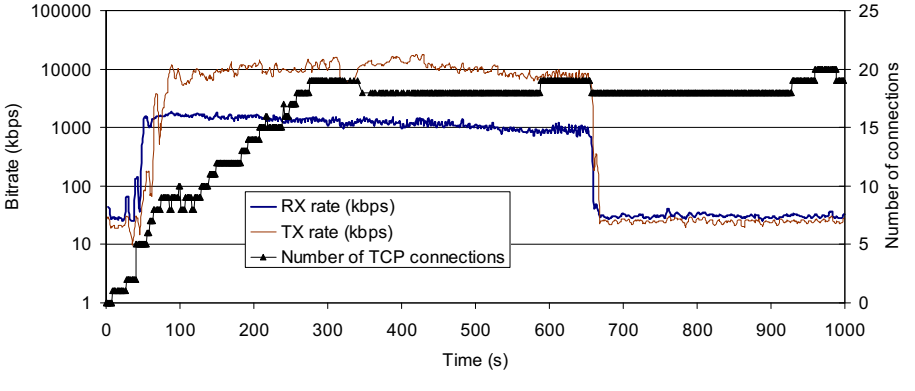


Fig. 8. Optimistic setting

5 Related Work

Basically three different approaches to evaluate P2P-networks exist: analytical, simulation-based, and experimental. Experimental approach can further be divided into live network trials with real users or experiments conducted in a controlled environment.

The analytical approach is utilized in [24], where traces are collected from real networks and then used to build an analytical model in order to study inter-torrent collaboration. Often the analytical model is developed to enable simulations, as is with the fluid model in [25]. More recent simulations have been presented for example in [26], and in [27] an extensive study about the state of P2P simulations is provided.

The experimental approach in live networks is taken e.g. in [2], where a BitTorrent network is studied over several months. Many interesting findings are presented, but the scope of the deployments concerning real users differs significantly from the controlled environment, where more rapid testing is possible. Marciniak et al. [28], experiments in a real Internet environment provided by PlanetLab [29]. Number of nodes is around 40 and does not change during the test session (the measurements in this case did not require it). The most important advantage of this approach is, naturally, the real-world environment, which is too complex to wholly reproduce by emulation. On the other end of the scale, P2PLab [12] uses network emulation and heavy virtualization of the nodes allowing tests with thousands of nodes. The authors of P2PLab consider a PlanetLab-like environment as difficult to control and modify. Also the test results may be difficult to reproduce in it. The TorrentLab [26] includes both network simulation and tools for conducting live experiments. It does not, however, permit the configuration of the network's routers or to emulate network level phenomena. An automated BitTorrent testing framework is presented also in [30], where the download performances of different BitTorrent clients are compared. The framework lacks, however, the possibility to use network emulation, wireless access networks, and a keyword driven test framework. LKDT [22] is a generic keyword driven distributed framework on linux that addresses limitations of other types of frameworks but does not satisfy peculiarities of a P2P system testing framework with multiple types of remotely located components. Another large-scale testbed is called OneLab [31] that is built on the foundation of Planet lab, but connects to more cutting edge access networks. Thus it is suitable for Future internet research, such as P2P overlay networks, but to the best of our knowledge the P2P networks have not been studied in it so far.

The presented framework can be seen as a phase in development flow that is after simulations, but before the experiments involving real users. The benefits of this approach are fast feedback on new developed features (e.g. protocol enhancements), reproducibility of the results, possibility to adjust network level features, and automated testing.

The aforementioned experimental works leave a gap: they either lack full control over the nodes, wireless access networks, network emulation, or keyword driven test execution framework. Our work takes advantage of all these features and is thus, to the best of our knowledge, the most advanced fully controllable P2P testbed so far.

6 Conclusion

This paper presented a novel testbed for P2P application testing with full control over the network and the peer nodes. The testbed enabled wireless network testing, keyword driven test execution, easy access network configuration of the peers, and thus it is key enabler for testing complex scenarios that involve variable network conditions and rapid changes in the topology of the P2P network.

In order to demonstrate the capabilities of the testbed, we executed three test cases with different access network configurations. From the results it can be seen that network conditions of peers have direct effect to the client. We performed also other scenarios and it was found out that the first peer joining the swarm had the best download rate, whereas the last one suffered a lot from the limited amount (7) of upload slots of the peers.

The future work will consist of three main topics: the improvement of the testbed and its control functions, more complex trials with Nextshare platform including dynamic network configuration and more realistic arrival patterns for peers, and finally we aim to increase the amount of virtual peers to enable larger scale tests. These allow us to help improving Nextshare and enable research on BitTorrent enhancements.

Acknowledgments. This work was supported by EU FP7 P2P-Next project.

References

1. BitTorrent, http://www.bittorrent.org/
2. Pouwelse, J.A., Garbacki, P., Epema, D.H.J., Sips, H.J.: The Bittorrent P2P File- Sharing System: Measurements and Analysis. In: Proc. of the 4th International Workshop on Peer-to-Peer Systems, IPTPS (2005)
3. P2P-Next project web page, http://www.p2p-next.org
4. VTT Converging Networks Laboratory, http://www.cnl.fi/
5. Canoni, R., Di Gennaro, P., Manetti, V., Venter, G.: Virtualization Techniques in Network Emulation Systems. In: Bougé, L., Forsell, M., Träff, J.L., Streit, A., Ziegler, W., Alexander, M., Childs, S. (eds.) Euro-Par Workshops 2007. LNCS, vol. 4854, pp. 144–153. Springer, Heidelberg (2008)
6. Net:Netem - The Linux Foundation, http://www.linuxfoundation.org/en/Net:Netem
7. Keller, A.: Manual: tc Packet Filtering and netem, ETH Zurich (July 2006), http://tcn.hypert.net/tcmanual.pdf
8. Hemminger, S.: Network Emulation with NetEm. In: Proceedings of the 6th Australia's National Linux Conference (LCA 2005), Canberra, Australia (April 2005)
9. Jha, S., Hassan, M.: Engineering Internet QoS. Artech House, Norwood (2002)
10. NIST Net Home Page, http://snad.ncsl.nist.gov/nistnet/
11. Dummynet home page, http://info.iet.unipi.it/~luigi/dummynet/
12. Nussbaum, L., Richard, O.: Lightweight emulation to study peer-to-peer systems. Concurrency and Computation: Practice and Experience 20(6), 735–749 (2008)
13. Nussbaum, L., Richard, O.: A Comparative Study of Network Link Emulators. In: 12th Communications and Networking Simulation Symposium, CNS 2009 (2009), http://www.loria.fr/~lnussbau/files/netemulators-cns09.pdf

14. Net:Iproute2 - The Linux Foundation, http://www.linuxfoundation.org/en/Net:Iproute2
15. Home HTB, http://luxik.cdi.cz/~devik/qos/htb/
16. Mod_python – Apache/Python Integration, http://www.modpython.org/
17. Garrett, J.J.: Ajax: A New Approach to Web Applications (February 2005), http://www.adaptivepath.com/ideas/essays/archives/000385.php
18. Prototype JavaScript framework: Easy Ajax and DOM manipulation for dynamic web applications, http://www.prototypejs.org/
19. JavaScript Diagram Builder, http://www.lutanho.net/diagram/
20. Fewster, M., Graham, D.: Software Test Automation. Addison-Wesley Professional, Reading
21. Kaner, C., Bach, J., Pettichord, B.: Lessons Learned in Software Testing: A Context Driven Approach. John Wiley & Sons, Chichester
22. Hui, J., Yuqing, L., Pei, L., Shuhang, G., Jing, G.: LKDT: A Keyword-Driven Based Distributed Test Framework. In: International Conference on Computer Science and Software Engineering, vol. 2, pp. 719–722 (2008), http://doi.ieeecomputersociety.org/10.1109/CSSE.2008.1036
23. World Speedtest.net Results, http://www.speedtest.net/global.php
24. Guo, L., Chen, S., Xiao, Z., Tan, E., Ding, X., Dang, X.: Measurements, analysis, and modeling of BitTorrent-like systems. In: Proc. of the 5th ACM SIGCOMM Conference on Internet Measurement, Berkeley, USA (2005)
25. Qiu, D., Srikant, R.: Modeling and Performance Analysis of BitTorrent-Like Peer-to-Peer Networks. In: SIGCOMM 2004, Portland, Oregon, USA, August 30–September 3 (2004)
26. Barcellos, M.P., Mansilha, R.B., Brasileiro, F.V.: TorrentLab: investigatingBitTorrent through simulation and live experiments. In: ISCC 2008 (July 6-9, 2008)
27. Naicken, S., Livingston, B., Basu, A., Rodhetbhai, S., Wakeman, I., Chalmers, D.: The state of peer-to-peer simulators and simulations. ACM SIGCOMM Computer Communication Review 37(2), 95–98 (2007)
28. Marciniak, P., Liogkas, N., Legout, A., Kohler, E.: Small Is Not Always Beautiful. In: 7th Internation. Workshop on Peer-to-Peer Systems (February 2008), http://www.cs.toronto.edu/iptps2008/final/55.pdf
29. PlanetLab, An open platform for developing, deploying, and accessing planetary- scale services, http://www.planet-lab.org/
30. Deaconescu, R., Rughinis, R., Tapus, N.: A BitTorrent Performance Evaluation Framework. In: 5th International Conference on Networking and Services (2009)
31. OneLab, Future Internet Testbeds, http://www.onelab.eu/

A Testbed for Validation and Assessment of Frame Switching Networks

Arthur Mutter[1], Sebastian Gunreben[1],
Wolfram Lautenschläger[2], and Martin Köhn[1]

[1] Universität Stuttgart - IKR, Pfaffenwaldring 47, 70569 Stuttgart, Germany
mutter@ikr.uni-stuttgart.de
[2] Alcatel-Lucent Deutschland AG, Bell Labs, Stuttgart, Germany

Abstract. Packet assembly at the network edge is one solution to reduce high packet rates in core network switches. Literature discusses this topic controversially because of three reasons: (1) potential negative impact of packet assembly on the traffic characteristics, (2) disruptive integration into existing networks and (3) lack of support of packet assembly in existing control plane environment.

In this paper, we introduce our testbed with 10 Gbps packet assembly nodes at the network edge and a GMPLS (Generalized Multi-Protocol Label Switching) control plane for its management. Our testbed integrates transparently in existing Ethernet based networks and allows comprehensive studies on the impact of packet assembly on the traffic characteristics by measurements. Beside feasibility, for early findings, we setup a measurement scenario and quantified the impact of packet assembly in terms of packet latency. For realistic traffic conditions, we found that the additional delays and jitter introduced by the assembly process are negligible.

1 Introduction

The popularity of the Internet causes growth of data volume and interface rates in packet core networks, i. e., 100 Gbps interfaces are currently on the way to standardization [11]. About half of the packets[1] in these networks is smaller than 200 Byte. Reasons are the acknowledgement packets of upper layer transport protocols, like the Transmission Control Protocol (TCP, [1]). With minimum size Ethernet packets, the packet rate on a 100 Gbps link is as high as 149 Million packets per second (preamble and inter-framing gap considered). A 100 Gbps capable network node has to be able to accept and deliver a minimum size packet every 6.72 ns. Current network processors operate at frequencies of several hundred MHz which translates to clock periods of 1 to 5 ns. This means a network processor has to accept and deliver a packet every few clock cycles. This puts a high burden on the realization. Consequently, higher data rates dramatically increase the processing and switching effort within core switches and raise their complexity.

[1] caida.org, the Cooperative Association for Internet Data Analysis, 2008.

T. Magedanz et al. (Eds.): TridentCom 2010, LNICST 46, pp. 84–99, 2011.

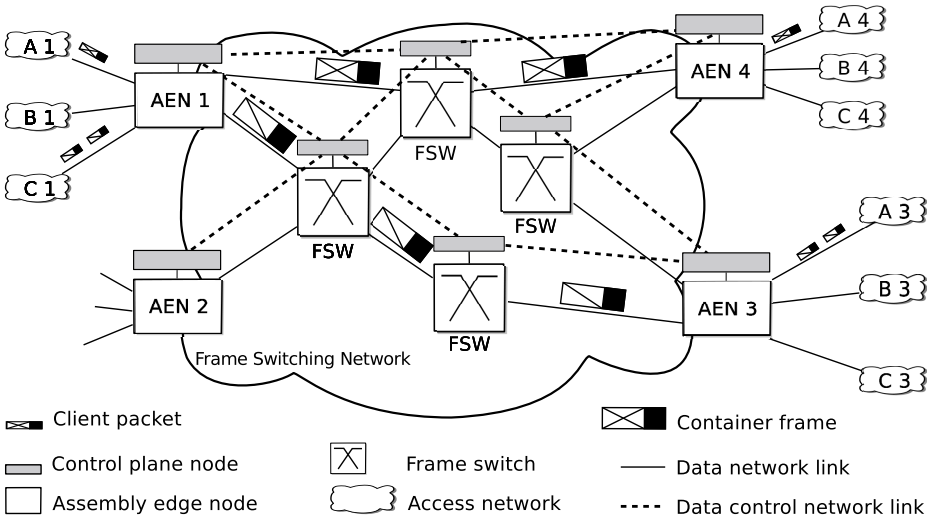


Fig. 1. Frame Switching network architecture

One approach to alleviate this problem is to reduce the number of packets in these networks, i. e., to reduce the packet rate. A prominent solution is the assembly of client layer packets into server layer containers at the network edge. The Frame Switching (FS) architecture relies on this principle [19]. FS is a connection oriented packet switching (CO/PS) network technology for metro and core networks. Fig. 1 depicts a FS network. It consists of edge nodes, namely Assembly Edge Nodes (AEN) and core nodes, namely Frame Switches (FSW). A path through the network includes two AENs – one at each end – and a sequence of FSWs between them. The CO character of this technology is ensured by a preceding connection setup phase. The signaling phase propagates traffic engineering properties along the path, e. g., relative Quality of Service (QoS) and required bandwidth.

At network edge, the AEN assembles in ingress direction multiple client packets of the same forward equivalent class (FEC) into containers and forwards these containers to the next FSW according to the next hop of the signaled path. Packets of one FEC belong together according to a superior order, e. g., belong to the same flow or need to traverse the same target AEN. The FSWs forward these containers according to a forwarding table and maintain relative QoS among different paths. The egress AEN disassembles the containers to individual packets and forwards them individually towards their destination client.

FS avoids any cross layer interactions and operates on layer 2 transparently to upper layer protocols. Because of the layering, we use the term *frame* for the containers carrying assembled packets. The originally foreseen layer 2 technology for FS networks was an extended Optical Transport Network (OTN, [12]). The server layer frame then corresponds to a G.709 frame [13]. However, the concept of packet assembly at the network edge is also applicable to other layer 2

technologies. The most prominent layer 2 technology for cheap and simple implementation is Ethernet [8].

Packet assembly maps multiple smaller packets into larger frames. In case of fixed size frames, packet segmentation is necessary to provide 100 % fill ratio. Additionally, in case of low load situations and fixed frame sizes, also padding is required. Besides segmentation and padding, frame assembly at the network edge changes the original traffic characteristic of assembled packets. Fig. 2 depicts this property. Arriving packets show a certain inter-arrival time characteristic (left side of Fig. 2). The AEN assembles these packets into one frame (middle). At the egress side, the AEN disassembles the frame to packets and releases them in a back-to-back manner without the original inter-arrival time characteristic (right side of Fig. 2).

It is assumed that the additional latency or jitter introduced by the assembly process leads to temporary buffer overflows at the network edge due to the burstification of the traffic or affect the upper layer protocol performance negatively. These assumptions are the major argument against any packet assembly technology in packet based networks. On the other hand, these assumptions are hard to quantify without a real network environment and real network conditions. Another open point of packet assembly networks is their controllability as packet assembly requires additional signaling because of the assembly process. This missing support of packet assembly in present control plane protocols leads to additional skepticism.

In this paper, we present, to the best of our knowledge, the first time a FS testbed including both the data plane and the control plane (CP). Our FS testbed enables studies on both topics, i. e., the packet assembly process and its impact on the traffic characteristics as well as the controllability of these networks. Our testbed relies on Ethernet technology and includes bidirectional AENs. Our AEN prototype supports various packet assembly schemes, i. e., combined timer and threshold based assembly. The resulting frames may be of fixed or variable size supporting packet segmentation and padding. The testbed and the AEN operate at 10 Gbps. Additionally, our testbed implements a CP for path management. Our CP bases on the DRAGON Generalized Multi-Protocol Label Switching (GMPLS, [30]) implementation and shows extensions to support the packet assembly functionality of FS networks. This especially includes the signaling protocols to signal service classes and the parameters of the assembly scheme.

1.1 Related Work

Considering the data plane, literature presents several architectures and implementations of assembly nodes as well as testbeds within the context of Optical Burst Switching (OBS) networks [21, 24, 32]. They focus in their studies mainly on technological problems and less on the realization of packet assembly. Thereby they support only variable length bursts and thus do not consider packet segmentation or padding. Additionally to the optical layer, Kögel et al. implement in [16] an assembly node on a network processor. However, on network processors, scalability and throughput are limited due to the software implementation.

In our earlier publication [23], we present our assembly edge node, exclusively focusing on the data plane part. We show its architecture and give detailed information about its implementation and supported features.

Kornaros et al. propose in [15] an assembly node architecture based on G.709 frames. For a detailed comparison to our assembly edge node we refer to [23]. However, the authors in [15] do neither report on any control plane interface for extensive path management support nor describe a testbed for validation and testing.

Selected testbeds related to the control plane of FS networks apply the GMPLS protocol suite. This includes both, Optical Burst Switching networks, e. g., [26, 29] as well as transport networks, e. g., [22]. The focus on the control plane studies in OBS lies on the interaction of both control planes: connection-oriented GMPLS CP for lightpath reservation and CPs for connection-less OBS for temporary lightpath occupancy. However, the frame switching network is connection oriented while OBS is connectionless and thus may apply any GMPLS CP directly. Additionally, none of the proposals in literature realizes the parameterization of the assembly edge node via the control plane. The focus on the control plane studies for transport networks lies on the support of optical technologies, inter-domain and multi-layer issues. These studies neither focus on the support of assembly units nor its impact on the control plane.

Summarizing, literature intensively studies the field of packet assembly on selected aspects. However, none of these studies provide a testbed, which (a) transparently integrates in existing networks, (b) provides equivalent capabilities for packet assembly strategies or (c) supports network operation by a standard compliant decentralized control plane.

1.2 Organization of the Paper

In Section 2, we introduce the principles of our testbed architecture. In Section 3, we introduce the prototype architecture and functionality of our frame assembly unit. Section 4 presents the control plane architecture of the frame switching network. Besides the functional validation of the testbed, we provide preliminary measurements to quantify the impact of the assembly process on the traffic characteristics in Section 5. Section 6 summarizes our work.

2 Introduction to Our Frame Switching Testbed

Our FS testbed uses Ethernet as the transport technology for the FS network between two AENs, cf. Fig. 1. We use Ethernet jumbo of fixed size (9 KByte) as container frames to transport the assembled packets. For easy and transparent integration in existing networks, the AEN also provide Ethernet interfaces to the client networks at the edge. The frame switches (FSW) along a path from one AEN to the destination AEN are manageable Ethernet switches. The following paragraphs detail the principles of the FS testbed architecture.

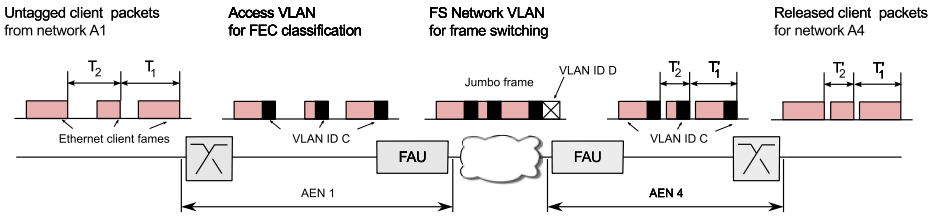


Fig. 2. Packet classification and assembly

The AEN shows two independent functions: switching and assembly. Therefore, the realization of an AEN shows two options. (1) One monolithic device incorporating both functionalities or (2) a modular device separating the functionalities. The advantage of (1) is the single device and potential resource savings due to common usage of components, e. g., a common packet buffer. However, the common components have to suffice a larger number of requirements and are therefore more complex.

We follow the modular approach (2) and separate the functionalities. Fig. 3 depicts our modular AEN architecture. It shows one Ethernet switch and multiple Frame Assembly Units (FAU). Both are interconnected to the control plane via a Control Channel Interface (CCI). Our modular approach is highly flexible as it allows adding FAU ports to an AEN incrementally.

The Ethernet switch serves two purposes. (a) Classification of packets on a per port basis, i. e., packets on one port belong to one FEC. (b) Switching of classified packets to the correct FAU.

The classification process at the AEN Ethernet switch assigns all packets on certain ports to one Virtual LAN (VLAN, cf. Ethernet VLAN extension of [7]). Packets that belong to the same VLAN have the same VLAN identifier (VLAN ID). The specific VLAN ID is selected in the connection setup procedure (cf. Section 4). The selected VLAN ID per port serves as a classification criteria for the subsequent FAU. The FAU assembles packets per FEC (i. e., per VLAN ID) and creates frames (cf. Section 3). The switch in the egress AEN will also use this VLAN ID to switch the packets to the appropriate client network. Additionally, the utilization of VLAN at this point enables to decouple the classification process in the FAU from the Ethernet MAC addresses of the clients. This drastically simplifies configuration.

Ethernet is a connectionless packet-switching technology with a separate control plane (Spanning Tree Protocol, [6]) for link management. This forbids any traffic engineering in Ethernet based FS networks. The virtual LAN extension of Ethernet (VLAN, [7]) alleviates this problem and provides connections to FS networks. In the core one connection corresponds to one VLAN. The ports of the FSW define the path of a connection, i. e., two ports per FSW along the path belong to the same VLAN. Please note that the VLAN ID of the classification process is not necessarily the same as the VLAN ID in the FS network. Ethernet limits the number of VLAN to 2^{12}. This restricts the application of this concept in large networks. However, the Provider Bridge (PB, [9]) and the Provider

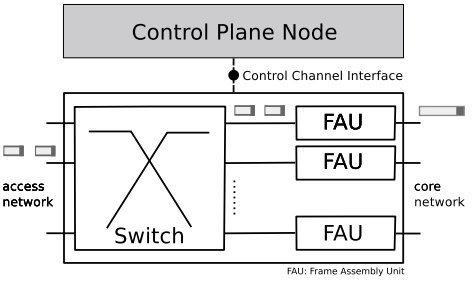


Fig. 3. Architecture of the assembly edge node (AEN)

Backbone Bridge (PBB, [10]) proposals alleviate this limitation by additional Ethernet VLAN and frame headers, respectively. Consequently, this paper does not elaborate on this problem.

Fig. 2 depicts from left to right the whole classification, encapsulation and forwarding process from the connection A1 to A4 of Fig. 1. Additionally, it distinguishes the different VLAN IDs. The Ethernet packets of the client network A1 arrive at the AEN 1 switch. The AEN 1 switch classifies these packets per port and assigns these packets the VLAN ID C. The VLAN tagged packets leave the AEN 1 switch and face the FAU for packet assembly. The FAU assembles packets of one FEC (i. e., same VLAN ID) to Ethernet jumbo frames of 9 KByte. The FAU assigns these Ethernet jumbo frames a VLAN ID (here VLAN ID D), which is used in the FS network to switch the jumbo frames. The FAU in the destination AEN 4 disassembles the Ethernet jumbo frames. This FAU forwards the original client packets including the classification VLAN ID (VLAN ID C) to the AEN 4 switch. The switch removes the classification VLAN ID and forwards the individual packets to the client network A4.

The big advantage of this MAC in MAC encapsulation is the transparency to any upper layer protocol. For instance, ARP (Address Resolution Protocol, [27]) and IP (Internet Protocol, [28]) protocols are transparently switched without interfering with the FS transport layer.

Additionally, each connection belongs to a selected class of service. The FS testbed realizes service differentiation on a Differential Service basis (DiffServ, [3]). The information on the service class is included in the header of the assembled frame (3 Bit user priority of the VLAN header, [7]). The Ethernet switches in the FS network (i. e., FSW) use these priority bits to relatively prioritize the jumbo frames.

The next sections introduce the architecture and implementation of the FAU as well as the implemented control plane instance to manage this testbed.

3 FAU Prototype Architecture

The FAU assembles packets to frames in ingress direction and disassembly frames to packets in reverse direction. Fig. 4 depicts the ingress direction of the FAU prototype architecture. The lower part shows the pipelined architecture of the

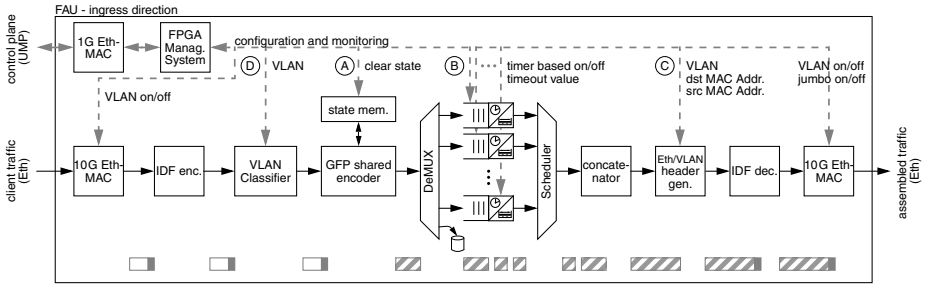


Fig. 4. Architecture of FAU prototype: ingress direction

packet processing. The upper part shows the interface to the control plane for runtime configuration. The following two sections describe both parts in detail.

We realized the FAU on an evaluation board with two Xilinx Virtex-4 FX100 FPGAs, two optical 10 Gigabit Ethernet interfaces for data plane and a 1 Gigabit Ethernet interface for CP connection. We designed the FAU prototype in VHDL supporting simultaneously seven FECs per direction.

3.1 Data Plane Part

The description in this section introduces the data plane part of the ingress direction of the FAU. For a more detailed introduction the reader is referred to our earlier publication [23].

The description of Fig. 4 follows the assembly process from left to right, i. e., from an access to a core network. The first stage, the 10 G Ethernet interface, receives packets, checks their frame checking sequence (FCS) and forwards them without the FCS. The next stage converts the packets into an internal data format (IDF) and therefore adds an IDF header.

The VLAN classifier classifies client packets according to their VLAN ID (12 Bit field in the VLAN header). Packets belonging to one VLAN belong to the same forwarding equivalent class (FEC). The VLAN classifier stage stores the information to which FEC a packet belongs in the IDF header.

The next stage encodes packets with the ITU-T Generic Framing Procedure (GFP, [14]). GFP encapsulates each packet by adding meta information (the GFP core and payload headers) that allows delineation at the destination. GFP supports packet segmentation as the delineation process relies only on the meta information and is independent of container frame boarders. Additionally, GFP operates stateful per FEC. Consequently, a new connection requires a clear state of the GFP encoder before operating.

The assembly stage consists of many assembly components. Each component serves one FEC. Packets not assigned to any FEC are dropped here. Each assembly component supports combined timer and threshold based packet assembly. The buffer of each assembly component collects packets of one FEC. In every component a control block monitors the fill level of this buffer. When the size threshold is exceeded or in case of timeout, the control block signals the scheduler

that a frame is ready to be sent. Since frame switching uses constant size frames, the assembly components segment packets to fill frames completely. Further, it appends padding if the amount of collected packet data is below the frame size.

If more than one assembly component signals a ready frame, the scheduler assigns the outgoing link in Round Robin manner. On request of the scheduler the assembly component transmits the GFP encoded packets or segments of a packet to the concatenator stage to build the frame. The concatenator stage removes the IDF headers of the GFP encoded packets, aligns the data and creates a continuous data block, which represents the frame's payload.

The next stage finalizes the Ethernet jumbo frame by adding to the payload an Ethernet and a VLAN header according to the frame's FEC. The IDF decoder stage removes the IDF header of the frame. The last stage, the 10 G Ethernet interface, appends the jumbo frame a FCS and transmits it to the core.

The data plane part of the egress direction is similar. [23] shows it in detail.

3.2 Control Plane Part

At several stages of the data path, the FAU requires status information and configuration data. For runtime configuration, the FAU foresees a management interface for interactions with the control plane. We developed an FPGA Management System (FMS) as CP interface. Communication with the CP is realized by utilizing an additional 1 Gigabit Ethernet interface (cf. Fig. 4). The FMS enables online read and write access to register and memory values via our self-developed low-level configuration protocol UMP (Universal Hardware Platform Management Protocol). UMP resides on address value pairs.

After powering up the testbed, the CP configures the 10 Gigabit Ethernet interfaces of each FAU to enable support for Ethernet VLAN header extension as well as Ethernet jumbo frames. Additionally, it assigns to each Ethernet interface a unique MAC address that is used as source address in the Eth./VLAN header generator stage. The number of simultaneously supported connections (i. e., FECs) is limited by the available hardware resources in the device. A new connection requires the necessary configuration steps to be done in a fixed order. The setup requires in the ingress FAU the following two steps to be done for the specific FEC:

1. Configure individual stages in the FAU pipeline.
   - Clear state of this FEC in GFP encoder stage (A in Fig. 4).
   - Enable or disable timer-based assembly in the corresponding assembly component. If enabled, then set the timeout value for the timer in units of 10 ns (B in Fig. 4).
   - Set VLAN ID, VLAN priority and Ethernet destination address in the Eth./VLAN header generation stage for this FEC (C in Fig. 4).
2. Set the VLAN ID in the VLAN classifier stage for this FEC (D in Fig. 4).

The order of these two steps is mandatory as the last step acts like a gatekeeper for the incoming packets. Before the first packet enters the FAU, the FAU has to be configured properly. After this step, the classifier labels packets for a certain

FEC. To tear down this connection the control plane has again to keep a fixed order of the necessary two steps. The classifier step first stops incoming packets, while the second step empties the assembly component of this FEC.

1. Remove the VLAN ID mapping of this FEC in the VLAN classifier (D in Fig. 4).
2. Enable timeout based assembly in this FEC to transmit all remaining data in case of only threshold based assembly (B in Fig. 4).

In the egress FAU similar configuration steps are required for connection setup and tear down.

4 Control Plane for Frame Switching Networks

This section introduces the control plane for frame switching networks as well as the control plane implementation of our testbed. The major requirements on any control plane are traffic engineering capabilities and path management support. The GMPLS CP architecture [30] is a promising candidate to control future networks and new network technologies. The CP of the FS testbed resides on the GMPLS framework of the US project DRAGON and has been extended to support FS networks. Extensions include the signaling protocols as well as the interfaces between the control plane and the data plane node. The first section details the DRAGON GMPLS CP. The second section introduces both control plane extensions.

4.1 The DRAGON GMPLS Control Plane

The FS CP resides on the GMPLS implementation of parts of the US project DRAGON (Dynamic Resource Allocation via GMPLS Optical Networks, [18]). This project targeted to support non-GMPLS capable network nodes like Ethernet switches, optical and Time Division Multiplex (TDM) devices. Fig. 5 shows our setup. In the upper part are the elements of the DRAGON CP: The Virtual Label Switching Routers (VLSR), the DRAGON User Network Interface (UNI) and the Network Aware Resource Broker (NARB, [34]).

The NARB enables constraint based path computation and interacts with other NARB elements in inter-domain scenarios. For instance, constraint path computation occurs if a connection request already includes the VLAN ID along the path. For non-constraint path computation, NARB selects one VLAN ID among the announced. The NARB partly fulfils the functionality of the IETF Path Computation Element (PCE, [5]).

The DRAGON UNI provides the interface for rudimentary path management. It enables setup and teardown of paths (VLAN connection). It triggers the path computation and signaling steps, which configure the switch ports appropriately. The DRAGON UNI implements the Optical Internet Forum interface (OIF 1.0, [25]) and realizes an UNI for an overlay model with a GMPLS environment as described in [33]. Additionally for manual operation, the UNI implements a command line interface (CLI) to manage connections.

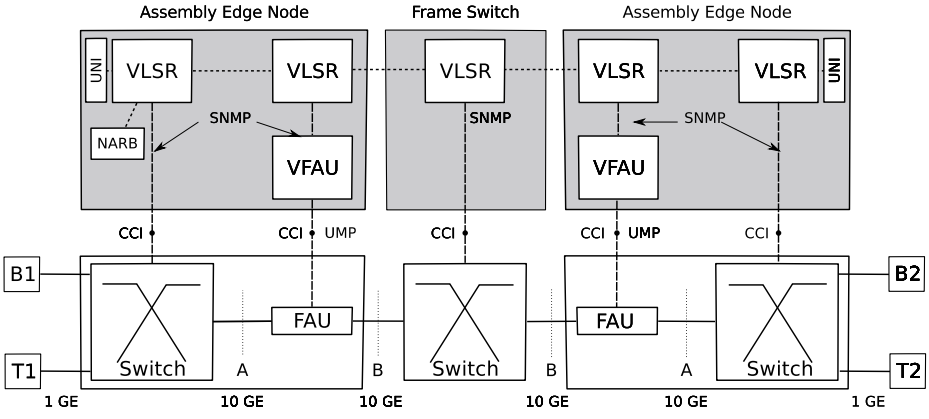


Fig. 5. Frame Switching Testbed Architecture

The VLSRs represent the non GMPLS-capable AENs (Switch, FAU) and the FSW in the CP and operate signaling and routing protocols, i. e., the Resource Reservation Protocol (RSVP-TE, [33]) and the Open Shortest Path Protocol (OSPF, [31]). On the control channel interface, VLSR and AEN/FSW exchange configuration data, i. e., requesting or changing the device state. The communication protocol is the Simple Network Management Protocol (SNMP, [4]). The manageable switches (in the AEN and FSW) support SNMP and implement the manageable objects for Ethernet VLAN extensions according to [2]. In detail, the VLSR performs the following actions on the CCI:

- Query of available or used VLANs per port.
- Assignment of ports to a VLAN.
- Removal of ports from a VLAN.
- Change of port state with respect to trunk and access ports.

While the manageable switches implement the corresponding MIB (Management Information Base) and the SNMP application interface, the FAU does not. The FAU implements the proprietary Ethernet based UMP protocol of Section 3.2. For easy integration in the framework (i. e., without changing the interface to the VLSR), Section 4.2 introduces our gateway, which translates SNMP messages to the proprietary UMP protocol.

4.2 Extensions to the DRAGON GMPLS Control Plane

We extended the DRAGON GMPLS control plane to support the parameters of the FAU, i. e., assembly scheme configuration and class of service (CoS). This includes changes in the signaling protocol RSVP and requires an application layer gateway at the CCI between VLSR and FAU. The following sections address these two extensions in detail.

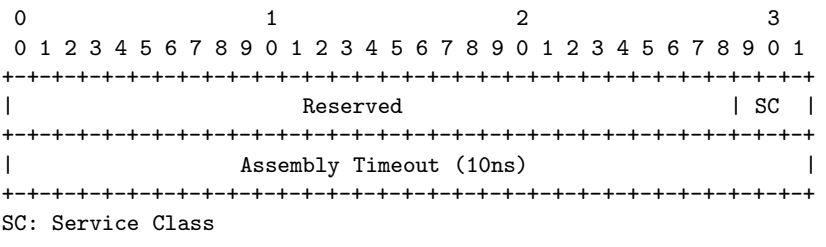


Fig. 6. Extended ATM service class object, [20]

Additional signaling parameters. The path setup procedure in our testbed requires signaling of the VLAN ID, CoS and the parameters of the assembly process. The VLAN ID corresponds to the signaling of labels in a GMPLS framework and the signaling protocol RSVP. The GMPLS control plane inherently supports this without any modification. However, RSVP does not support directly any CoS and the assembly parameters. To signal these parameters we implemented a modified RSVP ATM service class object, [20]. The ATM Service Class object extends the RSVP-TE Path message and includes 3 Bit to signal ATM service classes for any label switched path, i. e., UBR (Unspecified Bit Rate), VBR-NRT (Variable Bit Rate, Non-Real Time), VBR-RT (Variable Bit Rate, Real Time), CBR (Constant Bit Rate). As the aim is exactly the same as for FS networks, we reuse this object for the FS CoS. The 3 Bit for the ATM service classes perfectly correspond to the 3 Bit VLAN priority of the VLAN header.

We add the information on the assembly timeout parameter to the ATM service class object. Therefore, we extended the original ATM service class object by another 32 Bit indicating the timeout parameter in units of 10 ns. A value of 0 indicates pure threshold based assembly. Fig. 6 depicts the new object. We included this modified object in the RSVP protocol and were able to signal service class and assembly parameterization along the path. The assembly parameter of the size threshold is implicitly given by the constant frame size of Ethernet jumbo frames, i. e., 9 KByte.

Besides the extension of the signaling protocol RSVP, we also extended the UNI to include CoS information and the parameterization of the assembly timeout. These extensions include mainly the modification of the command line interface of the DRAGON UNI as well as the interface to the signaling protocol. As both do not affect standardized protocols, this paper skips the details here.

Control channel interface. The DRAGON VLSR implements the SNMP protocol at the interfaces to the switches and the FAU. Section 3 already introduced that the FAU does not support the high level language SNMP. Instead, it provides a lower level language to access registers and the memory of the FAU. This discrepancy requires a protocol gateway, which translates high level SNMP messages into low level UMP messages. Fig. 7 depicts schematically the gateway, i. e., the Virtual Frame Assembly Unit (VFAU).

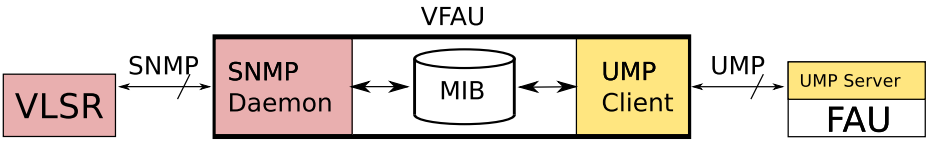


Fig. 7. Virtual FAU (VFAU)

On the interface towards the VLSR, the VFAU emulates a simple two port switch. For communication compatibility, the VFAU implements a SNMP daemon. For VLAN application in Ethernet networks, the VFAU implements the required Management Information Database (MIB, [2]) as the other switches in this network, too. On the interface towards the FAU, the VFAU implements a UMP Client. The UMP Client serves as a backend of the MIB. SNMP Messages, which operate on MIB entries (cf. Section 4.1), automatically trigger the corresponding UMP messages to put this in action on the FAU. The UMP messages include the necessary configuration data on the FAU, cf. Section 3.

Besides these configuration messages, the gateway also manages the resource state of the FAU. Each FAU is able to handle a limited number of FEC per direction, i. e., in our implementation we support 7 per direction. If the number of available resources is exhausted, the gateway signals this information to the VLSR and prohibits additional path setups until paths terminate. The information on the resource occupation state is configured in the FAU and stored and updated in the gateway.

5 Measurement Results

Our testbed in Fig. 5 and its realization in Fig. 8 not only demonstrate the feasibility of frame switching networks, but also enable studies on the impact of packet assembly. For validation of the functionality of the testbed and for preliminary results regarding packet assembly, we setup a measurement environment. Our testbed consists of three AENs (Fig. 5 shows only two AENs because of space limitations) and one core switch representing the FS core network.

The inherent burstification of the transported traffic is one of the major sources of skepticism by practitioners and network operators. The frame assembly may delay packets until the end of the assembly process (time/size threshold reached) at the assembly edge node. This waiting time follows a load dependent distribution, which leads to an additional jitter. These changes in traffic characteristics may cause problems on higher layers. Nonetheless, our preliminary results indicate that the additional delay introduced by the frame assembly is well controlled and remains small enough compared to other delay sources, e. g., queuing or propagation delay.

For our study, we used the setup of Fig. 5. We study the impact of frame assembly on a test flow in the presence of a background flow. The background flow represents the aggregated traffic of several users. It originates at B1 and

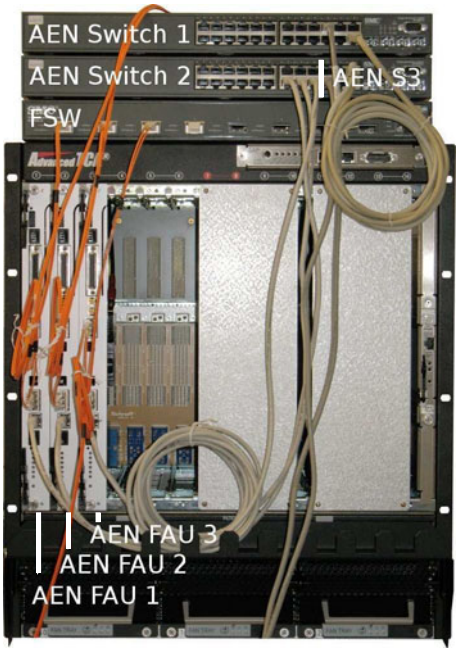


Fig. 8. Frame Switching Testbed

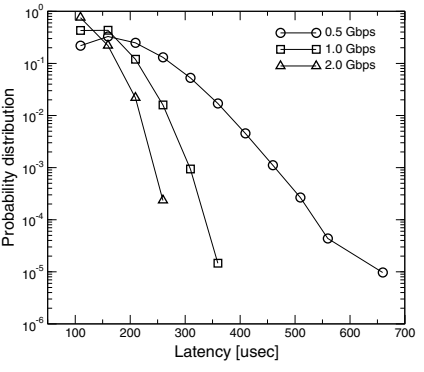


Fig. 9. Packet latency

terminates at B2. The test flow represents the traffic of one user or application. It originates at T1 and terminates at T2. Both flows share the same FEC and thus share the same resources in both FAUs. The target of our study is the experienced latency of the test flow with respect to a changing background flow.

The background traffic has an average rate of 0.5, 1, or 2 Gbps, respectively. We compose the background traffic by an overlay of randomly arriving 10 Mbps application streams and use the same traffic model as in [17]. The average rate of the test flow is 10 Mbps. It is composed by 500 Byte packets showing a constant inter-arrival time. At T2, we record the latency of the packets after traversal of the testbed. We use only threshold-based assembly with fixed size (9 KByte) jumbo frames. The timer-based assembly is switched off.

Fig. 9 shows the empirical probability distribution of the latency of the test flow packets with respect to different background load levels. All three curves show a constant offset of about 110 μsec propagation delay.

The maximum of each distribution shifts reciprocally with the traffic load, i. e., the distribution moves to the right with decreasing traffic load. These findings correspond to a load dependent waiting time during frame assembly. In low load situations, the latency shows a rather long tail, because only threshold based assembly scheme is used. However, any timer-based assembly would limit the maximum additional delay and cut the distribution.

Finally, we point out the small absolute values of the measured jitter. With our realistic traffic conditions, we observed jitter values far below 1 ms. As each packet receives this jitter only once at the network ingress. So the jitter does not accumulate with the network size. Consequently, we do not expect these jitter values to significantly influence the higher layers.

6 Conclusion

The contribution of our paper is twofold. We show the feasibility of frame switching networks and provide a testbed, which enables comprehensive studies on the impact of packet assembly on the traffic characteristics.

For transparent operation and easy integration in existing networks, our testbed resides on connection-oriented Ethernet and operates on 10 Gbps. The two major achievements of our testbed are the assembly edge nodes and a GMPLS control plane. The assembly edge node consists of a switching device and the frame assembly unit, which assembles packets to larger containers, named frames. The GMPLS control plane is responsible for path management.

Our frame assembly unit supports timer and threshold based assembly, fixed and variable size frames, packet segmentation for a 100 % frame fill ratio and the ITU-T Generic Framing Procedure for packet delineation. The architecture is modular to ease design adaptation to any packet oriented transport technology. We implemented the design in VHDL to show its feasibility and validated its functionality within our testbed.

Our control plane for Frame Switching networks applies the GMPLS control plane implementation of the DRAGON project. We extended the signaling protocols and the UNI interface to adapt to the special requirements of frame assembly at the network edge, i. e., QoS classes and assembly scheme parameters. The resulting control plane implementation is capable to setup and tear down end-to-end connections of different service classes between access networks. Each connection request includes the parameterization of the frame assembly units as well as of the intermediate switches.

For preliminary results, we setup a network scenario with realistic traffic conditions and measured the impact of the packet assembly scheme in terms of packet latency. We found that the absolute delay and jitter introduced by threshold based assembly is far below 1 ms. However, any timer-based assembly further bounds this delay. Consequently, we do not expect these delays to significantly influence the higher layers.

In our ongoing studies, we extend the measurement scenario towards additional traffic characteristics and load scenarios. Additionally, tests to measure the impact on transport and application layer protocols are planed.

Acknowledgement

The work reported in this paper has been partly funded by the European IP NOBEL Phase 2 FP6-IST-027305 and a bilateral cooperation with Alcatel-Lucent.

References

1. Allman, M., Paxson, V., Stevens, W.: TCP Congestion Control. RFC 2581, IETF (April 1999)
2. Bell, E., Smith, A., Langille, P., Rijhsinghani, A., McCloghrie, K.: Definitions of-Managed Objects for Bridges with Traffic Classes, Multicast Filtering and Virtual LAN Extensions. RFC 2674, IETF (August 1999)
3. Blake, S., Black, D., Carlson, M., Davies, E., Wang, Z., W.Weiss.: An Architecture for Differentiated Service. RFC 2475, IETF (December 1998)
4. Case, J.D., Fedor, M., Schoffstall, M.L., Davin, J.: Simple Network Management-Protocol (SNMP). RFC 1098, IETF (April 1989)
5. Farrel, A., Vasseur, J.-P., Ash, J.: A Path Computation Element (PCE)-Based Architecture. RFC 4655, IETF (August 2006)
6. IEEE Computer Society. 802.1D: IEEE Standard for Local and Metropolitan Area Networks–Media Access Control (MAC) Bridges (2004)
7. IEEE Computer Society. 802.1Q: IEEE Standard for Local and Metropolitan Area Networks–Virtual Bridged Local Area Networks (2005)
8. IEEE Computer Society. 802.3: IEEE Standard for Local and Metropolitan Area Networks–Carrier sense multiple access with collision detection (CSMA/CD) access method and physical layer specifications (2005)
9. IEEE Computer Society. 802.1ad: IEEE Standard for Local and Metropolitan Area Networks–Virtual Bridged Local Area Networks, Amendment 4: Provider Bridges (May 2006)
10. IEEE Computer Society. 802.1ah: Draft Standard for Local and Metropolitan Area Networks–Virtual Bridged Local Area Networks, Amendment 6: Provider Backbone Bridges (November 2007)
11. IEEE Computer Society. P802.3ba: IEEE Standard for Local and Metropolitan Area Networks–Carrier Sense Multiple Access with Collision Detection (CSMA/CD) Access Method and Physical Layer Specifications - Amendment: Media Access Control Parameters, Physical Layers and Management Parameters for40 Gb/s and 100 Gb/s Operation (2008)
12. ITU-T. Framework of Optical Transport Network Recommendations. Rec.G.871/Y.1301, ITU-T (October 2000)
13. ITU-T. Functional architecture of connectionless layer networks. Rec. G.809, ITU-T (March 2003)
14. ITU-T. Generic framing procedure (GFP). Rec. G.7041/Y.1303, ITU-T (August 2005)
15. Kornaros, G., Lautenschlaeger, W., Sund, M., Leligou, H.-C.: Architecture and implementation of a frame aggregation unit for optical frame-based switching. In: International Conference on Field Programmable Logic and Applications 2008, vol. 642, pp. 639–642 (September 2008)
16. Kögel, J., Hauger, S., Junghans, S., Köhn, M., Necker, M.C., Stanchina, S.: Designand Evaluation of a Burst Assembly Unit for Optical Burst Switching on a Network Processor. In: EUNICE 2005: Networks and Applications Towards a Ubiquitously Connected World (July 2006)
17. Lautenschläger, W., Frohberg, W.: Bandwidth Dimensioning in Packet-based Aggregation Networks. In: 13th International Telecommunications Network Strategy and Planning Symposium, Networks2008, Budapest (2008)
18. Lehman, T., Sobieski, J., Jabbari, B.: Dragon: a framework for service provisioning in heterogeneous grid networks. IEEE Communications Magazine 44(3), 84–90 (2006)

19. Leligou, H.C., et al.: Hybrid burst/packet switching architectures from IP NOBEL. In: Dingel, B.B., et al. (eds.) Optical Transmission Systems and Equipment for Networking, vol. 6388, p. 63880C. SPIE, San Jose(2006)
20. Malis, A.G., Hsiao, T.: Protocol Extension for Support of Asynchronous Transfer Mode (ATM) Service Class-aware Multiprotocol Label Switching (MPLS) Traffic Engineering. RFC 3496, IETF (March 2003)
21. Masetti, F., et al.: Design and implementation of a multi-terabit optical-burst/packet router prototype. In: Optical Fiber Communication Conference and Exhibit, pp. FD1-1–FD1-3 (March 17-22, 2002)
22. Munoz, R., Pinart, C., Martinez, R., Sorribes, J., Junyent, G., Amrani, A.: The ADRENALINE testbed: integrating GMPLS, XML, and SNMP in transparent DWDM networks. IEEE Communications Magazine 43(8), s40–s48 (2005)
23. Mutter, A., Köhn, M., Sund, M.: A generic 10 Gbps assemblyedge node and testbed for frame switching networks. In: Conference on Testbeds and Research Infrastructures for the Development of Networks and Communities, TridentCom 2009 (2009) (page accepted for publication)
24. Nejabati, R., Klonidis, D., Simeonidou, D., O'Mahony, M.: Demonstration of user-controlled network interface for subwavelength bandwidth-on-demand services. In: Optical Fiber Communication Conference, vol. 3 (March 6-11, 2005)
25. Optical Internetworking Forum (OIF). User network interface (uni) 1.0 signaling specification (oif-uni-01.0) (October 2001)
26. Pedroso, P., et al.: An interoperable GMPLS/OBS control plane: RSVP and OSPF extensions proposal. In: CNSDSP 2008, pp. 418–422 (July 2008)
27. Plummer, D.: Ethernet Address Resolution Protocol: Or Converting Network Protocol Addresses to 48.bit Ethernet Address for Transmission on Ethernet Hardware. RFC 826, IETF (November 1982)
28. Postel, J.: Internet Protocol. RFC 791, IETF (September 1981)
29. Rajaduray, R., Ovadia, S., Blumenthal, D.J.: Analysis of an edge router for span-constrained optical burst switched (obs) networks. Journal of Lightwave Technology 22(11), 26–93 (2004)
30. Generalized Multi-Protocol Label Switching (GMPLS) Architecture. RFC 3945, IETF (October 2004)
31. OSPF Extensions in Support of Generalized Multi-Protocol Label Switching (GMPLS). RFC 4203, IETF (October 2005)
32. Sun, Y., Hashiguchi, T., Minh, V.Q., Wang, X., Morikawa, H., Aoyama, T.: Design and implementation of an optical burst-switched network testbed. IEEE Communications Magazine 43(11), S48–S55 (2005)
33. Swallow, G., Drake, J., Ishimatsu, H., Rekhter, Y.: Generalized Multiprotocol-Label Switching (GMPLS) User-Network Interface (UNI): Resource ReserVation Protocol-Traffic Engineering (RSVP-TE) Support for the Overlay Model. RFC 4208, IETF (October 2005)
34. Yang, X.: Network Aware Resource Broker (NARB) Design and User Manual. Technical Report, University of Southern California (USC), Information Sciences Institute (ISI) (June 2006)

TridentCom 2010

Practices Papers Session 1: Network and Resource Virtualisation for Future Internet Research

Experimental Evaluation of OpenVZ from a Testbed Deployment Perspective*

Gautam Bhanage, Ivan Seskar, Yanyong Zhang,
Dipankar Raychaudhuri, and Shweta Jain

WINLAB, Rutgers University, North Brunswick 08902, USA
{gautamb,seskar,yyzhang,ray,sjain}@winlab.rutgers.edu
http://www.winlab.rutgers.edu

Abstract. A scalable approach to building large scale experimentation testbeds involves multiplexing the system resources for better utilization. Virtualization provides a convenient means of sharing testbed resources among experimenters. The degree of programmability and isolation achieved with such a setup is largely dependent on the type of technology used for virtualization. We consider OpenVZ and User Mode Linux (UML) for virtualization of the ORBIT wireless testbed and evaluate their relative merit. Our results show that OpenVZ, an operating system level virtualization mechanism significantly outperforms UML in terms of system overheads and performance isolation. We discuss both qualitative and quantitative performance features which could serve as guidelines for selection of a virtualization scheme for similar testbeds.

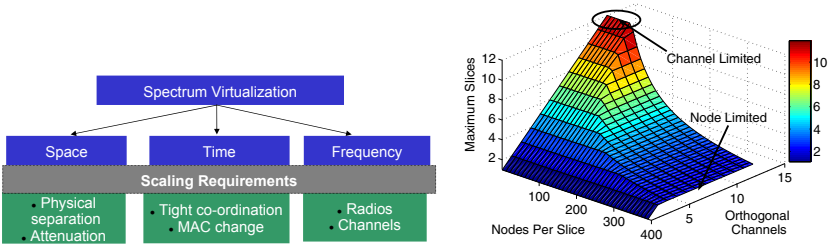
1 Introduction

Experimental validation of research ideas in a realistic environment forms an important step in identifying many practical problems. This is specially true for wireless networks since wireless communication environment is hard to accurately model through simulations. Public access testbeds like ORBIT [12,17,23], provide the research community with platforms to conduct experiments. ORBIT [12], typically uses a time shared experimentation model where each experimenter can reserve the grid nodes for a fixed duration (slot - approximately two hours) and has complete control of these nodes during the reservation period. An ever increasing demand for grid slots can only be met through sharing of the testbed whenever possible. Since spatial expansion is not an economically viable solution due to the limited space, prohibitive cost of setup and maintenance, we propose virtualization of ORBIT to support simultaneous experiments. Wired testbeds like VINI [9] and Planet lab already use node and network virtualization for the same reason. In our study, we will cater specifically to requirements for sharing a *wireless* testbed through virtualization.

Another important motivation for ORBIT testbed virtualization is the integration with the GENI [1] framework. This requires ORBIT to be virtualized for allowing integration with other shared testbeds such as PlanetLab and

* Research supported in part by the GENI initiative and NSF Grant#$CNS-072505$.

T. Magedanz et al. (Eds.): TridentCom 2010, LNICST 46, pp. 103–112, 2011.



(a) Spectrum sharing options. (b) FDM virtualization scaling.

Fig. 1. Options for sharing radio resources on ORBIT and potential capacity of the ORBIT grid with 800 interfaces(2/node), 12 channels(802.11a), 2VMs/Node

VINI [19,7]. GENI also requires combining of control and management across wired and wireless networks, providing researchers with a single programming interface and experimental methodology. Since the ORBIT testbed currently supports only a single experimenter mode of operation, virtualization is essential for integration.

While wired network virtualization may be achieved by pre-allocating memory, CPU cycles and network bandwidth, to achieve perfect virtualization of the wireless network, we need to perfectly isolate both the physical devices and the wireless spectrum while providing flexibility for experimentation. This additional requirement makes the problem of wireless virtualization much harder compared to the wired counterpart [21]. Figure 1(a) shows different options for sharing the radio spectrum. The authors in [21] attempt to solve the spectrum sharing problem by separating experiments in time. As observed, time sharing of a single channel can result in a less repeatable performance due to context switching overheads even though it could possibly reduce the wait time for experiments. Due to the availability of a large number of radio interfaces (800 - 2/node), we share the spectrum by allocating orthogonal channels to slices. ORBIT nodes are equipped with two wireless interfaces each and therefore two virtual machines may be run on each node thereby doubling the capacity of the grid. Figure 1(b) shows the potential capacity of the ORBIT grid with such a frequency division (FDM) based virtualization. We observe that the number of simultaneous experiments supported on the grid are limited either by the number of orthogonal channels[1] or the number of nodes allocated per experiment.

In order to provide meaningful experimentation in the virtualized wireless testbed, the choice of the vitualization platform is critical. In this work, we start by identifying the requirements and qualitative issues to consider when selecting a virtualization platform in Section 2. After discussing the relative merits of OpenVZ for our application, we present a comparative experimental evaluation of UML and OpenVZ in Section 4. Related work is discussed in Section 5. Finally, conclusions and future directions are presented in Section 6.

[1] Experiments that use other wireless technologies like zigbee and GNU radio may be run simultaneously provided they use non-interfering frequencies.

2 Background and Platform Selection

Production scale virtualization systems can be broadly classified as full, para and OS virtualization. Full virtualization [8,2](e.g.,VMWare, KVM) refers to a technique that emulates the underlying hardware and uses a software layer called hypervisor that runs directly on top of the host hardware to trap and execute privileged instructions on the fly[2]. Full virtualization is the least intrusive[3] form of system virtualization. In para virtualization [16,6](e.g., Xen, UML) the hypervisor layer exists within the host operating system to intercept and execute privileged instructions. Unlike full virtualization, para virtualization requires changes to the guest operating system. The most intrusive form of virtualization is operating system based [4](e.g.,OpenVZ) where the virtualized systems run as isolated processes in the host operating system. The host OS is modified to provide secure isolation of the guest OS. For the purpose of this study we lay out the main qualitative criteria and select candidates for performance evaluation based on their suitability for the ORBIT testbed.

Qualitative features of a virtualization scheme which are important from a wireless testbed administrator's perspective are as follows:

1. Ease of administration: Clean API to schedule node resources such as CPU, disk and memory on a per slice basis should be possible.
2. Shared or exclusive interface mapping: The setup should allow flexible mapping of virtual interfaces within the slice to physical interfaces or one or more virtual interfaces (on the hardware like virtual access points).
3. Control over network connectivity: Mechanisms should be available to bandwidth limit slices and control interaction between slices.

All types of virtualization schemes allow for such functions. However, in our experience the most flexible and easy approach for controlling the VMs is through operating system level virtualization such as OpenVZ. Such a setup also allows for the reuse and extension of regular system administration tools (such as IPTABLES, DHCP, SSH, LDAP) for controlling VMs.

From the perspective of an ORBIT experimenter we consider the following:

1. Support for standard and custom Linux distributions: Orbit nodes supports a wide variety of Linux distributions and users are free to use their own customized version. The virtualization platform running on ORBIT must support similar flexibility for the experimenter.
2. Root access within container: This feature is useful for an experimenter for providing complete freedom within the container.

[2] Native virtualization is a virtualization approach where the processor has support for virtualization e.g.,IBM System/370 and allows multiple unmodified operating systems to run together. Full virtualization does not include these systems.

[3] Intrusiveness refers to the degree of changes that need to be made to the guest OS to get it working with virtualization.

Table 1. Comparison of schemes from an ORBIT user perspective

Feature/Experiments	Full - Virtualization	Para - Virtualization	OS - Virtualization
Security Experiments	Yes	Yes	Yes
Network Coding	In Kernel	Overlay*	Overlay*
Mobility and Routing	Yes	Yes	Yes
Rate And Power Control	In Driver	Radiotap**	Radiotap**
Wireless Applications	Yes	Yes	Yes
Phy Measurements	Yes	Yes	Yes
MAC Parameter Control	Yes	Yes	Yes***
Transport layer Modification	In Kernel	*Emulation*$^{\nabla}$	*Emulation*$^{\nabla}$

* Transport layer experiments can be implemented as a part of overlays.

** Radiotap headers allow for per frame rate and power control.

*** For Atheros devices MAC parameters (txop, CW, AIFS)are supported per interface.

∇ Use a click like mechanism on top of IP for custom flow or error control

Multiple Linux distributions with root access in VMs are inherently supported in all three forms of virtualization. A more detailed comparison is shown in the Table 1. It is observed that all wireless experiments scenarios can be either directly supported or emulated (using open source radiotap libraries and overlays) with all the virtualization setups. Traffic control elements such as Click [13] can also be run on hosts to allow for bandwidth shaping and interface mapping. Appropriate API can also be exposed from the driver to allow experimenters to have a controlled interaction with the driver. The only experiments not supported in operating-system level virtualization is the option of customizing the host kernel itself to cater to individual VMs. Despite needing emulation to support experiments that would conventionally be done by direct changes in the host kernel, the possibility of obtaining very tight slice isolation [22] make OS-level virtualization a strong candidate for evaluation.

Based on these inferences, the choice of a virtualization mechanism for ORBIT is not limited to any one type. However, full virtualization such as KVM requires specific CPU virtualization extensions (E.g. Intel VT or AMD-V) which are currently not available with our ORBIT boxes, and hence is not considered for evaluation. We consider OpenVZ (OS level) and User Mode Linux (Para - level) virtualization for quantitative comparison with testbed deployment. Since UML based virtualization has been performed in a previous study [20], this study focusses on the performance analysis of OpenVZ. We ruled out Xen in this performance study due to incompatibility with the Via C3 processors used in the ORBIT testbed.

3 Experiment Setup

The Orbit testbed is a two-dimensional grid of 400 small form factor PCs with 1GHz Via C3 CPU, 512 MB RAM, 20 GB hard disk, three ethernet ports

(control, data and chasis management) and two WiFi interfaces[4]. We used Atheros 5212 chipset cards with the MadWiFi(0.9.4) [3] drivers for our experiments.

Figure 2 shows our experiment setup. OpenVZ uses the concept of a container also called virtual private server (VPS), an entity that performs like a stand alone server. It also provides a virtual network device named as *venetX* per VPS that acts as a point to point link between the container and the host system. We configure the *venet* devices from each of the two VPSs (on every node) to map to a corresponding WiFi card on the host. Effective virtualized and non-virtualized links are as shown in the figure. The UML virtualization setup is described in detail in [20] and is quite identical to the OpenVZ setup.

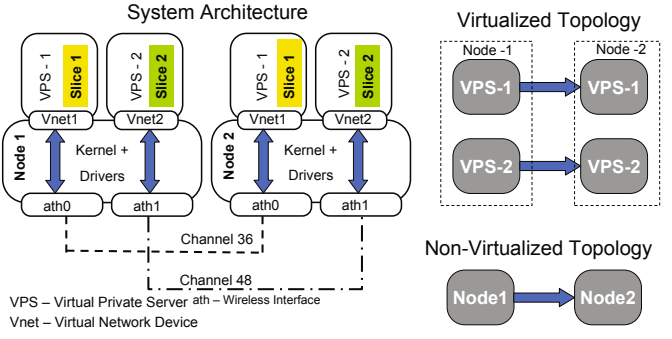


Fig. 2. Experiment setup for OpenVZ evaluation

We run each experiment for 3 minutes using UML and OpenVZ setups as well as with no virtualization. The operating mode of the WiFi cards was 802.11a with bit rate of 36Mbps set at channel 36. The debian linux distribution (*Woody*) was used for both guest and host operating systems.

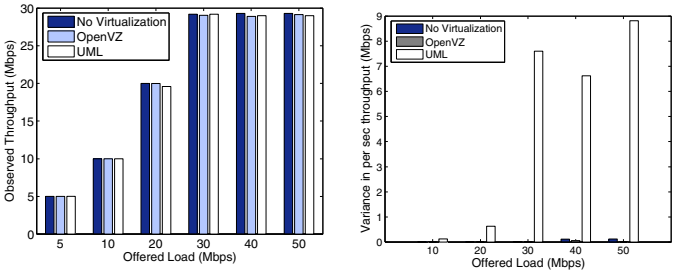
4 Performance Evaluation

We measure overheads in throughput and delay, followed by measurement of *slice isolation* achievable between slices. Quantitative evaluation presented in this section takes into account the importance of different measurement criterion. For instance, the isolation achieved between slices is far more important than sustainable peak throughput as it directly determines experiment repeatability.

4.1 Throughput Measurements

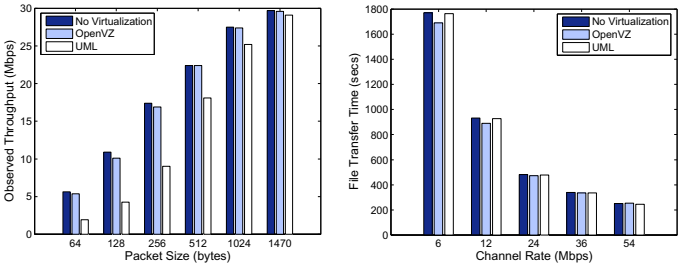
We use the *iperf* [5] tool to generate saturation UDP traffic and average the througphut over 3 min intervals. We plot the observed UDP throughput with

[4] It should be noted that though the results presented in the following section are hardware specific, performance trends will hold and scale with hardware capacity and load on the system.



(a) Average UDP throughput comparison. (b) Variance in UDP bandwidth.

Fig. 3. UDP throughput and variance in throughput as measured with different schemes. Performance is measured as a function of offered load per flow with a fixed packet size of 1024bytes. Variance in UDP bandwidth is measured over per second observed throughput at the receiver.



(a) Difference in UDP throughput with varying packet sizes. (b) FTP performance with 1GB file transfer.

Fig. 4. Measurement of UDP throughput with varying packet sizes and file transfer time with FTP. For the UDP throughput measurement, channel rate is constant at 36Mbps and packet size is varied. For the FTP experiment, packet size is constant at 1024 and channel rate is varied.

varying offered loads and fixed frame size of 1024bytes in Figure 3(a) and its variance in Figure 3(b). Throughput obtained in the virtualized case are averaged over the two links. We observed that both below and above channel saturation there is no distinct difference in throughput with or without virtualization. This trend indicates that both virtualization platforms perform efficiently under saturation conditions. However, the variance in throughput with UML increases with offered load specially near and above saturation. Typically, this suggests that the OpenVZ platform benefits from tighter scheduling and lower overheads compared to UML.

To determine the effect of varying packet sizes, we fix the offered load to 40Mbps and transmission rate to 36Mbps, and vary packet sizes from 128bytes - 1470bytes. Figure 4(a) shows that for packet sizes less than and equal to 1024 bytes, UML has a significantly higher packet processing overhead which

leads to a degraded performance. We attribute this degradation in performance with UML to the lack of support for virtualization in the host kernel.

Finally, we measure throughput performance of TCP by setting up a FTP transfer of a 1GB file with varying channel rates. Resulting file transfer times are as shown in Figure 4(b). For all channel rates, performance of UML is on par with OpenVZ and no virtualization due to the use of larger IP frames resulting in less performance overheads.

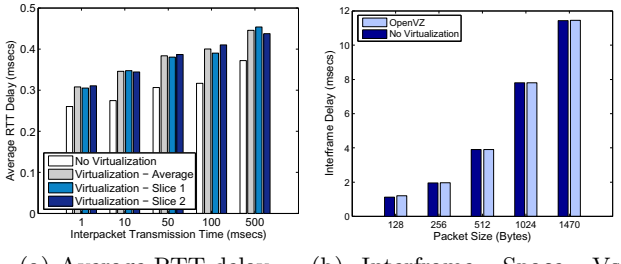
Thus for all three cases, we observe that OpenVZ has satisfactory performance, while UML's throughput performance suffers for small frame sizes.

4.2 Transmission Delay

Delay and jitter are typically important for experiments that measure performance of real time systems or data. We measure delay and jitter performance in terms of distribution of delay across slices and distribution of delay overhead with varying packet sizes.

To measure the distribution of delay across slices we generate ICMP traffic (ping) across both slices and measure the round trip times (RTT). In Figure 5(a) we present the average RTT over an interval of 300 secs for varying packet arrival rates using OpenVZ. We plot delay measurements without virtualization, average delay across both slices, and delay across individual slices. The results show that in all cases, OpenVZ adds a very small average overhead (of the order of $0.05msec$) in terms of absolute delay. The RTT delays for slices increase slightly with smaller sending rates due to slight decrease in CPU time spent on network tasks. Despite the overhead being negligible, we notice that the performance across both slices is always comparable. Efficient buffer copying mechanisms enable OpenVZ to operate with little or no delay overheads, and it is safe for making temporal measurements across slices. A separate study [20] has shown performance degradation in UML under similar experiment settings.

In order to evaluate the processing delay using OpenVZ, we measure the arrival time differences consecutive packets at the receiver with a constant sending



(a) Average RTT delay. (b) Interframe Space Vs Packet Sizes.

Fig. 5. Delay in different experiment scenarios. Minimum and average round trip time measurements are based on ping while interframe space measurements are based on difference in arrival times of packets at the receiver.

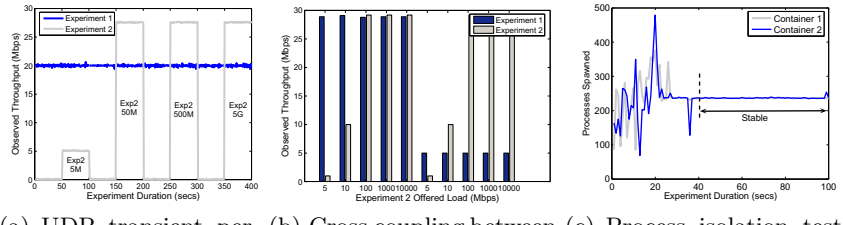
rate. This difference in arrival times is also directly proportional to the delay [10]. We present this result as an average over 10,000 consecutive frames of UDP traffic at 36Mbps in Figure 5(b). We repeat these experiments with various packet sizes. We observe that the delay increases with packet sizes due to increasing transmission times but there is little or no difference between the measurements with and without virtualization. Therefore we conclude that OpenVZ adds little overhead in packet processing and the overhead does not vary with packet size.

4.3 Slice Isolation

Isolation is an important requirement for a virtualized testbed since it directly determines the degree of repeatability achievable in a virtualized setting. Since OpenVZ has clearly outperformed UML in the previous experiments we will rule out UML for further experiments. To measure isolation we coin two performance measurement metrics: transient response and cross coupling between experiment.

We define transient response as the instantaneous change in throughput of an experiment running on one slice caused due to time varying change in offered load on another slice. To measure the transient response, we maintain the offered load for the experiment running on slice 1 at a constant value of 20Mbps and vary the offered load on slice 2 from 5 Mbps to 5 Gbps in steps. Results are presented in Figure 6(a). We see that there is little or no correlation in the throughput of the experiment running on slice 1 (over time) in response to the change in offered load in slice 2. Therefore we may conclude that OpenVZ provides reasonable isolation between slices.

We define cross coupling as the difference in throughput with virtualization as a percentage of the throughput without virtualization. To measure cross coupling we maintain the offered load of the experiment in slice 1 at constant values of 30Mbps and vary the offered load of the experiment on slice 2 from 5 Mbps to 10Gbps in steps. This experiment is then repeated with slice 1 fixed at 5Mbps. The throughput of each experiment averaged over 180seconds are as shown in Figure 6(b). We see that the results of the experiments in slice 1 are never affected by the change in offered load on slice 2 and therefore we concur that



(a) UDP transient performance. (b) Cross coupling between experiments. (c) Process isolation test with *Fork bombs*.

Fig. 6. Experiments for measuring the cross coupling and interference between experiments. First plot shows a performance with time, while the second plot displays results averaged over 180secs.

there is negligible cross coupling of experiments. It is important to note that these results are achieved without tweaking features of OpenVZ that allow the user to set custom cpu usage per slice.

Finally we present test results that measure process space isolation between the VPSs. As a part of these tests, each of the containers are triggered with fork bombs, and the number of processes spawned in each of the VPSs are as shown in Figure 6(c). We observe that the system quickly settles to an equilibrium where each of the containers share equal number of processes. Thus we observe that OpenVZ allows for successful containment of processes within each VM.

5 Related Work

There are several prior works that provide comparative analysis of virtualization platforms [18,15,11]. However, most of this work is in the context of server/machine virtualization. Authors in [18] study the scalability of four virtual platforms: Vserver [22], UML [6], Xen [16] and VMWare [8]. They perform a quantitative evaluation by measuring virtualization overhead, and isolation between VMs. They also measure startup time and memory occupancy of each virtualization platform. A similar study [11] has presented a comparative analysis of Xen, OpenVZ and VMWare Server using industry standard benchmarks for evaluating filesystem, network, multiprocessing and parallel processing performances. While these performance measures are important in our context as well, we concentrate more on the networking aspect of virtualization and platform suitability from a wireless testbed perspective.

The study in [20] discusses virtualization performance using UML by running two instances on a single Orbit node and isolating slices based on orthogonal channels. In our work, we extend this study by comparing the performance of OpenVZ based virtualization with the UML based scheme. Other previous wireless testbed [14] studies have more focus on the system architecture rather than features exported by the technology itself.

6 Conclusion and Future Work

This study presents a comparison of qualitative features and performance which are useful from the perspective of a virtualized wireless testbed deployment. Our qualitative comparison shows that all forms of system virtualization could be used for virtualization of a wireless testbed. Measurements presented in the paper show that OpenVZ consistently outperforms UML in terms of system overheads, slice isolation and its performance is closest to that of the native non-virtualized system. This performance can be attributed to a tight virtualization mechanism and efficient approach to packet handling. Having selected Open VZ as the platform for Orbit virtualization, integration with the orbit framework and measurement library are the most important next steps. From a measurement standpoint comparison with Xen and Vservers on newer Intel chipset based machines are important future research items.

References

1. GENI design principle, http://www.geni.net/
2. Kernel virtual machines, http://www.linux-kvm.org/page/Main_Page
3. Madwifi driver, http://www.madwifi.org/
4. OpenVZ instruction manual, http://wiki.openvz.org/
5. Tcp/udp traffic generation tool, http://dast.nlanr.net/Projects/Iperf/
6. A user-mode port of the kernel, http://user-mode-linux.sourceforge.net/
7. VINI, a virtual network infrastructure, http://www.vini-veritas.net/
8. VMWare player, http://www.vmware.com/products/player/
9. Bavier, A., Feamster, N., Huang, M., Peterson, L., Rexford, J.: vini veritas: realistic and controlled network experimentation. In: Proceedings of SIGCOMM, pp. 3–14. ACM, New York (2006)
10. Bhanage, G., Mahindra, R., Seskar, I., Raychaudhuri, D.: Implication of MAC frame aggregation on empirical wireless experimentation. In: IEEE Globecom 2009 Wireless Networking Symposium, Honolulu, Hawaii, USA (November 2009)
11. V., Chaudhary, M.C., Walters, J.P., Guercio, S., Gallo, S.: A comparison of virtualization technologies for hpc. In: Proceedings of AINA (March 2008)
12. Raychaudhuri, D., Seskar, I., Ott, M., Ganu, S., Ramachandran, K., Kremo, H., Siracusa, R., Liu, H., Singh, M.: Overview of the ORBIT radio grid testbed for evaluation of next-generation wireless network protocols. In: WCNC (March 2005)
13. Kohler, E., Morris, R., Chen, B., Jannotti, J., Kaashoek, M.F.: The click modular router. ACM Trans. Comput. Syst. 18(3) (2000)
14. M. Hibler, R. Ricci, L. Stoller, J. Duerig, S. Guruprasad, T. Stacky, K. Webby, J. Lepreau. Large-scale Virtualization in the Emulab Network Testbed. In: Proceedings of USENIX (2008)
15. Maier, S., Herrscher, D., Rothermel, K.: On node virtualization for scalable network emulation. In: Proceedings of SPECTS, Philadelphia, PA (July 2005)
16. Barham, P., Dragovic, B., Fraser, K., Hand, S., Harris, T., Ho, A., Neugebauer, R., Pratt, I., Warbeld, A.: Xen and the Art of Virtualization. In. In: Proc. of the 19th ACM Symp. on Operating Systems Principles (SOSP) (October 2003)
17. Peterson, L., Muir, S., Roscoe, T., Klingaman, A.: PlanetLab Architecture: An Overview. Technical Report PDN–06–031, PlanetLab Consortium (May 2006)
18. Quetier, B., Neri, V., Cappello, F.: Selecting a virtualization system for grid/p2p large scale emulation. In: Proceedings of EXPGRID Workshop (June 2006)
19. Paul, S., Seshan, S.: Virtualization and Slicing of Wireless Networks. Technical Report GENI Design Document GENI Wireless Working Group, pp. 6–17 (September 2006)
20. Singhal, S., Hadjichristofi, G., Seskar, I., Raychaudhuri, D.: Evaluation of UML based wireless network virtualization. In: Proceedings of NGI, Poland (March 2008)
21. Gregory, S., Anmol, C., Arunesh, M., Suman, B.: Wireless virtualization on commodity 802.11 hardware. In: Proceedings of Wintech, pp. 75–82. ACM, New York (2007)
22. Soltesz, S., Pötzl, H., Fiuczynski, M.E., Bavier, A., Peterson, L.: Container-based operating system virtualization: a scalable, high-performance alternative to hypervisors. SIGOPS Oper. Syst. Rev. 41(3), 275–287 (2007)
23. White, B., Lepreau, J., Stoller, L., Ricci, R., Guruprasad, S., Newbold, M., Hibler, M., Barb, C., Joglekar, A.: An integrated experimental environment for distributed systems and networks. In: OSDI 2002, Boston (December 2002)

Port-Space Isolation for Multiplexing a Single IP Address through Open vSwitch*

Ping Du[1], Maoke Chen[1], and Akihiro Nakao[2]

[1] National Institute of Information and Communications Technology (NICT), Japan
[2] The University of Tokyo, Japan

Abstract. Large-scale network testbeds raise the problem of the exhaustion of IPv4 address space. Before the IPv6 is widely deployed, multiplexing IPv4 address for guest slivers is necessary. NAT is one of the typical ways for the multiplexing. Violating the end-to-end feature of the Internet, the NAT approach has well-known drawbacks in performance scalability and in supporting diverse services and applications. In this paper, we propose a method to share the host's global IP address for all the guest slivers on a node and isolate their network usage in port-space. The idea is successfully implemented with Open vSwitch and deployed in the CoreLab platform. Benchmark result shows that the proposed solution is superior to NAT technique significantly.

Keywords: Network virtualization, resource isolation, Open vSwitch, testbed infrastructure.

1 Introduction

Large-scale network testbeds, such as PlanetLab [1], OneLab [2], EmuLab [3], CoreLab [4,5], etc., are widely deployed over the world, enabling researchers to run their experiments simultaneously without affecting each other's. Different requirements for resource isolation lead to different designs for testbed infrastructures. PlanetLab employs a resource container called Linux-VServer [6] for offering an isolated execution environment, while EmuLab loads arbitrary operating systems on bare hardware on demand. Our CoreLab applies both kernel-based virtual machine (KVM) [7] as a hosted virtual machine monitor (VMM) and OpenVZ [8] as a resource container.

No matter what kind of virtualization is applied for the resource isolation, testbeds are bound to face a common problem: it is necessary to isolate each sliver [1] from the others. In today's Internet, IP address is playing the role of identifying a logical peer over the Internet, and therefore a testbed node needs

* This work has been partly supported by Ministry of Internal Affairs and Communications (MIC) of the Japanese Government.

[1] Throughout this paper, we use the PlanetLab jargon such as *slice* and *sliver*. A slice is a collection of slivers distributed across the Internet, and a sliver is a unit of resources for enabling an execution environment.

T. Magedanz et al. (Eds.): TridentCom 2010, LNICST 46, pp. 113–122, 2011.

a large number of IP addresses to accommodate hundreds of slivers. IPv6 is an apparent solution; however, the current distribution of IPv6 is still sparse so that using IPv6 is still a nuisance. Therefore, multiplexing a global IPv4 address becomes a necessity in the deployment of a testbed.

Assigning private addresses to guest slivers and applying network address translation (NAT) is one of the typical solution for the IPv4 address multiplexing. It is well-known that the NAT violates the end-to-end context, although the commercial use of NAT is already prevalent. For network research testbeds, however, the violation of end-to-end brings obvious limitations. First, when a new application layer protocol designed by the researchers carries an IP address, the NAT fails to translate the address in application messages unless a new patch for NAT is applied, making it an application-layer gateway. Second, when both peers are located behind NAT boxes and using non-well-known port numbers for their services, they cannot communicate with each other unless they inform the NAT boxes to set up port mapping for them. Third, when there are problems during experiments, the additional indirection and address translations increases the time and coordination needed to debug and solve those problems. Fourth, keeping states of address-port mapping also makes NAT solution not scalable with heavy overhead.

In this paper, we propose an alternative way of IPv4 address multiplexing among slivers without NAT, attempting to directly applying the host's address for all the guest slivers and isolating their network activities through static separation of the port-space. We explore both the workaround with the `ebtables` [9] and the full-fledged solution with programming the Open vSwitch (or, briefly, OVS) [10]. We verify our solution in the practice of a real platform — the CoreLab, and compare the performance scalability with the previous NAT solution. The idea of multiplexing global IP address without translation is also used in PlanetLab but there the port-space is not statically isolated, which is not suitable for the case other than containers that share the system calls of the host.

The rest of the paper is organized as follows. Section 2 summarizes the work of NAT-based multiplexing and PlanetLab's dynamic sharing. Section 3 discusses our design challenges and proposes the workaround based on the `ebtables` tool and the OVS-based solution. Section 4 presents the experimental results of our proposal. Finally, Section 5 concludes our work.

2 Related Work

2.1 PlanetLab

PlanetLab applies Linux VServer [6], where each guest sharing the host kernel and all network resources such as CPU, memory are maintained in the same way as in ordinary OSes. Currently, all slivers on a PlanetLab node use the same IP address [11]. Each time, when a sliver is launching a session, the host selects an unused TCP/UDP port as its source port. Since all guests share the same kernel, it is easy for kernel to detect and avoid port conflict.

When there is port contention that more than two slivers want to bind the same ports simultaneously, a solution could be scheduling or using a resource allocator like SHARP [12] to arbitrate the port usage.

The shortcoming for above solutions is that current PlanetLab can only support short-term usage of a port. This is not sufficient if one would like to hold a port for a relatively long period.

Applying Linux VServer for PlanetLab brings another limitation in flexibility. Sharing the system calls of the host, slivers cannot run with a customized kernel, while adding new kernel modules often happens in the research for networking.

2.2 CoreLab

CoreLab applies KVM to support the slivers, getting rid of the limitations in flexibility. In the previous practice of CoreLab [5], each guest sliver is assigned a private IPv4 address and a specific range of port numbers for either TCP or UDP communications. Meanwhile the host OS is configured as a NAT gateway, translating the public address to proper guest's private address according to the destination port in any packet. The structure of deployment is depicted in Fig. 1. Therefore, the configuration includes a series of `iptables` DNAT (destination NAT) rules. For example, on a host with IP address 133.69.37.10, a guest OS has the private IP address 10.0.6.2 and runs an `sshd` service on the port 22, while port range 10000 ˜ 10100 is assigned to this guest. The host maps the address-port pair 10.0.6.2:22 to 133.69.37.10:10022, and a remote peer can access the guest's `sshd` service through the global address 133.69.37.10 and the port number 10022.

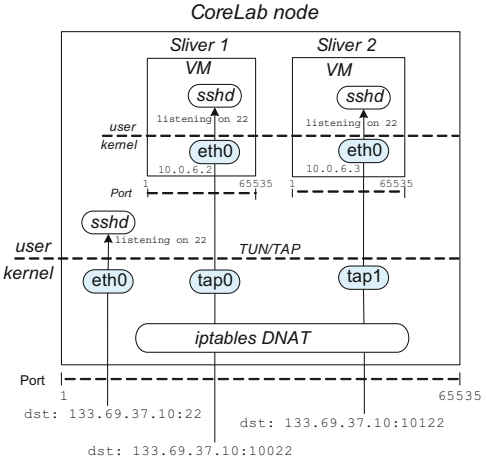


Fig. 1. Sharing IP address through iptables with NAT on CoreLab node

3 System Design

As we have mentioned before, sharing IP address through NAT has two shortcomings: (1) NAT breaks the end-to-end connectivity so that many applications

are unavailable; (2) NAT is a bottleneck of network performance since each session requires a piece of state recorded in the translator.

Current Internet distinguishes end systems (either physical or virtual) with IP address. One observation is that, though the 32-bit address defined by IPv4 is insufficient, the 16-bit port-space is inefficiently utilized. Based on this observation, we propose a solution that distinguishes slivers through port range instead of IP address, where different guest slivers can have the same address while using different ranges of ports.

3.1 Problem and Challenges

Because we don't incline to use the NAT mode to manage the connectivity between co-located virtual machines and the physical interface, the host should be configured in the bridge mode. Sharing the same IP address among the host itself and all its guests is equivalent to a case of IP anycast in the same connected network. If we had nothing done, once a packet that contains the anycast IP address as the destination comes, who (guest or host) would respond to the packet depends upon who would earliest receive the packet and process it.

We attempt to deliver a packet to a proper guest according to the port range that the packet's destination port falls in. This problem is equivalent to optimizing the anycast response according to port number. Today a typical IP anycast solution is announcing the anycast address via routing protocols and who's selected depends on the distance in routing. Such a solution is obviously not suitable for our case. We need an anycast optimization solution for a group of computers bridged together. Because the criteria for the optimization involve the transportation layer — the port number, our goal is similar to design a Layer-4 switch with anycast optimization. Xen [13] and Vmware [14] provides standard Layer-2 switching or Layer-3 routing functionality for a virtual environment. However, they are also lack of the Layer-4 switching functionality. This is a generic, unsolved problem.

In details, we face the following challenges.

1. For the last hop towards one in the anycast group, the link-layer address learned by the link layer peer decides the real destination. Unfortunately, however, ARP is only a protocol mapping Layer-3 address to Layer-2 but not a mapping considering port number. It is also a "slow" protocol whose cache entries have quite long lifetime, not able to be frequently updated — once a peer have learned the MAC address of one of the guests (or the MAC address of the host) for the shared IP address, it has to take some time to update to the MAC address of another.
2. There are some IP-supported protocols, like ICMP, that do not have "port" number in the payload of the IP datagram. Semantically, it is reasonable that one in the anycast group can receive all the non-port messages that are actively sent from outside and the non-port responses that are replied to its own requests.
3. Most TCP and UDP applications are designed so that a client randomly chooses source port number for a session that is not locally conflicting with

other existing sessions. Such a port number may not fall into the range of ports assigned to the system.
4. Some applications use a separated channel for data transmission rather than signaling. The port number for the data channel is often negotiated at the application layer and it is hard to limit an application program selecting only assigned ports for the negotiation without changing the program binary.

In order to resolve these challenges, we design two solutions, a bridge-based solution and a dynamically programmable switch. Although they are tested on the KVM-based CoreLab, we believe they are also suitable for any virtualization approach with independent kernel for guests, such as Xen and VMware running in a bridge mode.

3.2 *Ebtables* and Workaround

A straightforward idea is to redirect packets towards a guest according to the port number. The redirection is similar to the behavior of *iptables* but it should be done in the Layer-2. Therefore, the Ethernet filter *ebtables* [9] is considered.

Having the same IP address, each guest (as well as the host itself) can be distinguished from others through the MAC address. Therefore it is needed to establish a mapping from port range to a MAC address. The *ebtables* provides MAC address translation that fits the requirement. People may concern the scalability of the translation. Fortunately, the MAC address translation is fairly simple and each guest system involves only a static set of states, which is much lighter than IP NAT that stores per-session states.

Deploying the *ebtables* for the port-space isolation contains a couple of steps.

First, we hack the incoming frame with our rule in order a packet is able to be received by the proper guest. This is done with the destination MAC address translation of the *ebtables*.

Second, to avoid updating ARP entries for a peer, we disable any guest to respond ARP requests from the peer, defining the host as the only agent for itself and all its guests. For this purpose, we change the outgoing frame's source MAC address, and also let only the host itself reply ARP request.

ARP frame has the source hardware address twice, in both the frame header and also the ARP message body. It is necessary to change not only the guest MAC address in frame header but also that in the message body for any ARP frame.

However, the second action also disables a guest to receive ARP replies from an outside peer. Once an ARP reply from a peer arrives, we cannot recognize if this is requested by any guest or by the host itself. What we have to do is only delivering the reply to all of them by changing the ARP reply's destination address to the Ethernet broadcast address.

The above configurations haven't overcome the ICMP challenge. A workaround is forcing ICMP response to be broadcasted to all the guests, but the privacy is hurt. Furthermore, they haven't overcome the challenge of random port selection or that of the application-layer port negotiation. If application

chooses a port not available, now the system simply rejects the communication. A possible workaround is occupying the unassigned ports with a special daemon, similar to the `inetd`. Thus the application will automatically choose an available port. However, this method brings extra overhead.

3.3 Open vSwitch Solution

Open vSwitch (OVS) [10] can work as a network switch for virtual environment that hosts inside a physical machine and connects the various VMs. It provides a flexible, a 10-tuple flow-table [15] based forwarding engine which can to used to partition the forwarding plane. OpenFlow [16] can also provide a similar flow-table forwarding model. We applied OVS since it is more compatible with Linux-based virtual environment. With OVS, we can define flow entries to forward packets based on its port number.

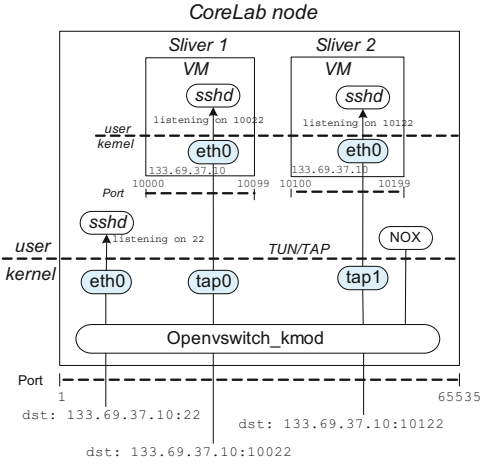


Fig. 2. Port-space isolation with OVS on CoreLab node

We design and deploy OVS solution in the practice of a real platform — the CoreLab. As shown in Fig. 2, all VMs and host are bridged to a datapath (a kind of bridge) of OVS. Besides OVS, we also deploy deployed a NOX [18], which is an open-source controller that speaks OpenFlow protocol on each CoreLab node. The flow entries are installed from NOX to OVS. We address the challenges as follows.

a. Port-space isolation

Since the virtual interface of each VM is written in software, all VMs can be configured with the same IP and MAC addresses as the host so that any ethernet frames from outside can be received by a VM or host without address translation. This can reduce the overhead of NAT and etables that require maintaining state of address translation. For the ARP packets, since all VMs share the same IP and MAC addresses with the host, when an ARP packet is received by an host, it will be flooded to all VMs.

To isolate the packets of different VMs, each VM is assigned with a range of port numbers. The port range of each VM can be got from the database of PLC [17] node. As a result, the corresponding flow entries (forwarding rules) are installed after a VM is launched. Each VM can only listen on the ports that assigned to it. The OVS switches packet based on the destination port number. For example, an ssh request packet with <dst: 133.69.37.10:10022> is delivered to VM 1 while the one with <dst: 133.69.37.10:10122> is delivered to VM 2.

b. ICMP packets

ICMP packets that do not have "port" number in the payload of IP datagram. When a host receives an ICMP request packet, it will respond it without redirecting it to VMs. When a VM sends out an ICMP request packet, it will install a flow entry for the reverse incoming ICMP response packets. For example, when VM 1 sends ICMP packets to remote host B, it will install a flow entry <nw_src=B,icmp_echo_response, actions: VM1>. The flow entry will expire after it has been idle for a specified period (i.e., 10 secs). When VM 2 is sending ICMP packets to a different remote host C, the ICMP response packets could be separated.

One possible problem is that when two VMs are sending ICMP packets to the same host B, the flow entry will be <nw_src=B,icmp_echo_response, actions: VM1|VM2>. In this case, all ICMP packets will be sent to both VM1 and VM2. An ideal solution is to install flow entry based on icmp_id. However, due to that current OpenFlow protocol does not support user-defined flow-space, we can redefine the unused vlan_id field as icmp_id. This is left for our future work.

c. Source port conflict

We can reduce the port conflict probability to zero by changing the VM configuration file (i.e., /proc/sys/net/ipv4/ip_local_port_range for Linux) to let the TCP/UDP can only select source port in a specified range. We also apply a simple patch to the VM kernel to prevent the VM from selecting a source port out of the range.

d. Multi-homing

When a host has multiple physical interfaces (and therefor multiple IP addresses), we create the same number of OVS datapaths. Each datapath is bridged to a physical interface. A guest VM boots with multiple virtual interfaces bridged to different datapaths. As a result, each guest VMs can share one or multiple global IP addresses with host. How to deliver packets through these multiple virtual interfaces is decided by each guest separately, which is out of the scope of this paper.

4 Performance Evaluation

In this section, we compare the performance of our implemented port-space isolation with OVS against the previous practice with NAT on CoreLab.

Figure 3 shows the experimental environment, which is configured with two CoreLab nodes. Each of them is with a 2.67GHz Intel CPU and 4GB memory.

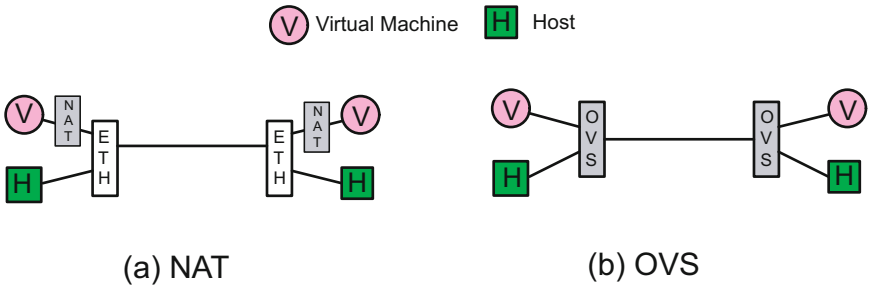


Fig. 3. Experimental environment for evaluating port-space isolation with (a) NAT (b) OVS

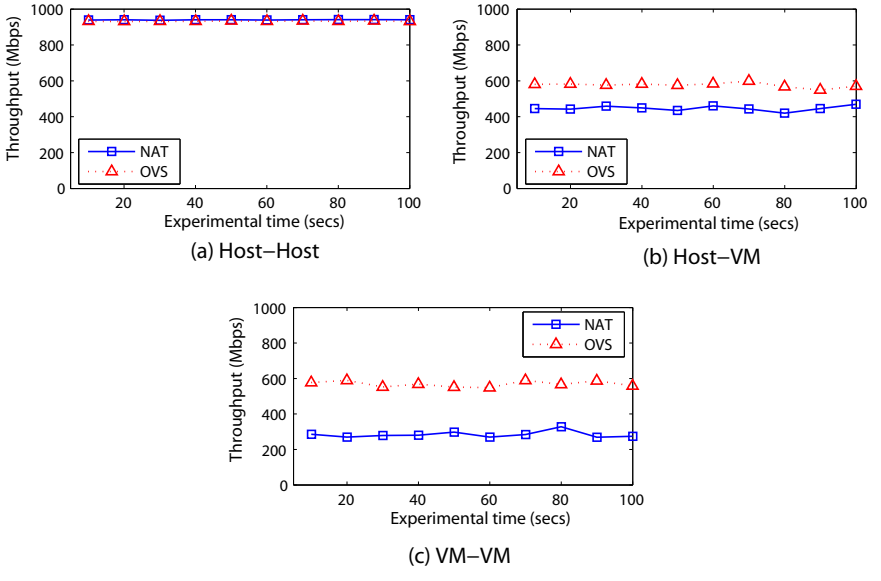


Fig. 4. Comparisons of throughput of (a) Host-Host (b) Host-VM, and (c) VM-VM under NAT and OVS solutions

They connect with each other over 1Gbps Ethernet link. The host OS is Fedora 8 with kernel 2.6.31. Each VM is with 512M memory and its virtual interface driver is `ne1000`. In the NAT case (Fig. 3(a)), the VM connects outside through NAT while the host connects outside directly. In OVS case (Fig. 3(b)), Both VM and host connect outside through OVS implementation in the Linux kernel.

We run `iperf` server and client on VM or host of different nodes and measure the effective TCP throughput between them. Figure 4(a) compares the throughput between two hosts. In the case of NAT, since the hosts are connected directly so that the throughput are that of two nodes of native Linux. In the case of OVS, the hosts are connected through OVS implementation in the Linux kernel. Both of them can achieve around 940Mbps throughput. The results show that OVS has very small performance cost comparable to that of native OS.

Figure 4(b) compares the throughput between a host and a VM. In the case of NAT, a packet of a VM will go through both NAT and KVM. As a comparison, in the case of OVS, a packet of a VM will go through NAT and OVS. Under the same cost of KVM, the throughput is around 650Mbps in the case of OVS while the throughput is only 400Mbps in the case of NAT. The results show that OVS has much smaller performance cost than NAT.

Figure 4(c) compares the throughput between two VMs at different Corelab nodes. In the case of NAT, the throughput (250Mbps) between two VMs is much smaller than the throughput (450Mbps) of Fig. 4(b) since the packets should go through NAT gateway one more time. As a comparison, in the case of OVS, the throughput (600Mbps) between two VMs is almost the same as the throughput between a host and a VM.

In the case of OVS, although the performance of cost of OVS is very small, the throughput (600Mbps) between two VMs is much lower than that (940Mbps) between two hosts. This is due to the performance cost of KVM, which requires improvements.

5 Conclusions

This study has identified the IP scarcity problem in a testbed environment. Since conventional NAT solution has many limitations. We have sought to solve the problem by port-space isolation. We have designed and implemented such a port-space isolation system through Open vSwitch on CoreLab. The experiment results show that it has better performance than conventional NAT technique.

In fact, the IP scarcity problem exists not only in a testbed environment, but also in most stub network such in an office LAN. The techniques of private IP address or IPv6 can make the Internet become heterogeneous. To make the Internet remain flat, this paper proposes a new angle for identifying and routing packets by ports instead of IP addresses in a local environment.

As a final note, we posit that this work can be immediately extended to generic network-namespace isolation for slices. In other words, while this paper shows how to isolate a port range for a slice, the other network-namespace such as IP addresses, protocols, and the other fields as well as a tuple of them can be allocated to a slice. OpenFlow [16] and Open vSwitch [10] allow us to partition network by flows and to define simple actions per flow at the central controller [18]. In CoreLab, we will combine network partitioning—not only in terms of only port range, but also of a generic flow space—and computational slicing together to enabling complex processing in a distributed fashion. Such extension is our immediate future work.

References

1. Planetlab, `http://www.planet-lab.org/`
2. Onelab, `http://www.onelab.eu/`

3. White, B., Lepreau, J., Stoller, L., Ricci, R., Guruprasad, S., Newbold, M., Hibler, M., Barb, C., Joglekar, A.: An integrated experimental environment for distributed systems and networks. In: OSDI (2002)
4. Corelab, http://www.corelab.jp
5. Nakao, A., Ozaki, R., Nishida, Y.: Corelab: An emerging network testbed employing hosted virtual machine monitor. In: ROADS (2008)
6. Linux-vserver project, http://linux-vserver.org/
7. Kvm, http://kvm.qumranet.com/kvmwiki
8. Openvz, http://wiki.openvz.org/
9. Ebtables, http://ebtables.sourceforge.net/
10. Pfaff, B., Pettit, J., Amidon, K., Casado, M., Koponen, T., Shenker, S.: Extending networking into the virtualization layer. In: ACM HotNets (2009)
11. Port use and contention in planetlab, https://www.planet-lab.org/files/pdn/PDN-03-016/pdn-03-016.pdf
12. Fu, Y., Chase, J., Chun, B., Schwab, S., Vahdat, A.: Sharp: An architecture for secure resource peering. In: ACM SOSP (2003)
13. Dragovic, P.B.B., Fraser, K., Hand, S., Harris, T., Ho, A., Neugebauery, R., Pratt, I., Warfield, A.: Xen and the art of virtualization. In: SOSP (2003)
14. Vmware vnetwork distributed switch, http://www.vmware.com/products/vnetwork-distributed-switch/
15. Openflow switch specification, http://www.openflowswitch.org/documents/openflow-spec-v0.8.9.pdf
16. McKeown, N., Anderson, T., Balakrishnan, H., Parulkar, G., Peterson, L., Rexford, J.: Openflow: enabling innovation in campus networks. SIGCOMM Comput. Commun. Rev. 38, 69–74 (2008)
17. The trutees of princeton university. myplc, http://www.planet-lab.org/doc/myplc
18. Gude, N., Koponen, T., Pettit, J., Pfaff, B., Casado, M., McKeown, N., Shenker, S.: Nox: towards an operating system for networks. SIGCOMM Computer Communication Review 38, 105–110 (2008)

FEDERICA: A Virtualization Based Infrastructure for Future and Present Internet Research

Mauro Campanella*

Consortium GARR
Via dei Tizii 6, 00185 Roma, Italy
Mauro.Campanella@garr.it

Abstract. The Europe wide infrastructure managed by the FEDERICA project demonstrates how an infrastructure based on computers and network physical resources, both capable of virtualization, is a flexible, neutral and efficient facility for future and present Internet research. The facility can create virtual resources sets, grouped in virtual infrastructures (slices) according to users' specification. The user has full control on the resources in the assigned slice which can be used for many types of research, from Future Internet clean-slate architectures to security and distributed protocols and applications. The infrastructure has European size and it is capable of federating with other facilities worldwide.

Keywords: Virtualization, NRENs, Future Internet, polymorphic test infrastructure, cloud infrastructures.

1 Introduction

FEDERICA [1] is a European project started in January 2008. It has been engineered to provide support to the research on current and Future Internet technologies and architectures. The project is linked to the European FIRE initiative [4] and the European Future Internet Assembly [5]. Other similar initiatives exists worldwide, e.g. GENI [6] in the United Stated and AKARI [7] in Japan.

Research and experimentation on novel technologies and architectures require new experimental environments that combine flexibility, neutrality, a minimum set of constraints for the researchers, reproducibility and allow full control of the testing environment. The project has distinguishing characters from similar projects, e.g. Onelab [3], in particular, it does not mandate a specific operating system, and the user has full control of the physical networking layers down to the data link layer,

The project is based on the constant developments of National Research and Education Networks (NRENs) [8] in Europe and worldwide, which created a strong multi-domain hybrid network infrastructures with advanced capabilities.

* On behalf of the FEDERICA consortium.

T. Magedanz et al. (Eds.): TridentCom 2010, LNICST 46, pp. 123–132, 2011.

The FEDERICA project uses the NRENs infrastructures and virtualization technologies to create a research facility available to the public and private sector.

In the following, section 2 describes the FEDERICA project framework and architecture. Section 3 details the operational infrastructure. Section 4 lists status challenges and new possibilities. Section 5 concludes the article.

2 The FEDERICA Project

The project is co-funded under the 7th European Community Framework Program. It started 1st January 2008 and lasts 30 months. The project partners include a wide range of stake-holders on network research, NRENs, DANTE, TERENA, academic and industrial research groups and vendors.

2.1 Project Goals and Objectives

The main goal is to support research in present and Future Internet. To achieve this goal, the project set its objectives to:

- Engineer and implement a Europe-wide Infrastructure to be used as a distributed testing facility
- Research in virtualization of e-Infrastructures integrating network and computing resources
- Facilitate technical discussions amongst specialists, in particular arising from experimental results and disseminating knowledge and NREN experience of meeting users requirements
- Contribute with real test cases and results to standardization bodies, e.g. IETF, ITU-T, OGF, TM Forum/IPsphere.

2.2 Requirements

According to its goals, the infrastructure has to support research on the widest range of new technologies and protocols, in particular in networking. To achieve such goals, the infrastructure must:

- Avoid to impose specific technological or architectural constraints to the researchers, as an example avoid mandating the IP protocol. The facility has to create environments that are technology agnostic and neutral (transparent) to new protocols or technologies.
- Ensure reproducibility of the experiments. Given the same initial conditions, the behaviour of a virtual resource should be the same, as a basic principle to obtain the same experimental results. This requirement is considered of particular importance.
- Provide to the user complete control and configuration capabilities within the assigned resources, allowing disruptive testing.
- Open to interconnect or federate with other e-Infrastructures and Internet.

2.3 Infrastructure Framework

The requirements point to two key framework principles for the infrastructure:

- The utilization of a combination of network and computing physical resources.
- The use of virtualization technologies applied both to the computing and network resources.

Virtualization is defined here as the capability to create a virtual version of a physical resource, both in computing and network environments. The virtual resources (e.g. a virtual network circuit, a disk partition, a virtual computer) are typically created by segmenting a physical resource. Virtualization creates unconfigured (clean) virtual resources, e.g. an image of the hardware of a computing element on which (almost) any operating system can be installed, a point-to-point network circuit, a portion of disk space. Those resources can be then tailored and configured to users needs and even moved from a virtualization-aware platform to another.

The framework for such an infrastructure is based on two distinct layers (see Fig. 1 for a pictorial representation):

1. The virtualization substrate. The physical environment which contains all the hardware and software to instantiate the virtual resources.
2. The virtual infrastructures layer, containing all the virtual sets of resources (or slices).

The virtualization substrate is managed and controlled as a single administrative domain. The virtual infrastructures are in principle an unlimited, or very large

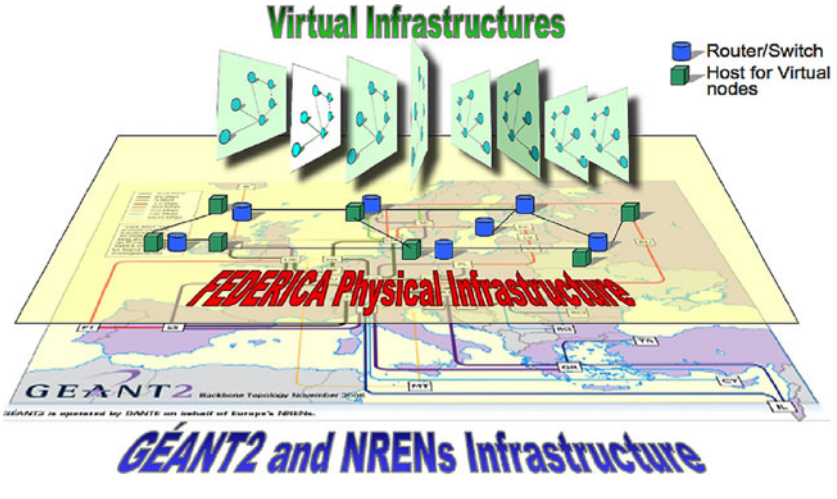


Fig. 1. Pictorial view of the FEDERICA infrastructure

number, restricted by the physical resources available and the requested characteristics. In case of federated facilities, the slices can comprise set of resources provided by different infrastructures.

The architecture defines only two basic resource entities:

1. Data connectivity. In form of a point to point circuit with or without assured capacity guarantees and with or without a data link protocol (a bit pipe).
2. A computing element, offering the equivalent of a computer hardware containing at least RAM, CPU and one network interface and mass storage. The computing element is capable of hosting various operating systems and also perform functionalities (e.g. routing). The computing element is defined as the basic entity. The RAM, NICs, storage are considered characteristics of the entity. This differs from other proposal, in particular for cloud services, where storage is considered a basic entity.

The figure represents the slice in vertical format for sake of clarity and to show that there is no dependency or hierarchy between them. Each slice may contain a virtual resource coming from any part of the substrate.

3 The Infrastructure Implementation

Following the framework outlined above, the FEDERICA infrastructure is implemented in two layers. The substrate an its made of network and computing physical resources. Each physical resource can produce virtual resources, slicing itself. Resource grouped by a topology in sets, or slices, are handled to the the user and compose the other layer.

As a design principle, the infrastructure favours testing of functionalities, protocols and new ideas, rather than providing a laboratory suited to very high performance studies.

The network resource in the wide area are provided by the participating NRENs through the GÉANT [9] infrastructure. Each NREN hosts a point of presence (PoP). A mesh of one Gigabit Ethernet circuits connects the PoPs. The circuits are initially at one Gbps, the capacity has been chosen as a compromise between cost on the wide area network and total capacity. It has been adequate for now to users' requirements. This capacity can be sliced, still creating high-speed links, and although expensive is contributed by the participating NRENs. Most of the circuits are created over the GÉANT SDH equipment using generic framing procedure and virtual concatenation.

Each PoP hosts network and computing elements. The network equipment in the four core PoPs is a programmable high-end router/switch from Juniper Networks. The model is a MX480 with dual CPU and 1 line card with 32 ports at 1Gb Ethernet (8 optical and 24 copper). The MX functionalities include virtual and logical routing, MPLS, VLANs, IPv4, IPv6. Two MX480 are equipped with Ethernet linecards with hardware QoS capabilities to enforce precise QoS at the packet level when requested. Smaller multi-protocol switche/routers (Juniper EX series) are installed in non-core PoPs. The computing equipment (V-Node)

is based on off the shelf PC hardware, running virtualization software. Each PC contains 2 x Quad core AMD CPU running at 2 GHz, 32GB RAM, 8 network interfaces, 2x500GB disks. All the interfaces of the V-Nodes are connected to the Juniper routers.

The initial choice of the virtualization software for the V-nodes is the free version of the VMware [10] client software (ESXi) . This choice has been done after a review of other virtualization software (e.g. XEN). In particular it has been evaluated the completeness and structure of the Application Programming Interface, the availability of usage examples, available expertise, free tools to import and convert node images. The capabilities and performance of the free version have been adequate for the current requirements. The major drawback is the more difficult management of the V-Nodes without the commercial software. Such complexity is considered adequate for the initial phase of the project, but will be reviewed as a function of the number of slices and requests for a possible upgrade to the commercial versions of the software.

These building blocks of the substrate pose very few constraints to the user. The virtualization software in the V-nodes can host a large variety of operating systems and tools . It is possible to create an image from a fully configured system, avoiding the need of configuration in the slice. In the current status of the infrastructure the most significant one is that the data link layer is fixed to Ethernet framing. Future development of FEDERICA, according to users' requirements, will implement access to optical equipment to overcome this limitation.

3.1 Topology

The topology is composed of 13 distributed physical sites. Amongst these PoPs, a full mesh of four is equipped with Juniper MX router/switches and it is considered the core. The 9 non-core nodes are equipped by EX switches. The core nodes are equipped by 2 V-Nodes, the non-core PoPs host one node each. The FEDERICA physical topology is depicted in Fig. 2. The design placed particular importance on the resiliency and load balancing of the network, based on GÉANT infrastructure, and resources availability at partners locations. To minimize the load on the physical network resources and hence the interference between virtual resources, the network topology has a high level of meshing.

The FEDERICA substrate is a single administrative domain that contains all the physical resources (point to point circuits, nodes) in all PoPs. The domain does not contain the optical equipment of GÉANT used to transport the circuits between PoPs. It is configured as an IPv4 and IPv6 Autonomous System with both public and private addresses. The infrastructure is permanently connected to Internet using the Border Gateway Protocol and receives full routing tables in the four core PoPs.

The infrastructure is centrally managed and monitored by a Network Operation Centre. The NOC has also the task to create the slices. The monitoring system uses mainly the simple network management protocol to retrieve information for physical and virtual resources. In a PoP a packet flow based monitoring is available

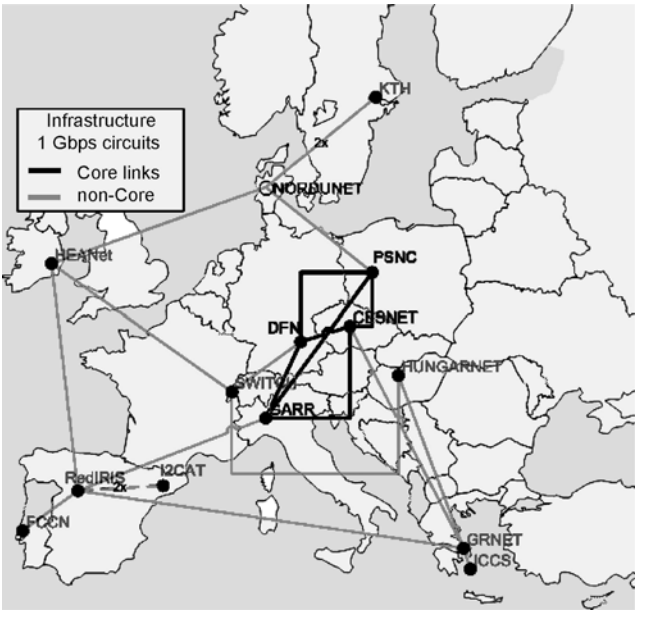


Fig. 2. FEDERICA network topology (November 2009)

3.2 Reproducibiity

The reproducibility and the stability of the behaviour of the virtual resources is a fundamental requirement for quantitative evaluations of new ideas. As an example, a virtual circuit may not be capable of offering a constant, fixed amount of bit per second, and a virtual computer image may not provide a constant CPU usage.

The project has focused the engineering of the infrastructure to provide reproducibility of a single virtual resource. The difference is due to two main independent causes:

- The behaviour of the physical resource supporting the virtual resources, e.g. due to its workload
- The virtualization technology itself, usually a layer placed between the physical resources and the virtual ones

Computing elements in FEDERICA have been chosen to provide specific functionalities in hardware, like virtualization-aware CPUs. The added hardware capabilities ensure a smoother sharing of the CPU by concurrent executing images. As additional measure, a virtual node is usually assigned to a idle core. As network interface cards do not support virtualization of flows in hardware, the V-node has been equipped with the maximum number of interfaces. A running image received an idle interface if available. If a node requires a high level of reproducibility, to such node can be fully dedicated a single core, a fixed amount

of physical memory, a fraction of disk and a network interface. Not all the nodes requires such guarantees. In addition to the packet based QoS technologies of the switch/routers, some circuits are connected to packet Quality of Service capable line cards in the Juniper MX. In the rest of the infrastructure, the resources have been adequately increased, to avoid overbooking and minimize contention. It is possible then to create a slice with a set of resources, which exhibits, singularly, a known behaviour in all conditions.

Assuring the reproducibility of a set of connected resources is out of the scope of the infrastructure and it is instead a user responsibility. The complexity increases rapidly with the number of resources involved and it is strongly dipendent on the technologies and architecture. The classic problem of guaranteeing an end-to-end quality of service of an IP flow exemplifies the issue. In case of virtual infrastructures, as in the case of Internet traffic, often the requirements do not mandate strict guarantees, but rather a best effort behaviour. Virtual resource performance measurements are ongoing in FEDERICA.

3.3 Access Policy

The infrastructure is available to public and private researchers on Future Internet technologies. The access to the infrastructure is controlled by a User Policy Board (UPB). The UPB receives all requests for access, analyzes their technical feasibility (not the scientific content) and the current availability of resources and then prioritize them. The motivation for mandating a controlled access is mainly related to security (identification of the user, agreement on an acceptable use policy) and reproducibility (quality of service) guarantees.

3.4 Resource Virtualization and Slice Creation

The process to create a virtual computing system is rather straightforward and can also accept an image provided by the user or on available template of various operating systems. The virtualization capabilities in the network are also evolving, as described in [2]. The article reviews the current research in a Network Virtualization Environment (NVE) and the many challenges associated. The initial choice in FEDERICA is to use Virtual LANs and use QoS techniques for circuit virtualization; multi protocol label switching (MPLS) may be applied when needed.

The slice creation procedure definition is developing to incorporate the users' feedback. The current implementation of the infrastructure is based on manual or semi-automated provisioning of the virtual resources. The manual process is a choice in the step that maps virtual to physical resource. Even if the infrastructure contains a small amount of resources, this step is fundamental to ensure that the performance requirements of the virtual resources is respected and the infrastructure is efficiently used.

The current slice creation process consists of the following steps. First, the researcher that wants to perform an experiment over the FEDERICA infrastructure is required to provide the NOC with the desired topology, including requirements for the nodes and the network (each V-node RAM size, CPU power, mass

storage space, topology and bandwidth between the V-Nodes, routing or switching functionalities, protocols). The request may be for un-configured resources, that the user will configure directly, even substituting protocols, or resources with an initial configuration, e.g. IP routing.

Once the NOC receives the slice description and resource requirements, the NOC maps the logical topology requested on the physical topology of the substrate and chooses the sites (PoPs) from which physical resources will be allocated. Besides instantiating all the resources requested by the user, the NOC needs to instantiate an extra virtual machine, that act as a gateway between Internet and the slice: the Slice Management Server. Access control of the Slice Management Server is performed by means of identity credentials managed by a RADIUS server.

The next step for the NOC is to instantiate Ethernet VLANs to connect the slice resources and create the topology required by the researcher. Finally, the NOC needs to setup the Slice Management network for the user that will connect the Slice Management Server to the management interface of each one of the managed resources in the slice (V-Nodes, logical routers, software routers). The connection is performed creating virtual interfaces in all resources and one in the Management Server in the same IP subnet (usually private) and creating an additional VLAN linking them. This subnet is initially the only IP path for the user to connect to the slice resources when accessing from Internet the Management server.

When the NOC has created the slice, it communicates to the researchers the information to access it: the public IP address of the Virtual Slice Management Server, the security credentials, the credentials to access in case of the Juniper logical routers and/or the software routers, and finally the IP addressing scheme of the Virtual Slice Management Network.

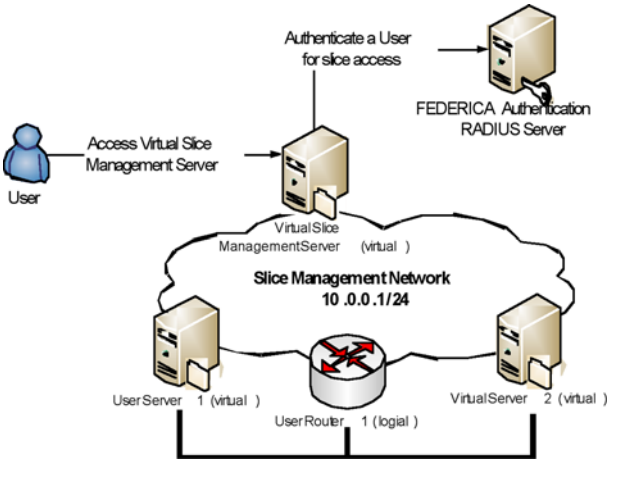


Fig. 3. Slice access example)

In the example in Fig. 3 the user has requested a slice containing two virtual servers connected through a router (created on a Juniper). The NOC created the three resources, connected them through a VLAN (black line at the bottom of the Figure), instantiated the Virtual Slice Management Server and created the Slice Management Network. The slice management network (cloud at the centre of the Figure) is needed to access the resources independently of user's slice topology and the virtual resource configuration. The researcher connects to the Virtual Slice Management Server using the credentials provided by the NOC, and is authenticated by the FEDERICA Authentication RADIUS Server. VMware virtual machines may also be configured to be accessed through remote Virtual Network Console (VNC) connections. By exploiting this mechanism users would have access to the console of their virtual servers, but they would also be able to interact with graphical user interfaces and to even access the BIOS of the server.

All the steps are performed either manually or using a heterogeneous set of tools (web portal for users, VMware Infrastructures application, the remote console of the devices, VNC clients, monitoring tools). A tool bench that provides a unified environment to operate the FEDERICA infrastructure and configure also the slices is being developed, and will be progressively deployed and used by the NOC and the FEDERICA users.

4 Status and Challenges

The FEDERICA infrastructure is now supporting a variety of users, who experiment on monitoring, new routing protocols, advanced control and management of physical and virtual circuit based topologies, energy-aware routing. The current feedback is positive and demonstrates the wide range of applicability of an architecture based on virtualization.

The experience has also suggested a list of developments, which also represent a set of research challenges:

- An increased level of control of each virtual resource behaviour. The performance of the virtual resources is still a function of the hardware and software used.
- Monitoring of the substrate and the resources in the slices and their relationship. Monitoring is considered a fundamental resource for the management of the substrate and for users' analysis of their experiments.
- Automation of the procedures and virtual resource description. An increased level of automation is needed to improve the management of the infrastructure and the efficiency in slice provisioning and management. The resource description and the schema to describe the relation between them and with the physical resource ia a fundamental element to achieve a greater automation and the creation of a FEDERICA service. The standardization of the resource representation is fundamental also for the fast-developing "cloud computing" [11] [12] architecture and services, allowing a possible synergy.
- increase the federation capabilities with other facilities to offer a richer environment to the users.

5 Conclusion

An infrastructure substrate based on virtualization both in computing and network resources is a novel approach to provide an ideal environment for innovative research and services on present and Future Internet. The virtual infrastructures created can be tailored to a very large variety of testing scenarios and present a simple interface to the user. The time needed to experimentally validate an idea can be reduced as well as the debugging and analysis phases.

Such infrastructures demonstrate the capability of current technologies to decouple the functionalities from their physical location, creating cloud infrastructures and granting new possibilities (e.g. the mobility of the routing functionality. The developments of the infrastructure require research on reproducibility behaviour, resource mapping, monitoring and resource standardization.

Acknowledgments. The FP7 project FEDERICA is partially supported by the European Commission under the Grant Agreement No.: RI- 213107. The authors acknowledge the fundamental contribution of all project partners.

References

1. Federated E-infrastructure Dedicated to Research Innovating in Computing Architectures, http://www.fp7-federica.eu
2. Chowdhury, N.M.M.K., Boutaba, R.: Network Virtualization: State of the Art and Research Challenges. IEEE Communications Magazine, 20–26 (July 2009)
3. A European Project based on PlanetLab Europe, http://www.onelab.eu/
4. Future Internet Research and Experimentation, http://cordis.europa.eu/fp7/ict/fire/
5. Future Internet Assembly, http://www.future-internet.eu/
6. Global Environment for Network Innovation, http://www.geni.net
7. AKARI Architecture Design Project, http://akari-project.nict.go.jp/eng/index2.htm
8. For a detailed analysis of NRENs in Europe and their role, see the documents (in particular the TERENA compendium), http://www.terena.org/publications/
9. GÉANT, the European Research and Education backbone, http://www.geant.net
10. VMware, http://www.wmware.com
11. Mell, P., Grance, T.: For a definition of Cloud Computing, see NIST Definition of Cloud Computing. Version 15 (10-7-09), http://csrc.nist.gov/groups/SNS/cloud-computing/cloud-def-v15.doc
12. IEEE Computing now: System Virtualization: Cloud Computing: Opportunities and Challenges (October 2009), http://www.computer.org/portal/web/computingnow/articles

Towards a Virtualized Sensing Environment

David Irwin, Navin Sharma, Prashant Shenoy, and Michael Zink

University of Massachusetts, Amherst
{irwin,nksharma,shenoy}@cs.umass.edu, zink@ece.umass.edu

Abstract. While deploying a sensor network is necessary for proof-of-concept experimentation, it is a time-consuming and tedious task that dramatically slows innovation. Treating sensor networks as shared testbeds and integrating them into a federated testbed infrastructure, such as FIRE, GENI, AKARI, or CNGI, enables a broad user community to benefit from time-consuming deployment exercises. In this paper, we outline the challenges with integrating sensor networks into federated testbeds in the context of ViSE, a sensor network testbed we have integrated with GENI, and describe our initial deployment experiences. ViSE differs from typical embedded sensor networks in its focus on high-bandwidth steerable sensors.

Keywords: Testbed, Sensor Network, Federation, Radar.

1 Introduction

Apart from the additional burden of developing, operating, and maintaining an experimental shared testbed, researchers do not typically co-opt their field-deployed sensor networks to serve as shared testbeds for at least three reasons. **First**, the benefits of multiplexing node resources are unclear, since computational, bandwidth, and energy constraints restrict embedded platforms to operating a few passive sensors that collect similar data regardless of the application. Thus, controlling the data gathering process at each sensor node has little value, and end-user applications may simply "share" the collected data at the network's sink.
Second, embedded platforms generally do not support the hardware mechanisms, e.g., MMU or TLB, or software abstractions, e.g., virtual memory or processes, necessary for either basic control plane functions or efficient multiplexing, making it undesirable or even impossible. Software platforms for embedded nodes often do not provide the separation between kernel-space and user-space that testbed's use as the linchpin for safe execution of untrusted user software.
Third, the specialized nature of a sensor network deployment often lends itself to only a single application rather than multiple diverse applications. For example, the choice of sensors and node placement for a network designed specifically for monitoring the vibrations of a volcano [3] are unlikely to be suitable for, say, tracking the movements of nearby wildlife [4].

However, there are a range of "large" sensor types that exist today that do not adhere to the extreme power and form factor constraints of embedded sensor

T. Magedanz et al. (Eds.): TridentCom 2010, LNICST 46, pp. 133–142, 2011.

nodes and, instead, (i) expose programmable sensor actuators to applications, (ii) support the basic mechanisms/abstractions for safe software execution, and (iii) incorporate multiple types of sensor supporting a range of applications. Most notably, networks of steerable sensors are distinct from their embedded counterparts by allowing applications to dictate the type, quality, and quantity of data they collect by steering the sensor to different points in space.

Unlike passive sensors that collect the same data regardless of the application, the data gathering process for each steerable sensor is highly application-dependent. Further, the energy required to move a steerable sensor's mechanical motor necessitates the use of higher-power computing components that support both the hardware mechanisms and software abstractions critical for a testbed control plane. Finally, node platforms are generally powerful enough to operate a range of sensors useful for different applications. Examples of steerable sensors include both steerable weather radars and pan-tilt-zoom video cameras. Recent work has proposed using networks of small steerable weather radars to fill coverage gaps in the NEXRAD radar system [5], while the U.S. Border Patrol uses networks of pan-tilt-zoom cameras to monitor both the northern and southern border [6].

Thus, the basic characteristics of steerable sensor networks do not preclude exposing them as shared testbeds for use by external researchers. As with their embedded counterparts, steerable sensor networks are costly to deploy, operate, and maintain, which magnifies the potential benefits of sharing them with external users. For instance, he hardware cost alone for CASA's steerable radars is $250,000 and does not include infrastructure, operational, or labor costs [5], and the cost for the Border Patrol's 20-mile "virtual fence" using pan-tilt-zoom cameras is over $20 million [6]. In general, much like early mainframe computing systems, the cost and expertise necessary to build and maintain sensing systems significantly restricts the scope of users available to experiment with them.

We also believe that continuing advancements in low-power embedded sensor nodes will eventually mitigate, or even eliminate, the characteristics that discourage testbeds using small form factor field-deployed sensor nodes. For instance, in Section 4.1, we briefly discuss a prototype of a small form-factor sensor node we have built capable of (i) separating control plane functions from user functions using two distinct processors and (ii) supporting multiple types of "high-power" sensors off harvested energy. Steerable sensor networks provide an early opportunity to address challenges, such as control plane separation, multiplexing shared sensor actuators, and dealing with unpredictable energy availability, that are, or will be, relevant to all types of sensor network testbeds.

The focus of this paper is the design and implementation of the ViSE [1] testbed and its integration with ORCA [7,8], a candidate control framework for GENI [9]. ViSE focuses on *virtualizing* steerable sensors to benefit from the strong resource and fault isolation of modern virtualization platforms. Virtual machines isolate ViSE's control plane from untrusted users, and untrusted users from each other.

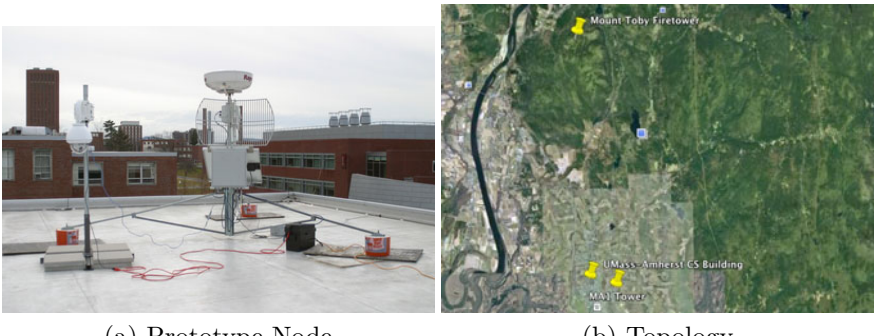
[1] See `http://geni.cs.umass.edu/vise`. ViSE stands for **Vi**rtualized **S**ensing **E**nvironment.

Each ViSE node includes a weather radar, a pan-tilt-zoom camera, and a weather stations that applications programmatically control to sense aspects of their surrounding environment. Thus, ViSE users request slices composed of virtual machines bound to not only isolated slivers of each node's CPU, memory, storage, and bandwidth, as in other GENI testbeds [10,11], but also one or more attached sensors.

ViSE reflects our view that new innovation in sensing systems is necessary to take advantage of new innovation in the Internet and cloud computing, since sensors must transmit their data over the Internet to the computers that ultimately process the data to some end. Section 2 gives an overview of ViSE, while Section 3 details ViSE's current integration with Orca/GENI, using Orca's extensible slice controller implementation, as well as deployment issues we have observed in practice. Section 3 then briefly outlines three specific challenges for ViSE, and other sensor network testbeds, moving forward. Finally, Section 4 discusses concludes.

2 ViSE Testbed Overview

The ViSE testbed currently includes one Internet-accessible node that acts as a gateway to three sensor nodes located on the roof of the UMass-Amherst Computer Science Research Center at 140 Governors Drive in Amherst, MA (42 23' 42.33" N, 72 31' 50.96" W at elevation 62 meters), on the MA1 Radar Tower on the UMass-Amherst campus (42 23' 30.95" N, 72 31' 2.53" W at elevation 120 meters), and on the Mount Toby firetower in Sunderland, MA (42 29' 17.00" N, 72 32' 14.96" W at elevation 385 meters). The distance between Mount Toby and the UMass-Amherst Computer Science Research Center is 10.37 kilometers and the distance between Mount Toby and the MA1 Tower is 10.83 kilometers. There is no link between the MA1 Radar Tower and the Computer Science Research Center since there is no line of sight.



(a) Prototype Node (b) Topology

Fig. 1. ViSE contains nodes (a) spread throughout the Amherst, MA area (b)

Figure 1(a) shows the ViSE node on the roof of the UMass-Amherst Computer Science building. The relative location of each node is depicted in Figure 1(b). We are planning to add one additional node on the Pelham firetower, approximately 10km east of the MA1 tower, and other nodes on campus buildings in the near future. Each node includes three distinct sensors, a Davis VantagePro2 Weather Station, a Sony SNC-RZ50N Pan-Tilt-Zoom Camera, and a Raymarine RD424 Radome Radar Scanner. Testbed nodes use 802.11b over directional antenna for communication. Importantly, programmatic control is available for each sensor, allowing users to build sensor control into their experimental applications. The ViSE testbed is part of the GENI prototype and integrates with GENI's Orca control framework, which we describe in the next section.

ViSE's primary usage scenario is as a platform for experimenting with closed-loop control of adaptive sensor networks using non-traditional steerable sensors. Experiments actuate sensors to capture data at a specific time, location, spatial region, etc., stream that data over a wireless network to compute clusters for analysis, and use the new results to actuate and refocus sensors on important regions as conditions change. For example, recent work [12] explores how shared high-bandwidth sensor systems can intelligently prioritize and compress data when not enough bandwidth exists to transmit all of the sensor data. ViSE also targets the study of multi-sensor experiments, such as performing longitudinal climate studies or studying the fidelity of the long-distance wireless link under different atmospheric conditions including rain, snow, wind, or fog [13]. Finally, ViSE is also useful for experimenting with long-distance wireless communication using directional antennas—the testbed includes two links over 10 kilometers long. Setting up long-distance links, like those in ViSE, is cumbersome since they require line-of-sight from elevated vantage points.

3 GENI/Orca Integration

As with other GENI testbeds [7,10,11], ViSE uses virtualization to separate its control plane from each user's data plane. The control plane has mechanisms to start and stop user VMs, create and destroy VM root disks, attach and detach sensors to VMs, and configure network connections. Users currently request a *slice* of the testbed by logging into ViSE's web portal and issuing a slice request. Each ViSE slice consists of a virtual machine on each node bound to an isolated *sliver* of each node's CPU, memory, storage capacity, network bandwidth, as well as one or more attached sensors [2]. Note that since, currently, ViSE has a chain topology every slice *must* include a virtual machine on each ViSE node—otherwise, users would have no means of accessing virtual machines at the end of the chain. ViSE users may log into the gateway of their slice using a secure shell session at `vise-testbed.cs.umass.edu` on a specified port. ViSE currently uses `ssh` as the method for users to bootstrap their own services and/or inject their own code into a slice, although we are working on integrating the Gush experiment workflow tool [14]. Once inside the gateway, ViSE nodes operate

[2] See [9] for complete description of terminology.

within a private IP address space: the node on the roof of the Computer Science Research Center has IP address 192.168.2.25, the node on the firetower at Mount Toby has IP address 192.168.2.26, and the node on the MA1 Tower on the UMass-Amherst campus has IP address 192.168.2.27.

3.1 Requesting Slices

To gain access to ViSE, each user sends a `ssh` public key to `vise@cs.umass.edu` and receives in return a unique username and password to access ViSE's web portal. Slice requests made through the portal under each login account are tagged by an Orca slice controller, discussed below, with the public key associated with that account. As with other testbeds, on slice creation, each user's `ssh` public key is copied to the root disk image of each of its virtual machine, allowing it to access the machine with its corresponding private key. ViSE currently uses Linux VServers as its virtualization technology because it simplifies attaching privileged sensing devices, such as cameras and radars, to virtual machines. We initially used Xen as our virtualization technology. However, we were forced to switch to VServers since the device driver for our radar's analog-to-digital convertor uses DMA operations that Xen does not currently support. Since ViSE focuses on exposing sensors to users, the OS-level virtualization provided by VServers is sufficient. Note that VServers and Xen exhibit similar resource and fault isolation capabilities [15].

We implement ViSE's web portal as a simple front-end to a customized Orca *slice controller*. The portal uses PHP to log user slice requests to a backend MySQL database, which the ViSE-Orca slice controller polls for new requests every tick of its internal clock. By default, Orca's internal clock ticks every 10 seconds, although, as with a conventional operating system, the tick rate is configurable. Upon detecting a new request, the slice controller reads the request from the database and issues it to Orca's instantiation of the GENI *Clearinghouse*. GENI testbeds that use Orca are able to delegate the right to allocate their resources to the Clearinghouse. As a result, GENI users—including ViSE's web portal and slice controller—request slices from the Clearinghouse, and not from the testbed itself. Note that Orca does not *require* testbeds to delegate their resources to the Clearinghouse. If necessary, testbed's may retain control of resource allocation. However, the Clearinghouse serves as a focal point for a GENI facility to implement GENI-wide resource allocation and authorization policies.

3.2 Clearinghouse Integration

If testbed's retain control of resource allocation then GENI cannot implement coordinated global policies, such as ensuring that slice requests spanning different testbeds and networks at multiple institutions are allocated atomically with the same start time. One alternative to a Clearinghouse approach that implements GENI-wide policy is to force the burden on individual end-users to request resources from each individual testbed. Using this approach, if testbed's do not coordinate with each other, end-users will have no guarantee of being

granted resources from each testbed at the same time. ORCA currently operates a GENI Clearinghouse for multiple institutions, including Ohio St. University, Duke University, University of Massachusetts at Amherst, Northwestern University, and the University of Houston, at the Renaissance Computing Institute in Chapel Hill, North Carolina. Once the Clearinghouse approves or declines a ViSE slice request, it sends the response, in the form a signed ticket, to the ViSE slice controller, which then logs a state change for the request to the web portal's database to notify the portal, and hence the user, of the request's current status.

The Clearinghouse allocation policy for ViSE always approves requests from the ViSE slice controller for the next available unallocated time slot. If the request is approved, the portal notifies the user of the start time of its slice. To activate the slice, the ViSE slice controller redeems the ticket it received from the Clearinghouse with ViSE's *aggregate manager*. The slice controller also attaches important user-specific configuration properties to this redeemed ticket; in ViSE, the most notable property is the public key the user initially registered, described above. ViSE's architecture in combination with GENI/ORCA's architecture, separates the slice controller, which accepts and issues slice requests, from the aggregate manager, which uses ViSE's control plane to allocate and configure virtual machines, to accommodate slice controllers operated locally by end-users. Thus, rather than ViSE operating a single monolithic web portal that end-users leverage to issue slice requests, end-users may operate their own slice controllers that programmatically issue slice requests for ViSE or other testbeds.

The underlying assumption of ORCA's architecture is that the only thread common to all GENI testbeds is that they share resources and do not use them forever. As a result, all delegations and requests in ORCA are represented as leases with well-defined start and end times. Since ORCA does not make any assumptions about the attributes of a resource, integrating ViSE's non-traditional sensing resources with its slice controller, Clearinghouse, and aggregate manager implementations was straightforward. ORCA includes extensible implementations of each server that exposes lease handler interfaces that execute, for each request, at the start and end of its lease at both the slice controller and the aggregate manager.

3.3 Slice Controller Integration

For ViSE's slice controller, we implement lease handlers that update the web portal's database when leases start and end to notify users of the status of their slice through the web portal. For ViSE's aggregate manager, we implement lease handlers that setup and teardown virtual machines on each ViSE node, attach sensors to them, and write the user's public key to each VM's root disk. The aggregate manager setup handler snapshots a copy-on-write template virtual machine disk image using Linux's logical volume manager to create the root disk for each virtual machine in a slice. The template disk image includes simple scripts and code segments that provide examples for how to control each sensor. Once active, the aggregate manager notifies the slice controller, which logs the notification to its database.

Initially, we are limiting the *degrees of freedom* available to users through the ViSE web portal, even if Orca's default slice controller provides them. Examples of degrees of freedom include the ability to (i) request leases with variable lengths and variable start times, (ii) extend leases with a variable duration, (iii) lease virtual disks or sensors independently of virtual machines, (iv) cancel a lease before its end time, (v) add or remove nodes from a slice on or before a lease extension, or (vi) increase or decrease the size of a node's sliver, i.e., share of CPU, bandwidth, memory, etc., attached to each virtual machine. Instead, ViSE's portal currently forces users to issue slice requests for four hour durations starting in the next available slot. Users cannot extend leases beyond this four hour period or submit additional requests before their current lease has finished to prevent hoarding.

Some of the degrees of freedom above do not apply to ViSE, although they may apply to other testbeds. In particular, as previously mentioned, we allocate all ViSE nodes in each slice (v) rather than allocate partitions of the network. Initially, we are only allocating dedicated nodes, so (vi) does not currently apply, although we are investigating multiplexing actuators, which we discuss in Section 4.2. While (i), (ii), (iii), and (iv) may be useful, they burden the user with formulating complex requests, and the Clearinghouse with implementing sophisticated allocation policies that require mechanisms to prevent users from hoarding resources. We plan on integrating these functions for users as necessary. Our approach is to start simple as we attract users, and allow their experiences to motivate the degrees of freedom we ultimately add and support.

4 Challenges

While Orca's minimalist design based on leases, which only assumes that users do not use resources forever, simplified our initial integration of ViSE with GENI, we have identified at least three challenges moving forward. We outline current challenges in ViSE related to control plane separation, fine-grained actuator multiplexing, and unpredictable energy supplies below. Although ViSE focuses predominantly on steerable sensors, we view these challenges as also being applicable to embedded sensors as well.

4.1 Control Plane Separation

As with other testbeds, sensor network testbeds must separate their control plane from each user data plane. ViSE accomplishes this separation on each node using virtual machines. However, since ViSE nodes connect wirelessly, it currently does not separate control plane communication. The control plane operations occur over the same wireless channel used by each testbed user. In related work, we built a small form factor sensor platform that uses two distinct nodes and radios for control plane and data plane communication [16]. The advantage of the dual node/radio approach is that the properties of the control plane's node/radio can be well-matched to its needs. For example, in our work we matched a mote-class control plane node/radio with a more powerful embedded node, capable of running Linux, for the data plane.

The control plane node could always remain on because its energy demands were small compared with the demands of the data plane node, which required an appropriately sized solar panel. Since the control plane only executes a small number of operations, the more powerful computing platform of the embedded node was not necessary. Further, the control plane radio had low bandwidth, but long range, rather than the high bandwidth, but short range, radio of the embedded node. The low bandwidth radio is appropriate for a control plane that only sends short commands to nodes, while the long range is appropriate for providing greater connectivity in the case of node failures. Finally, separating the control plane onto a different processor decouples the control plane from the software on the main embedded node, allowing faults to be tolerated on the embedded node.

We are investigating using the same approach in ViSE. However, in the case of ViSE, we have prototyped a solution using our embedded node from the work above—the Linux Gumstix platform—as the control plane node, since for ViSE the Gumstix draws significantly less power than our x86-class nodes. For our radio, we have prototyped using a GPRS modem connected to AT&T's cellular network. Much like the mote-class control plane node in our prior work, the GPRS modem exhibits low bandwidth but a wide range. An advantage in ViSE of using a separate control plane radio is that it allows testbed applications to control the main wireless radio's operational settings, such as its operating frequency, bit rate, or MAC-layer protocol. Currently, ViSE does not allow applications control of these settings because ViSE's control plane uses them, and, if they are changed ViSE's control plane becomes disconnected. However, long-distance wireless researchers may find the control useful. ViSE also uses automated reboot switches, triggered by cell phone text messages, as a last resort to rebooting a node that cannot be contacted, since physical access to the nodes is time-consuming in good weather and nearly impossible in poor weather or during winter.

4.2 Fine-Grained Actuator Multiplexing

ViSE currently allows testbed users dedicated control of each steerable sensor. However, we are integrating support for fine-grained multiplexing of sensor actuators, which we call MultiSense [17]. MultiSense enables multiple virtual machines to each control and steer their own *virtual sensor*. The virtual machine hypervisor, or privileged management domain, then multiplexes steering requests at a fine-grain to provide the illusion that each virtual machine controls a dedicated, albeit, slower sensor. MultiSense optimizes a proportional-share scheduler, which we call Actuator Fair Scheduing or AFQ, for steerable sensors by adding support for judicious batching and merging of requests, anticipatory scheduling, and state restoration. Our results from a MultiSense prototype show that for ViSE's pan-tilt-zoom camera it enables a tracking application to photograph an object moving at nearly 3 mph every 23 ft along its trajectory at a distance of 300 ft, while supporting a security application that photographs a fixed point every 3 seconds. Of course, MultiSense diverges from strict fairness

between virtual machines by reordering steering requests, in part, by batching together requests that are "near" each other. An in-depth description of MultiSense and a complete experimental evaluation is available in [17]. Once we integrate MultiSense with ViSE, we will be able to allocate sensor "slivers" in the same way that we allocate slivers of CPU, memory, or bandwidth.

4.3 Unpredictable Energy Supplies

Finally, while there is currently A/C power available for each ViSE node, we have also connected nodes to both solar panels, wind turbines, and batteries. We are studying how to operate nodes purely off harvested energy to enable us to deploy nodes where external A/C power is not available. One consequence of operating a testbed using unpredictable energy supplies is that both testbed users and the GENI Clearinghouse expect predictable behavior. For example, ViSE's aggregate manager delegates the right to allocate its resources to a GENI Clearinghouse for a fixed period of time. Recall that this delegation is important for implementing GENI-wide policies, such as atomically allocating slices that span multiple testbeds. However, determining the duration of this time period is dependent on ViSE's available energy supply, which is not entirely predictable.

One option is to choose the duration based on the current reserves in the battery to ensure that the Clearinghouse will not allocate nodes to users when there is no energy to run them. An alternative option, which we are exploring, is to use weather forecasts to increase the duration that ViSE can delegate to GENI. If weather forecasts are accurate, they should provide a means for increasing the duration of possible requests ViSE can satisfy. Additionally, the simple policies above allocate resources assuming that testbed users operate nodes at their full utilization. If nodes are not operated at their full utilization, then additional energy is available for subsequent experiments. With a GENI-wide Clearinghouse, these updates must propagate to the Clearinghouse to allow subsequent testbed users to take advantage of the additional resources. One avenue for future exploration is the performance impact of these updates on a federated Clearinghouse.

5 Conclusion

In this paper, we have described the motivation behind ViSE, a steerable sensor network testbed we are building in Western Massachusetts, described ViSE's integration with Orca, a candidate GENI control framework, and outlined three challenges ViSE presents moving forward. We are actively working on ViSE's integration with other substrates and testbeds, including other GENI testbed's, cloud computing environments, and networks, such as NLR and Internet2, to allow cross-testbed slices.

Acknowledgments. This material is supported by the NSF grant CNS-0834243.

References

1. Werner-Allen, G., Swieskowski, P., Welsh, M.: Motelab: A Wireless Sensor Network Testbed. In: Information Processing in Sensor Networks, pp. 483–488 (2005)
2. Ertin, E., Arora, A., Ramnath, R., Naik, V., Bapat, S., Kulathumani, V., Sridharan, M., Zhang, H., Cao, H., Nesterenko, M.: Kansei: A Testbed For Sensing At Scale. In: Information Processing in Sensor Networks, pp. 399–406 (2006)
3. Weren-Allen, G., Lorincz, K., Johnson, J., Lees, J., Welsh, M.: Fidelity And Yield In A Volcano Monitoring Sensor Network. In: Operating System Design and Implementation, pp. 381–396 (2006)
4. Juan, P., Oki, H., Wang, Y., Martonosi, M., Peh, L., Rubenstein, D.: Energy-efficient Computing For Wildlife Tracking: Design Tradeoffs And Early Experiences With Zebranet. In: Architectural Support for Programming Languages and Operating Systems, pp. 96–107 (2002)
5. NSF Center For Collaborative Adaptive Sensing of The Atmosphere, CASA (2009), `http://casa.umass.edu` (accessed)
6. Magnuson, S.: New Northern Border Camera System To Avoid Past Pitfalls. National Defense Magazine (2009)
7. Irwin, D., Chase, J., Grit, L., Yumerefendi, A., Becker, D., Yocum, K.: Sharing Networked Resources With Brokered Leases. In: USENIX, pp. 199–212 (2006)
8. Chase, J.: Orca Control Framework And Internals. In: Duke University Technical Report (2009)
9. The GENI Project Office. GENI System Overview. The GENI Project Office, Technical Report GENI-SE-SY-SO-02.0 (2008)
10. Bavier, A., Bowman, M., Chun, B., Culler, D., Karlin, S., Muir, S., Peterson, L., Roscoe, T., Spalink, T., Wawrzoniak, M.: Operating Systems Support For Planetary-scale Network Services. In: Network Systems Design and Implementation, pp. 253–266 (2004)
11. White, B., Lepreau, J., Stoller, L., Ricci, R., Guruprasad, S., Newbold, M., Hibler, M., Barb, C., Joglekar, A.: An Integrated Experimental Environment For Distributed Systems And Networks. In: Operating System Design and Implementation, pp. 255–270 (2002)
12. Li, M., Yan, T., Ganesan, D., Lyons, E., Shenoy, P., Venkataramani, A., Zink, M.: Multi-user Data Sharing In Radar Sensor Networks. In: Embedded Networked Sensor Systems, pp. 247–260 (2007)
13. Kang, W., Stankovic, J., Son, S.: On Using Weather Information For Efficient Remote Data Collection In WSN. In: Workshop on Embedded Networked Sensors (2008)
14. Gush (2009), `http://gush.cs.williams.edu/` (accessed)
15. Soltesz, S., Potzl, H., Fiuczynski, M., Bavier, A., Peterson, L.: Container-based Operating System Virtualization: A Scalable, High-performance Alternative To Hypervisors. In: EuroSys, pp. 275–288 (2007)
16. Sharma, N., Gummeson, J., Irwin, D., Shenoy, P.: SRCP: Simple Remote Control For Perpetual High-power Sensor Networks. In: European Conference on Wireless Sensor Networks, pp. 358–374 (2009)
17. Sharma, N., Irwin, D., Shenoy, P.: VSense: Virtualizing Stateful Sensors With Actuators. University of Massachusetts, Amherst, Technical Report UM-CS-2009-033 (2009)

TridentCom 2010

Practices Papers Session 2: Future Internet Testbeds for Wireless Sensors, Media and Mobility

The w-iLab.t Testbed

Stefan Bouckaert, Wim Vandenberghe, Bart Jooris, Ingrid Moerman, and Piet Demeester

Ghent University - IBBT, Department of Information Technology (INTEC), Gaston Crommenlaan 8, Bus 201, 9050 Ghent, Belgium
Stefan.Bouckaert@intec.ugent.be
http://www.ibcn.intec.ugent.be

Abstract. In this paper, the W-iLab.t wireless testbed is presented. The testbed consists of nearly 200 sensor nodes and an equal amount of WiFi nodes, which are installed across three floors of an office building. The testbed supports wireless sensor experiments, WiFi based mesh and ad hoc experiments, and mixed sensor/WiFi experiments. It is explained how changes in the environment of the sensor nodes can be emulated and how experiments with heterogeneous wireless nodes are enabled. Additional features of the testbed are listed and lessons learned are presented that will help researchers to construct their own testbed infrastructure or add functionality to an existing testbed. Finally, it is argued that deep analysis of unexpected testbed behavior is key to understanding the dynamics of wireless network deployments.

Keywords: testbed, wireless sensor, wireless ad hoc, wireless mesh, emulation.

1 Introduction

As a research group, frequently involved in interdisciplinary projects with industrial partners, validation of developed algorithms and protocols for wireless ad hoc, sensor and mesh networks on actual (prototype) hardware has been an important way of proving validity of theoretical and simulated novel concepts, and demonstrating the feasibility of network architectures [1,2]. Very often, our wireless experiments revealed minor or major flaws in theoretical assumptions [3], requiring time intensive debugging sessions and algorithm modifications that would not have been required if simulation results were the final product of our research.

Over the years, multiple different small scale wireless sensor and wireless mesh testbeds were set up and torn down in the scope of various projects, master theses and doctoral theses. While a lot of lessons were learned from these experiments on diverse types of hardware, there are also several drawbacks associated with the deployment of multiple individual testbeds. *(i)* Buying new hardware setups for every project is costly, and therefore limits the deployment scale. *(ii)* Different hardware architectures require different development approaches. As an example, in the case of IEEE802.11 based mesh and ad hoc research, experiments have been performed using off-the-shelf WiFi routers with custom built

T. Magedanz et al. (Eds.): TridentCom 2010, LNICST 46, pp. 145–154, 2011.

firmware, custom built multi-interface mesh nodes, PDAs, tablets, laptops and desktop computers with various wireless NICs, and integrated system boards. While experience with diverse network platforms is gained, there is a substantial overhead associated with creating new development environments. *(iii)* Results obtained from different test set-ups cannot easily be compared. *(iv)* Rebuilding old test set-ups is time-consuming and has a negative impact on the reproducibility of test results.

In order to overcome the drawbacks of these individual test set-ups and to enable wireless tests on a larger scale, the w-iLab.t testbed was designed and installed at the buildings of the IBCN research group and IBBT research institute in Ghent, Belgium. The w-iLab.t inherited its name from the larger IBBT iLab.t [4] test infrastructure, where the testbed is a part from. The w-iLab.t testbed consists of nearly 200 sensor nodes and an equal amount of WiFi nodes, which are mounted to the ceilings in the offices and hallways. Although the primary focus of the testbed is to support large scale wireless sensor and actuator network deployments, the testbed architecture supports WiFi mesh and ad hoc test, and mixed sensor/ad hoc experiments as well. In the remainder of this paper, the w-iLab.t testbed is presented. The design choices are motivated and the possibilities are demonstrated. Furthermore, we present lessons learned which can help testbed designers to analyze behavior of their own testbed set-up, inspire testbed administrators to add time saving functional blocks to their set-up, or act as a guideline during the initial design phase of a new testbed.

2 Goals and Requirements

One of the major drivers to perform real life experiments, is the fact that a purely mathematical or simulation based approach for designing wireless network solutions is not entirely representative for the real life performance of the same solutions when deployed in realistic environments. The reason for this discrepancy is a result of simplified traffic pattern and end user models, wrong assumptions about signal propagation and interference, interactions with other (wired or wireless) network devices and errors introduced by hardware and wireless drivers. While the latter errors should not be solved by upper layer network designers in theory, the success and applicability of a developed algorithm depends on the algorithm's ability to cope with unpredictable behavior introduced by any of the above elements.

Through careful simulations and well designed small scale testbeds, networking algorithms and protocols can efficiently be debugged to a certain extent. Multi-hop environments can be emulated on a desktop by interconnecting a small number of wireless nodes through coaxial connections, RF splitters and RF attenuators [5], without the need for a large test infrastructure. However, even with the most advanced simulation models or desktop testbeds it is hard to represent a real networking environment, especially when it comes to simulating interaction with user-level programs and operating systems, evaluating network scanning techniques and channel selection mechanisms, or when

modeling dynamic network environments with moving users and external interference. Additionally, measuring user satisfaction and quality of experience is only possible with large scale testbeds deployed in a realistic environment. Therefore, similar to [6] and [7], it was chosen to install the testbed in an office environment across three $18m$ by $90m$ office floors and thus create a natural network topology. On top of this default topology, additional topology control measures (cf. Section 4.2) can be taken to vary the perceived node density in the testbed.

In addition to allowing experiments in a realistic office setting, multiple technical and practical requirements were set before designing the testbed:

- Future network environments are expected to be increasingly heterogeneous. Therefore, the testbed should support tests with wireless sensor and actuator nodes, WiFi based mesh and ad hoc nodes, and mixed scenarios. Since sensor nodes are continuously evolving, it should be possible to easily replace or install additional sensor nodes at the test locations.
- It should be possible to install new software to any sensor or mesh/ad hoc node from a remote location, and to reboot the nodes remotely in case of node failure. The nodes are preferably powered by external power sources, as to avoid the frequent replacement of batteries.
- Sensor nodes react to environmental changes. Testing protocols that depend on environmental changes is not easily done with current testbeds, as, for example, it is not very convenient to test the reaction of a protocol detecting fire through a fast rise in temperature by holding a flame close to a temperature sensor. Therefore, the testbed infrastructure should be able to emulate environmental changes instead of relying on manual interventions, without necessarily requiring specific simulation protocols to be compiled with the software under test.
- Researchers should be able to use the testbed from any location. Personalized log in is needed to provide access control and to guarantee a fair share of access to the testbed for each user.
- Advanced logging functionalities are needed, both for wireless sensor network experiments and wireless mesh and ad hoc experiments.
- Deploying the network devices at the test locations must be as fast and simple as possible, requiring the least possible number of cables to be installed in the offices, reducing the installation cost and minimizing damage to the building.

In the next section, it is explained how the w-iLab.t architecture is able to fulfill all of the above requirements.

3 Testbed Architecture

3.1 Node Location

The testbed node locations are distributed across thee similar floors of office space. Figure 1 shows the location of the nodes on the third floor. Nodes are

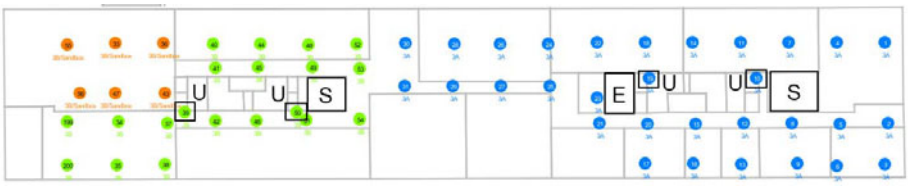


Fig. 1. Third floor of the testbed location. Office area is approximately 90m x 18m. S: staircase. E: elevator. U: utility shaft.

mounted near the ceiling of both hallways and individual offices which are separated by thermal insulated wooden walls causing little RF attenuation. Several other interesting construction elements are indicated on the floor plan: the elevator and elevator shaft are indicated by E and cause severe RF attenuation. Staircases enclosed in concrete walls (S) and concrete utility shafts (U) which run across the different floors cause an increased RF loss as well. Since the office ceiling is made of metal rasters and the floors of aluminum tiles, there is a large inter-floor signal attenuation inside the building. Therefore, it was chosen to deploy nodes in the utility shafts at every floor, thus constructing inter-floor paths with low signal attenuation.

3.2 Hardware Components and Initial Testbed Installation

The TMote Sky sensor mote, which is used as the primary type of sensor device in our testbed, is programmed through a USB interface. Since USB technology is not designed to support cable lengths longer than 3 to 5 meters without intermediate USB hubs, a large sensor network cannot be deployed in an office environment using USB cables only. In contrast, Ethernet technology allows longer cable lengths, but is not commonly supported on sensor nodes.

The chosen solution for our testbed was to deploy cheap embedded Voyage Linux operated Alix 3c3 system boards [8] at all node locations. These embedded system boards, which we call *iNodes*, are equipped with an ethernet NIC, a serial port, VGA output, compact flash storage, onboard audio, two mini PCI slots and two USB ports. Using the iNodes as relay devices allows the sensor nodes to be programmed remotely.

Two additional advantages are associated with the use of these iNodes: *(i)* by installing two miniPCI 802.11a/b/g compatible atheros based wireless NICs and adding dual band antennas, the control hardware for the sensor tests can be used as test equipment for WiFi based mesh and ad hoc tests. *(ii)* The power consumption of the iNodes is low: each iNode consumes only 6.5W in idle state, rising to 7.8W if both WiFi interfaces are enabled and continuously transmitting with a processor load of 100%. The low power consumption allows the iNodes to be powered using only power over ethernet (PoE). As such, only a single ethernet cable and a PoE converter per node are needed to power the iNodes and connect them to a central administration server. This reduces installation

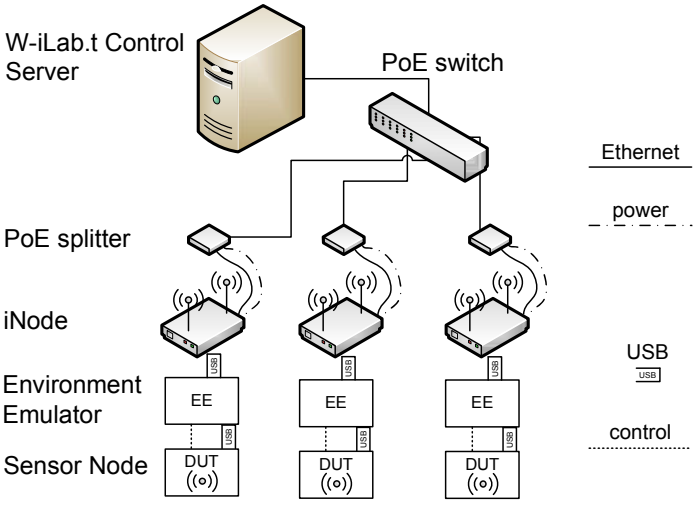


Fig. 2. Logic overview of the w-iLab.t architecture

complexity and cost, and allows for remote power switching of the iNodes, and by extension, sensor nodes.

In order to emulate changes to the physical environment of the node, an in-house designed circuit board called *environment emulator* (EE) is added between the USB port of the sensor device under test (DUT) and the iNode. The EE is built around a micro controller, a three port USB hub, and a voltage regulator/measurement chip. It is plugged into a USB port of the iNode, and is equipped with two additional USB ports. The most important goals of the EE are the following: first, one port is used to connect the DUT, the other port allows to connect additional EEs in cascade, thus allowing multiple (heterogeneous) sensor nodes to be tested using the same back-end testbed infrastructure. Second, the EE can replace the USB power from the DUT with its own internal power source. Thus, the EE is able to emulate depleting batteries, energy harvesting power sources and node failures. Third, the power that is consumed by the DUT is measured with a sample frequency of 4kHz, allowing to measure the exact power consumption of any sensor node while executing a certain protocol. Fourth, general purpose digital and analog IO pins are connected to the DUT, allowing to emulate real time digital and/or analog sensor input via programmable events. Fifth, a seven segment LED display and status LEDs provide additional feedback when e.g. flashing the sensor nodes, writing information to the logs, or during events occurring during normal node operation.

The hardware components of the w-iLab.t testbed architecture are summarized in Figure 2: the iNodes are powered and connected to a control server through a gigabit PoE switch. Two WiFi cards are installed at the iNodes, allowing to perform WiFi mesh and ad hoc experiments, and the USB ports of

the iNodes are used to connect the sensor node via an environment emulator, which allows advanced testbed manipulation and logging.

3.3 Using the Testbed

Wireless Sensor and Actuator Experiments. The w-iLab.t testbed is accessible by authorized users via a web based interface, which allows users to monitor the testbed status, to upload sensor firmware, to select which nodes will be running what type of firmware during a specific experiment, to schedule an experiment at a specific time for a specific duration, to get an overview of past, current and future tests, and to retrieve results and additional information on completed tests.

The testbed is organized in several geographical zones and sub-zones such as *'third floor'*, *'first half of the third floor'* or *'entire testbed'*. The user can schedule tests in one or multiple zones, or may deploy different code on each individual node. Zone reservations are non blocking, meaning that if one user is running a test on one zone, another user might run a simultaneous test in another, non-overlapping zone. To avoid interference from other experiments, a single user can reserve the whole testbed but only use part of it.

The W-iLab.t control server software is based on the MoteLab [6] software. The software was modified and expanded to support the use of the EE and to allow a more advanced collection and easy representation of test results. Modifications include *(i)* added support for EE scenarios. The user is able to configure events to be triggered at (a selection of) EEs at a user specified time. For example, the user might specify a scenario in which several buttons are pressed at some sensor nodes, while other sensor nodes observe an emulated rise in temperature or fail because of (emulated) battery depletion. The EEs are synchronized and execute the scenario with a maximum error of $100\mu s$. *(ii)* A result processing toolbox, comprising a *sniffer*, *visualizer* and *analyzer* module. Events and sensor node logging information are stored in an SQL database together with the precise timestamps and other test data such as the individual power consumption of the sensor nodes. If the sniffer is enabled, certain sensor nodes are configured as promiscuous nodes and keep a log of all captured frames on a user defined channel. The visualizer and analyzer are universal GUIs allowing both real-time and post-experiment visualization of e.g. packet flows, sensor values or other user measured data, either on a map of the sensor testbed, or by producing a scatter diagram of measured values.

As such, a user is able to easily define tests and emulated scenarios, schedule sensor experiments, and analyze and visualize test results in real-time or after the experiment.

Wireless ad hoc and Wireless Mesh Experiments. As previously stated in Section 3.2, two WiFi NICs are installed at every iNode. In order to enable mixed WiFi node / sensor node experiments and to keep a uniform interface, it was decided to integrate the support for the WiFi nodes into the same web interface as used for the sensor nodes. Moreover, this fully integrated approach assures that no scheduling conflicts can occur between wireless sensor and wireless mesh

experiments. Additionally, when running WiFi experiments, the user should be allowed to operate the devices using a custom Linux kernel, custom drivers and custom application software.

Implementing the above flexibility for WiFi tests might endanger the operation of the sensor testbed: in the default testbed set-up, the iNodes execute a daemon which interprets management information from the central control server, controls the EE and installs the firmware to the DUT. Hence, there could be a certain risk involved in allowing the iNodes to be used for experiments: if a WiFi experiment goes wrong or a user deliberately or unwillingly removes or corrupts crucial files needed for booting the iNode or controlling the sensor nodes, the sensor testbed might become unstable or stop functioning.

These potential issues were avoided as follows. Three subcomponents are required to operate the WiFi testbed: the w.iLab-t central control server acting as a *Preboot Execution Environment* (PXE) server, a user defined *Network File System* (NFS) share, and the iNodes themselves. Two partitions are installed on the iNodes: a first partition holding the original iNode software for controlling sensor experiments, and a second partition used for WiFi experiments only, possibly in combination with a user specified kernel. Whenever an experiment is scheduled, the iNodes reboot using the management functionalities of the PoE switch and contact the PXE server to determine which partition to boot. In case of a WiFi or mixed experiment, the iNode is instructed to load the second partition. The user might specify the location of a custom image using specific kernel located on the NFS share, and also specifies the location of the libraries, binaries and other files or scripts needed to perform the experiment. As a new experiment starts, a user defined start script is executed that e.g. might copy the required files from the share to the iNodes, and/or execute a specific program. Not all nodes need to run the same code, allowing experiments with different node roles.

After the WiFi experiment completes, the iNodes automatically reboot and are instructed to load the first partition. As the first partition is booted again, the second partition is restored to its default state, providing clean iNodes for the next test using WiFi nodes.

Each time a scheduled experiment runs, a logging directory is created on the user defined NFS share. For each iNode in the test, a subfolder is automatically created that uniquely identifies the iNode by its hostname. The respective directories are then mounted to a logging directory on the iNodes. All output that is redirected to this directory on the iNodes is stored on the NFS share. This results in a flexible, fully user specified logging system. Furthermore, as the clocks on iNodes are synchronized through the Precision Time Protocol, logging output can be correlated by adding timestamps to the log messages.

4 Additional Features and Lessons Learned

4.1 Defining New Experiments

The W-iLab.t infrastructure allows fast and easy deployment of newly developed code on a large number of devices. Therefore, it is tempting to not only use

the testbed for large scale deployment of stable algorithms, but also during the development phase for testing incremental adjustments. This results in the testbed not being available for the tests for which it is actually meant, and causes the sensor nodes and/or flash cards of the WiFi node to undergo a large amount of program/erase cycles during a single day, shortening the lifetime of the flash chips in the testbed. Therefore, early development is still performed on isolated small scale set-ups. Additionally, one zone in the testbed is reserved as a *sandbox* area which is meant for functional testing of new code before switching to another testbed zone. While the use of the 'normal' testbed zones is limited by a user-based quota, the sandbox area is not, thus promoting its use.

As for WiFi experiments, it was learned that when no user specified kernel is used, particular care should be taken in keeping the software on the personal testbeds and large scale testbed synchronized. More specifically, different versions of wireless drivers have shown to cause significant changes to stability and throughput, and result in syntax changes, leading to unexpected results. While obvious, simple driver settings such as disabling antenna diversity when only a single antenna is connected to the wireless NIC are often forgotten but result in considerable stability increases.

Furthermore, it was found that when analyzing a protocol, a researcher often has to create a lot of similar tests, where only a few parameters are changed. For example, in a sensor experiment, one might want to re-run a test on a different transmission power, or change the transmission interval of a certain protocol. Therefore, the option to use parameters in test definitions was added to the testbed: a user might schedule a single test, but with different parameters which are determined at scheduling phase. The system will translate these parameters to individual tests and schedule all of them. This way, a very large amount of test data is collected through a single scheduling action.

4.2 Topology Control

As previously stated, the w-iLab.t testbed is not located in a separate room but deployed in an office environment. This way, the use of noise injection [9] topology control techniques or attenuators was hoped to be avoided. While this assumption proved to be correct for the sensor network experiments, it was found that it is hard to create topologies with a large number of hops using the WiFi nodes, as their transmission power cannot be set to a value below $0dBm$ due to driver restrictions. After determining the receive sensitivity of the WiFi cards through a measurement campaign using a variable attenuator and modeling the RF propagation characteristics of the office environment, it was decided to add fixed attenuators to all WiFi interfaces of the testbed on the second and third floor of the testbed, with attenuation values of $10dB$ and $20dB$ respectively. The result of this attenuation is a variation of perceived node density at the different floors. Note that the effect of the $10dB$ attenuators on sending and receiving interfaces may be canceled by changing the output transmission power from $0dBm$ to $20dBm$, and that variation of the transmission power of the attenuated nodes allows to emulate environments ranging from sparsely connected (only the

direct neighbors are within transmission range) to very densely connected (over 60 nodes in transmission range).

4.3 Cautionary Perspective on Testbed Experiments

While new testbed experiments are often characterized by unexpected issues such as protocol failures, node failures or driver errors, it is important to realize that every error happens for a reason. Although this is an obvious observation, authors discussing testbed experiments all too often resort to educated guesses on why a certain error was observed, such as "*we believe that the errors are introduced by the wireless driver*". There are two reasons for these often vague descriptions: first, it takes a huge amount of time to debug all aspects of a testbed deployment, while theoretic calculations and simulations might already be available and are considered to provide adequate proof of an algorithm's or protocol's functionality. Second, the tools to analyze the complex behavior of the testbed might lack.

With respect to the above, some recommendations are the following. *(i)* Test should preferably be run with some nodes acting as a sniffer, since the actual transmitted data is often key to solving problems and better understand the actions (not) taken by the protocol under test. *(ii)* Additionally, even when analyzing upper layer protocols, (basic) knowledge of RF propagation and interference is recommended. *(iii)* Finally, using open source software allows deep analysis of observed behavior.

It should never be forgotten that one of the reasons of using testbeds is to be able to study the behavior of a protocol in a realistic environment. If discovered issues are put aside because the are "*probably* due to X or Y", then the effort of implementing a fully working solution should probably not have been made to begin with.

5 Conclusion

The w-iLab.t testbed supports large scale sensor deployments, WiFi based mesh and ad hoc tests, and mixed sensor/WiFi experiments, and is therefore able to analyze the behavior of future heterogeneous network deployments. Nearly 200 node locations are available, situated across three floors of an office building. Through an easy-to-use web-based interface, researchers are able to control the deployment of the software to be tested based on network zones or may address individual nodes. Moreover, the environment emulator allows to emulate sensor network scenarios, provides advanced logging and control, and allows the modular addition of other type of sensor nodes. Test results can be visualized on a map or in graphs in real-time or after the test. The possibility to generate multiple tests based on the same code has proved to be a time-saving functionality, and attenuating WiFi signals is a feasible technique to create a sparser topology in the testbed.

The listed testbed experiences may inspire researchers to design a brand new testbed, or modify or expand their existing testbeds. In order to get a better

understanding of the dynamics involved in a real-life deployment, it is necessary to try and explain all erratic behavior observed while conducting testbed experiments. This will eventually lead to the development of robust wireless deployments that are expected to be part of our lives tomorrow.

Acknowledgment

Stefan Bouckaert thanks the Institute for the Promotion of Innovation through Science and Technology in Flanders (IWT-Vlaanderen) for funding part of this research through a grant.

References

1. Bouckaert, S., Bergs, J., Naudts, D., De Kegel, E., Baekelmans, J., Van den Wijngaert, N., Blondia, C., Moerman, I., Demeester, P.: A mobile crisis management system for emergency services: from concept to field test. In: Proceedings of the WiMeshNets 2006 Workshop, part of the Third Intl. Conference on Quality of Service in Heterogeneous Wired/Wireless Networks, Waterloo, Canada (August 2006)
2. Vandenberghe, W., Lamont, K., Moerman, I., Demeester, P.: Connection management over an ethernet based wireless mesh network. In: WRECOM, Wireless Rural and Emergency Communications Conference, Rome, Italy (February 2007)
3. Bouckaert, S., Naudts, D., Moerman, I., Demeester, P.: Making ad hoc networking a reality: Problems and solutions. Journal of Telecommunications and Information Technology 1(1), 3–11 (2008)
4. iLab.t technology center, `http://ilabt.ibbt.be/`
5. Kaba, J.T., Raichle, D.R.: Testbed on a desktop: strategies and techniques to support multi-hop manet routing protocol development. In: MobiHoc 2001: Proceedings of the 2nd ACM International Symposium on Mobile ad hoc Networking & Computing, pp. 164–172. ACM, New York (2001)
6. Werner-Allen, G., Swieskowski, P., Welsh, M.: Motelab: a wireless sensor network testbed. In: Fourth International Symposium on Information Processing in Sensor Networks, IPSN 2005, pp. 483–488 (April 2005)
7. Handziski, V., Köpke, A., Willig, A., Wolisz, A.: Twist: a scalable and reconfigurable testbed for wireless indoor experiments with sensor networks. In: REALMAN 2006: Proceedings of the 2nd International Workshop on Multi-hop ad hoc Networks: From Theory to Reality, pp. 63–70. ACM, New York (2006)
8. PC Engines. Alix system board, `http://www.pcengines.ch/alix.htm`
9. Kaul, S.K., Gruteser, M., Seskar, I.: Creating wireless multi-hop topologies on space-constrained indoor testbeds through noise injection. In: 2nd International Conference on Testbeds and Research Infrastructures for the Development of Networks and Communities, TRIDENTCOM 2006 (2006)

From Kansei to KanseiGenie: Architecture of Federated, Programmable Wireless Sensor Fabrics

Mukundan Sridharan[1], Wenjie Zeng[1], William Leal[1], Xi Ju[2], Rajiv Ramnath[1], Hongwei Zhang[2], and Anish Arora[1]

[1] Dept. of Computer Science and Engg., The Ohio State University, Columbus, OH, 43210, USA
{sridhara,zengw,leal,ramnath,anish}@cse.ohio-state.edu
[2] Dept. of Computer Science, Wayne State University, Detroit, MI, 48202, USA
{xiju,hongwei}@wayne.edu

Abstract. This paper deals with challenges in federating wireless sensing fabrics. Federations of this sort are currently being developed in next generation global end-to-end experimentation infrastructures, such as GENI, to support rapid prototyping and hi-fidelity validation of protocols and applications. On one hand, federation should support access to diverse (and potentially provider-specific) wireless sensor resources and, on the other, it should enable users to uniformly task these resources. Instead of more simply basing federation upon a standard description of resources, we propose an architecture where the ontology of resource description can vary across providers, and a mapping of user needs to resources is performed to achieve uniform tasking. We illustrate one realization of this architecture, in terms of our refactoring the Kansei testbed to become the KanseiGenie federated fabric manager, which has full support for programmability, sliceability and federated experimentation over heterogeneous sensing fabrics.

Keywords: wireless sensor network, federation, fabrics, resource specification, ontology, experiment specification, GENI, KanseiGenie.

1 Introduction

Several edge networking testbeds have been realized during this decade, in part due to the recognition within the networking community that testbeds enable at-scale development and validation of next generation networks. The role of edge networks —and edge networking testbeds— is likely to only increase, given the growth of wireless networks of sensors, vehicles, mobile communicators, and the like. In this paper, we focus our attention on an emergent architecture for next generation Wireless Sensor Network (WSN) testbed federation.

WSN Testbeds. Many WSN testbeds are in use today, of which Kansei [3], Orbit [7], NetEye [4], and PeopleNet [8] are but a few. Two recent experimentation trends are worth noting: One, experiments are being repeated in multiple

T. Magedanz et al. (Eds.): TridentCom 2010, LNICST 46, pp. 155–165, 2011.

testbeds, to learn about (potentially substantial) variability of performance in different backgrounds, radio types, and size scales. And two, a number of experiments involve long running deployments—they often yield long lived sensing services—which in turn implies that testbeds are increasingly hosting concurrent experiments. These trends motivate the emergent requirement that testbeds need to be *federations of programmable fabrics*.

By programmable WSN fabrics, we mean that individual sensor arrays offer not just resources on which programs can be executed, they also provide network abstractions for simplifying WSN application development and operation. Examples include APIs for scheduling tasks, monitoring system health, and in-the-field programming and upgrade of applications, network components, and sensing components. Fabrics can also support and manage the concurrent operation of multiple applications. Figure 1 compares the traditional WSN model with the emerging fabric model of WSNs. By federated WSN testbeds, we mean multiple WSN testbeds that are loosely coordinated to support geographically and logically distinct resource sharing. A federation provides users with a convenient, uniform way of discovering and tasking desired WSN resources.

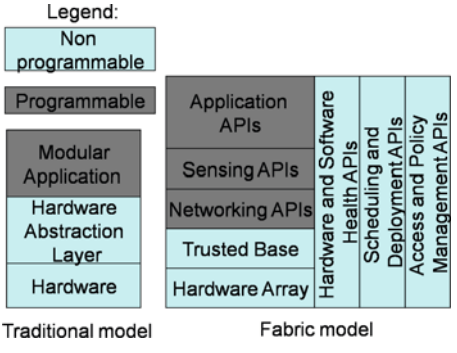


Fig. 1. Traditional and Fabric model

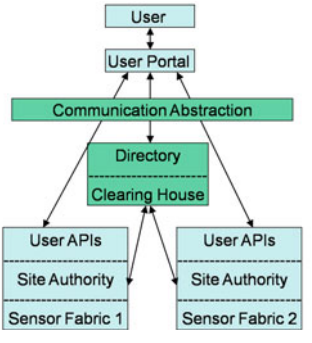


Fig. 2. Federated fabric/GENI model

GENI. The Global Environment for Network Innovation project [2] concretely illustrates an architecture where WSN fabrics are a key component. GENI is a next-generation experimental network research infrastructure currently in its development phase. It includes support for control and programming of resources that span facilities with next-generation fiber optics and switches, high-speed routers, city-wide experimental urban radio networks, high-end computational clusters, and sensor grids. It intends to support large numbers of users and large and simultaneous experiments with extensive instrumentation designed to make it easy to collect, analyze, and share real measurements and to test load conditions that match those of current or projected internet usage.

Figure 2 depicts the GENI architecture from a usage perspective. In a nutshell, GENI consists of three entities: Researchers, Clearinghouses and Sites (aka resource aggregates). The Clearinghouse keeps track of the authenticated users,

resource aggregates, slices, and reservations in the federation. A Researcher (interacting typically via a specially designed Portal) queries a Clearinghouse for the set of available resources at one or more Sites and requests reservations for those resources that she requires. To run an experiment, she configures the resources allocated to her slice, which is a virtual container for the reserved resource, and controls her slice through well-defined interfaces.

Overview of the paper. A federation of WSN fabric testbeds needs to address two core issues: One, an efficient and flexible method for resource description, discovery and reservation. And two, a convenient, uniform way of tasking and utilizing federation resources. While standardizing resource descriptions would simplify addressing these two issues, we find that the diversity of sensor characteristics and the lack of a compelling standard model for describing wireless networks complicate the federation of WSN fabrics.

In this paper, we propose a software architecture that we call KanseiGenie for federating WSN fabrics that is compatible with GENI. KanseiGenie is based on the position that, on one hand, different WSN fabric aggregates can advertise resources based on different resource ontologies. On the other hand, users can obtain uniform experimentation support from a portal supporting a given federation. Central to the KanseiGenie architecture is a mapping between a uniform experiment specification and a non-uniform resource specification, which is handled by the portal.

The rest of the paper is organized as follows: In Section 2, we detail the requirements of each actor in a WSN federation. In Section 3, we discuss the need for Resource and Experiment Specification in federations and their design challenges. Then, we present the KanseiGenie architecture and its implementation in Section 4, and make concluding remarks in Section 5.

2 Requirements of Federated WSN Fabrics

As explained in Section 1, the federated WSN fabric model distinguishes three actors: the Site that owns WSN aggregate resources, the Researcher who deploys applications via a Portal, and the Clearinghouse (CH) that enables discovery, management, and allocation of resource. In this section, we analyze actor-specific requirements, to make user experimentation easy, repeatable, and verifiable.

2.1 Clearinghouse Requirements

Broadly speaking, a Clearinghouse has two functions: One, identification and authentication of various actors in the system (the details of which are out of the scope of this paper and hence will not be discussed here); And two, resource discovery and allocation using a resource description language. The resource description language should be feature rich and extensible to capture the underlying heterogeneity of WSN fabrics and their constraints. A CH design should be efficient and robust to manage multiple Site Resource Specifications, large number of resources and their lease states. In a global infrastructure such

as GENI, a CH might have work in both master-slave and peer-to-peer modes with others CHs. A Portal/Researcher might request resources from multiple CHs directly or a single one which in turn communicates with other CHs.

2.2 Site Requirements

To support federated experimentation, a Site needs to support sliceability, programmability, experimentation services for the resources it controls. In the GENI model of experimentation, each Researcher owns a virtual container, aka a Slice, to which resources can be added or removed, and experiments deployed. It follows that a Researcher should have secure intra-slice communication that is isolated from other slices. To enable multiple slices to co-exist on the same fabric, sliceability may require Sites to virtualize resources both at the node and network level. For example, memory, processing time, or communication channels on the same device/network may have to be shared between slices. The challenge in virtualization is to provide as much control to the users (as low in the network stack as is possible) while retaining the ability to share and safely recover the resource. Virtualization in WSN fabrics is nontrivial, sometimes even impossible, given the limited resource on sensor nodes and the interference caused by concurrent wireless communications. Sites are also expected to provide programming services that reliably deploys applications composed by Researchers on the WSN fabric, testbed status and experiment execution monitoring, logging, trace data injection, and workflow control services.

2.3 Portal Requirements

A Portal in our architecture is a way to provide the user with a standard set of tools that can be used to exploit the federated resources. We believe that the Portal has an important role in networking sub-domains like WSN, where a high level of domain knowledge is needed to exploit the resources efficiently. A Portal needs to provide an uniform resource utilization framework. The framework should also allow the Researcher to select one or more of the standard User Services provided by the Site/Portal to be instrumented on the slice. This is challenging since the federation may consist of fabrics with a great variety of available platforms, sensors, radios, operating systems and libraries. For instance, while some sensor platforms such as mica family and TelosB are programmed on bare metal, others such as iMote2, Sunspots, and Stargates host their own operating system. The execution environments in these platforms vary from a simple file download and flash reprogramming, to command line interfaces and virtual machines.

3 Specification Languages

To use WSN fabrics, a Researcher queries the CH to discover what resources are available, selects a set of resources and obtains a lease for them, and configures the resources and User Services needed for the experiment. Each of these steps

needs a flexible, feature-rich, and extensible language to convey the goals of the Researcher to the system. We refer to the language used to publish, query, request, and allocate resources as the RSpec language, and the language used to configure resources and script workflow as the ESpec language.

3.1 Resource Ontologies for Resource Description

Resources available at multiple Sites are usually different from each other. Although describing resources in terms of a taxonomical flat-schema could hide minor differences by shoe-horning similar resources into the same category, building an exhaustive schema for WSNs, which will be conformed to by every Site, Clearinghouse, and Portal, is both difficult and sometimes impossible. The current GENI proposal [1] mitigates this problem somewhat with a partial standardization approach, a uniform core RSpec and multiple domain-specific extensions. Although this solution in principle is better than a single standardized RSpec, agreement on the extensions is difficult in domains like WSNs. Roscoe in [13] lays out the drawbacks of this approach. A key challenge in using a standardized specification is anticipating in advance all of the possible things that must be expressible. He argues that the resource request instead should be viewed as a *constraint satisfaction problem* and not a simple database query. For example, in a wireless network, while a node and channel might be two separate resources, it is usually impossible to allocate just the node but not the associated channel, or vice versa. It is better to explicitly capture such dependencies and relationships in the RSpec which would lead to a better resource allocation.

Thus a RSpec language should be able to specify constraints and handle multiple decentralized specifications. One way of doing this is to represent resources in terms of ontologies, using a resource description language such as NDL [5], which could describe not just the resources but the also constraints. With this approach, in order to communicate, two entities need only to agree on the language in which resources are specified, but not on the ontology.

Using Multiple Resource Ontologies for WSN Resources. Our primary motivation to support multiple ontologies in WSN federation is the difficulty in specifying things *uniquely*. We offer two examples cases. The first pertains to definitions, or rather the lack of it in the WSN space. There is no agreements on even simple terms like *nodes* and *sensors*. More over, defining something as either an allocatable *entity* or as its *property* could differ significant between Sites, depending on what platforms they provide. For example, is a channel an allocatable resource or simply a property of a radio?

Our second example deals with the difficulty of specifying wireless network topologies. A network topology in a WSN can be specified either implicitly or explicitly. An example of implicit specification would be specifying the transmit power level of each node. An example of explicit specification would be specifying the number and/or the list of neighbors for each node and letting the fabric choose the transmit power level and other radio parameters. A Site provider could choose one or other while defining the Site ontology, depending on the

hardware platform. One cannot force a Site provider to adopt an ontology which is not compatible with the hardware.

A question that directly follows from our argument for multiple ontologies is, why not simply use a union of them? This approach could result in RSpecs that cannot be instantiated by the Sites. For example, consider the two ways of (as discussed above) specifying a network topology; Even if a hardware could support both specifications, a single RSpec, which uses both implicit and explicit specification cannot be instantiated by the Site. Thus the union approach could lead to the risk of producing consistent RSpecs.

Given the intense debate in the WSN community, forcing Sites to use a single ontology is likely to throttle innovation and will result in a needlessly bulky ontology that is not easily extensible. Since most Researchers are expected to interact only through a Portal (or two) of their choice, we envision that Portals will serve as the unifying agent for resource specifications. Since all Sites will use the same language for their descriptions, the Portal can map the different ontologies used by the Sites to a single ontology to be used by Researchers. We note that there are several extant techniques and tools to map and align ontologies [9,12].

3.2 Experiment Specification and Work Flow Control

One of the roles of a Portal is to enable uniform experimentation across all federation fabrics. One way of satisfying this requirement is by providing an Experiment Specification (ESpec) language that enables Researchers to configure slices in a generic manner. Intuitively, besides the resource that the experiment is to be run on, an ESpec should also include the selection of User Services that is required by the experiment. WSN applications typically run in multiple well defined phases, with each phase involving a possibly different configuration. In addition to declarative elements, the experiment specification language also includes procedural descriptions (or workflow elements). This enables iterative experimentation, where a researcher programs repeated experiments, and the configuration of each experiment depends on the outcome of the previous ones. ESpec thus provides Researchers with a flexible and feature-rich way of interacting with resources, rather than just a GUI or a command line interface. They become particularly relevant for future scenarios where applications will primarily involve machine-to-machine, as opposed to human-to-machine, interaction. The idea then is to standardize the Experiment Specification language and not the format of interaction.

4 KanseiGenie

KanseiGenie is a refactoring of the Kansei testbed, to support a GENI compatible federation of geographically separated Sites, each hosting one or more WSN fabric arrays. Each sensor device is represented as a Component that defines a uniform set of interfaces for managing that device. An Aggregate contains a set of Components of the same type and provides control over the set. (In WSN

experiments, Researchers normally interact with fabric arrays through the aggregate interface rather than individual component interfaces). An Aggregate also provides other internal APIs needed for inter-component interactions.

4.1 Architecture and Implementation

In keeping with the GENI architecture, KanseiGenie consists of actors for a Site, a Clearinghouse, and a Portal. The current implementation of federation consists of a Site at The Ohio State University, which has four different sensor fabric arrays, and a Site at Wayne State University, which has two different sensor fabric arrays. The Sites and the Portal (which is hosted at Ohio State) run the KanseiGenie software developed at Ohio State. One of the Clearinghouse functions, namely resource management, is implemented using ORCA [6].

KanseiGenie Site. A KanseiGenie Site has four components: Aggregate of Aggregate Manager (AAM), the Web Service Layer (WSL), the individual Component Managers (CM) for each device type, and the Orca Site Authority module.

Aggregate of Aggregate Manager. Given that each fabric array is an aggregate, the KanseiGenie Site Authority (SA) is conceptually an Aggregate of Aggregate Managers that provides access to all the arrays. AAM is responsible for implementing the fabric APIs. AAM provides an AM interface for each sensor array through parameterization. Externally, AAM (i) administers usage of the resource provided by the Site according to local resource management policies, (ii) provides the interface through which the SA advertises its shared resource to one or more authenticated CHs and, (iii) provides a programming interface through which Researcher (via the Portal) can schedule, configure, deploy, monitor and analyze their experiments. Internally, the AAM provides mechanisms for inter-aggregate communications and coordination. The fabric APIs provided by an AM are organized into the four functional planes, namely, Resource/Slice management APIs, Experimentation APIs, Operation & management APIs, and Instrumentation & Measurement APIs.

Web Service Layer. WSL provides a wrapper for AAM and acts as a single-point external interface for the KanseiGenie SA. The WSL layer provides a programmatic, standards-based interface to the AAM. We utilize the Enterprise Java Bean framework to wrap the four functional GENI planes.

Component Manager. Each sensor device in KanseiGenie has its own Manager (although for some primitive devices such as motes the Manager is itself implemented on other more capable devices). The Component Manager implements the same APIs as that of AAM and is responsible for executing the APIs on the individual devices. Currently KanseiGenie supports Linux-based PCs/Laptops (Redhat and Ubuntu), Stargates, TelosB, and XSMs. CMs for Imote2 and SunSpots are under development.

KanseiGenie Portal. The Portal contains a suite of tools for the life cycle of an experiment. It provides an easy interface for experiment specification; at present, this is a user-friendly GUI; a user programmable interface is under development. It automates tasks for resource specification creation, requesting, and subsequent deploying of the experiment, for all Sites in the federation. It uses the Orca Slice Manager (explained below), to reserve resources requested by the Researcher. Once the reservation is done, it interacts with the AAM web services interface to configure and run experiments. Of course, a Researcher could directly program against the AAM web interfaces to gain more fine-grained control of experiments, i.e., write his own portal as need be. The Portal is implemented using the PHP programming language.

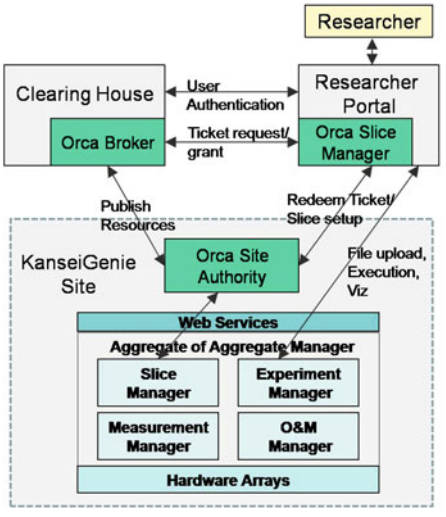


Fig. 3. KanseiGenie Architecture

KanseiGenie Clearinghouse. CH has two main functions; (i) Resource management: CHs are responsible for managing the resources on behalf of Sites. They keep track of resources and their leases. We use ORCA [6] (see below for more details) Resource Broker to implement our CH. (ii) Identity, Authentication and Trust: CH is also responsible for maintaining the overall security of the system. They authenticate/issue credentials for Sites, Portals and Researchers. In a federation, CHs implement trust-chaining to authenticate Researchers and Brokers from other domains. KanseiGenie, consistent with the GENI/ORCA effort, plans to use Shibboleth [10] for this purpose.

ORCA-based Resource Management System. ORCA consists of 3 sub-entities, each one correspondingly embedded into the three KanseiGenie actors (Portal, Site and Clearinghouse) respectively, to implement the resource management function. (i) The ORCA Slice Manager interacts with the Researcher and gets the resource request, forwards it to the ORCA Broker and gets the lease for the resources. Once a lease is received, the Slice Manager, forwards it to the Site Authority to redeem the lease. (ii) The ORCA Site Authority keeps inventory of all the resources that need to be managed. It delegates these resources to one or more Brokers, which in turn lease the resources to Researchers through the Slice Manager. (iii) The ORCA Broker keeps track of the resources delegated by various Site Authorities. It leases resources (if free) to the ORCA Slice Manager on request. A number of different allocation policies can be implemented using a policy plug-in.

4.2 Portal-Based Federation in KanseiGenie

Apart from being the single point of access for a Researcher, the Portal plays an important role in KanseiGenie federation architecture. The Portal has three important functions in federation, namely Resource Specification Mapping, Experiment Specification Mapping, and Federated Slice Stitching. KanseiGenie has thus far chosen the Portal as the main federating agent in the system. This design suits the view that a Portal realizes *application domain specific support*, and that for different domains, different Portals may suit. In other words, we view the KanseiGenie Portal as suitable for WSN experiments (and perhaps only some classes of WSN experiments) as opposed to sufficing for all GENI-Researcher needs.

In this view, Clearinghouses are treated as being generic rather than domain specific. Roles which are less domain specific, e.g. embedding Resource Specifications or slice stitching, can be moved from Portals to CHs (or even to Sites) assuming the method of stitching desired is communicated to them. Now, should CHs evolve to become domain specific, they may import more roles from Portals. Taken to the extreme, this would suggest that a top level CH be directly or indirectly capable of unifying all resources in GENI.

Resource Specification Mapping. As explained in Section 3 our position is that Sites may use their own Resource Specification dialects (ontologies). To provide a unified experience to the Researchers, we put the onus of interpreting multiple ontologies on the Portal. The Portal discovers resources from multiple sites, understands the Resource Specification ontologies and remaps the different ontologies into a unified ontology at the Portal. The current research [12,11] suggest that this remapping of ontologies can be done automatically with very high probabilities for related ontologies. However, we do the mapping manually since it is an one-time operation per Site. After a resource request is created (in the Portal ontology), it first gets *translated* into the individual Site's ontology and then sent to the CH to obtain the lease.

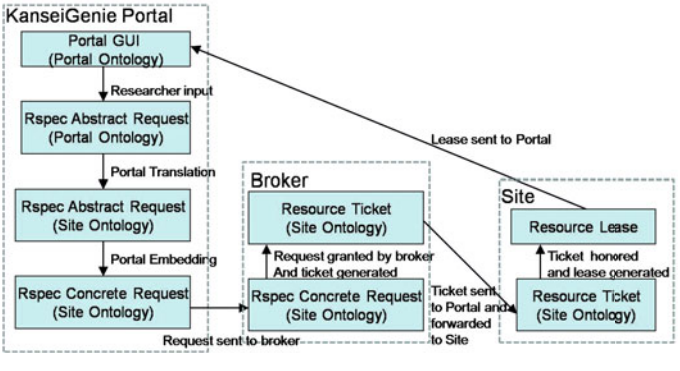


Fig. 4. Resource Specification mapping, translation and lease generation

In the current implementation, the Portal is also responsible for RSpec *embedding*), i.e. the embedding of an abstract RSpec into a concrete RSpec. It is often convenient for a Researcher to request networked resources in an abstract manner. For instance, requesting a 5-by-5 connected grid with 90% link delivery radio is much easier than pinpointing specific sensor devices that match the required topology. Since the resources published at the CHs are specified concretely, a Portal needs to convert or embed (many be with the help of the Site) the abstract RSpec into an concrete RSpec. The RSpec generation, translation, embedding, ticketing, and lease generation process is shown in Figure 4.

Experiment Specification Mapping. Researchers use the ESpec language to script an ESpec which is fabric/platfrom neutral. The Portal maps the ESpec created onto Site/fabric specific experiment manager APIs. Thus, a Researcher may "stitch $fabric_1$ *slice* and $fabric_2$ *slice*", "inject data on $fabric_1$ *slice* from $file_1$", etc., without worrying about the details of $fabric_1$ *slice* and $fabric_2$ *slice*; She may likewise repeat the same experiment on different fabric slices easily. In KanseiGenie, we attempt to use the same APIs for different aggregates whenever possible; nevertheless, whenever the notion of a service (say logging on a Virtual Machine fabric versus a Mote fabric) is different between fabrics, the complexity of mapping the configuration onto the corresponding APIs is handled by the Portal.

Federated Slice Stitching. When conducting a federated experiment, a Researcher expects seamless communication between the resources in the federated slice, which means that sub-slices from different Sites needs to be stitched together to form a federated slice. While individual Sites provide the stitching services as part of the AAM, the Portal possesses the knowledge for implementing stitching (such as VLAN numbers, IP addresses, ports, web URLs, etc.). Note that multiple types of stitching might be needed (and possible) depending on the sensing fabrics involved and their capabilities, e.g., it is easy to stitch a federated slice consisting exclusively of virtual machines connected by wired virtual LANs, while it is much harder to stitch two wireless network slices to create a single federated wireless slices.

5 Conclusion

In this paper, we described the KanseiGenie software architecture for federated WSN fabrics. We argued for letting WSN fabrics choose their own Resource Specification ontologies and resolve the diversity in resource specifications by letting the Portal map the Site specific description to a local ontology with which the Researchers can interact. We also illustrated the need for a Experiment Specification language to enable Researchers to uniformly interact with multiple WSN fabrics in the federation; Experiment Specifications further enable scripted experimentation and complex staging between fabrics. As KanseiGenie grows to accommodate other sites, it remains to be seen whether a need to develop other Portals will emerge or some of the federation functionality in the Portal will migrate to Clearinghouses/Sites.

Finally, we share here a few of the lessons from our experience with federated testbed design. (i) RSpec and ESpec design, their completeness and adaptability is very important for the growth and research in testbed federations. (ii) A powerful Portal, such as in our design, is not always a necessity, but definitely a blessing in highly domain-specific federations such as WSN. The domain knowledge needed to exploit WSNs is very high and a fully functional Portal is a very useful tool for Researchers. (iii) The design and implementation of the Aggregate and Component Managers is a non-trivial aspect of WSN fabric design. The AMs needs to make the least assumptions about platform support, while simultaneously providing non-trivial guarantees to the Researcher. (iv) A significant amount of work is involved in maintaining a healthy testbed. A systematic approach to fault-detection and correction, and an autonomous health monitoring system are an essential part of any long living WSN infrastructure.

References

1. GENI draft, `http://groups.geni.net/geni/attachment/wiki/GEC2/TF_RSpec.pdf`
2. GENI: Global environment for network innovation, `http://www.geni.net`
3. Kansei wireless sensor testbed, `http://kansei.cse.ohio-state.edu`
4. NetEye wireless sensor testbed, `http://neteye.cs.wayne.edu`
5. Network description language, `http://www.science.uva.nl/research/air/projects/ndl/`
6. Open resource control architecture, `https://geni-orca.renci.org/`
7. Orbit network testbed, `http://www.orbit-lab.org/`
8. PeopleNet mobility testbed, `http://peoplenet.cse.ohio-state.edu`
9. Protege: Ontology editor-knowledge framework, `http://protege.stanford.edu/`
10. Shibboleth: Software package for identity and authorization management, `http://shibboleth.internet2.edu/`
11. Johnson, H.L., Cohen, K.B., Hunter, L.: A fault model for ontology mapping, alignment, and linking systems. In: Pacific Symposium on Biocomputing (2007)
12. Noy, N.F., Musen, M.A.: Prompt: Algorithm and tool for automated ontology merging and alignment. In: Proc. of the 7th National Conference on Artificial Intelligence, pp. 450–455. AAAI Press/The MIT Press (2000)
13. Roscoe, T.: Languages not Formats: Tackling network heterogeneity head-on. In: Proc. of 3rd the Workshop on Future Directions in Distributed Computing (2007)

OpenEPC: A Technical Infrastructure for Early Prototyping of NGMN Testbeds

Marius Corici, Fabricio Gouveia, Thomas Magedanz, and Dragos Vingarzan

Fraunhofer Institute FOKUS
Kaiserin-Augusta-Allee 31, 10589, Berlin, Germany
{marius-iulian.corici,fabricio.gouveia,thomas.magedanz,
dragos.vingarzan}@fokus.fraunhofer.de

Abstract. The challenging and ever increasing requirements of future applications demand new concepts for better control and management of network resources. The Third Generation Partnership Project (3GPP) introduced in their latest specifications the Evolved Packet Core (EPC) architecture for transparently unifying the parameters of different technologies, like the UMTS, WLAN, non-3GPP access technologies and a future Evolved Radio Access Network, called Long Term Evolution (LTE), with the use of multiple application platforms such as IP Multimedia Subsystem (IMS) and the Internet. This paper describes a testbed implementation of the Evolved Packet Core named OpenEPC which provides a reference implementation of 3GPP's EPC developed by the Fraunhofer Institute FOKUS. OpenEPC is a set of software components offering advanced IP mobility schemes, policy-based QoS control, and integration with different application platforms in converging network environments. This initiative, in addition to fostering research and development, enables academic and industry researchers to rapidly realize state-of-the-art NGMN infrastructures and application testbeds.

Keywords: Open EPC, Testing, Testbeds, Next Generation Mobile Network, Operator core network, EPC.

1 Introduction

The Evolved Packet Core (EPC) ([1], [2]), formally known as Service Architecture Evolution (SAE), was developed from the necessity to converge different types of networks for the different services which are provided on top, thus having as main goal the transparency of access technology features to the core network of the service provider. In the next years, not only the integration of these accesses and services networks is envisaged, but also the adoption of a new wireless access technology: the UMTS Long Term Evolution (LTE). LTE brings new requirements, and therefore enhancements to the current architectures should also take place in order to achieve better performance. Following these principles, EPC is designed to fully support and seamlessly integrate in the core network LTE and other future technologies. EPC is a converged network architecture that enables the efficient connection of

T. Magedanz et al. (Eds.): TridentCom 2010, LNICST 46, pp. 166–175, 2011.

the mobile devices to the network and their communication with different service platforms like IMS or Internet. It guarantees the required resources are delivered with seamless service continuity among accesses. This functionality offers to the service platforms an independent management of the various wired and wireless networks following the indications for the Next Generation Mobile Networks (NGMN) Alliance [3].

This paper describes the OpenEPC testbed that is being developed by the Fraunhofer Institute FOKUS. OpenEPC aims to facilitate research and development of EPC, such as new access network integration, handover optimizations, EPC functional extensions, and new NGMN service prototyping. The implementation of the OpenEPC takes advantage of the previous successful implementation of the Open IMS Core ([11], [12]) as a prototypical open source implementation of the IMS core elements that started out as an internal project at the Fraunhofer Institute FOKUS in Berlin. Since the launch of that testbed, the solution has successfully worked as a reference implementation in the Open IMS Playground [7], a vendor and operator independent IMS test-bed providing coaching, consulting and proof of concept implementations as well as performance and interoperability tests for major vendors and fixed and mobile network operators.

2 OpenEPC

In comparison with the above related initiatives on the EPC/LTE, the OpenEPC [6] initiative targets not only trials and research on this topic, but also offers the platform for an easy integration of various research use cases. One of the main advantages of the OpenEPC, derived directly from the goals of EPC standards, is that it represents a simplified flat-IP architecture, with a single core network, able to integrate multiple access networks, with a clear separation between the data and the control plane, which offers an easy interaction with various service platforms and network independence to the services themselves. OpenEPC implements a set of standard components permitting the cost-efficient creation of NGMN testbeds. These testbeds are then used to prototype, measure, monitor, test, and perform research and development for NGMNs. It enables a quick start on the heart of emerging NGMNs, namely the EPC architecture, because of its standards conformance and of its configurability to match different needs for testing and extensibility. Open EPC aims to provide its users with a basic understanding and practical hands on experience with EPC, as well as conformance testing and proof-of-concept studies. With OpenEPC it is possible to develop functional extensions of individual and/or multiple EPC components and new NGMN showcases. In addition, OpenEPC supports research and development into challenging aspects of upcoming NGMN infrastructures and services, like the integration of new fixed and wireless access technologies, new approaches to mobility, QoS and security, optimizations of the architecture, design of new seamless wireless applications and the investigation of new business models in the NGMN world.

Having as basis the standards of 3GPP and containing already different optimizations to the available standards which follow the NGMN requirements, OpenEPC addresses to:

- Operators are using OpenEPC to prepare for the upcoming all-IP NGN and NGMN world and have an open and vendor independent testbed infrastructure.
- Manufacturers of individual EPC components are using OpenEPC to test their products in concert with a standards based NGMN environment.
- Manufacturers of full EPC platforms are using OpenEPC for practical research on new concepts and protocols in an easier to maintain platform.
- Application developers are using OpenEPC to certify that their applications work in NGMNs and take advantage of the functional capabilities offered by EPC to the applications domains.

Research institutions and universities are using OpenEPC for practical NGMN research, including usage of OpenEPC as black box for applications prototyping, or extending individual or multiple EPC components and/or developing new EPC components and protocols to provide new capabilities for integrating new networks or enabling new applications.

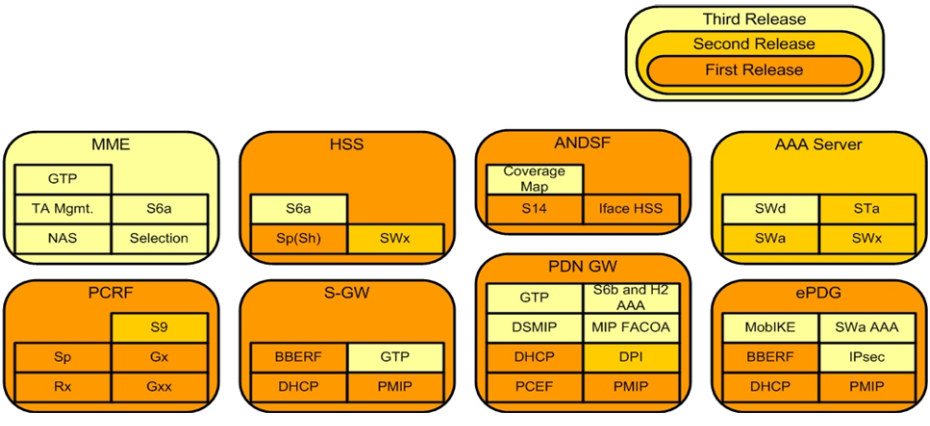


Fig. 1. OpenEPC Architecture

The current version of the OpenEPC is based on the 3GPP Release 8 standards offering the bases for research in the area of integrating different access technologies and different service platforms. Its features provide a policy based implementation for access network discovery and selection, attachment to different access networks, mobility and QoS management based on the network configuration and subscriber profiles. OpenEPC has been successfully tested for IPv4 and IPv6; the remaining heterogeneous IPv4/IPv6 scenarios will be added in the future. All the central software components of OpenEPC have been developed in C, based on a new high modular and configurable software framework, designed especially for the required EPC protocols and architectures (internal code-name Wharf). This allows the easy configuration and customization of components by selecting the functionality

required from a set of modules. With this approach, OpenEPC supports customized components which can dynamically include/exclude or combine functionality and interfaces. For instance, a deployment could include an S-Gw which performs BBERF functionality of the Policy and Charging Control architecture or a PCRF which does not include S9 interface for roaming scenarios. Such customizations are supported by simply modifying the configuration file and enabling or disabling such modules, each module being also capable of self-configuration by detecting what additional functionality it can employ. The components that require dynamic configuration or provisioning of parameters (like PCRF, HSS or ANDSF) offer a web-based front-end, while the state information is stored in a database back-end. In complete configurations a common front-end can be used to configure all components.

2.1 OpenEPC Components

The current first version contains the following entities, as depicted in Figure 1: HSS, ANDSF, PCRF, PDN-GW including PCEF and PMIP mobility, S-Gw and ePDG including BBERF functionality and PMIP functionality.

OpenEPC HSS. It provides storage and provisioning enablers for the subscriber profile, as defined in the EPC specification, by extending the IMS subscriber profiles already supported. It also performs the non-standard Subscriber Profile Repository (SPR) functionality which sends upon request the subscriber profile to the PCRF, ANDSF or any other authorized EPC component which requires subscription data to function. For this part it offers the not yet standardized interface Sp, which permits the support of personalization through subscriber profile of the policy and charging control functionality, included in the specifications, but still not developed. Added to that, through the same interface, the HSS is connected to the ANDSF to provide personalization for the mobility policies transmitted to the UE. The definition of the OpenEPC interface has been developed based on the Open IMS Core Sh interface, used between HSS and an IMS Application Server, with the addition of specific Data References and Attribute Value Pairs (AVPs). Concretely, the Data-Reference AVP has been extended to include parameters which are fetched from HSS and are used for policy control and for access network discovery and selection.

OpenEPC PCRF. It allows the application functions to request resources and priority treatment from the core network for one or more data flows which pertain to the application exchanged between the UE and the network. The Rx interface is used to convey the information received from the applications as part of the active profile of the UE. Also it connects to the HSS using the Sp interface, as to make the policy decisions according to the subscription profile of the users. These decisions are then pushed to be enforced on the gateways of the specific accesses using the Gxx interface and to the PDN Gw using the Gx interface. In order to receive notifications for events, the PCRF subscribes to profile modifications to the HSS (over the Sp interface) and to bearer modifications to the PDN GW and the access network gateways (using Gx and Gxx interfaces). The behavior of the OpenEPC PCRF is controlled through policies which can be provided dynamically through the provisioning

front-end. The policy language used complies with OMA Policy Expression Language extended with the specific PCRF tags. The interfaces of the OpenEPC PCRF are separated into different modules without a very tight interconnection, which makes them separately or all-together deployable depending on the specific requirements of an OpenEPC testbed. In practice, modules like the PCRF can detect which other modules are available and as such deploy a less or more complex set-up.

OpenEPC PDN GW. It includes a Proxy Mobile IP (PMIP) stack, capable of acting on both IPv4 and IPv6 and configured as Local Mobility Anchor (LMA). It allocates the IP address of the UE at the initial attachment from the provisioned client IP address pools. For all the subsequent attachments, the same IP address is transmitted. It also maintains the mobility tunnels and enforces the forwarding of the data packets to the access network specific gateway for download traffic and to the correspondent addresses of the other entities involved in the data sessions for the upload traffic. It also supports the Policy and Charging Enforcement Function (PCEF) as a module. This enables the allocation of the default QoS values (upon attachment of a new subscriber to the packed data network) and service and subscriber specific QoS (when a UE accesses a service). The enforcement of these rules is done through fully configurable and already established Linux network tools.

OpenEPC Serving GW. It includes the PMIP Mobility Access Gateway (MAG) functionality. It is also integrated with a modified version of the ISC-DHCP server, which offers support for both IPv4 and IPv6. The attachment of the UE to the EPC is detected at network level through a request for an IP address received by the DHCP server. During the tunnel establishment with the PDN GW the IP address to be allocated to the UE is received and it is then forwarded as the address to be transmitted by the DHCP server to the UE. The S-GW includes also a Bearer Binding and Event Reporting Function (BBERF) module for policy enforcement and event notifications related to the data traffic. It connects to the PCRF through the Gxx interface and receives QoS rules which are then enforced using standard Linux tools like iptables, traffic control etc.

OpenEPC ePDG. It includes the same functionality as the OpenEPC Serving Gateway, but with a special interest into non-3GPP access. For further releases IPsec connection to the UE will be deployed together with the EAP-AKA authentication interfaces, as to completely discriminate between the trusted and the un-trusted non 3GPP accesses [9].

OpenEPC ANDSF. It is based on several not yet standardized functions necessary to provide the demonstrative role of this entity in the EPC: the interface to the HSS for subscription profile fetching and the enabler to make decisions not only based on the location of the UE, but also on its required parameters as well as on the network's operational requirements. As standardized, it communicates to the UE to provide access networks available in the vicinity of the mobile device using the OMA DM communication mechanism and the Management Objects format of 3GPP [8].

Mobility Manager. For the UE, a minimal Mobility Manager was implemented which is able to retrieve data from the ANDSF in both PUSH and PULL modes, to make decisions on which access network to select for an initial and

subsequent attachment and to execute the afferent procedures, including handovers. The client platforms which are currently enjoying the largest deployments, like Windows™ and Linux for laptops/netbooks, or Android, Maemo 5 and Windows Mobile for smart-phones, are supported, with support for more extensible based on requirements. For deeper integration and more advanced demonstration of the EPC features, the Mobility Manager features an open interface, demonstrated for example by the integration with the MONSTER [10] client platform.

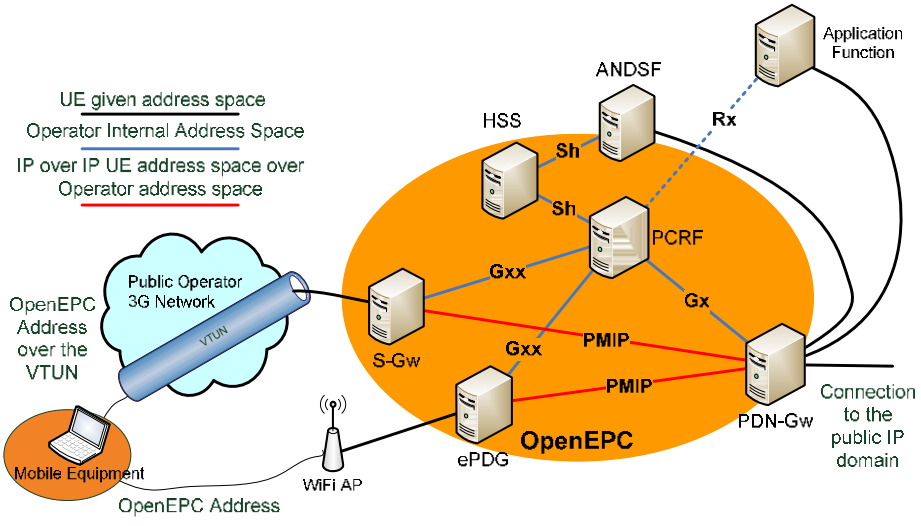


Fig. 2. OpenEPC Testbed

2.2 Interconnection with the Access Networks and Applications

As depicted in Figure 2, for the interconnection with the access networks, the OpenEPC uses the functionality described for the S-GW and the ePDG. Currently, the communication is sustained over a public 3G access network. A direct link layer connection is simulated through a Layer 2 tunnel (VTUN) which encapsulates all the link and network layer packets (e.g. DHCP, IP etc.) in UDP over IP packets. This special setup gives the opportunity to test the EPC procedures on a realistic environment, without interacting with the real access network provider and as such practically circumventing the high-costs usually associated with such testbeds. A local WiFi access network can also be easily deployed and connected to the OpenEPC ePDG, making the scenarios that can be demonstrated as realistic as possible for the testbed. This brings the total cost and complexity of running an EPC testbed to the minimum, while, of course, more realistic setups can be employed in case direct access to the SGSN/GGSN or eNodeB components if feasible.

OpenEPC provides inter-connection between the applications and services layer and the network layer. It provides IP connectivity far beyond the concept of an IP pipe by supporting extended features like QoS resource reservation, prioritization and information on events happening at the link and network level (e.g. the UE

lost connectivity or a handover to another access network occurred) upon request of the applications. These features are realized through two interfaces. The SGi interface is the IP interface from PDN-Gw to applications layer through which user data is sent. The Rx interface from PCRF to application layer is the signaling interface based on Diameter that allows the application layer to request special treatment to certain service flows and to get notifications upon events occurring in the access network. OpenEPC provides both interfaces and example functionality to demonstrate the interconnection to both Internet and the 3GPP IMS. The OpenEPC can be also used in conjunction with the IMS, providing for the IMS services QoS, IP mobility and security. The OpenIMSCore [11] provides the perfect extension to the OpenEPC in this area. IMS can be used for Voice over IP (VoIP) or other multimedia services. Independent of the implemented services, the P-CSCF of OpenIMSCore supports the Rx interface towards the PCRF and requests service authorization and resource reservation from IP connectivity layer. Moreover the OpenEPC platform can also be used to connect to **plain Internet**, in order to make use of the advantages of the communication between the applications and the EPC, providing a better control of network usage.

3 Proof of Concept Implementation

For showing the capabilities of the OpenEPC in the area of seamless mobility between heterogeneous access networks, advanced triggers for mobility, QoS decisions and enforcement, interconnection with the service delivery platforms and personalization of services multiple testing scenarios were implemented, from which the following were selected as the most representative. These are presented here as validation proofs of the OpenEPC platform, as they can be demonstrated in the current state.

Mobility Scenarios. In EPC, a network provider is able to restrict the access of a UE to specific access networks by modifying the subscription profile. This scenario presents an initial connectivity case in which one subscriber attaches to one access network, as exemplified in Figure 3, to the 3G network. After the initial attachment, it connects to the ANDSF and requests the default attachment policy. For this purpose, the ANDSF fetches the client's subscription profile from the HSS, which was modified during the inactive time the UE as to restrict it to connect to the 3G networks. Then the ANDSF evaluates inter-system policy in conjunction with the subscriber profiles and makes a mobility policy decision in order to find another suitable access network to maintain the data traffic. It finds that a WiFi access is available in the vicinity of the mobile device and it sends as a response to the UE with a policy indicating that an immediate handover to the WiFi is required, which the UE executes. The modification of the subscription profile can happen also during the service of the mobile device. In that case, the ANDSF receives a notification from the HSS containing the modifications of the profile and upon this trigger it makes the decision for mobility. The ANDSF alerts the UE in this case, which practically triggers an immediate policy PULL between the UE and the ANDSF. Using the new policies, the UE executes the handover procedures. The mobility may be triggered also by the loss of signal to the access network to which the UE

is connected to. In this case the UE either uses the policies of handover as they were previously transmitted by the ANDSF or requires new policies. It is to be noted that in OpenEPC the policy transmission to the mobile device can be either synchronous with the handover trigger or asynchronous based on the location change of the UE. In the second case, when the network or the UE notices a change in location new policies are either pushed or pulled to the UE.

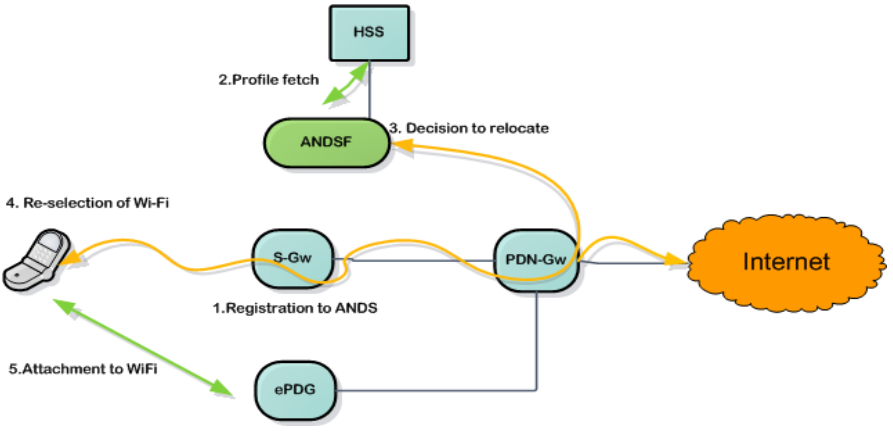


Fig. 3. Subscription Based Mobility

This scenario was tested with and without having a real-time video application established between the UE and a server providing the service. As specified by the PMIP standard, the same IP address was allocated to the two interfaces of the mobile device which made the service to be seamless to the user. This opens the possibility for the operators to deploy seamless services across multiple access technologies without having to extend the functionality of neither the mobile devices nor of the service platforms.

QoS Control Scenarios. In OpenEPC, the subscription profile can be used to provide different QoS levels for the same service For instance a list of prioritized services is provisioned, or different QoS profiles can be provisioned for different users [4], [5]. This allows the OpenEPC network provider to offer differentiated service levels according to subscriptions which may be bound with different charging schemes. An UE connected to one access network, for example to the 3G network as depicted in, initiates the establishment procedures with the Application Function (AF). After the session parameters are negotiated, the preferred QoS levels are indicated by the application function to the PCRF which fetches the subscription profile from the HSS. Based on this subscription profile it makes the policy based decision whether the user is allowed to use the negotiated level of resources. Then the decision is enforced on the PDN GW and on the access network gateway. During the testing process, a SIP based video service was used as a stub for services. The PCRF offered the required level of resources for one subscriber, but a smaller level for another one. For the subscriber which received the required level of resources the

service was available at a good quality of experience while for the other it deteriorated fast. If other mechanisms of detecting the service are available, like Deep Packet Inspection ones, in which the EPC detects the service, then the network operator may increase or decrease the quality depending on the service, the service provider and the momentary conditions in the access networks.

Service Adaptation to Network Context. Different wireless access technologies offer different capabilities to which the services have to be adapted. OpenEPC provides the connection between the access networks and the service layer which allows to inform the applications when the characteristics of the access network through which a UE communicates change due to a vertical handover. This allows the services to adapt to the change and to transmit its data according to the requirements of the UE in the new access network. In this OpenEPC scenario, a UE is attached to one access network; as depicted in Figure 4, it may be the 3G network. A service is initiated with an Application Function (AF) having a negotiated level of resources. The AF in this case subscribes to an access network type event for the specific UE, as the service may be received with different qualities depending on the access technology. A handover is triggered by the ANDSF, in the case of the first version of the OpenEPC, manually by the operator of the testbed. A handover indication is transmitted to the UE, which requests a handover to a WiFi access network. The UE attaches to the WiFi access network and the event of the new attachment is received by the PCRF. The PCRF informs the subscribed application function that the UE has changed the access network and receives in return the new parameters which are to be enforced for the target access network. The new QoS level is enforced on the gateways and the AF changes its parameters too and continues to provide the service uninterrupted.

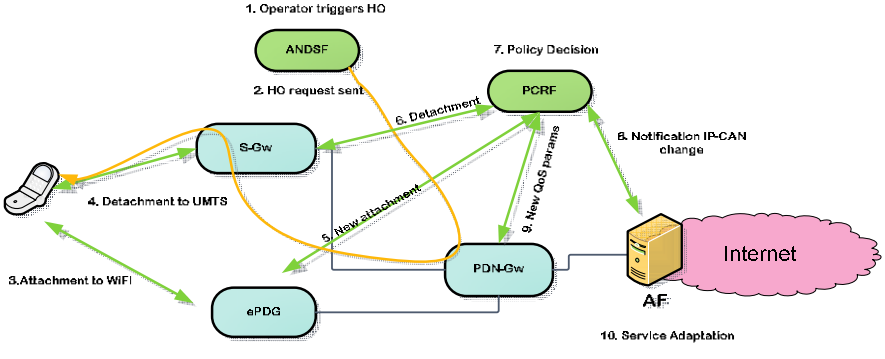


Fig. 4. Service Adaptation to the Network Environment

Internet Services over OpenEPC. OpenEPC is prepared to offer more than just best-effort IP pipes for any type of IP services. It can connect to any internet service delivery platform, exporting the same capabilities as given to IMS. As a testing scenario for the OpenEPC, an HTTP Proxy based on the Squid project and enabled with the Rx interface was used. Two mobile devices were used. One which uses the HTTP Proxy and one that does not. As a service, a public third party TV broadcast from the

Internet was used. For the user which connected over the HTTP proxy the service received the required quality while for the user which did not use the proxy, as the service remained transparent to the EPC, the resources usually allocated for the default communication were used. Then the PCRF makes the decision on which level of resources are to be allocated, based on the type of service, the subscription information and the momentary available resources in the access networks. This allows operators to dynamically discriminate between the different services which the UE are allowed to use depending also on the location which gives the possibility of introducing new charging schemes.

4 Conclusions and Further Work

The OpenEPC platform, here presented conforms on one side to the standards of 3GPP and it also provides different researched optimizations which enable the operators to easily test and to enhance their requirements for the device manufacturers. It offers also to the device manufacturers, the basis for further development of concepts and proof of concept demonstrations for a fast development of standards and new products. Continuing in these two directions, the OpenEPC will develop to further include more standards related to the Next Generation Mobile Network Core architecture with a high regard for new service platforms and access networks. It will also contain new concepts regarding Self Organizing Networks, more flat network architectures and Future Internet research activities.

References

1. 3GPP TS 23.401, General Packet Radio Service (GPRS) enhancements for Evolved Universal Terrestrial Radio Access Network (E-UTRAN) access
2. 3GPP TS 23.402, Architecture enhancements for non-3GPP accesses
3. Next Generation Mobile Networks (NGMN), http://www.ngmn.org
4. 3GPP TS 29.212, Policy and Charging Control over Gx reference point
5. 3GPP TS 29.213, Policy and Charging Control Signalling Flows and Quality of Service (QoS) Parameter Mapping
6. OpenEPC – Open Evolved Packet Core, http://www.openepc.net/en/openepc/index.html
7. The FOKUS Open IMS Playground, http://www.fokus.fraunhofer.de/ims
8. 3GPP TS 24.312, Access Network Discovery and Selection Function (ANDSF) Management Object (MO)
9. TS 36.401 Evolved Universal Terrestrial Access Network (E-UTRAN); Architecture Description
10. myMONSTER Toolkit, http://www.mymonster.org/
11. Open IMS Core, http://www.openimscore.org/
12. Vingarzan, D., Weik, P., Magedanz, T.: Development of an Open Source IMS Core for emerging IMS test-beds, academia and beyond. Journal for Mobile Multimedia (2006)
13. 3GPP TS 29.272, Evolved Packet System (EPS); Mobility Management Entity (MME) and Serving GPRS Support Node (SGSN) related interfaces based on Diameter protocol

FIRST@PC MediaX: A Service-Oriented Testbed for Realistic Media Networking Experiments

Sang Woo Han, Bum Hyun Baek, and JongWon Kim

Networked Media Lab.,
Dept. of Information and Communications,
Gwangju Institute of Science and Technology (GIST),
261 Cheomdan-gwagiro, Buk-gu, Gwangju, 500-712, Korea
{swhan,bhbaek,jongwon}@gist.ac.kr

Abstract. As an effort to devise and experiment diverse types of media-oriented service compositions supported by Future Internet infrastructure, this paper introduces an attempt to build a service-oriented testbed named as FIRST@PC (Future Internet Research on Sustainable Testbed based on PC) MediaX (Media eXperiment). Following the SOA (service oriented architecture) paradigm, FIRST@PC MediaX targets a flexible and cost-effective testing environment where media-oriented service compositions are flexibly realized on top of virtualized computing/networking resources. In this paper we will discuss on-going efforts on designing and building this testbed with several PC-based devices for media acquisition, media processing, display (networked tiled display), and networking. Specially, the preliminary implementation of agent-based software toolkit called as OMX (Open Media eXperiment for service composition) is explained and verified by testing a HD-media service scenario that combines multiple HD videos.

Keywords: Service-oriented testbed, service composition tool, media services, service composition experiment, and Future Internet.

1 Introduction

Ever-increasing demand for immersive media contents has raised flexible multimedia systems that understand widely dispersed media contents/tools and dynamically compose media-centric services with diverse computing/networking environments [1]. To realize the vision, several work accommodated the fundamental basis of SOA (service oriented architecture) [2] in building large-scale multimedia applications based on media-oriented service composition [3]. This paradigm made a broad impact to multimedia communities, which led to move from monolithic multimedia applications to more flexible component-based ones. Note that the service composition in SOA decomposes complex tasks into smaller independent entities (e.g., Web services), and then supports a flexible service composition in a variety of ways. Unlike the Web service composition, a

T. Magedanz et al. (Eds.): TridentCom 2010, LNICST 46, pp. 176–185, 2011.

composed media service should handle the support challenge of networking continuous data flows (e.g., audio/video streams) that have strict restrictions on timing and resources [3].

Testbeds are considered as a key tool to test semantical correctness and basic functioning of a new technological idea. These testbeds are running with a control framework that controls and manages testbed resources for user experiments. Testbeds additionally provide experimenters with tools and methods to build and execute their own experiments. For example, GENI Future Internet testbed effort [4] develops a suite of network research infrastructure with virtualized resource substrate, which encourages experimenters to easily develop new protocols and services using open, large-scale, and realistic experimental facility. PlanetLab, ProtoGENI, ORCA, and OMF (cOntrol and Management Framework) are current candidates of GENI control framework.

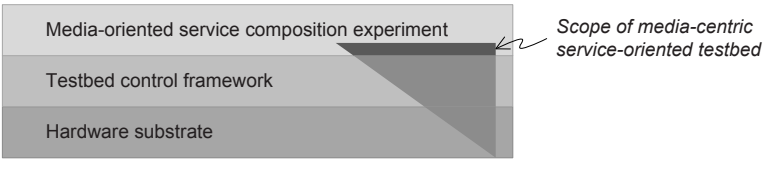


Fig. 1. Technical focus of service-oriented testbed

To facilitate the diverse experimental needs for media-centric service composition, as shown in Fig. 1, in this paper, we focus on a media-centric service-oriented testbed that carries out media-centric service composition experiments, by leveraging the support of testbed control framework (e.g., from GENI) and by specializing on tools and hardware for massive media processing and delivery. Like this, in order to flexibly devise and build new media-centric services, we attempt to build a service-oriented testbed named as FIRST@PC (Future Internet Research on Sustainable Testbed based on PC) MediaX (Media eXperiment)[1]. With this experimental testbed, we hope to incubate innovative and creative ideas for futuristic media-centric services. Key challenges are in extending existing testbeds further to support complicated service composition and in stably operating the developed testbed through new control and management tools for media-centric service composition, albeit with following features:

- *Flexible service composition:* Experimenters should be able to flexibly compose services based on the functional service dependency relationship among component services;
- *Dynamic service adaptation combined with monitoring:* The composed service should be able to adapt itself according to the monitored service status and resource utilization.

[1] In [5], a very early design for FIRST@PC testbed is presented by combining hardware-accelerated programmable networking [6] and virtualization [7] to support the experiments on media-oriented service composition.

The rest of this paper is organized as follows. Section 2 introduces basic concept of FIRST@PC testbed and key building blocks. Section 3 describes the building progress about FIRST@PC MediaX testbed with special attention to OMX (Open Media eXperiment for service composition) toolkit. After explaining the HD-media service composition experiment for verification in Sec. 4, we conclude this paper in Sec. 5.

2 FIRST@PC: Media-Centric Service-Oriented Testbed

In this section, we explain the basic idea about FIRST@PC testbed from conceptual illustration to key building blocks.

2.1 Testbed Conceptual Design

A conceptual illustration of FIRST@PC testbed (in short TB) is depicted in Fig. 2. Administrators and experimenters are accessing the computing/networking resource substrates in an aggregated form called *resource aggregation (RA)*. The controlled access to resource substrates are managed by *slice*

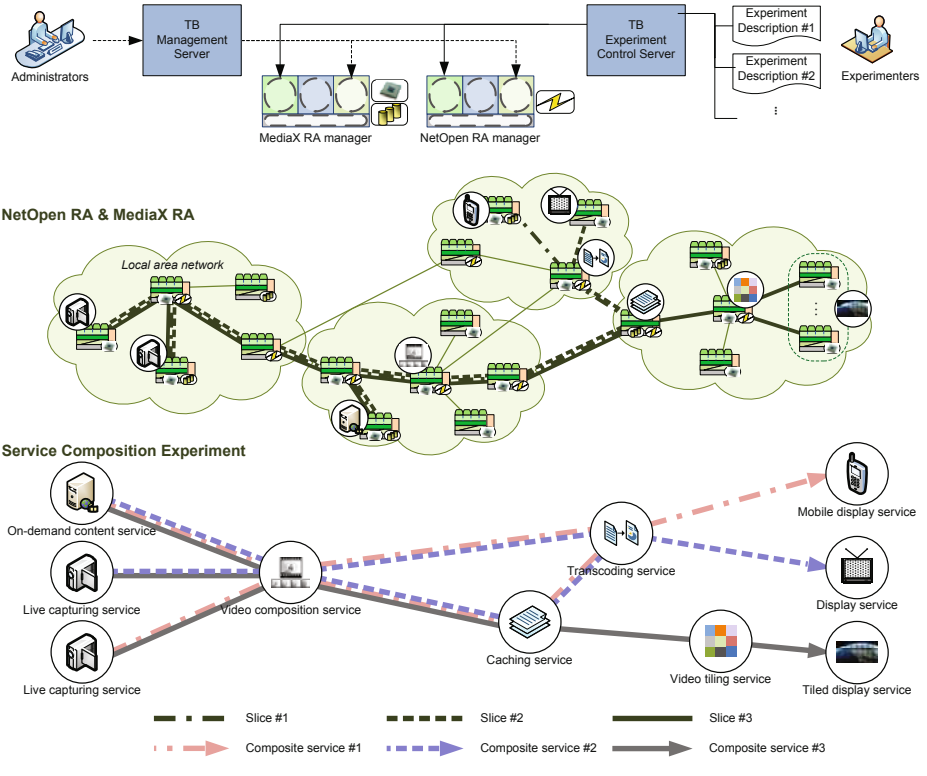


Fig. 2. A conceptual illustration of FIRST@PC testbed

control, which is actually coordinated among the corresponding RA managers, TB management server, and TB experiment control server. Two kinds of RAs are assumed: MediaX (for computing and storage) RA represented by two small CPU and storage icons and NetOpen (for networking) RA represented by communication link icon. *MediaX RA* prepares high-performance computing and GPU-based media processing acceleration capability with high-volume storage for diverse media-centric service composition experiment. *NetOpen RA* supports OpenFlow-compatible [8] hardware-accelerated (e.g., NetFPGA [6]) programable and virtualized networking capability. To make a service composition, an end-to-end slice is created where multiple nodes in both MediaX and NetOpen RAs are connected in a service path. Note that, in Fig. 2, three slices #1, #2, and #3 are illustrated to depict three connected composite services in a Internet-based broadcasting scenario.

2.2 Testbed Key Building Blocks

FIRST@PC testbed mainly comprises of two key building blocks: RAs, and coordination servers (RA managers, TB management server, and TB experiment control server).

Each *RA (resource aggregation)* represents a group of PC-based computing and/or networking resource substrates. Typical resource substrates include computing (e.g., CPU and GPU power), storage (e.g., memory and disk array), and networking (e.g., NetFPGA and wireless network interfaces) resources. A node should support *virtual nodes* that can take utilize selected portion of virtualized (hopefully isolated) resources. Note that virtual nodes are can be associated either with slice control for experiments or resource management for TB operation & management.

Among the coordination server, first, each RA manager manages his RA by allocating resources on behalf of individual nodes and helping the configuration of RA nodes. It also supports open but authorized access for resources according to the privilege of experimenters and administrators. The TB management server is responsible for operating and managing TB by involving with slice control, resource management, and resource monitoring. Finally, TB experiment control server enables experimenters to describe service composition experiments and supports service control and its status monitoring.

3 Preparing FIRST@PC MediaX Testbed for Experiments

In this paper, we focus on running experiments on media-centric service composition only with MediaX RA, temporarily setting aside the programmable and virtualized networking capability of NetOpen RA. After explaining how to conduct the media-centric service composition experiments, we discuss on-going realization of FIRST@PC MediaX testbed with special emphasis on the agent-based OMX toolkit for service composition experiments.

3.1 Target Service Composition Experiments

With the FIRST@PC MediaX testbed, we are currently working to realize the experiment on service mapping coordination based on a template-matching approach, as investigated in SpiderNet [9] and SeSCo [10]. Generally, for the media-centric service composition, the input media sources that end users want to receive are going through composition processes to be displayed in various user-defined presentation formats (e.g., layout including size and resolution, and visual effects). The service composition experiment matches component services with available resources of appropriate QoS characteristics (e.g., delay, jitter, loss, and playout continuity). This process is conducted by connecting component services into a composite service according to the service dependency graph and composition algorithm. The connection is actually made by binding the interfaces of component services together, which is equal to making an end-to-end slice for service composition.

As the first target experiment of FIRST@PC, we are considering a personalized and interactive broadcasting scenario. As depicted in Fig. 2, we define two underlying networks specialized for media production and distribution. Future media production enables real-time and online distributed video editing, where numerous media sources (e.g., live 4K video and panoramic multiple HD video) are dynamically integrated together. These integrated media streams are converted (with the aid of transcoding services and media upscaling/downscaling services) and delivered (e.g., multicasting services) to match the target end systems. Our target experiment assume that such heterogenous display devices receive common contents from live capturing service(s) and/or an on-demand content service. Three composite services are defined for mobile display, TV, and tiled display, respectively. Especially, for the composite service #3, the most powerful tiled display can service multiple media contents by selectively binding on-demand content service, live capturing service, video editing service, caching service, video tiling (partitioning into several tiles) service, and networked tiled display service.

These composite services, as a connected set of composable component services, are represented by a directed acyclic service dependency graph to express the functional dependencies. To run the service composition experiments, we first need to describe the targeted service composition with the service dependency graph. For this, it is required to clarify the relationship between the component services and used resources. However, since this clarification is not an easy task, the experimenter may rely on his interpretation and interactively guide the service composition. To avoid the manual interactive composition, we can upgrade the experiment to adopt a composition algorithm to automatically coordinate the service mapping between the service dependency graphs and physical distributed component services. We may further extend to the dynamic composition experiments, where the dynamic change in the connection path is automatically negotiated to overcome the unexpected congestion of underlying network links. We may also experiment the impact of load balancing method that evenly distributes total resources for dynamic composition of multiple media distribution.

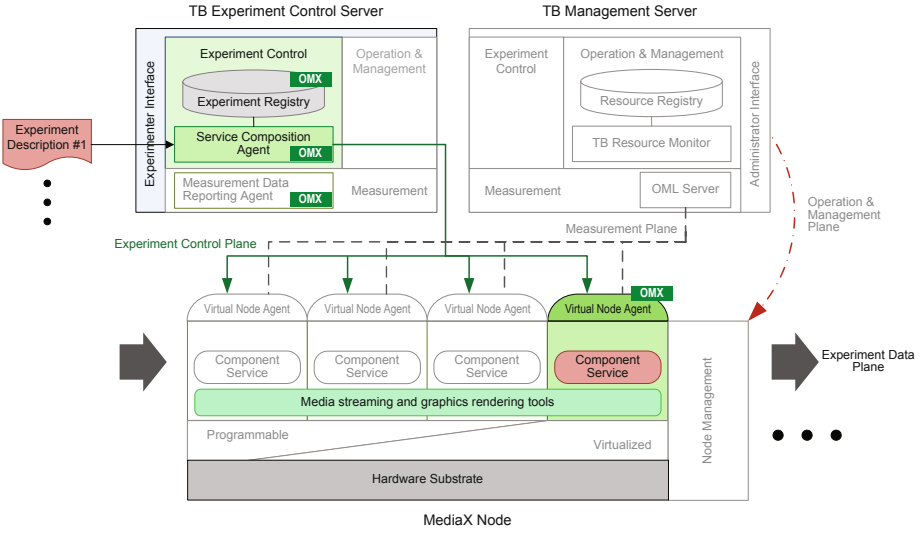


Fig. 3. FIRST@PC MediaX architecture (tentative)

3.2 FIRST@PC MediaX Architecture

Fig. 3 illustrates FIRST@PC MediaX architecture[2]. Experimenters formally specify the targeted service composition with quantitative requirements such as required component services and their configuration parameters (with numerical permissible ranges). They also describe adaptation rules to dynamically change composite services (e.g., service migration) according to events. The TB experiment control server assists the experimenters by feeding registry information about the capabilities of MediaX RA nodes to estimate the capabilities of instantiated component services. Measurement reports collected from real-time computing/networking performance data are used to configure component services and later enforce adaptation rules. On the other side, the TB management server enables administrators to register MediaX RA nodes and periodically monitor their status. We currently use OMF [11] for TB management and manage resources (e.g., resetting nodes, retrieving node status information, and installing new OS images). We also use the companion OML (Orbit Measurement Library) [12] to monitor TB resources (i.e., gather CPU/memory usage and the IO traffic amount).

To efficiently support multiple concurrent experiments, MediaX RA nodes need to provide virtualization of computing/networking resources. We are working on OpenVZ container-based virtualization that can virtualize resources of a common operating system for multiple virtual nodes. In a node, the node management governs the allocation of resources to virtual nodes. In each virtual

[2] Blocks in gray text in white background are not yet fully implemented. Also, blocks tagged with OMX belong to OMX toolkit, explained in Section 3.3.

node, an agent starts and stops component services and monitors its current status (e.g., inactivated, activated, or defected). Utilizing the allocated (virtualized) resources, component services provide elementary media functions and they are connected by taking the order from the TB experiment control server.

3.3 OMX Toolkit for Service Composition

To assist the composition experiment, the OMX (Open Media eXperiment for service composition) toolkit provides software agents (i.e., including experiment control agent, node agent, and service agent) with experimenter interfaces. Currently, the OMX toolkit (version 0.1) manually connects component services according to the service path[3] drawn by experimenters. To implement OMX toolkit, we use JADE (Java Agent DEvelopment framework), a FIPA-compliant multi-agent middleware.

The service composition description, written in XML, includes <name> representing the service identifier, <max_instances> representing the permissible number of the service instances to be executed, and <control_interfaces> specifying control interfaces (e.g., shell command) and their input parameters. For each virtual node, experimenters write Java code to describe their experiments by linking with the OMX agent that implements callback methods such as `start`, `stop`, `serviceLinkAdded` (when the service to be connected is determined), and `serviceLinkRemoved` (when the service to be disconnected is determined). Also, the OMX toolkit provide an experimenter UI (user interface) to facilitate the service composition experiment. The experimenter UI shows all available nodes and component services supported in each node. To draw a service dependency graph, an experimenter selects nodes by drag-dropping it from the node list, chooses a component service to be run, and connects a node with another.

4 Service Composition Experiment with FIRST@PC MediaX Testbed

4.1 Testbed Setup

FIRST@PC MediaX testbed, depicted in Fig. 4, includes MediaX nodes categorized into three different purposes: media servers, adaptors, and a networked tiled display. Media servers feed MPEG2-encoded live video. Adaptors do real-time media processing with GPU-based computing assist. The networked tiled display realizes ultra-high resolution by tiling multiple LCD displays. Most MediaX nodes are connected via 1Gbps LAN while some are connected via 10Gbps to handle multiple uncompressed HD videos.

4.2 Service Composition Experiment

A service composition experiment for sharing HD-media and desktop screen is specified by a service dependency graph depicted in Fig. 5(a). Live capturing

[3] The service path, a special case of a service dependency graph, shows how all component services are connected into a single successive end-to-end chain.

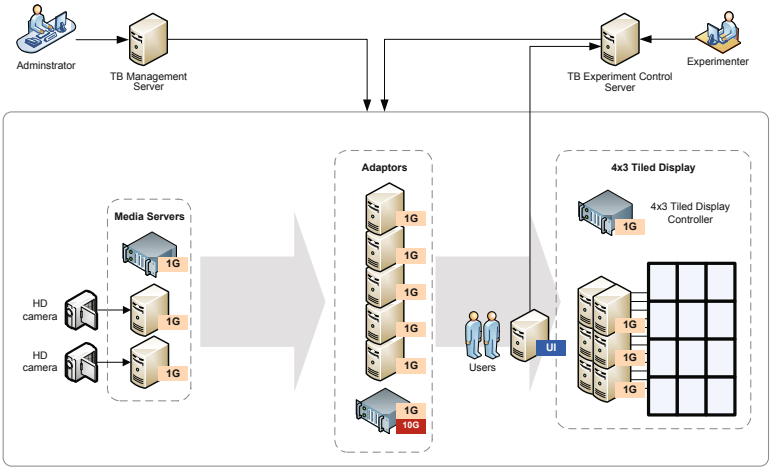
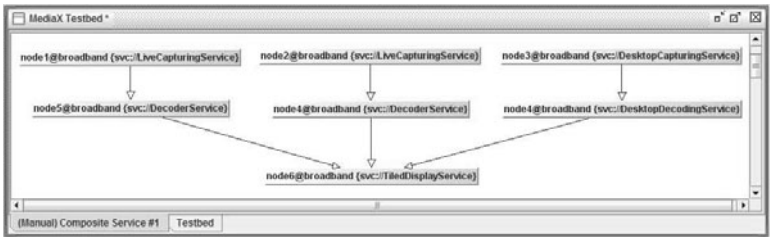
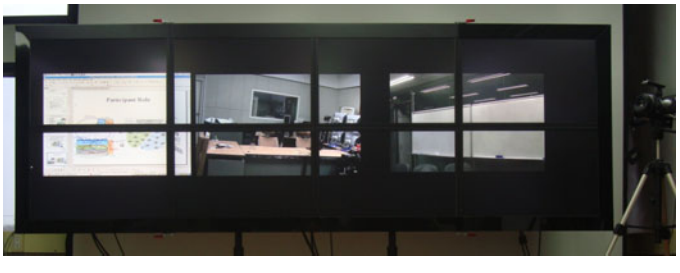


Fig. 4. Deployment diagram of FIRST@PC MediaX testbed



(a) Service graph drawn in the OMX toolkit interface.



(b) HD videos and shared desktop screen rendered.

Fig. 5. Experimental results of multiple HD video service composition

services acquire HD live videos and delivers the MPEG2-TS video stream to corresponding decoding services (realized by a VLC media player). Decoded uncompressed video are then sent to the networked tiled display. Also, an interactive graphics producer service captures desktop screen and transports graphic

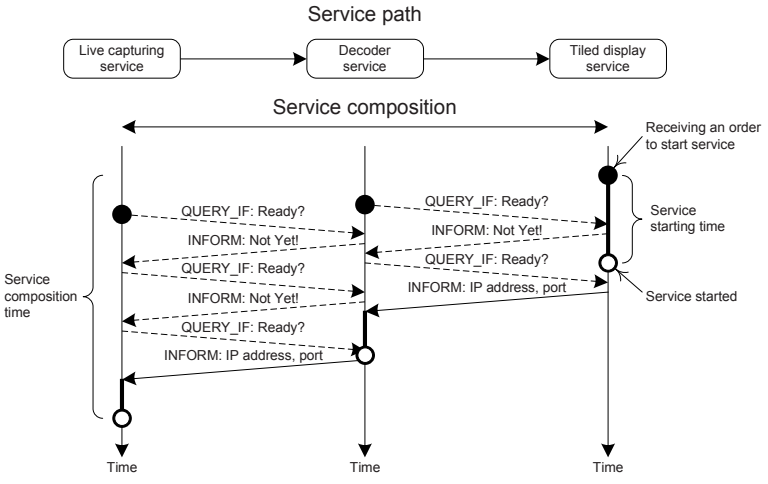


Fig. 6. Agent communication messages exchanged between service agents

streams (using TightVNC). An interactive graphics consumer service receives graphic streams and converts them for networked tiled display. All these composite service results are depicted in Fig. 5(b).

Fig. 6 illustrates the communication messages exchanged between the agents for involved component services. Composition is performed in backward sequential order from destination to source services along the service path. When a component service is asked to start, it consumes time to initialize its function. While the tiled display service prepares the service execution, the decoder service periodically sends QUERY_IF messages to inquire about the service status. The tiled display returns INFORM message with its IP address and port number when it is started. The decoder service prepares streaming to the IP address and port number. Similar procedures are repeated between the live capturing service and the decoder service.

5 Conclusion

An on-going effort to design media-centric service-oriented testbed called FIRST@PC MediaX was described where media-oriented service compositions are flexibly realized on top of virtualized computing/networking resources. Current design and realization on top of several PC-based devices for media acquisition, media processing, display (networked tiled display), and networking were extensively explained with the preliminary experimental testing based on the OMX toolkit.

Acknowledgments. This paper is one of results from the project (2009-F-050-01), "Development of the core technology and virtualized programmable platform for Future Internet" that is sponsored by MKE and KCC.

References

1. Daras, P., et al.: Why do we need a content-centric Future Internet? Proposals towards content-centric internet architectures. European Commission, Networked Media Unit, Information Society and Media (May 2009)
2. Huhns, M., Singh, M.: Service-oriented computing: key concepts and principles. IEEE Internet Computing 9(1), 75–81 (2005)
3. Nahrstedt, K., Balke, W.T.: Towards building large scale multimedia systems and applications: challenges and status. In: Proc. of Int. Workshop on Multimedia Service Composition (November 2005)
4. GENI Project Office: GENI System Overview. GPO Documents, GENI-SE-SY-SO-02.0 (September 2008)
5. Han, S.W., Kim, N., Kim, J.: Designing a virtualized testbed for dynamic multimedia service composition. In: Proc. Int. Conf. on Future Internet Technologies (CFI 2009), Seoul, Korea (June 2009)
6. Gibb, G., Lockwood, J., Naous, J., Hartke, P., McKeown, N.: NetFPGA–An open platform for teaching how to build Gigabit-rate network switches and routers. IEEE Trans. on Education 51(3), 364–369 (2008)
7. Bhatia, S., Motiwala, M., Muhlbauer, W., Mundada, Y., Valancius, V., Bavier, A., Feamster, N., Peterson, L., Rexford, J.: Trellis: A platform for building flexible, fast virtual networks on commodity hardware. In: Proc. ROADS 2008 (December 2008)
8. McKeown, N., Anderson, T., Balakrishnan, H., Parulkar, G., Peterson, L., Rexford, J., Shenker, S., Turner, J.: OpenFlow: enabling innovation in campus networks. ACM Computer Communication Review 38(2), 69–74 (2008)
9. Gu, X., Nahrstedt, K.: Distributed multimedia service composition with statistical QoS assurances. IEEE Trans. on Multimedia 8(1), 141–151 (2006)
10. Kalasapur, S., Kumar, M., Shirazi, B.: Dynamic service composition in pervasive computing. IEEE Trans. on Parallel and Distributed Systems 18(7), 907–918 (2007)
11. Rakotoarivelo, T., Ott, M., Jourjon, G., Seskar, I.: OMF: a control and management framework for networking testbeds. In: Proc. of Int. Workshop on Real Overlays and Distributed Systems (ROADS 2009), Big Sky, MT (October 2009)
12. Singh, M., Ott, M., Seskar, I., Kamat, P.: ORBIT Measurements framework and library (OML): motivations, implementation and features. In: Prof. Tridentcom 2005 (February 2005)

TridentCom 2010

Practices Papers Session 3: Wireless and Mobile Networking Testbeds

Design of a Configurable Wireless Network Testbed with Live Traffic

Ruben Merz, Harald Schiöberg, and Cigdem Sengul

Deutsche Telekom Laboratories, An-Institut der Technischen Universität Berlin
{ruben.merz,cigdem.sengul}@telekom.de, harald@net.t-labs.tu-berlin.de

Abstract. Testbeds have become standard tools for performance evaluation in wireless networks as they allow for evaluation with realistic wireless propagation and interference conditions, hardware constraints and timing requirements. However, in most testbeds, traffic and user mobility is generated by either using synthetic load generation tools, or replaying traffic traces. Evaluations using live traffic generated by real users, possibly moving around the network, are typically not possible. This is the main motivation of our research: building a wireless research testbed that enables experimentation with live traffic. In this paper, we present the design and architecture of a campus-wide wireless network testbed towards this goal. Our testbed enables both transparent Internet access and seamless mobility to the network users, and supports full network reconfigurations in the presence of live traffic. The reliability of user traffic is guaranteed by failure avoidance mechanisms that are invoked whenever a disruption occurs in the network.

Keywords: Wireless testbeds, live traffic, online configurations.

1 Introduction

Wireless networking research has gained prominence over the years due to its potential for realizing "anytime, anywhere networking". However, the transition from research to production level deployment has been somewhat slow as many wireless networking ideas do not typically move out of simulation environments. Therefore, despite the efforts for more realism and accuracy in wireless simulation [1,2,3], evaluation with testbeds needs to become more prevalent to understand the impact of realistic wireless propagation and interference conditions, hardware constraints and timing requirements. However, evaluation with testbeds is extremely tedious: research ideas are throttled by high implementation barrier, cumbersome and time-consuming experiment set-up and debugging, and validation challenges. These difficulties preclude testing in real systems and limit a broader adoption of testbed-based experimentation and evaluation.

To address these difficulties, wireless testbeds such as Orbit [4,5] and Emulab [6] allow for automating network and topology configurations and provide

T. Magedanz et al. (Eds.): TridentCom 2010, LNICST 46, pp. 189–198, 2011.

better experiment management and repeatability. Although these works ease deployment (e.g., by providing remote access to nodes and operating system distributions, tools to automatically script measurements), the implementation of an idea still remains significantly difficult. For instance, even the reconfiguration of network protocols and parameters need to go through invasive kernel changes. Therefore, current testbeds need to facilitate both implementation and deployment. Furthermore, current wireless testbeds do not typically serve live user traffic. In addition to allowing realistic input for evaluations, serving real users also urges researchers to build *complete systems*.

To the best of our knowledge, no testbed exists that provides both real-user traffic and easy online reconfigurations for wireless networking experimentation. We believe this limits full evaluation of research proposals; i.e., answering how a system behaves under different conditions, and finding out the underlying reasons for such behavior and the advantages of a system compared to others. To this end, a research testbed should allow for reconfiguring the network and running experiments with live traffic, characterizing the current network, traffic and mobility conditions, correlating system behavior to these conditions, and reproducing past conditions and system behavior. Bringing all these together presents a much challenging agenda. However, we assert that it is also much required to be able to fully evaluate systems. In this paper, we present a first step - the design, architecture and implementation of a testbed, the Berlin Open Wireless Lab (BOWL) [1], which reliably supports live traffic in a research network.

The main challenges that we address in this work stem from the conflicts between the expectations of network users and researchers. Real users expect reliable network access and privacy. On the contrary, a research network is an unstable environment with frequent outages due to experimental software and reconfigurations. These two different viewpoints create the following trade-offs in a research environment:

- **Reliable Internet access for users vs. network programmability for researchers:** Researchers require network programmability to run controlled experiments. In turn, any disruption to the user traffic in the presence of experiment reconfigurations or an outage caused by experiment failures should be avoided. Essentially, failure avoidance mechanisms should automatically push the network to a safe state and gracefully recover affected nodes.
- **Privacy for users vs. providing a rich set of measurements for researchers:** Since user traffic is not controlled, all the parameters of the underlying layers, e.g., TCP, routing and medium access control (MAC) layers should be captured for any meaningful measurement study. Furthermore, application level information might be necessary to reproduce an experiment. However, the measurement and tracing of system information should be performed without jeopardizing user privacy.

[1] http://bowl.net.t-labs.tu-berlin.de/

The main contribution of our research is providing two essential capabilities that address these trade-offs:

- Automated and online reconfigurations: it includes turning on/off functionality of a specific protocol and changing protocols at different network layers respecting inter-layer dependencies.
- Hosting real user traffic: this requires avoiding disruption to the user traffic during reconfigurations. Failure avoidance mechanisms are triggered to automatically push the network to a safe state during reconfigurations. The network comes out of this state once a reconfiguration is complete.

To the best of our knowledge, this is the first wireless testbed of its kind.

The rest of the paper is outlined as follows. In Section 2, we present the implementation of our outdoor wireless testbed, both its network and node design, as well as the software components. In Section 3, we present an example configuration of our network: the mesh network configuration. Section 4 discusses the related work and we conclude in Section 5 with a summary and future work.

2 The Berlin Open Wireless Lab Network

To create our testbed, we deployed 46 nodes outdoor, on the rooftop of several buildings in the TU-Berlin campus, covering roughly 40 hectares. The maximum link distance is one kilometer. In addition, we deployed 12 nodes indoor on two floors of the Deutsche Telekom Laboratories. In the remainder of this section, we walk through our network and node design, software architecture and the main components for reconfigurations with live traffic.

2.1 Network Architecture and Node Design

Our network architecture and the node design are shaped mainly by our design objectives: to be able to perform remote management as well as failure avoidance and recovery in the presence of disruptions to the network (e.g., due to failures or launching new experiments), and to provide a rich set of configuration options. To achieve high *topology configurability*, our nodes are deployed densely in the campus, allowing switching off a significant fraction without losing connectivity. This enables effective evaluation of power control and interference issues in dense networks. Additionally, our nodes have several wireless interfaces. Also, the antenna poles are extensible and can easily host additional hardware.

Our mesh nodes are built around an Avila Gateworks GW2348-4 motherboard [7]. Each node has an Intel XScale 533 MHz CPU (ARM architecture), 64 Mbyte of RAM and four wireless interfaces - Wistron CM9 with an Atheros 5213 chipset. One wireless interface is always operated in the ISM band at 2.4 GHz, running IEEE 802.11g, where either a 2.4 GHz 12 dBi omnidirectional antenna or a 17 dBi sector antenna is attached. This interface provides an access point (AP) functionality for users to access the network and the Internet. The three other wireless interfaces are operated in the UNII 5 GHz band, where a 5 GHz 12 dBi omnidirectional antenna is attached.

For most of the rooftop nodes, physical access is severely limited. Therefore, each node is remotely accessible through a wired, 100 Mbit/s Ethernet, connection. This Ethernet connection plays a critical role for remote management, failure avoidance and reconfigurations. In practice, the outdoor nodes and the indoor nodes are on two separate VLANs that connect to a central router. Two other VLANs are also attached to this central router: one connects to the Internet through the TU-Berlin network and the other one is used for management. For failure avoidance, additionally, a hardware watchdog daemon automatically triggers a reboot in case of malfunction or connectivity loss. Finally, a majority of the nodes contain a second independent unit for managing the router in case of a malfunction or as a passive measurement and monitoring device. This second unit is built around an Asus WL-500GP router. A custom-built circuit interfaces the two boards and allow us to remotely power on and off each board from each other. The nodes are powered by Power over Ethernet (PoE). Hence, for the rooftop nodes, only one cable per node is necessary. Both units run a customized version of the OpenWrt operating system, a GNU/Linux distribution for embedded devices [8].

2.2 Software Architecture

Building on its hardware and network architecture, BOWL brings together several software components to achieve a reconfigurable live network:

- **Online reconfigurations** provide network, topology, parameter and protocol reconfigurations.
- **Failure avoidance mechanisms** switch nodes to a fail-safe state to maintain reliability during disruptions.
- **Remote management and monitoring** remotely and automatically manages nodes as well as checks their status.
- **Measurements and tracing** provide user, system and network information.
- **Visualization and control interface** provides both flow and connectivity based views of the network in real-time.

In the rest of the section, we explain these components in further detail.

Online Reconfigurations
To run experiments in a wireless network, researchers need the capability to perform automated and online reconfigurations. To this end, the BOWL testbed can be run in different *network configurations* for each experiment. We currently support three basic configurations: an infrastructure network (the default configuration), a mesh network or a mixture of the two.

In each configuration, the 2.4 GHz wireless interface is used as the "client interface" to provide an AP functionality for Internet access. However, it is up to the experimenter to use the live traffic or to direct it straight to the Internet over the wired interface. In the infrastructure configuration, the client interface is transparent and bridged to the central router. Authentication and access control

are handled on this central router, as well as DHCP. We provide transparent and seamless mobility and roaming between the APs. This is achieved by mainly running the "client interfaces" in promiscuous mode.

Additionally, the BOWL testbed supports configurations for changing the network topology, exploring the parameter space of a given protocol, or comparing different protocols under similar network conditions. To modify the *topology*, we support connecting additional nodes to or disconnecting nodes from the network remotely. The level of network connectivity can also be modified by changing the per-node transmission power levels. Going a step further, we also support modifications to *protocol parameters*. For instance, we can change a MAC layer functionality: turn on and off RTS/CTS in IEEE 802.11. We also enable *entire protocol* switches at a given layer of the network stack. As a proof of concept, we currently support routing protocol switches in our mesh configuration. These examples are explained in more detail in Section 3.

For efficient execution, all these reconfigurations rely on the "remote management and monitoring" component, which will be described later in this section. Furthermore, during all these reconfigurations, user traffic is expected to be disrupted. How to protect user traffic from such disruptions is discussed next.

Failure Avoidance Mechanisms

The core of our software design consists of providing one stable and safe default configuration, or "rescue" mode to fall back to during disruptions and a very flexible "live" mode to facilitate research. This is implemented by installing a so-called "rescue system" and possibly multiple so-called "guest systems", each of which are fully self-contained Linux systems. To this end, we extended OpenWrt for our purposes. The rescue system is installed on the internal flash memory of all the nodes and is started by the boot loader. Guest systems are installed on additional flash memory storage.

The rescue mode boots the rescue system, runs the default infrastructure configuration (i.e., the user traffic is bridged to the central router over the wired interface) and provides the functionality to install and launch guest systems. The live mode boots a guest system and runs a configuration corresponding to an experiment. Both modes must always implement the AP functionality.

In addition to the rescue and live modes, BOWL supports a "transient" mode, where the data flows by default through the Ethernet interface and can be replicated over the wireless network for testing (i.e., the duplicate traffic is discarded at the exit point of the network). Like the AP functionality, the transient mode must be implemented by any guest system.

Failure avoidance makes nodes to fall back to either rescue or transient modes. It can be triggered on-demand whenever we expect a major disruption or automatically. If the watchdog triggers a reboot, a node reboots in the rescue system. The expected delay is in terms of a few seconds. The management of these different states are handled by the "remote management and monitoring" component, which is explained next.

For the implementation of the online reconfiguration and failure avoidance mechanisms, we make extensive use of the Click modular router [9,10].

Remote Management and Monitoring
The management architecture was designed using a centralized approach that contains two parts: (1) **A central node manager** keeps various pieces of node state and information in a database and marshals commands and information to the nodes. (2) **A node controller** runs on each node and executes commands initiated by the central node manager, as well as collects state and information, which, in turn, are sent back to the central node manager.

All software is written in Ruby [11], as it is readily available on several platforms which cuts down development time in case we upgrade or change our hardware. The second reason is the availability of Distributed Ruby (DRb), a remote method invocation (RMI) implementation. The database uses Postgresql[12].

The central node manager is a process that uses a database back end to maintain information about each node controller. Communication is marshaled to the nodes using DRb, which allows for both state and code distribution. In addition, the central node manager implements an event-based callback framework. Finally, the node manager supports exporting data to other components (e.g., a visualization component) out of the database. To support privacy, sensitive data is mangled: for example, the MAC addresses are anonymized.

Each node runs an instance of the node controller to communicate with the central node manager. The node controller implements two main concepts: so-called adaptors and an event framework. An adaptor maintains a particular functionality on the node (e.g., DHCP), control various daemons and relay information to the node controller itself. Each adaptor runs in its own thread within the controller process. In the default configuration, the default adaptor is the association adaptor, which maintains information about the associated clients. The central node manager learns about new clients through this adaptor.

The event framework allows the controller to react to changes in the environment and the state of its adaptors. The typical communication flow between a node controller and central node manager is as follows: (1) Changes in an adaptor may result in an event. (2) This is relayed by the node controller to the central node manager. (3) Next, this may lead to the execution of callbacks. (4) The callbacks, in turn, may be dispatched back to the node in question.

Measurement and Tracing
Measurements typically consist of data points, collected by the nodes and transferred into the measurement database for further processing (e.g, for visualization). Measurement and tracing processes currently run separately from the node controllers. However, like any other program, these processes are also generally configured and started through the node controllers. To collect measurements from the nodes to a central location, we extensively use the Orbit measurement framework and library (OML), which comprises a measurement library and server developed in the context of the Orbit testbed [13]. We currently support two ways of collecting data. Connectivity measurements collect signal strength information by monitoring IEEE 802.11 beacons or other traffic, and traffic measurements collect data regarding the existing flows in the network.

Visualization and Control Interface. We implemented a live testbed map, which displays the current situation in the network (e.g., existing *real user* connections, current node configurations and protocol parameters). Using this map, changes like new nodes or disruptions can be easily displayed. The data to the network visualization component is acquired from two different interfaces: (1) the database of central node manager, which maintains different node states and (2) the measurement component, which, for instance, provides connectivity strength among different nodes and flow information.

The network is also remotely configurable through either a command line interface or a "Control" frame embedded in the map, which can interface to the node manager. In the next section, we give examples of currently possible reconfigurations and implemented controls based on the mesh network configuration.

3 A Configuration Example: The Mesh Experiment

In this section, we show how the BOWL testbed can be configured as a wireless mesh network [14]. In the current implementation, any node can act as an AP and one node acts as the gateway. Figure 1 depicts a snapshot of the network in this configuration on 27 October, 2009. In the mesh configuration, the mesh network is transparent to the clients and is seen as a regular layer 2 network. The mesh interfaces use a different IP address range than the clients and the central router. In the mesh, IP packets are encapsulated using IP in IP tunneling and sent towards the mesh gateway. They are decapsulated at the gateway before being delivered to the central router. The exterior IP addresses of the tunnel belong to the mesh IP address range. The AP nodes use ARP spoofing to answer ARP requests from the clients to obtain the hardware address of the first hop. Also, the gateway uses the same technique to answer ARP requests from the first central router for the hardware addresses of the clients.

This tunnel setup also enables seamless mobility. At an AP node, a so-called location table indicates which node is the current gateway of the network[2]. The destination IP address of the tunnel is then set to the mesh IP address of the gateway. At the gateway, the location table indicates to which AP node is a client currently attached. Hence, the destination IP address of the tunnel is set to the mesh IP address of this AP node. These location tables need to be maintained only on nodes that function as an AP or a gateway. The location tables are currently updated centrally by using the Ethernet interface.

In this mesh configuration, we support several challenging reconfiguration scenarios, e.g., a routing protocol switch from OLSR to DSR and vice versa. In DSR, a route is discovered only when a new flow is initiated. Therefore, additional steps need to be taken to first find routes for flows that exists in the network during reconfiguration. In this case, the transient mode is a perfect fit as it redirects the client traffic through the wired interface but also duplicates it on the active wireless interfaces. When switching between DSR and OLSR, the transient mode is activated right before starting the routing protocol switch. This

[2] This setup is straightforward to extend to multiple gateways.

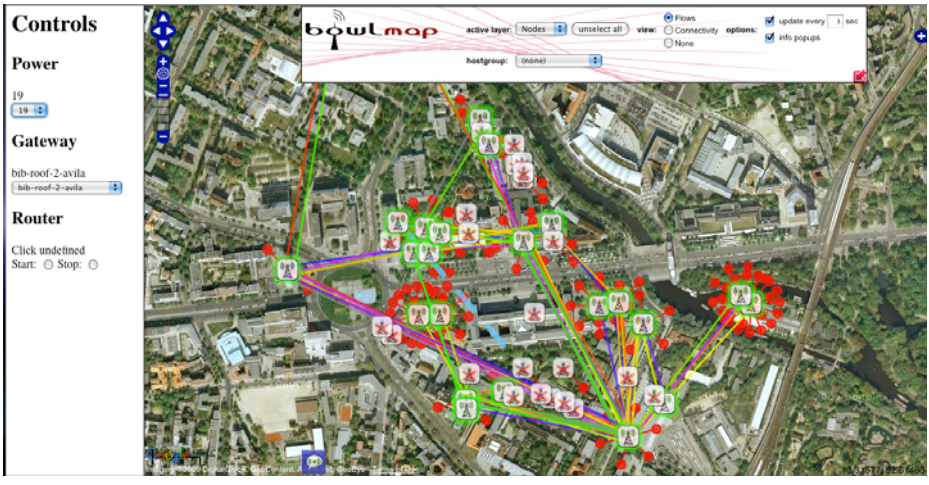


Fig. 1. Snapshot of the network in the mesh network configuration. The green nodes are online APs; the red nodes are unreachable as they are newly deployed, or not configured. The red dots are the clients attached to the access points. The map depicts the flow view, where each flow is is shown with a different color line. The control frame on the left enables transmit power control, gateway change, and both wired-only and wireless-only communication through "start and stop mesh routing".

duplicated traffic bootstraps DSR and fills route caches on the nodes. Similarly, transient mode is used when the gateway functionality is assigned to another node, or when the transmission power is modified. In both cases, the connectivity to the gateway might be impaired and hence, requires client traffic to be safely delivered over the wired interface until connectivity to the gateway is restored.

4 Related Work

The existing work aiming at evaluating wireless research can be roughly categorized between simulators, emulation testbeds and testbed deployments. Simulators are typical tools of choice [3] in wireless research due to their relative ease-of-use. However, a recent survey also shows that simulations may have several pitfalls due to faulty practices, *but* more importantly, due to abstracting details, especially at the physical layer.

Some of the limitations facing wireless scientific research and some suggestions to overcome these are presented in [15]. The closest to our work in essence are PlanetLab in the wired world and TFA [16], the Netbed/Emulab [6] and Orbit [4,17,13,18,5] in the wireless world. PlanetLab [19] also tries to bring together researchers that develop and test services, and clients that want to use these services. The TFA network [16] covers residential users with the goal of providing reliable connectivity to low-income households, whereas our goal is

to enable evaluation and comparison of real systems in an open experimental environment. Netbed/Emulab provides a test-bed management framework with a Web-based front end used to create and manage experiments. Note that Emulab is an indoor network, which allows testing protocols with artificial traffic. Despite these differences, Emulab provides a very mature service interface for distributing code, controlling applications and gathering log files, which can be extended to match our requirements.

To compensate for the lack of realistic network conditions in Emulab, FlexLab [20] proposes to loosely couple Emulab with PlanetLab. This can only capture limited aspects of user behavior, namely traffic characteristics but not mobility. Another effort is presented in [21]. However, the goal is limited to re-introducing "laboratory notebooks" to the networking community and automating re-running recorded experiments or their variations.

The Orbit testbed [4,5] is an indoor two-dimensional grid of 400 IEEE 802.11 radios. Nodes can dynamically form specified topologies with reproducible wireless channel models. However, Orbit does not support real users. Similarly, the Hydra testbed [22] is a purely research testbed with no support for real users. To the best of our knowledge, no experimental research environment exists, which can meet all three design goals (1) supporting real users, (2) enabling to build real systems and (3) facilitating real system evaluation.

5 Conclusion and Future Work

This works presents the Berlin Open Wireless Lab (BOWL) network, a configurable testbed with support for live traffic. The current network deployment comprises 46 multi-radio IEEE 802.11 nodes deployed outdoor on the rooftops of the TU-Berlin campus. The live traffic is generated by TU-Berlin students and staff using the network for Internet access. Thanks to its unique software architecture, the network allows researchers to reliably and remotely reconfigure the network and run experiments with live traffic.

Currently the following reconfiguration scenarios that can be remotely activated in the BOWL network include a mesh network configuration with a single gateway, gateway activation and gateway location update, and switching protocols and updating protocol parameters. This work is a first step towards an experimental research environment with real user traffic. For future work, we plan to extend the software architecture to better support the collection of measurements, validation of configurations and experiment results.

Acknowledgments

We would like to thank Anja Feldmann for her continuous support to the BOWL project, Thomas Hühn who made the deployment of this network a reality, Mustafa Al-Bado for the implementation of node installation validation tools, Benjamin Vahl and Julius Schulz-Zander for their work on the live map, and our system administrator, Thorsten Fischer.

References

1. The ns-3 network simulator, http://www.nsnam.org
2. Kotz, D., Newport, C., Gray, R.S., Liu, J., Yuan, Y., Elliott, C.: Experimental evaluation of wireless simulation assumptions. In: MSWiM 2004: 7th ACM Int. Symp. on Modeling, Analysis and Simulation of Wireless and Mobile Systems, pp. 78–82. ACM, New York (2004)
3. Kurkowski, S., Camp, T., Colagrosso, M.: MANET simulation studies: The incredibles. Mob. Comp. and Comm. Rev (MC2R) 9, 50–61 (2005)
4. Orbit, http://www.orbit-lab.org/
5. Raychaudhuri, D., Seskar, I., Ott, M., Ganu, S., Ramachandran, K., Kremo, H., Siracusa, R., Liu, H., Singh, M.: Overview of the orbit radio grid testbed for evaluation of next-generation wireless network protocols. In: IEEE WCNC 2005 (2005)
6. School of Computing, University of Utah: Emulab - total network testbed, http://www.emulab.net/
7. Avila network platform gw 2348-4, http://www.gateworks.com/products/avila/gw2348-4.php (retrieved November 2009)
8. Openwrt, http://openwrt.org/
9. Kohler, E., Morris, R., Chen, B., Jannotti, J., Kaashoek, M.F.: The click modular router. ACM Trans. Comput. Syst. 18, 263–297 (2000)
10. The click modular router, http://read.cs.ucla.edu/click/
11. The ruby programming language, http://www.ruby-lang.org/
12. Postgresql open source object-relational database system, http://www.postgresql.org/
13. Singh, M., Ott, M., Seskar, I., Kamat, P.: Orbit measurements framework and library (oml): Motivations, design, implementation, and features. In: TRIDENTCOM, pp. 146–152 (2005)
14. Akyildiz, I.F., Wang, X., Wang, W.: Wireless mesh networks: a survey. Comput. Netw. ISDN Syst. 47, 445–487 (2005)
15. White, B., Lepreau, J., Guruprasad, S.: Lowering the barrier to wireless and mobile experimentation. ACM SIGCOMM CCR 33, 47–52 (2003)
16. Camp, J., Knightly, E., Reed, W.: Developing and deploying multihop wireless networks for low-income communities. In: Digital Communities (2005)
17. Ott, M., Seskar, I., Siraccusa, R., Singh, M.: Orbit testbed software architecture: Supporting experiments as a service. In: TRIDENTCOM, pp. 136–145 (2005)
18. Ganu, S., Kremo, H., Howard, R., Seskar, I.: Addressing repeatability in wireless experiments using orbit testbed. In: TRIDENTCOM, pp. 153–160 (2005)
19. Peterson, L., Anderson, T., Culler, D., Roscoe, T.: A blueprint for introducing disruptive technology into the internet. In: Hotnets (2002)
20. Ricci, R., Duerig, J., Sanaga, P., Gebhardt, D., Hibler, M., Atkinson, K., Zhang, J., Kasera, S., Lepreau, J.: The flexlab approach to realistic evaluation of networked systems. In: NSDI (2007)
21. Eide, E., Stroller, L., Lepreau, J.: An experimentation for replayable networking research. In: NSDI (2007)
22. Mandke, K., Choi, S.H., Kim, G., Grant, R., Daniels, R.C., Kim, W., Heath, R.W., Nettles, S.M.: Early results on hydra: A flexible mac/phy multihop testbed. In: IEEE VTC-Spring, pp. 1896–1900 (2007)

Design, Implementation and Testing of a Real-Time Mobile WiMAX Testbed Featuring MIMO Technology*

Oriol Font-Bach[1], Nikolaos Bartzoudis[1], Antonio Pascual-Iserte[1,2], and David López Bueno[1]

[1] Centre Tecnològic de Telecomunicacions de Catalunya (CTTC), Parc Mediterrani de la Tecnologia (PMT), Av. Carl Friedrich Gauss 7, 08860 Castelldefels, Barcelona, Spain
[2] Dept. of Signal Theory and Communications - Universitat Politècnica de Catalunya (UPC), Campus Nord, Jordi Girona 1-3, 08034 Barcelona, Spain
{ofont,nbartzoudis,dlopez}@cttc.cat, antonio.pascual@upc.edu

Abstract. Multiple input multiple output (MIMO) is a technology that enhances wireless systems capacity, data rate, and coverage by utilizing the spatial diversity provided by multiple antennas. However, these benefits come at the expense of increased computational complexity. Implementing a broadband MIMO wireless communication system in a real-time testbed is a challenging task, entailing numerous pitfalls. This paper presents several implementation aspects of a real-time MIMO testbed based on the mobile WiMAX standard. The focus is mainly laid on the bit-intensive baseband digital signal processing at the receiver.

Keywords: MIMO, testbeds, IEEE 802.16e, real-time systems, FPGAs, DSP.

1 Introduction

Deploying a broadband wireless communication standard such as the IEEE 802.16e-2005 [1] (i.e., mobile WiMAX) in a real-time testbed implies several design, implementation and testing challenges, especially considering the top-up computational complexity introduced by the MIMO technology. The massive parallelism required for the baseband signal processing in real-time testbeds, makes the FPGA devices the obvious candidate for implementing such systems. The inherent processing parallelism of FPGA devices and the availability of a wide range of pre-verified IP-cores make them a preferable choice compared to DSP microprocessors. At the same time, the cell-processors though demonstrating a remarkable performance [2] are still considered to be an immature solution due to the C-coding parallelism limitations, the insufficient IP libraries and the lack of development boards.

This paper presents the challenging and demanding task of implementing a point-to-point mobile WiMAX system in a real-time testbed having a 2x2 MIMO configuration.

* This work was partially supported by the Catalan Government under grant 2009 SGR 891; by the Spanish Government under projects TEC2008-06327-C03 (MULTI-ADAPTIVE) and 2A103 (MIMOWA) from MEDEA+ program (AVANZA I+D TSI-020400-2009-44) and "Torres Quevedo" grants PTQ-08-01-06441, PTQ06-02-0540, PTQ06-2-0553; and by the European Commission under projects NEWCOM++ (216715) and BuNGee (248267).

T. Magedanz et al. (Eds.): TridentCom 2010, LNICST 46, pp. 199–208, 2011.

The system uses matrix A encoding in an open-loop configuration (i.e. without feedback), based on Alamouti's space-time block code [3] in a per carrier basis. The 20 MHz channel bandwidth of this testbed exceeds the WiMAX Forum specifications for the IEEE 802.16e-2005 standard (i.e. wave-2) positioning the system presented herein on the forefront of applied research utilizing real-time MIMO testbeds.

Setting up the whole testbed is a quite hard research and engineering task. The most critical part of the mentioned development is found in the design, simulation, implementation and real-time debugging of the receiver which, for this reason, is widely detailed in this paper.

2 Short Review of MIMO Testbeds

The great majority of the existing testbeds supports off-line signal processing, making use of Matlab or other signal processing software [4], [5]. Apparently, off-line testbeds are not able to process in real-time the received signals. However, their flexibility makes them appealing to researchers since such testbeds allow them to explore various real-world signal processing concepts. Their offline operation renders these testbeds incapable to explore medium access control protocols and the reception of long data frames because of timing and memory constraints, respectively. Besides, offline testbeds are not able to realize closed-loop strategies.

Although we have found in the literature low-bandwidth MIMO testbeds based on the IEEE 802.11n [6] and the 802.16d standard [7], [8] (i.e. no mobility), we have not encountered literature for real-time MIMO testbeds implementing the IEEE 802.16e standard using a 20 MHz bandwidth. A combination of mobile and fixed WiMAX testbed is presented in [9]; nevertheless the scope of the project is different since commercial equipment is used to assemble the entire physical layer of the testbed. Real-time MIMO testbeds implementing the IEEE 802.16e-2005 standard are mainly deployed by industrial initiatives (e.g. Alvarion), which are currently offering bandwidths up to 10 MHz (e.g. WiMAX wave-2 specifications [1]).

3 Description of the Experimental Setup

A point-to-point MIMO testbed typically comprises i) a transmitter with baseband signal processing units, digital-to-analog converters (DACs) and RF up-converters, ii) a multi channel emulator or sets of transmit and receive antennas (indoor channel), and iii) a receiver with a series of RF down-converters, analog-to-digital converters (ADCs) and baseband digital signal processing units. A graphic-overview of our real-time mobile WiMAX testbed setup for a point-to-point 2x2 MIMO system is shown in figure 1. The parameters of the OFDM downlink (DL) frame, consisting of a single burst with a fixed predefined format (i.e, FCH and DL-MAP are not decoded), are shown in table 1 and depicted in figure 2.

Taking into account the baseband sampling frequency (i.e. 22.4 MHz), the rate of Alamouti's space-time code (STC) coding (i.e. unity) and the total number of PUSC subcarriers (i.e. data+pilot+dc-carrier = 1681), the actual peak of un-coded useful data rate is $22.4 \times 2 \times 1 \times 1681/2048 \times 2048/2560 = 29.4175$ Mbits/s and the spectrum efficiency for a 20 MHz channel bandwidth is 1.47 bits/Hz/s assuming the use of QPSK modulation at each data carrier.

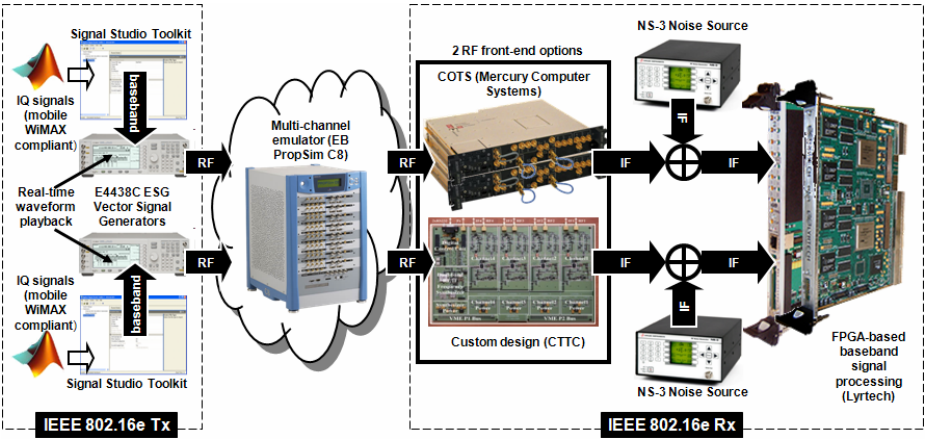


Fig. 1. CTTC's point-to-point real time 2x2 MIMO testbed featuring mobile WiMAX

Table 1. A synopsis of system parameters

Parameter	Value
Wireless telecommunication standard	IEEE 802.16e-2005
Tx antennas x Rx antennas	2x2
RF frontend operating band (GHz)	2.495 - 2.690
IF frequency ‖ Ch. Bandwidth (MHz)	156.8 ‖ 20
Channel models	ITU Ped. B - Veh. A
A/D sampling clock frequency (MHz)	89.6
Sampling frequency Fs (MHz)	22.4
Modulation type	QPSK
Duplex mode	TDD
FFT size	2048
Supported permutation scheme	DL PUSC only
Data ‖ pilot ‖ null subcarriers	1440 ‖ 240‖ 368
Sub-channels	60
Subcarrier frequency spacing f (kHz)	10.94
Useful symbol time ‖ Guard time (μs)	91.4 ‖ 22.85
Frame duration (ms) ‖ OFDM symbols	5 ‖ 48
Open loop configuration: STC type	Matrix A Alamouti
De-interleaving, Ch. coding, multiuser	not supported

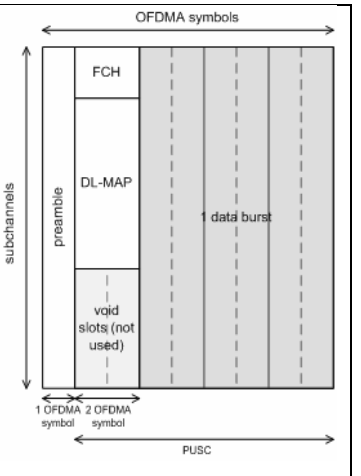


Fig. 2. Frame format

MIMO Signal Transmission: The baseband part of the transmitter was designed using Matlab. The separate I and Q baseband outputs of this model (corresponding to the two transmitter's branches) are written to data-files, which are fed to two instances of Agilent's Signal Studio Toolkit. The data-files are then uploaded to two Agilent vector signal generators (i.e. ESG4438C), which are appropriately connected for a MIMO signal transmission. This connection requires careful offline adjustments (e.g. time-alignment of the output signals). The two ESG4438C are utilizing their embedded arbitrary waveform generator to playback in real-time the baseband I and Q waveforms, up-convert the signal and finally provide the RF output centered at 2.595 GHz. The accuracy of these instruments guarantees a very high performance (e.g. excellent Error Vector Magnitude (EVM) profile). To verify the transmitter's

compliance with the IEEE 802.16e standard, we have used Agilent's Vector Signal Analyzer (VSA) to demodulate the received RF signal.

The Channel: The connection between the transmitter and receiver can be done via a direct cable, over the free radio channel using antennas, or through a channel emulator. The two testing and measuring scenarios were: i) antenna transmission using an indoor radio channel, and ii) use of a channel emulator (i.e. EB Propsim C8) to generate an outdoor static or mobile channel. Measurements over the indoor radio channel were conducted only to prove functional conformity and thus they would not be analyzed herein. The channel emulator is configured with a 2x2 MIMO model, emulating the ITU Vehicular-A standard channel model (i.e. 6 tap tap-delay-line). The channel is assumed to be quasi static for the duration of an OFDM frame.

The Receiver: A fully integrated, dual-band multi-channel WiFi RF transceiver [10] (i.e. designed in CTTC) was upgraded to match the WiMAX testing scenario. Certain critical building blocks were replaced (i.e. RF and IF filters, the local oscillator and the sampling frequency synthesizer). Both the WiFi operation at 2.4 GHz and the WiMAX one at 2.6 GHz performed satisfactory in a 2x2 MIMO configuration (i.e. proof of concept validation). The testbed specifications were expanded in terms of scalability and performance by acquiring high-end, multi-channel broadband RF downconverters (i.e. MCS Echotek Series RF 3000T). Table 2 summarizes the main specifications of the two available RF front-end solutions.

Table 2. Performance-comparison of the RF front-end solutions

Parameter	Custom receiver	COTS receiver
RF input frequency range	2.4-2.7GHz & 5.15-5.35 GHz	20MHz-3GHz
IF output frequency range (3dB BW)	135..173 MHz	107.5..172.5 MHz
Frequency resolution	<150Hz	1 Hz
Internal reference accuracy	<±1ppm adj.	<±0.5ppm
Phase Noise	-83dBc/Hz@10KHz	-112dBc/Hz@10KHz
Noise Figure	9.5dB	8.25 dB
Gain control range	72 dB	85 dB
Image rejection	30 dB	95 dB
Spurious output levels	-30 dBc (LO not inc.)	-85 dBc
Input Third-Order Intercept Point (IIP3)	-15 dBm	0 dBm

The receiver also includes two noise signal generators (i.e. Applied Instruments NS-3) for testing purposes and a powerful baseband signal processing platform from Lyrtech Inc., which is assembled in a cPCI chassis. This includes an ADC board (i.e. 8 channels, 105 MSPS, 14-bit resolution, Xilinx Virtex-4 LX160 FPGA device), a signal processing board with 4 Xilinx Virtex-4 devices (i.e. 2 LX160 and 2 SX35) and 4 Texas Instruments TMS320C6416 DSPs. The platform also offers various I/O connectivity options (e.g. 8 Gbps data transfer between boards).

4 Signal Model and Impairments

The WiMAX signal is frame-based, composed of data and silence periods. The receiver is continuously monitoring the signal during the silence periods through a

synchronization algorithm to detect the beginning of a data period. In real systems, on top of the noise, some system-wide signal-impairments appear during the silence due to the instrumentation used (e.g. signal generators, channel emulator). These "parasitic" signals impair the operation and performance of the system. Each one of these has been studied in order to remove its undesirable effects. The specifications of the signal generators, the channel emulator and RF downconverters, allow us to ignore the impact of the following signal-impairments: i) I/Q gain and phase imbalances due to variations in components between the analog I and Q processing branches, ii) inaccuracy between the sampling clocks of the transmitter and receiver, iii) random phase noise due to oscillator instability. Thus, the resulting received signal model at the output of the RF downconverters at the *i*th antenna can be expressed as:

$$r_i(t) = \mathrm{Re}\left\{ x_i^{(R)}(t) \cdot e^{j2\pi(f_{IF}+\Delta f)t} \right\} + A_i + B_i \cos\left(2\pi(f_{IF}+\Delta f)t + \varphi_i\right) + w_i(t), \quad (1)$$

where $x_i^{(R)}(t)$ represents the useful part of the received baseband signal, f_{IF} is the intermediate frequency (IF), Δf is the carrier frequency offset (CFO), A_i represents the DC level introduced by the baseband boards, $B_i\cos(2\pi(f_{IF}+\Delta f)t + \varphi_i)$ represents the carrier located at the center of the useful signal-spectrum as a result of the coupling of the local oscillators at the transmitter and/or the receiver, and $w_i(t)$ is the Gaussian noise. In our case, the IF is 156.8 MHz and the sampling frequency is 89.6 MHz (oversampling by a factor of 4). This means that after the ADCs, one of the aliases of the discrete signal will be located at 22.4 MHz. The $x_i^{(R)}(t)$ can be expressed as:

$$x_i^{(R)}(t) = \sum_{p=1}^{n_T} x_p^{(T)}(t) * h_{i,p}(t), \quad (2)$$

where $x_p^{(T)}(t)$ is the equivalent baseband signal transmitted from the *p*th transmit antenna and $h_{i,p}(t)$ is the equivalent baseband of the time impulse response of the MIMO channel between the *p*th transmit and the *i*th receive antennas. The total number of transmit antennas is assumed to be n_T.

Several countermeasures were employed to compensate as much as possible the effects of the received noise and some undesired spurious signals introduced by the channel emulator (e.g. prototyping of additional IF SAW filters).

5 The Mobile WiMAX Receiver

The digital front-end is one of the most critical processing stages of the receiver with determinant contribution to system's performance, highly susceptible to signal impairments and thus error-prone. It comprises from the automatic gain control (AGC), the digital down converter (DDC) and the synchronization block (figure 3). Any deviation from the expected signal-specification may either render the system's output invalid or seriously compromise its performance.

The AGC block adjusts the gain of the programmable gain amplifier to fit the dynamic range of the signal to the operating range of the ADCs preventing their saturation (excluding a back-off margin accounting for the OFDM signal crest factor) preventing their saturation and minimizing quantization errors. The DDC extracts the in-phase and quadrature components of the signal. This is achieved by a numerically

controlled oscillator (NCO), initially tuned at the nominal frequency of 22.4 MHz, a digital mixer and a low-pass filter with a decimation factor of 4 (i.e. output baseband sampling frequency of 22.4 MHz). The filter was specifically designed to reject the DC level introduced before the ADCs by the baseband board chassis.

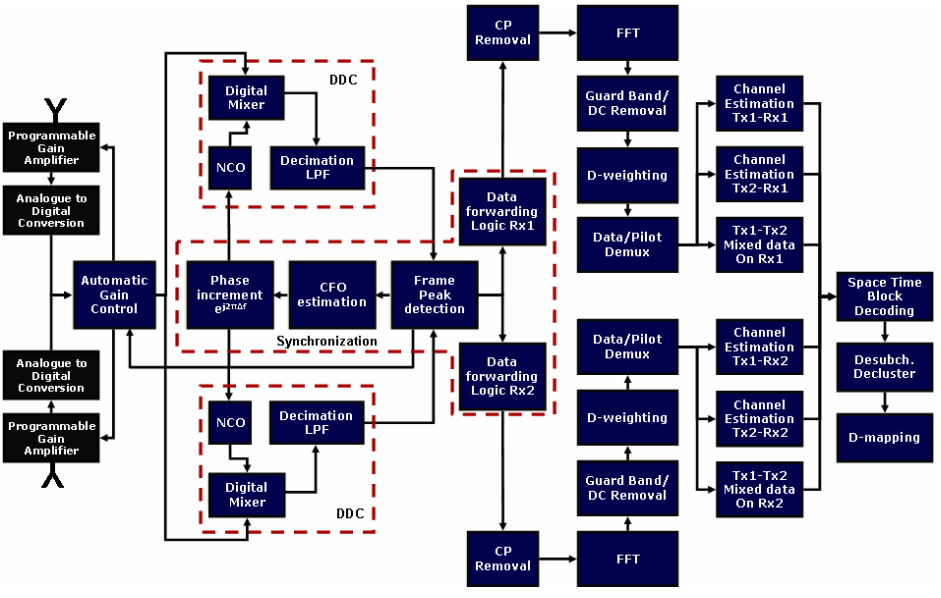


Fig. 3. Overview of the processing components of the receiver

The synchronization block has two main functions. First, it detects the beginning of the data period and, accordingly, the position of the OFDM symbols to apply the FFT. Second, it has to estimate any residual CFO that will be used to finely tune on-the-fly the NCO of the DDC. Both functions can be implemented jointly by calculating the cross-correlation of the received samples. Each OFDM symbol is composed of a Cyclic Prefix (CP) of 512 samples and a useful part of 2048 samples. Since the CP samples copy that last samples of the useful part, it is expected that a peak in the modulus of the cross-correlation $r[n]$ between both sets of samples will appear if the position of the correlation window is correct. An additional mechanism will also prevent the false peak detection due to the presence of the carrier from the coupling of the local oscillators during the silence periods. In addition, the phase of the cross-correlation at the peak will allow calculating the CFO. This can be mathematically expressed as follows (where n_R denotes the number of receive antennas):

$$r[n] = \frac{\sum_{i=1}^{n_R}\sum_{k=0}^{511} x_i^*[n+k]\cdot x_i[n+k+2048]}{\sqrt{\sum_{i=1}^{n_R}\sum_{k=0}^{511}\left|x_i[n+k]\right|^2}\sqrt{\sum_{i=1}^{n_R}\sum_{k=0}^{511}\left|x_i[n+k+2048]\right|^2}}, \tag{3}$$

$$n_{\max} = \arg\max_n \left|r[n]\right|^2, \quad \Delta f = \frac{1}{2\pi}\frac{22.4\cdot 10^6}{2048}\cdot\arg\left\{r[n_{\max}]\right\}$$

In the following processing stages, the OFDM demodulation takes place; this includes the CP removal, the FFT, and the removal of the guard band and the DC carriers. The subsequent blocks are related with the organization, randomization and grouping of the carriers according to the IEEE 802.16e frame definition, and the pilots' extraction which are scattered along the signal bandwidth (where these pilots are used to estimate the channel response). When the system is configured as a MIMO one, each processing-chain at the receiver has to estimate the corresponding channels from all transmit antennas. This is carried out by extracting the channel frequency response at the pilots' positions and then interpolating them by using a second order polynomial interpolation as indicated in formula (4).

$$f(x) = y_{p1} + \frac{(y_{p2} - y_{p1})}{(x_{p2} - x_{p1})} \cdot (x - x_{p1}) + \frac{\frac{(y_{p3} - y_{p2})}{(x_{p3} - x_{p2})} - \frac{(y_{p2} - y_{p1})}{(x_{p2} - x_{p1})}}{(x_{p3} - x_{p1})} \cdot (x - x_{p1}) \cdot (x - x_{p2}) \quad (4)$$

One of the final blocks is related with the decoding of Alamouti's code (matrix A), which is applied on a per-subcarrier basis in 2-transmit antenna systems. The two equations (5) show the operation to be applied for estimating the transmitted symbols at the kth subcarrier. Two consecutive OFDM symbols, $2l$ and $2l$+1, have to be processed jointly, where the samples at each antenna of the receiver after the FFT are represented by R_i and the estimated channel frequency response is denoted by $H_{i,p}$:

$$\hat{S}_N = \frac{\sum_{i=1}^{n_R} H_{i,1}^*[k] \cdot R_i[k,2l] + H_{i,2}[k] \cdot R_i^*[k,2l+1]}{\sum_{i=1}^{n_R} |H_{i,1}[k]|^2 + |H_{i,2}[k]|^2}, \quad \hat{S}_{N+1} = \frac{\sum_{i=1}^{n_R} H_{i,2}^*[k] \cdot R_i[k,2l] - H_{i,1}[k] \cdot R_i^*[k,2l+1]}{\sum_{i=1}^{n_R} |H_{i,1}[k]|^2 + |H_{i,2}[k]|^2} \quad (5)$$

The design has passed through numerous optimization stages in order to boost the performance and minimize the processing complexity (i.e. FPGA slices, RAMB16s utilization). The 2x2 MIMO receiver was fitted in two Virtex-4 LX160 devices, using the Xilinx ISE 9.2 design-suite (FPGA1: 81% slices, 93% RAMB16s, 100% DSP48 and FPGA2: 49% slices, 71% RAMB16s, 57% DSP48). The compilation time of the MIMO configuration reached a peak aggregate of 40 hours in a dedicated 64-bit server. The dense device utilization resulted in a volatile implementation, since the place and routing process only manages to meet timing constraints in an arbitrary way. This limitation is posed by the ISE software and the only solution is to apply more stringent timing constraints and divide the design targeting 3 FPGA devices.

6 Results and Conclusions

The data at the end of the processing chain of the mobile WiMAX receiver is captured and visualized in real-time with the help of the Chipscope Pro software from Xilinx. This software generates monitoring cores which are attached to the target design running in the FPGAs of the testbed. The accuracy of the received constellation points is investigated at the receiver by calculating the EVM, which is a metric of their dispersion from the ideal positions [11]. To compare the performance of the implementation we have also captured and stored in a data file a real signal in the output of the ADCs

which we have then fed to the respective Matlab model of the receiver. Thus, by comparing both results, we can effectively measure which is the degradation in the system performance due to the numerical approximations (finite bit precision) performed by the hardware implementation of the testbed.

An indicative comparison of how the QPSK constellation of the received signal is visualized using Matlab on the one hand and on-board real-time data-capturing on the other, is shown in figures 4, 5 (i.e., SNR = 15.34dB) and figures 6, 7 (i.e. SNR = 21.69dB) respectively. The SNR represents the function of the total signal power over the total noise power across the bandwidth of the received signal.

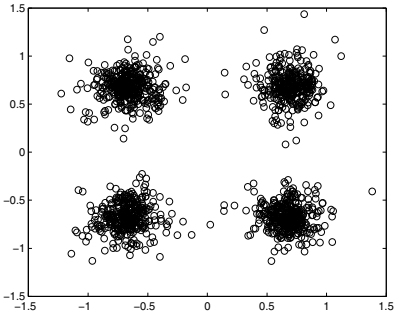


Fig. 4. Matlab-SNR=15.34dB

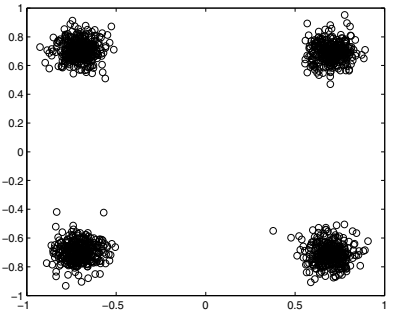


Fig. 6. Matlab-SNR=21.69dB

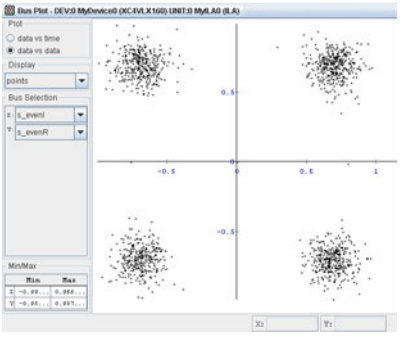


Fig. 5. On-board-SNR=15.34dB

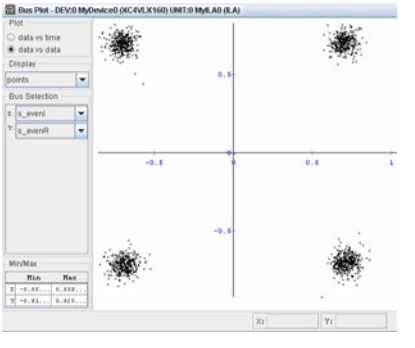


Fig. 7. On-board-SNR=21.69dB

The two previous figures show a capture of the received QPSK constellations at two indicative SNR values. In order to further evaluate the precision and performance of the implemented system, more detailed numerical results have been obtained. These correspond not only to the aforementioned EVM, but also to the raw BER (i.e., assuming no channel coding) for both the Matlab model (i.e., "ideal" receiver) and the actual hardware implementation of the receiver. Note that the EVM is approximately calculated since the deviations of the received constellation points are measured with respect to the taken QPSK de-mapping decisions, instead of the actual ideal error-free constellation points. In this sense, this approximate EVM will be more accurate at

Fig. 8. CTTC real-time testbed. Left: channel emulator, oscilloscope. Rack: (top to bottom) spectrum analyzer, signal generators, RF frontends, baseband boards.

high SNR, i.e., when less errors occur in the decisions. The results are obtained for a fixed channel and for different values of the received SNR by adjusting the attenuation of two noise generators. The EVM and BER curves, which are shown in figures 9 and 10, demonstrate the robustness and precision under implementation losses within an acceptable margin.

The deviation observed in the raw BER curves is due to certain approximations made in the processing chain at the HW receiver, which ultimately affected the EVM calculation; the loss of precision due to the fixed-point implementation of critical building blocks of the system (e.g. channel estimation) deteriorate the constellation on top of the impairment introduced by the observed SNR. In high SNR scenarios, this fixed-point arithmetic-conversion deterioration will only affect few points in the constellation, thus, increasing the raw BER but not equally the approximate EVM, as the QPSK d-mapping decision could be wrong.

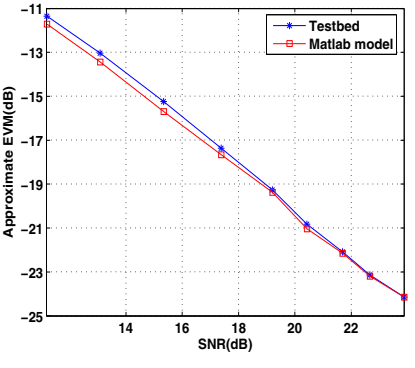


Fig. 9. Performance comparison between the Matlab model and the real-time testbed, through the approximate EVM-SNR curves

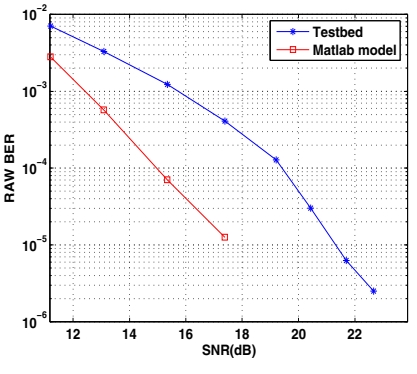


Fig. 10. Performance comparison between the Matlab model and the real-time testbed, through the raw BER-SNR curves

This paper presented a real-time MIMO testbed featuring a mobile WiMAX system. This is rightfully considered a challenging task requiring a resourceful budget, manpower, time and hands-on expertise on advanced signal processing aspects. The testbed comprises the necessary technology, equipment and specifications that allow the implementation of top performance systems. The experimental setup of the 2x2 MIMO testbed operating in CTTC is shown in figure 8. The deployment of an IEEE 802.16e-2005 2x2 MIMO real-time system operating beyond the typical WiMAX specifications (i.e. wave-2), with a 20 MHz channel bandwidth is giving us the opportunity to test and experiment state-of-the-art research concepts.

The next development step will include the implementation of a real-time transmitter replacing the signal generators with a custom FPGA development. This will open several new research and implementation possibilities, such as the inclusion of more advanced MIMO exploitation schemes based on real-time feedback. This feedback from the receiver to the transmitter could deliver information about the current channel conditions and would allow the transmitter to adapt its transmission scheme to such channel, thus, boosting the system performance and making it more efficient. Some schemes to be analyzed are based on antenna selection or adaptive beamforming based on codebooks. Moreover, the feedback will enable us to explore more advanced configurations, such as multi-user networks based on opportunistic transmission, e.g., allocating carriers to users in a dynamic way.

References

1. Mobile WiMAX air interface, P802.16Rev2/D9 (Revision of IEEE Std 802.16-2004 and consolidates material from IEEE Std 802.16e-2005, IEEE Std 802.16-2004/Cor1-2005
2. Kuhling, D., Ibing, A., Jungnickel, V.: 12x12 MIMO-OFDM realtime implementation for 3GPP LTE+ on a Cell Processor. In: 14th European Wireless Conference, Prague, Czech Republic, June 22-25, pp. 1–5 (2008)
3. Alamouti, S.: A simple transmit diversity technique for wireless communications. IEEE Journal on Selected Areas in Communications 45(9), 1451–1458 (1998)
4. Ramirez, D., Santamaria, I., Perez, J., Via, J., Tazon, A., Garcia-Naya, J.A., Fernandez-Carames, T.M., Gonzalez Lopez, M., Perez-Iglesias, H., Castedo, L.: A Flexible Testbed for the Rapid Prototyping of MIMO Baseband Modules. In: 3rd International Symposium on Wireless Communication Systems, Valencia, Spain, September 6-8, pp. 776–780 (2006)
5. Hu, S., Wu, G., Guan, Y.L., Law, C.L., Yan, Y., Li, S.: Development and performance evaluation of mobile WiMAX testbed. In: IEEE Mobile WiMAX Symposium, Orlando, USA, March 25-29, pp. 104–107 (2007)
6. Haene, S., Perels, D., Burg, A.: A Real-Time 4-Stream MIMO-OFDM Transceiver: System Design, FPGA Implementation, and Characterization. IEEE Journal on Selected Areas in Communications 26(6), 877–889 (2008)
7. Ramirez, D., Santamaria, I., Perez, J., Via, J., Garcia, J.A., Fernandez, T., Perez, H.J., Gonzalez, M., Castedo, L., Torres, J.M.: A comparative sutdy of STBC transmissions at 2.4 GHz over indoor channels using a 2x2 MIMO tested. Wireless Communications and Mobile Computing 8, 1149–1164 (2008)
8. Jiménez, V.P.G., García, M.J.F.-G., Armada, A.G., et al.: A MIMO-OFDM Testbed, Channel Measurements, and System Considerations for Outdoor-Indoor WiMAX. EURASIP Journal on Wireless Communications and Networking 2010 , Article ID 871291, 13 pages (2010)
9. Mignanti, S., Castellano, M., Spada, M., Simoes, P., Tamea, G., et al.: WEIRD Testbeds with Fixed and Mobile WiMAX Technology for User Applications, Telemedicine and Monitoring of Impervious Areas. In: Proceedings of TridentCom 2008, Innsbruck, Austria, March 18-20 (2008)
10. Nieto, X., Ventura, L., Mollfulleda, A.: GEDOMIS: A Broadband Wireless MIMO-OFDM Testbed. Design and Implementation. In: Proceedings of. TridentCom 2006, Barcelona, Spain, March 1-3, pp. 121–131 (2006)
11. Shafik, R.A., Rahman, M.S., Islam, A.H.M.R.: On the Extended Relationships Among EVM, BER and SNR as Performance Metrics. In: 4th International Conference on Electrical and Computer Engineering, Dhaka, Bangladesh, pp. 408–411 (December 2006)

ASSERT: A Wireless Networking Testbed

Ehsan Nourbakhsh, Jeff Dix, Paul Johnson, Ryan Burchfield, S. Venkatesan, Neeraj Mittal, and Ravi Prakash*

Distributed Systems Lab, Computer Science Department,
The University of Texas at Dallas 75080 USA
{ehsaan,jxd028100,paj041000,ryanb,venky,neerajm,ravip}@utdallas.edu

Abstract. As wireless networks become a critical part of home, business and industrial infrastructure, researchers will meet these demands by providing new networking technologies. However, these technologies must be tested before they can be released for mainstream use. We identify the key design considerations for a wireless networking testbed as *a*) accuracy *b*) controllability *c*) mobility *d*) repeatability *e*) cost effectiveness *f*) data collection *g*) resource sharing *h*) multi-nodal capability *i*) scalability. In this paper we portray how we have used coaxial cables and our custom hardware of RF switches and programmable attenuators to create Advanced wireleSS Environment Research Testbed (ASSERT), addressing the above requirements. ASSERT supports various types of wireless devices, providing researchers in academia and industry with the necessary experimentation tools to validate their designed protocols and devices.

Keywords: ASSERT, Wireless, Sensor, Testbeds, Repeatability.

1 Introduction

As wireless networking is becoming more pervasive, there has been a greater desire to develop communication hardware and protocol stacks that have a number of desirable properties like increased throughput, reduced latency, reduced energy consumption, quality of service, security, etc. Consequently, several academic and industrial research groups are actively working towards improving the performance of wireless networks. Due to their inherent complexity, accurate theoretical analysis of the performance of large wireless networks is quite challenging. Hence, several researchers have resorted to simulation experiments to evaluate the performance of large wireless networks. Most simulators make a set of simplifying assumptions about the communication medium and the communication protocols [1, 2, 3, 4]. This enables them to run experiments within a reasonable amount of time. However, sometimes these assumptions can bias experiments in a significant way. It is no surprise that often the results of

* ASSERT was designed in collaboration with Crane Wireless Monitoring Solutions and funded by the US Department of Defense, Defense Microelectronics Activity (DMEA).

T. Magedanz et al. (Eds.): TridentCom 2010, LNICST 46, pp. 209–218, 2011.

simulation experiments differ significantly from the actual performance of wireless networks.

Over the last few years several research groups have initiated the development and deployment of wireless networking testbeds such as Netbed [5], Kansei [6], Trio [7], ExScal [8] and other testbeds at U.C. Berkeley [9]. The underlying assumption of all these endeavors is that experiments conducted on a testbed composed of actual wireless devices communicating over the air will yield results representative of performance in field deployments. Several of these testbeds are deployed in general-purpose laboratories in academic buildings. As these laboratories are not shielded from external wireless interference, the experimenters have little or no control over the environment in which experiments are conducted. As a result, it is almost impossible for experimenters to independently reproduce the results obtained by other research groups. Building a Faraday cage large enough to house wireless networks of non-trivial diameter is prohibitively expensive. Outdoor deployments, unless sufficiently ruggedized, can deteriorate quickly due to variations in temperature and humidity. Moreover, it may not be possible to conduct outdoor experiments during inclement weather. Also, innovative ideas need to be employed to conduct mobile networking experiments if one does not have a lot of manpower available.

Based on the above discussion, one can count some general requirements for an ideal testbed, similar to considerations that De et al. in [10] proposed for a multihop testbed. The ideal testbed shall: *a*) accurately reflect wireless network behavior (**accuracy**) *b*) provide enough control to configure topology and environment conditions (**controllability**) *c*) emulate mobility of the nodes (**mobility**) *d*) conduct experiments that are reproducible and easily repeatable (**repeatability**) *e*) be cost effective in terms of hardware, manpower, space and time requirements to set up, run experiments on and maintain (**cost effectiveness**) *f*) provide necessary tools to collect and analyze data (**data collection**) *g*) be able to share the available resources to conduct multiple experiments without interfering with each other (**resource sharing**) *h*) have **multi-nodal capability** (i.e., it will support many types of nodes) *i*) have the ability to scale to a large number of nodes (**scalability**).

In the next sections, we will present some clear examples of challenges in designing and building our large-scale testbed called ASSERT (**A**dvanced wirele**SS** **E**nvironment **R**esearch **T**estbed). In Section 2 we show how other testbeds have worked towards some of the above requirements. Section 3 will present some of the main characteristics of our work and show how we are able to satisfy the above requirements. Following this overview, we will describe the architectures selected for both our hardware and software implementations of the ASSERT in Section 4. Some final conclusions and future work are discussed in Section 5.

2 Related Work

Creating an environment to test and validate new protocols and hardware designs has been a challenge for wireless researchers. The desired environment

should be precisely controllable and the possibility of repeating the same experiment is vital. The first category of testbeds, such as MoteLab [11] and ORBIT [12], attempted to create such environment by focusing on using the antenna of unit under test (UUT), resulting in over-the-air transmissions. Researcher is not able to control the exposure of the UUT to background noise and interference from other nodes, so **controllability** requirement is not addressed . The distance between nodes is limited to the physical placement of the devices, and **mobility** is not provided by design. Although the size of the network they are able to create is large, they are not able to partition them effectively, thus failing to address **resource sharing** requirement. Noise injection and MAC filtering can be used to create topologies, but as [13] pointed out **repeatability** and reproducibility of results from noise injection is reduced if nodes with marginal Signal-to-Noise Ratio (SNR) are involved. Using MAC filtering, as suggested by some researchers to emulate mobility, will also fail to address **mobility** because in this method either a packet is able to go through or is completely dropped, unlike actual movement. Mint-m [14] and Mobile Emulab [15] address the **mobility** and **controllability** requirements by using added attenuation between UUTs and antenna and small robots to move the UUTs around the test area. Pharos [16] uses a similar approach of using robots, but the environment is set outdoors. These approaches still do not eliminate the problem of exposure to background noise or interference from other testbed devices. This would fail to address **controllability** requirement. Also, **mobility** is either limited to the speed of the robots or is not supported as a design feature.

A second category of efforts such as work done in CMU [17] digitize the outgoing signal of the UUTs and use existing RF propagation models to emulate effects such as distance and multipath on the signal. The resulting altered signal is fed to the destination devices. This approach provides the required **controllability** requirement, but its **accuracy** is limited to the precision of the applied RF propagation model. Furthermore, signal alternation requires sophisticated calculations so higher number of nodes will make the processing power requirements harder to achieve, failing to address the **cost effectiveness** requirement.

The third category of testbeds focus on simply using coaxial cables to connect different nodes. Attenuators and RF switches can be used to form topologies and emulate distance between nodes. MeshTest [18] is one example of such set up with a design close to our work. However, their method requires complex design for higher number of nodes. High cost of the switch matrices used might be another obstacle to building larger networks, failing to address **cost effectiveness** requirement.

3 Design Overview

Our testbed currently consists of forty devices, known as "sites" in this context. Each site consists of a microprocessor and its peripherals, also a Field Programmable Gate Array (FPGA) as well as a set of 16 attenuators and corresponding RF connectors. The site processor is running Linux and is connected

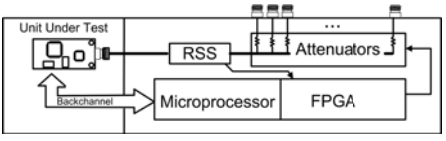


Fig. 1. Block diagram of one site and the related RF, data and command lines

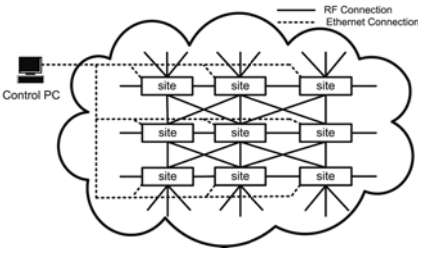


Fig. 2. ASSERT RF Grid and control plane

to an Ethernet network for network storage and communication. Each connector can be connected to another site using a coaxial RF cable. The main functionality of the FPGA is to set the attenuators on each RF connector as instructed by the microprocessor and poll the RSS meter on a millisecond basis. The attenuators are set to values either calculated based on a theoretical model, or based on real life recorded attenuation values such as work done by Lee et al. [19] addressing the **accuracy** requirement. This real life readings can be gathered once for some generic scenarios and then used multiple times to compose more complex terrains.

Each UUT is connected to a site, and the site connects it to up to 16 other sites through its internal switches and attenuators. This 1-to-16 connection means we are able to add more sites and enhance our testbed only by adding sites, addressing the **scalability** requirement. We synchronize the clock on sites through our clock distributors so that we are able to set the attenuations on these 16 connectors, called ports, almost instantly on all sites. This will allow us to control how the sites are "virtually" moving in reference to each other with a high precision, addressing the **mobility** requirement. A block diagram of a site is demonstrated in Figure 1. As shown in Figure 2, a front-end computer called Control PC is connected to all sites through Ethernet and is responsible for getting experiment definition from user, sending control information and gathering the results. Our design choices ensure we are able to scale to at least one thousand nodes without unreasonable processing power requirements.

3.1 Reproducibility and Repeatability

As demonstrated in [20], reproducibility of experimentation can be difficult due to inconsistency in environmental conditions. Many wireless networking testbeds operate in schools and laboratory environments, where non-testbed devices can interfere with an experiment. But for a testbed to provide reproducible experiments, we need the ability to control the noise around the testbed, so that we can compare different testbed experiments. As we have stated above, our solution to this problem is to place all communication between devices on coaxial cables. By properly shielding a wireless device, and connecting it to our coax network, we have control over which devices can "see" each other in the network. We can

also prevent outside leakage from other wireless networks operating on the same band.

We would like to emulate effects the environment has on signal reception, such as multipath. Each site processor will calculate the attenuation for each link due to the selected pattern by the user and set the link's attenuators accordingly on both sites on a millisecond basis. All the parameters for the link's attenuation patterns, including random number generator seeds, are stored in a Control PC's database and can be retrieved again to rerun the experiment. Using the same concept, we can also emulate mobility by changing each link's attenuation dynamically to emulate the changing environment, or neighbors.

3.2 Multi-nodal Capability and Data Collection

As evident in the previous sections, there are only some general expectations we have from the unit under test. We consider the UUT a black box, a consideration which is vital for the **multi-nodal capability** requirement. We expect the UUT to transmit in the frequency range our RF equipment is designed to work. We also expect it to have an antenna that can be replaced by a coaxial cable, so that we can connect the RF transmitter to one of our testbed sites. The other optional requirement is to have a RS-232 interface so that the UUT can receive commands, such as *reset* or *load image*, from the site. This interface can be also used for the data collection mechanisms we have provided. All logged data written by the UUT to the RS-232 serial interface is kept on file along with RSS meter readings, with timestamps of each log message, and is returned by the testbed software as part of the experiment results. This combination of correlated data can be an important tool for the researcher, addressing the **data collection** requirement.

3.3 Experiment Setup and Execution

We provide more details about creation and execution of an experiment through a common use case of our testbed. We currently have twenty five Crossbow MICA2 motes installed in our testbed. The first step a user has to take to create a new experiment is to create the corresponding topology script. This script specifies which sites are involved in this experiment, what are the attenuation values and how these values change over time. As an example, user might select ten motes and name them according to their own plan. Then use the attenuation values gathered by Lee et al. [19], recorded specifically for MICA2 motes, to place nodes realistically. The user might also decide that a node A slowly moves away from its neighbors at time T. This movement can be realized in the script by using a step function to increase attenuation of all its links at time T until it is disconnected. Same can be done to add new nodes or adding background noise to some links using a normal distribution attenuation pattern. The integrated signal generator of the sites can be used to act as a constant noise source. By changing quality of the links from these noise sources to each site we can control amount of noise the UUT inside that site experiences.

The other parameters that the testbed will ask from user are the number of times the experiment has to run, and the image to be flashed on all UUTs. It is worth mentioning that the user only compiles one generic image, and the testbed adjusts it for each UUT based on experiment setup. Once all sites are flashed and have received the experiment details formatted as an XML file, the experiment starts. Each site will report to the Control PC when the end time of the last attenuation indicated in the XML descriptor has passed. Once Control PC has got the successful termination signals from all the sites in an experiment, the user is notified and the log files created during the experiment are made available to them. If any of the sites encounter a fatal error during an experiment, it will notify the Control PC. The Control PC will then terminate the experiment early, and notify the user that a problem has occurred.

The previous methods give the necessary support for the user to easily and quickly set up one or multiple runs of an experiment, or repeat an existing one. This is to address the **controllability** requirement mentioned earlier. It also is addressing the **cost effectiveness** requirement. We are significantly reducing the amount of time that the experimenter has to spend to create an experiment on the testbed. As we reserve the required sites during the experiment and also set the attenuation levels to maximum on unused links, we are able to partition the testbed and run multiple experiments in different parts of it. This will increase the utilization of the testbed, addressing the **resource sharing** requirement.

4 System Architecture

4.1 Hardware Architecture

It is best to think of ASSERT hardware as a graph (as in Figure 2), with nodes representing sites in the testbed, and edges representing RF links between sites. Each site consists of a custom digital board (Figure 3) and a custom RF board (Figure 4). The digital board has a processor, memory, FPGA and serial interfaces where the UUT can be connected. The processes executing on the digital board can control the operations of the UUT, monitor the experiment, and gather results as described in Section 4.2. The RF board connects to the digital board and provides an interface through which the antenna port of the UUT can connect to the RF board. The UUT interface leads to a 1×16 power divider/combiner. Each output port of the power divider/combiner leads to a programmable attenuator (controllable by the digital board) which can provide signal attenuation between 0dB and 63.5dB, in steps of 0.25dB. The attenuators from two different RF boards can be connected via a coaxial cable forming an RF link between two sites. Thus the signal on this link can be attenuated in the range 0dB to 127dB: a maximum of 63.5dB attenuation provided by each of the two programmable attenuators on the path.

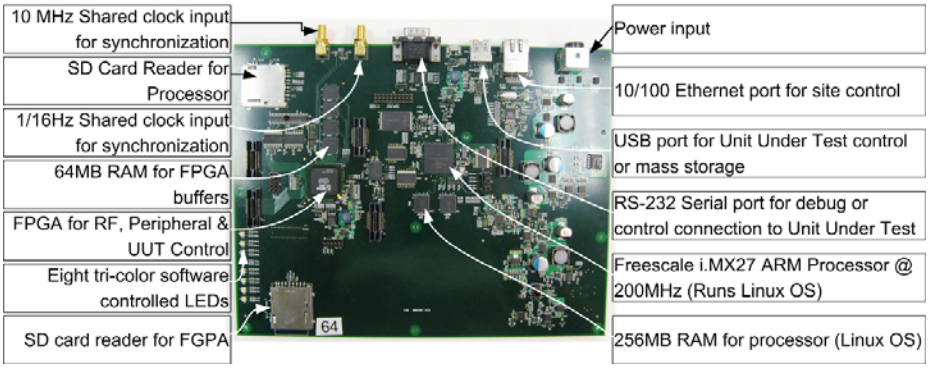


Fig. 3. One Site Board. Each Site Board is paired with one RF board

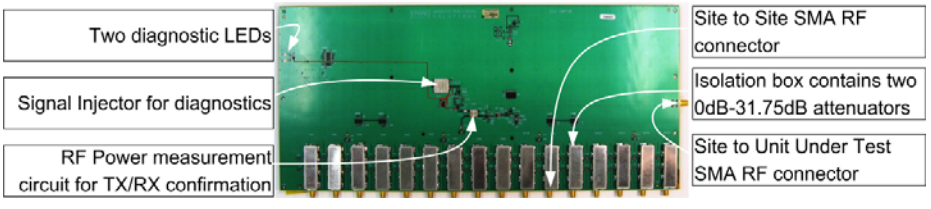


Fig. 4. One RF Board

Continuing with the graph theoretic description, the digital and RF boards together correspond to a node with a maximum of sixteen incident edges. The components on the RF board can currently support all communication bands between 720 MHz and 1125 MHz. In the near future we plan to extend the support to the 2.4 GHz ISM band. The sites, each with a degree of sixteen, are connected to form a mesh. The quality of links can be manipulated by changing the level of signal attenuation as rapidly as one db every millisecond.

4.2 Software Architecture

The software is divided into *slices*, with each slice implementing a specific functionality. We now describe the software architecture in the context of the creation and execution of a user's experiment.

The *diagnostics slice* runs both periodically and on demand to check the integrity of RF links on the testbed. This involves selecting sites sequentially, having them send signals along each incident link, and measuring the received signal strength for different values of attenuation along the link. This slice registers a link in the database between two sites if both can hear the signal generated by each other. The quality of this link can be determined as the strength of the received signal compared with the strength of generated signal.

The *user interface slice*, running on the central controller provides a graphical user interface to the users. The network topology can be selected from a library of topologies (like mesh, star, ring, etc.) provided by the user interface. If the

topology the user wishes to emulate is not present in the library, the user can specify it as a graph with vertices and links between vertices. The user can also specify the characteristics of the wireless links between vertices. Once again, link characteristics can be specified either by selecting from a library of fading patterns, or by providing the formula to define the link characteristics.

Once the user has specified the desired topology, the user interface slice invokes the *experiment control slice*. The experiment control slice first gathers the current state of the testbed by querying the *system state slice*. The system state slice returns the state of all the sites as well as the set of reservations currently running on the testbed. Then the experiment control slice invokes the *topology mapper slice*. The topology mapper slice computes the topology based on user input as a subgraph of the portion of the testbed that is not running any experiment. If the topology mapper slice is successful, the experiment control slice invokes the *reservation slice*. The reservation slice reserves the corresponding testbed sites for the experiment. These reservations are implemented as leases of finite duration. If the experiment needs to run longer than the lease duration, the lease must be renewed prior to its expiration.

Then the experiment control slice invokes the *attenuator control slice* at all the reserved sites. As part of this invocation, the experiment control slice informs each reserved site about the properties of the links incident on it. To implement the desired link characteristics the sites at either end of the emulated wireless link work cooperatively. Consistent with the producer-consumer model employed by operating systems, acting as a producer each site generates a sequence of attenuation values along with the time offset from the beginning of the experiment when these attenuation values are to be used on a link. Acting as a consumer, the FPGA on the site hardware reads these values and sets the attenuation values accordingly once it is informed that the experiment has started. Concurrently, the experiment control slice informs the unit under test to start executing the experiment through the *UUT control slice*. Thus, as the unit under test is running the experiment and sending and receiving messages along the emulated wireless links, the attenuator control slice is manipulating the characteristics of these links.

For the entire duration of an experiment running on ASSERT, the *system state slice* monitors the state of the sites. The *logging slice* records all error and control messages generated by all the participating sites. The *UUT logging slice* records information that the UUT writes to serial port of the site as it is running. This data can be used for debugging the UUT by the researcher. Finally, on the completion of the experiment the experiment control slice invokes the *data retrieval slice* to gather the results of the experiment from all the participating sites. which are then conveyed to the user via the user interface.Experiment control slice instructs the reservation slice to release all the sites reserved for the experiment.

As a security measure, and to block access to a user's code by other users, the sites can be re-flashed to original configuration, wiping all user data and settings from them. Similar measures are also designed in the Control PC stopping users

from having access to data of others. Furthermore, all the data including experiment details and results can be stored on one single portable hard drive. Also physical and remote access to the facility can be restricted, effectively providing a secure environment for performing experiments that require extra security. In a nutshell, all the software slices together can be thought of as the operating system for ASSERT.

5 Conclusion

The Advanced wireleSS Enviroment Research Testbed (ASSERT) has a small footprint, and emulates mobility and link deterioration inside a room in a repeatable manner. All the experiments are controlled through front-end computers, and network topology can be modified through a sequence of keystrokes and mouse clicks. This takes significantly less time than physically changing the topology in existing over-the-air wireless networking testbeds. ASSERT is immune to interference from other devices in the laboratory and the environment. It will be possible for experimenters to inject noise or the desired interference into the system, and observe their impact on the system being studied. Communication between nodes in the testbed does not leak into the environment. We perform all the signal manipulation purely in the RF domain. This allows us to scale to higher number of nodes. Furthermore, while an RF emulator does all the calculations in a central location for all nodes, our solution is decentralized as we are able to break down the attenuation changes into tasks for each site. With ASSERT it is possible to conduct experiments in licensed bands like the cellular service band without interfering with the services offered by the owners of these licensed bands. Through sophisticated custom hardware and easy-to-use control software ASSERT has many valuable features that allow it to reduce the cost of testing wireless networking protocols at scale.

Future Work: ASSERT currently consists of forty nodes, it is designed to scale to at least one thousand nodes without any design changes. We are converting our user interface (UI) from current small Java program to a web based application, so that the experiment set up and data gathering is done by the experimenter from their browser. The other goal is to add more preset topologies, so that the experimenter has an extended database of already existing topologies, while they are always able to define their own topology.

References

1. Kotz, D., Newport, C., Gray, R.S., Liu, J., Yuan, Y., Elliott, C.: Experimental Evaluation of Wireless Simulation Assumptions. In: Proceedings of MSWiM 2004 (2004)
2. Pawlikowski, K., Jeong, H., Lee, J.: On Credibility of Simulation Studies of Telecommunication Networks. IEEE Communications Magazine 40(1) (2002)

3. Cavin, D., Sasson, Y., Schiper, A.: On The Accuracy of MANET Simulators. In: Proceedings of the Second ACM International Workshop on Principles of Mobile Computing (2002)
4. Skehill, R., Scully, P., McGrath, S.: Characteristics, Results and Findings of IEEE 802.11 in an RF Isolated Testbed. In: Proceedings of PIMRC 2007 (2007)
5. White, B., Lepreau, J., Guruprasad, S.: Lowering the Barrier to Wireless and Mobile Experimentation. In: Proceedings of HotnetsI (2002)
6. Ertin, E., Arora, A., Ramnath, R., Naik, V., et al.: Kansei: A Testbed for Sensing at Scale. In: Proceedings of the Fifth International Conference on Information Processing in Sensor Networks (2006)
7. Dutta, P., Hui, J., Jeong, J., Kim, S., Sharp, C., Taneja, J., Tolle, G., Whitehouse, K., Culler, D.: Trio: enabling sustainable and scalable outdoor wireless sensor network deployments. In: Proceedings of IPSN 2006 (2006)
8. Arora, A., Ramnath, R., Ertin, E., et al.: Exscal: Elements of an extreme scale wireless sensor network. In: Proceedings of 11th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications (August 2005)
9. Soda Hall wireless and sensor network testbeds, `http://www.millennium.berkeley.edu/sensornets/`
10. De, P., Raniwala, A., Sharma, S., cker Chiueh, T.: Design Considerations for a Multihop Wireless Network Testbed. IEEE Communications Magazine 43(10) (2005)
11. Werner-Allen, G., Swieskowski, P., Welsh, M.: MoteLab: A Wireless Sensor Network Testbed. In: Proceedings of the 4th International Symposium on Information Processing in Sensor Networks (2005)
12. Raychaudhuri, D., Seskar, I., Ott, M., Ganu, S., et al.: Overview of the ORBIT Radio Grid Testbed for Evaluation of Next-Generation Wireless Network Protocols. In: IEEE Wireless Communications and Networking Conference 2005, vol. 3 (2005)
13. Kaul, S., Gruteser, M., Seskar, I.: Creating Wireless Multi-hop Topologies on Space-Constrained Indoor Testbeds Through Noise Injection. In: TridentCom 2006 (2006)
14. De, P., Raniwala, A., Krishnan, R., Tatavarthi, K., et al.: MiNT-m: An Autonomous Mobile Wireless Experimentation Platform. In: Proceedings of the 4th International Conference on Mobile Systems, Applications and Services (2006)
15. Johnson, D., Stack, T., Fish, R., Flickinger, D.M., Stoller, L., Ricci, R., Lepreau, J.: Mobile Emulab: A Robotic Wireless and Sensor Network Testbed. In: Proceedings of INFOCOM 2006 (April 2006)
16. Stovall, D., Paine, N., Petz, A., Enderle, J., Julien, C., Vishwanath, S.: Pharos: An Application-Oriented Testbed for Heterogeneous Wireless Networking Environments. The University of Texas at Austin, Tech. Rep. TR-UTEDGE-2009-006 (2009)
17. Judd, G.: Using Physical Layer Emulation to Understand and Improve Wireless Networks. Ph.D. dissertation, Carnegie Mellon University, School of Computer Science (available as Tech Report CMU-CS-06-164) (2006)
18. Clancy, T., Walker, B.: MeshTest: Laboratory-Based Wireless Testbed for Large Topologies. In: TridentCom 2007 (2007)
19. Lee, H., Cerpa, A., Levis, P.: Improving Wireless Simulation Through Noise Modeling. In: Proceedings of IPSN 2007, p. 30 (2007)
20. Burchfield, R., Nourbakhsh, E., Dix, J., Sahu, K., Venkatesan, S., Prakash, R.: RF in the Jungle: Effect of Environment Assumptions on Wireless Experiment Repeatability. In: IEEE International Conference on Communications, ICC (2009)

AMazING – Advanced Mobile wIreless playGrouNd

João Paulo Barraca, Diogo Gomes, and Rui L. Aguiar

Instituto de Telecomunicações – Polo de Aveiro
{jpbarraca,dgomes,ruilaa}@av.it.pt

Abstract. We describe a wireless testbed composed by 24 wireless nodes that can be used to perform a broad range of studies in the area of next generation networks. This paper addresses the difficulties and constrains faced by the authors throughout the deployment process of such testbed. Flexibility and controllability were key concerns driving the testbed design. The testbed can be remotely managed through a series of remotely accessible web services performing low-level management. Validation results are presented, showing the interference levels of the testbed as well as its maximum throughput capabilities.

Keywords: testbeds, wireless network, outdoor systems.

1 Introduction

Wireless Networks, and especially mobile networks, have been a hot-topic in the research community during the past years. Most of the associated research work has been carried out through simulations due to difficulties associated with conducting real experiments. Such difficulties include (but are not limited to) cost of equipment, complexity of scenarios, availability of a lab with the required conditions, human resources required to conduct the experiment and reproducibility difficulties in wireless environments.

Simulation tools have therefore been used to conduct such studies in a controlled manner with far less effort. However, with increasingly complex systems, the research community has questioned in recent years the accuracy of such simulations, based on models of the wireless reality. This led to an increased interest in deploying testbeds were wireless studies could be conducted under more realistic – but still controllable - conditions. Such testbeds constitute an intermediate step between simulations/emulations and a full-scale prototype. The objective of such testbeds is to provide the means for the research community to validate their concepts and theories in an environment that better matches a real scenario.

Several testbeds exist around the world, with different strengths and weaknesses. In this paper, we discuss a new testbed deployed for supporting research on next generation networks (NGN), which intends to differentiate itself from the existing ones by increased controllability (for the experimenter) and high reproducibility of the experiments. The AMazING testbed is an outdoor system, Operating System agnostic, which the authors have designed and deployed in the rooftop of Instituto de Telecomunicações in Aveiro, Portugal with a reasonably low deployment cost.

T. Magedanz et al. (Eds.): TridentCom 2010, LNICST 46, pp. 219–230, 2011.

In section 2 we review previous work in wireless testbeds and identify some of the problems associated with such testbeds. In section 3 we establish the requirements we had for our testbed and describe the solutions developed both in terms of Hardware and Management Platform. We present the results of a brief performance evaluation on the throughput of the deployed testbed in section 4 and conclude the paper in section 5 with an assessment of the most important features of the testbed and draw some guidelines for future work.

2 Wireless Testbeds

Much research effort was put and is increasingly being put into the development of wireless solutions. Over the time this created optimized solutions for multiple scenarios such as metropolitan area, campus, industry, our homes, or even our personal space. Despite all the effort spent in developing new solutions and evaluating new scenarios, there are still doubts about the limits and applications that wireless can support. This is in part due to the fact that the wireless medium is much less reliable, unpredictable, and hard to evaluate than wired mediums.

Typically, solutions for wireless networks are first validated in a discrete simulation environment. After this initial validation by simulation, it is emulated with a real world prototype and then implemented in a final product (if useful). There are many such simulation platforms, with NS-2 [1], dominating the academia world. These simulation tools are vital to the advance of the state-of-the-art in most networking areas, because they allow protocol evaluation to be performed in a reasonably controlled platform, unaffected by external variables and thus enabling some controlled repeatability. All simulators implement simplified representations of the real world models, which need to be validated [2]. Due to their inherent limitations, these simplified models constrain the quality of the results obtained. With current models and the complexity of existing systems, practice shows that simulation results should be increasingly taken with a grain of salt, in particular because minor (unseen) details may result in misleading or incorrect answers. The amount of synthetic validation through simulation increased in many cases without a clear increase in the quality of the results presented [3].

Fast deployment in a more real environment can rapidly exclude unreal results.

Thus, in recent years there has been a major interest around wireless testbeds. With decreasing equipment prices, it became easier to evaluate solutions in scenarios closer to the real world. Moreover, with testbeds appearing as a reasonably efficient tool, increased effort in testing, stressing, observing, tuning and validating solutions became normal. For high profile scientific work, simulations alone are no longer sufficient to support the results. Several research institutions, enterprises and universities have built testbeds aiming to create a reproducible evaluation environment. These initiatives usually have in their core either multi-purpose experimental platforms or simply custom proof-of-concept platforms. The first aim to provide an environment able to evaluate a multitude of solutions and scenarios while proof-of-concept platforms target advancing a particular research objective and exist until the solution reaches market or is abandoned.

The recent history of multi-purpose experimental wireless testbeds starts in the late 90s. Ad-hoc networks were on the rise and the MONARCH [3] project was created. It consisted of 2 fixed nodes at CMU premises, and 5 mobile nodes installed in rented cars circulating in a nearby road. While a little crude and with low repeatability, this early effort had a major impact. in many networking areas related to mobility and wireless communications, such as MobileIPv6 and Dynamic Source Routing. At UCLA the NRL [4], aimed at developing ad-hoc solutions, comprised 20 laptops and 60 PDAs using 802.11b cards. It spans the entire campus and was integrated with a simulation platform able to evaluate hybrid scenarios. MIT Roofnet [5] consists of an experimental 802.11b/g network of 20 off-the-shelf nodes providing connectivity to users in Cambridge. As another example, the University of California created a wireless testbed aimed at low-level radio research, with more than 100 nodes, with very different capabilities and access technologies. ORBIT [6], that includes over 400 nodes in a 20x20 meter area, is currently one of the most complex wireless testbeds. On a different scale, but also relevant there is OneLab [7] add-ons to PlanetLab, that has extensions to support evaluation of wireless solutions over its general-purpose network infrastructure. In most of the existing wireless testbeds nodes are placed very close to each other. In such testbeds it is very difficult to create scenarios in which sets of nodes are hidden from each other. Such scenarios are only possible in such testbed through the use of MAC filtering mechanisms that can severely interfere with the experiment results.

There are many other examples of wireless testbeds around the world. This diversity is now leading to efforts of federation of these testbeds, trying to reuse as much as possible common tools and data representation.

All the testbeds mentioned have inherent flaws and limitations. Most rely on MAC filtering techniques, which is unable to discard radio interference effects. Often they are only able to support static scenarios, and the few that have mobile nodes are not able to provide reasonable repeatability of the experiments, impairing the study of mobile phenomena. NRT aims at supporting integration with a simulator but it is unknown when this will be fully functional.

3 The AMazING Testbed

In the past the authors have experienced with custom purpose testbeds for next generation networks, such as the ones used for project IST-Daidalos [8]. This experience provided valuable knowledge about the requirements for multiple purpose test systems able to evaluate new NGN concepts through prototypes. It became apparent the need for a new testbed, that could be used by multiple NGN research projects without ever limiting the complexity of the experiments through artificial management rules and hardware constrains.

In particular it was clear from our previous experience that a NGN wireless testbed should abide to two key requirements:

Provide Administrative Privileges for Users: Most of the existing testbeds are either private, and therefore fully accessible by their owners, or public but provide a limited set of functionalities to users. Ultimately meaning that only a limited set of experiments can be conducted, which is worrisome for NGN research. To restrict access to foreign users is usually an administrative decision, based on the need to

setup a monitoring and management infrastructure, capable of controlling the testbed usage. We have made a requirement of our testbed to fully disclose access to the nodes, making it possible to foreign users accessing all the functionalities made available by the hardware platform deployed.

Provide a Reproducible Environment: – A NGN testbed, albeit not completely isolated, has to be very predictable, with small variance in the tests. This could only be obtained in an open environment in which no objects would flow through the node and where radio interference would be set to a very low level (bellow what could be considered interference). Most testbeds exist in laboratory environments,and are affected by normal people movement,regular electromagnetic interference, and interference from multiple experiences.
Besides these key requirements, several other issues must be taken inconsiderations:

x86 Compatible Nodes: Of the several existing computer architectures the most popular amongst developers is the x86 architecture. This fact is easily realized by the sheer amount of publicly available software implementations in the area of networking, and most NGN research uses network nodes based on x86. By making it a requirement, we are guaranteeing that a broader set of testbed users can easily deploy existing software.

Support for Multiple Radio Interfaces: The focus of the testbed is in creating a wireless testbed for NGN, thus potentially covering several wireless technologies, existing and future. Each node must therefore be flexible enough to support different radios. Ultimately this must be translated into mainboards with multiple hardware interfaces, in which the radios can be connected to.

Access to Low-level Radio Interfaces: In addition to the existence of multiple radios, it is also required that each of the chosen radio interfaces provides programmatic access to low-level aspects such as the MAC (a common research issue). Ultimately the desirable radio interface should be a fully software-defined radio, but this is not simple to support currently.

Extendable: The nodes processing capabilities must be extendable, and easily linkable to current and future core network solutions. This implies that a smooth interface for the testbed nodes should exist, and that the wireless interfaces of the nodes should be easily accessible from core machines.

Reduced Radio Interference: An obvious corollary for the reproducibility requirement. The amount of interference a wireless testbed is usually subject to, can lead to important variations in the results obtained. It is necessary that the nodes receive as few interference as possible, and that they cause as little interference on their neighbors as possible.

Low Power and Cost: The amount of nodes immediately leads to concerns related to power consumption. It is a good practice for each node to consume as less power as possible. To this end, there is a strong concern on the amount of power supplies involved, as well as on the power requirements of the computers and radio interfaces used. Furthermore, it is expected that the testbed should be deployed with low cost CoTS (Commercial of-The-Shelf) devices.

Based on these requirements several hardware-based solutions were evaluated, and a management platform was specified. In the next sections the chosen hardware

platform and management platform solutions are detailed, together with the reasons behind each of the choices.

3.1 Hardware and Deployment Aspects

The key design decision is related with the need to have a clean, predictable environment, with an easy access location. The solution found was to place the testbed outdoors in the rooftop of a building, reasonably insulated from common interference sources, and isolated from human movement interference. This also had the advantage of providing a reasonable large area for the testbed deployment.

3.2 Hardware Deployed

This key decision led to the AMazING (Advanced Mobile wIreless Network playGround) testbed, which consists of 24 fixed nodes located at Instituto de Telecomunicações (IT) - Aveiro rooftop, as depicted in Figure 1. The testbed spans over 1200m2, and nodes are distributed across the area forming a grid with approximately 8m between each neighbor.

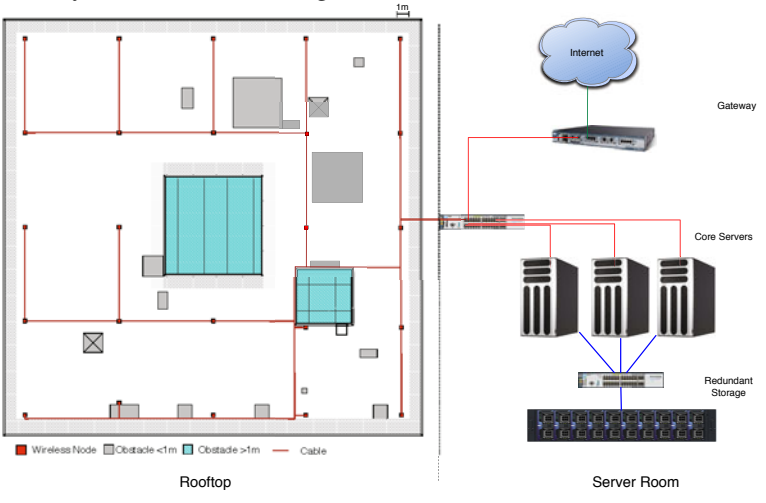


Fig. 1. Architecture of the AMazING testbed with detail over the spatial deployment on the rooftop

The system is exposed to the weather conditions of a south European coastal region (frequent sun, and humidity frequently higher than 85%), therefore imposing the use of watertight, and UV resistant, enclosures (IP65). However, Polycarbonate enclosures have a very low heat transfer coefficient (~0.21 W/(m2K)), which is insufficient for the heat produced by the systems. Solar heat gain was also taken in consideration, especially because temperature in this region can easily reach more than 40ºC during summer time. The solution we found (see Figure 2) was to use one

fan and two openings, protected with particle filters. In order to reduce dangerous water condensation, the main board was placed in an inverted position and near the top of the enclosure. Heat generated by the electronics (in particular the CPU) keeps the board always above dew point. The final solution involved developing a protective cap made of waterproof maritime plywood, to reduce solar load, and allow air heat exchange with the environment. Figure 3 depicts the temperature differential between outside environment and the interior of a node during three days of sunny weather. As it can be seen, the difference remains stable and inside temperature only varies by less than 10ºC.

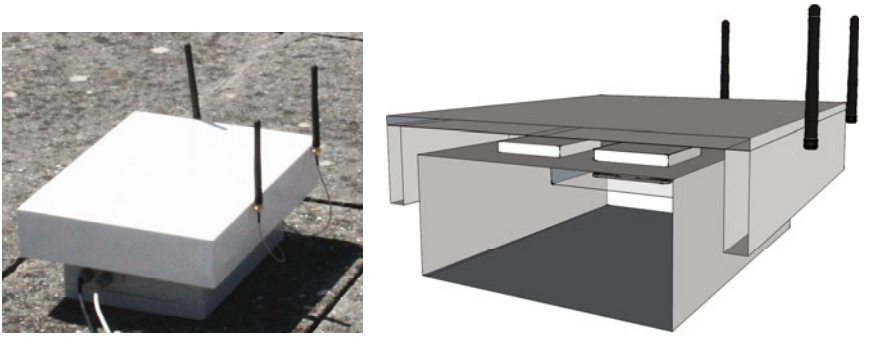


Fig. 2. Node structure with side cut and actual node in the testbed

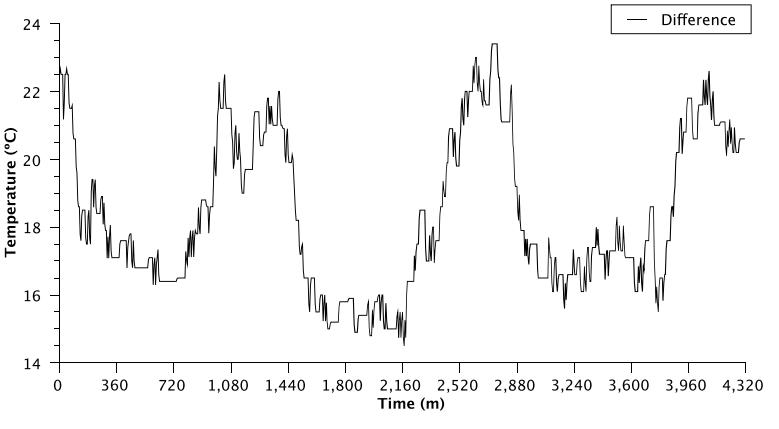


Fig. 3. Temperature variation over 3 days period

At the core of the testbed, there are several support servers and a redundant storage that serves all files to the testbed. Servers provide processing power to analyze results and extend the testbed by integrating simulators such as NS3. Also, they configure nodes according to the experiment trough the Control Module and provide a common time source [9]. Additionally they can be used to run Virtual Machines configured by

testbed users in order to support specific experiments. The redundant storage device provides a 4TB NAS in a RAID5 configuration, connected to the servers and nodes using Gigabit Ethernet. This device provides storage support to store OS images, experiment configuration, the results obtained and the virtual machines.

Nodes have at its core a Commell LE-365 board, with a VIA EDEN 1Ghz CPU and 1GB of RAM. These are powered by a single 120A, 8-16V variable power supply, which is shared by all nodes. Power is distributed point-to-point from the power supply to each of the nodes. The LE-365 board provides a high number of expansion possibilities such as 8 USB ports, a RS-232 port, a 2 ports SATA controller, 2 miniPCI slots and 1 CF interface. This myriad of interfaces makes it possible to easily expand the node interfaces, which is perfect for a multi-radio testbed. In particular because new network interfaces can in the future be added to the available interfaces. A Gigabit Ethernet NIC is also available and is used solely for the purpose of managing the nodes and collecting statistics. The availability of a Gigabit Ethernet control network is important due to several aspects: it reduces command delay, it supports raw packet collection from multiple node, and provides support for NFS with low latency. This interface can also be used for extending the testbed nodes capabilities.

Currently the CF interface is occupied by a 4GB CF card while two 802.11 cards occupy the miniPCI interfaces. One of the cards is a Compex WLM54SuperAG, which has an Atheros AR5414 processor at its core. This card can output 20dBm and supports channel bonding both in the 2.4Ghz (802.11bg) and 5Ghz (802.11a) ranges. The additional card (Compex WLM200NX) is also Atheros based (AR9220) but besides supporting 802.11a,b,g,i,e,h,j, high sensitivity (Extended Range), Power Control, and Channel Bonding (40Mhz) like the previous card, it also adds support for 802.11n in a 2x2 MIMO spatial multiplexing configuration. The open-source ath5/9k drivers control both cards and provide high flexibility due to almost direct access to the networking ASIC.

Electromagnetic interference is reduced due to the high insulation of the roof pavement materials, location of the building (in the edge of the university campus), and the characteristics of the external users wireless usage (there are no residential or industrial neighbors).

3.3 Management

Given the fact that the testbed nodes are enclosed inside the protecting cases in an outdoor environment, it was necessary to carefully plan how the nodes were to be managed without the need to direct intervention. Additionally, one of the requirements we set for our testbed was the need to support full administrative (root) access to the nodes.

Based on these two requirements, the management framework of our testbed was built with a mixture of hardware and software based management mechanisms. The former provide a basic control of the nodes that is independent of the operating system, and can override users access, while the later are required to be installed in the operating system currently active in the node. The two types of mechanisms are nonetheless undissociable, with the hardware based mechanism only taking action if the software based mechanisms have failed, based on a centralized management infrastructure that monitors all the testbed.

The operation concept of the testbed is a "leased-time" model. Experiments take continuously hold of the testbed for a given period of time, deploying their own operating system and tools. Access to the testbed is provided to registered users, which have gone through a screening process in order to check the purposes and requirements of their experiments. After registration, users are provided a user account in the management platform in which they can upload their custom build Operating System image or choose from one of the existing ones. Access to the testbed is based on a timesharing scheduling mechanism and in accordance to the requirements set by the users.

The node operating system is remotely loaded using Pre Execution Environment (PXE) [10]. An operating system image is loaded to the node using PXE and a mixture of various file-sharing protocols such as TFTP, NFS, iSCSI and SMB depending on the image being loaded. Testbed users are supposed to be able to setup their own operation images with any Linux distribution or even Microsoft Windows. A central Linux based server is made available for distribution of the images using the aforementioned protocols, and additionally core machines can be setup by the testbed user to host any additional tools required for the deployment of the Operating System Images and experiment tools, as well as any other type of core infrastructure required to support a NGN experiment. This approach intends to give testbed users maximum flexibility in terms of choice of Operating System and experiments. But it creates an important challenge for reducing the risks associated to this full access. The testbed hardware must be monitored in order to avoid damage such as CPU overheating, or for violations on the time used for conducting a given experiment.

The most basic control mechanism we have setup is a power control web-service that controls several relays in the power distribution network. Through this very simple mechanism it is possible to remotely power-cycle the machines regardless of their status (running, crashed, booting, etc). The web-service is available not only to the testbed administrator, but also to the testbed user. In this case it will only allow control over the leased nodes for the experiment, during the experiment duration. Experiments that overdue their reservations periods can be shutdown selectively using this power control mechanisms. Since users are encouraged to run their tools and data collection mechanisms over the network, it is not expected serious loss of results from the cold power off solution. A remote power off command will only came as a last resort.

A spectrum analyzer placed centrally on the testbed monitors the spectrum. The information collected by such spectrum analyzer is used in the management platform to identify violations of spectrum usage that can be stopped by shutting down the offending nodes

One of the requirements we placed for our nodes was the availability of hardware watchdogs in the mainboard. This is required in order to easily (and automatically) recover from crashes and bad booting processes. Our objective is that once the system boots, a hardware watchdog will reboot the system after 4 minutes if no watchdog driver is meanwhile loaded. The mainboard we choose, the Commell LE-365 board, uses a Winbond W83697HF watchdog. This watchdog has drivers both for Linux and Windows operating systems that were set as our primary operating systems.

In addition to providing these drivers to the testbed users, prebuilt images (Debian based) are available for customization or for direct use by users. These prebuilt

images, besides having support for the hardware watchdog, have several other monitoring and management modules, and automatically mount a NFS share where tools and data can be placed (accesses through the Ethernet interface). Such functionality makes it possible for non computer oriented users to have a simple access to the testbed, through the use of the standard images provided by us, but still being able to use custom software deployed over the NFS share. In the future, we plan to extend the prebuilt image concept to the Windows operating system.

Another important driver (available only on the Linux image) is the so-called FlyingInterfaces, which allows direct access (from the core servers) to the wireless interfaces. This driver enables hosts in the core servers to use the network interfaces of the wireless nodes as local devices. Because testbed nodes are multiradio, by using virtual machines, maximum testbed capacity is actually doubled (2x24). Each testbed node runs a low complexity daemon, which exports network device IOCTL access over the network. Remote equipments run a different module providing a virtual interface, which mimics all functions of the testbed equipment wireless interface. Tools like Linux's *iwconfig* or *iwpriv* (used in Linux to control and monitor wireless devices) report exactly the same results when running both in remote equipments (using virtual devices) and in testbed nodes (using the real device).

The AMazING Control component (see Figure 4) is thus comprised by both a server (running in the support servers) and a client (running at each node). Its tasks include: monitor and management operations, node configuration, and trigger based execution of a given experiment. Its design is simple so that the impact on the node operation is minimal, thus not introducing artificial glitches in the experiment results.

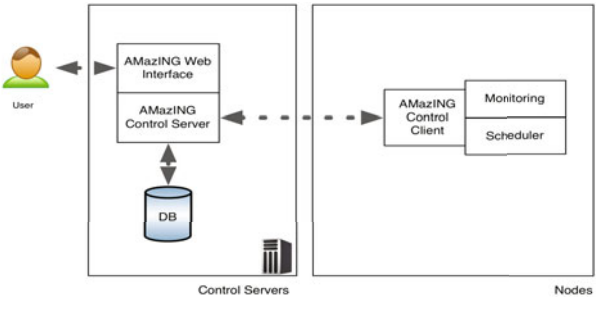


Fig. 4. Architecture of the AMazING Control component

The Monitoring function of the component provides basic statistics about the operation of the node, such as CPU load, memory consumption, temperature, and fan speed. While irrelevant for most experiments, operational metrics are vital to the proper maintenance of the system, mostly to avoid and early detect hardware failures. Values for each of the monitored metrics are reported periodically while the node is operating, and independently of the testbed utilization.

The Scheduling function is an event driven processing module, which triggers actions (command or script executions) at specific time events. Lists of events are specified upon the creation of the experiment and distributed to all nodes upon initialization. After the list of tasks is finished, the Scheduling function will

periodically try to get new event definitions. Nodes are synchronized with the support servers within a few milliseconds using NTP [9], allowing for coordinated quasi-simultaneous events across the entire testbed.

4 Validation

As a preliminary validation of our testbed and surrounding environment, we demonstrate some simple experiments obtained with the described setup. In the first case we deployed a WiSpy dongle near one of the nodes and monitored the radio levels over the 2.4Ghz band. This dongle is an easy to use spectrum analyzer, which while not comparable with laboratory equipments (which are also tens of times more expensive), allows for some basic measurements in the range of -6.3 to -100dBm. The model we used has a resolution of 373KHz and sweeps the radio band in the range of 2.400 to 2.483 GHz. Then we plot signal strength as a function of time and frequency and assess interference level at a particular location. Figure 5 depicts the signal strength near Node 1, when no node is active and when Node 1 is acting as an Access Point in channel 1. As it is depicted, average signal level on the testbed area is below -92dBm. Peak signal value measured when nodes are active, but idle, is a little higher, reaching -80dBm. If compared with the situation of nodes transmitting in channel 1 (2.412 GHz) it can seen the different between the signal power of the nodes (at 20dBm) and the environment peak noise values.

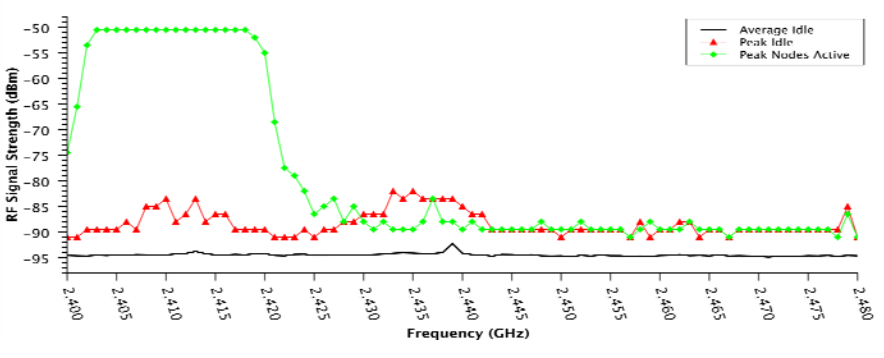


Fig. 5. Peak and average radio power measured with nodes active and idle

While Node 1 is configured as an Access Point in channel 1, we conducted an experiment to determine how radio signal and throughput varies along one of the sides of the testbed. For this, we sent TCP flows from Nodes 2, 3, 4 and 5 towards Node 1 during 1 minute and observed the throughput at intervals of 10s. All these nodes are located in the right side of the building, actually forming a line, each node distancing 8m from the previous. Node 5 is 32m from Node 1. As depicted in Figure 6, neighbor nodes are able to sustain a throughput of up to 23Mbps when using 802.11g. This value decreases, as expected, with the distance as the receiving power also decreases. The farthest node (Node 5) is able to reach Node 1 and exchange data at less than 4Mbits/s. Interestingly, when considering 802.11b, rates drop to less than

6Mbps, but values do not seem to decrease with distance. Node 5 is able to exchange data with Node 1 with 6Mbits when using 802.11b, which is higher than when using 802.11g. According to the specification of the Atheros chip, 20dBm of SNR are required for a link rate of 11Mbps, indicating that noise level at this node is indeed very low.

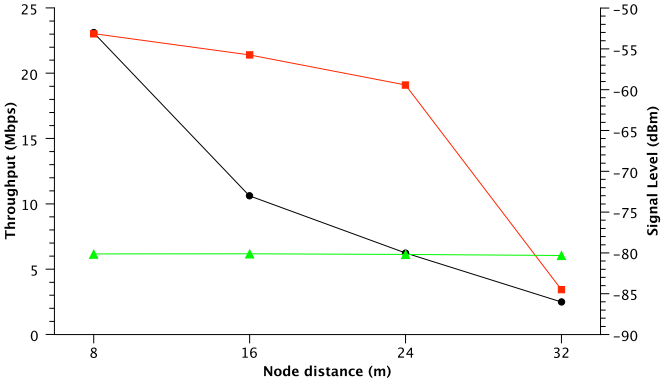


Fig. 6. Throughput (green is 802.11b, red is 802.11g) and SNR (black) as a function of distance from the AP node

SNR measured at the receiving node drops rapidly as the distance increases. At 8m, Node 2 indicates a receiving power of -55dBm while at 32 meters, Node 5 indicates that the signal generated by Node 1 is received with -86dBm. Considering free space propagation [1] of an omnidirectional antenna, at 32 meters the receiving power should be higher than it is, which suggests additional attenuation. In this case attenuation is intentional and is mainly caused by the occlusion of the Fresnel zones, which we achieved by placing antennas very near the floor at about 25cm high.

Considering the radius of the Fresnel zone as given by

$$F = \sqrt{\frac{n\lambda d^2}{2d}}$$

where n is number of the Fresnel zone, d is the distance between end-points and λ the wavelength of the signal. The maximum radius of the 1st Fresnel zone has the values present in Table 1.

Table 1.

	8 m	**16 m**	**24 m**	**32 m**
2.400 GHz	0.5	0.71	0.87	1.00
5.000 GHz	0.35	0.49	0.60	0.69

As depicted, even at close range (8m) and for the 5Ghz band, the 1st Fresnel zone intercepts the rooftop at a large extent. The minimum occlusion will be of 29% for the node at 8m when using the 5Ghz band, while the maximum occlusion is of 85% for nodes at 32m when using the 2.4Ghz band. The resulting effect is additional radio attenuation, which helps in creating custom topologies where nodes have no, or only minimized, direct radio connectivity.

5 Conclusions

Testing NGN has led us to the development of a general-purpose testbed, which is characterized by its extreme flexibility, and large reproducibility.

We have developed a usage model where users have full access to the nodes devices, and can even expand its capabilities by locating in the core functions that eventually access the wireless interfaces of the nodes. The outside nature of the testbed had posed specific deployment challenges, but the advantages achieved in terms of reduced noise and interference are well worth these added constrains. The management infrastructure developed allows for a coarse hardware control of the testbed, while software modules can perform fine control.

Confronting with previous custom-built NGN testbeds, this testbed seems to-provide a much faster deployment and evaluation process, thus simplifying the complex evaluation process of such research.

In the future we intend to expand the testbed to other buildings in the campus and explore the use of robots as mobile nodes.

Acknowledgments

This work has been partially funded by IST OneLab2 under grant agreement 224263.

References

1. DARPA/NSF: The network simulator - ns-2, `http://www.isi.edu/nsnam/ns/`
2. Ivanov, S., Herms, A., Lukas, G.: Experimental validation of the ns-2 wireless model using simulation, emulation, and real network. In: Proceedings of the 4th Workshop on Mobile Ad-Hoc Networks (2007)
3. Kurkowski, S., Camp, T., Colagrosso, M.: MANET simulation studies: the incredibles. SIGMOBILE Mob. Comput. Commun. Rev. 9(4), 50–56 (2005)
4. Maltz, A., et al.: Experiences designing and building a multi-hop wireless ad hoc network testbed, CMU-CS-99-116, CMU, School of Computer Science (1999)
5. Aguayo, D., et al.: MIT Roofnet: Construction of a Production Quality Ad-Hoc Network. In: Proc of MOBICOM 2003 (2003)
6. Raychaudhuri, D., Ott, M., Secker, I.: ORBIT Radio Grid Tested for Evaluation of Next-Generation Wireless Network Protocols. In: Proc. of the First international Conference on Testbeds and Research Infrastructures For the Development of Networks and Communities (2005)
7. OneLab1 D4D.1, OneLab wireless mesh multi-hop network (2007)
8. Gomes, D., Matos, A., Fonseca, E., Aguiar, R.L.: Deploying and testing a NGN testbed. In: Proc. 2009 TridentCom, Washington, D.C. (April 2009)
9. Mills, D.: Network Time Protocol (Version 3), RFC 1305. IETF (1992)
10. Intel Corporation: Preboot execution environment (PXE) specification (2002)
11. Peterson, L., et al.: PlanetLab architecture: An overview, Technical Report PDN- 06-031, PlanetLab Consortium (2006)
12. Saunders, S.R., Simon, S.R.: Antennas and Propagation for Wireless Communication Systems. John Wiley & Sons, Inc., New York (1999)

Challenges in Deploying Steerable Wireless Testbeds

Eric Anderson[1,2], Caleb Phillips[1], Gary Yee[1], Douglas Sicker[1], and Dirk Grunwald[1]

[1] University of Colorado, Department of Computer Science
430 UCB, Boulder, CO 80309, USA
{caleb.phillips,gary.yee,sicker,grunwald}@colorado.edu
[2] Carnegie Mellon University, Department of Electrical and Computer Engineering
5000 Forbes Ave, Pittsburgh PA 15213
andersoe@ece.cmu.edu

Abstract. Phased array antennas enable the use of real-time beam-forming and null-steering to further increase control of signal strength and interference in wireless networks. Understanding the potential of this platform for both mesh and single-hop networks is becoming more important as smart antennas begin to appear in emerging networking standards. Prior attempts to test non-standard antenna platforms have typically focused around simulations, fixed (non-steerable) directional antenna testbeds, and small scale temporary setups utilizing 1 or 2 phased array antenna nodes over the span of a few hundred meters. This paper presents the challenges encountered – and solutions developed – in building WART, a permanent, campus-wide testbed for wireless networking with beamforming antennas.

Keywords: Smart antennas, steerable, directional, phased array, antennas, outdoor, wireless, testbed, 2.4GHz.

1 Introduction

Directional antennas, both fixed and steerable, are proving to be important in the next generation of wireless networking protocols. These antennas give nodes further control over both signal strength and interference, allowing optimization techniques which can yield greater network throughput with fewer errors. While protocols incorporating directional or "smart" antennas have been proposed, their evaluation has been limited. Those researchers who have attempted real-world evaluation of their ideas have often used one-off testbeds assembled to perform a small number of experiments [11,9,8]. Most proposals, however, rely solely on simulation or theoretical analysis (for instance, [14,13]).

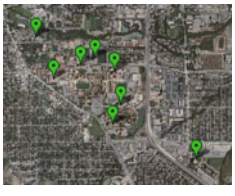
In this paper we introduce the University of Colorado Wide-Area Radio Testbed (WART) as a platform for studying uses of directional, steerable, and smart antennas in wireless networking[1]. Given the widely-recognized difficulty of accurately simulating radio environments, real-world experiments are essential to evaluate wireless networking protocols. In the case of directional applications, which are especially dependent on the vagaries and environmental effects of radio propagation, this is even more important [2,4].

[1] An expanded version of this paper is available in a companion technical report at [3].

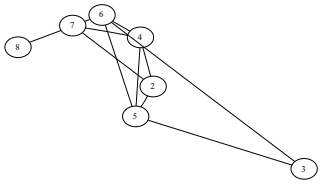
T. Magedanz et al. (Eds.): TridentCom 2010, LNICST 46, pp. 231–240, 2011.

WART is currently the only permanent facility for studying smart antennas in a large and diverse urban environment. The system consists of eight phased array antenna nodes mounted to the rooftops of university buildings, spanning an area of 1.8 x 1.4 kilometers. The entire testbed is linked together via wired Ethernet and can be controlled from a single administration point. This architecture ensures that WART can not only offer the geographic scale and realism of large scale distributed testbeds [1], but can also give its users the degree of control and ease of management only seen in dense indoor testbeds such as ORBIT and Emulab [12,7].

The production and deployment of such a testbed, however, is itself an engineering problem. In addition to the capabilities of WART, this paper describes some of the logistical challenges encountered in planning, installing, and maintaining a centrally controlled wide area rooftop network.



(a) Campus testbed (1.8 x 1.4 km)



(b) Connectivity of nodes 2 through 8 without beam steering

(c) Installed antenna node

1.1 Design Goals

WART is intended to be a dedicated experimental testbed for studying the impact of directionality and beam-forming throughout the network stack. Given this objective, we chose three principle design goals for WART: (1) The testbed must be able to perform outdoor omni-directional, fixed directional, and beam-forming experiments; (2) The testbed must be able to test a diverse set of link distances of varying link qualities; (3) WART nodes must be simple to reconfigure for varying experiments and provide an easy recovery mechanism in case of failure. The node sites were chosen to provide a variety of link lengths, with line-of-sight between many but not all pairs of nodes.

The remainder of this paper describes the hardware, software, and centralized architecture of WART that helps fulfill the design goals of easy maintenance and administration.

1.2 Smart Antenna System

In this section we describe the hardware and software that comprise WART. These components give it the unique ability to perform smart antenna research at all network stack levels and address challenges with its administration and experimental setup.

Each smart antenna node consists of a phased array antenna and an embedded computer. The phased array antennas were designed and constructed by Fidelity Comtech. The antenna operates in the 2.4GHz ISM band and uses an 8 element uniform circular

array of dipole antennas that support a minimum 42° primary lobe. The ratio of the lowest null to the highest peak is ≈ 40dB, which allows for selectively "nulling out" interfering signals. The antenna arrays can be electronically switched between radiation patterns in $\approx 100\mu$seconds, allowing for precise dynamic reconfiguration. The wireless interface card used is an IEEE 802.11 Senao 5345MP MiniPCI adapter, which uses an (especially flexible) Atheros chipset.

The default Operating System (OS) image used by each WART node is a modified OpenWRT Linux (Kamikaze) distribution. The wireless driver is based on the Multiband Atheros Driver (MADWiFi) version 0.9.4.5 and has been extended to support the smart antennas' ability to change antenna patterns, along with various measurement and experimental Medium Access Control (MAC) features. Unix Network File System (NFS) is used to transmit data from the smart antenna node to long-term storage.

2 Administration and Maintenance Infrastructure

Experimental hardware and software is almost inevitably flawed, and faults which escape notice during testing regularly cause problems during live experiments. When problems do occur, equipment needs to be rebooted, experiments need to be re-started, scripts need to be edited, and sometimes new software needs to be installed. The (human) communication overhead of trying to identify and correct problems across all test locations quickly becomes prohibitive, even when the necessary fixes are small. In early tests we found that even when nothing went wrong, coordinating a four node experiment required at least a half-hour of overhead for setup, configuration checks, synchronization, starting the experiment, downloading the data afterwards, and running basic sanity checks on the data. Overall, the ratio of time expended to successful experiment time was very high.

Our primary requirement for the testbed infrastructure was that it enable centralized management. At its core, this infrastructure consists of a control plane network, a "management box" connected to each experimental antenna unit, and a collection of software tools. All of these will be described in upcoming sections.

2.1 Management System

Every experimental antenna unit is directly connected to a management box, as depicted in Figure 1. These boxes connect the experimental units to the control plane network. Additionally, the management boxes also provide network booting and remote power control to the antenna units. This approach greatly simplifies reconfiguration: Any software change, from one configuration file to a new operating system, can be made by uploading a new image to the management system and rebooting the experimental equipment. The equipment could boot from a remote server, but only if the intervening network had the configuration and performance to support it; such a requirement which would limit options substantially.

Each management box contains a single-board Soekris computer running Linux and can be installed indoors at a significant distance from the antenna unit. All of the current deployments have Ethernet connections, but they can accommodate other data connections with minimal configuration changes.

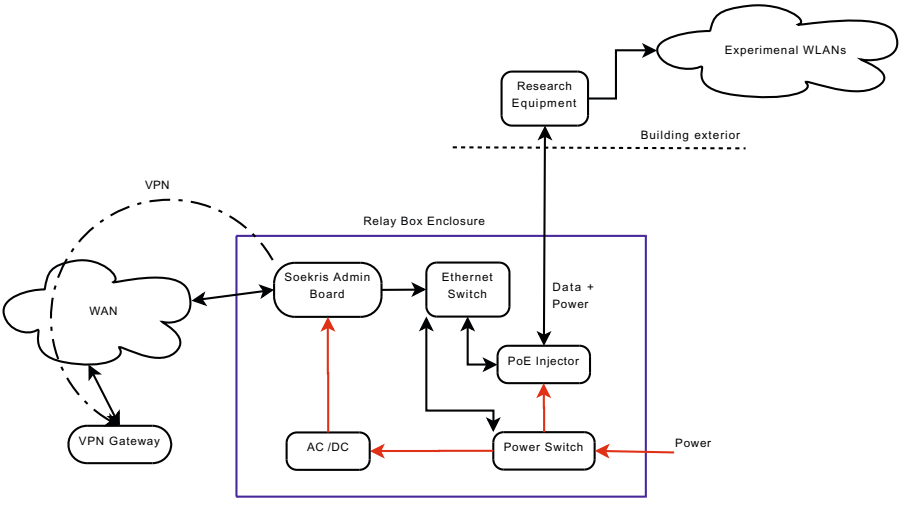


Fig. 1. Management box configuration

The organization of the control plane network relies heavily on the use of a virtual private network (VPN). Management units establish and maintain connections to a central VPN server, so that all the devices are accessible as a single IP network. Network Address Translation (NAT) is used to map hard-wired local addresses to node-specific VPN addresses. The practical effect of this configuration is that error-prone customization and dynamic reconfiguration are largely avoided, yet every device is reachable by a globally unique address, and most configuration information is stored at the central server where changes are easy to make. As a final failsafe, each management unit contains a remote power switch for each component allowing for a hard-reboot of any device.

2.2 Interchangeable Parts

Each phased array antenna unit or network power switch has exactly the same hardware and configuration as every other. Every management computer is the same as every other except for the contents of a removable compact flash card. This makes it easier to develop testing processes for each component and means that a faulty or suspect component can be replaced with no thinking or configuration required. In fact, it is often easiest to replace the entire management unit as a whole – except for the flash card – and then diagnose faulty equipment in the comfort of the lab. Hardware and configuration homogeneity make software management more practical. All of the files associated with the testbed – source code, configuration files, and compiled operating system images – are kept under version control. Because the antenna units are identically configured, there is one generic OS image for all of them.

2.3 Security

Since WART nodes are connected to untrusted networks, they are potentially susceptible to the same attacks that many other machines on the University of Colorado network experience on a day-to-day basis. Several steps have been taken to ensure that only authorized access is given to both the phased array antenna node and management board. Communication to the WART management nodes is restricted to nodes that are part of the same VPN. This requires having a certificate signed by the certificate authority, a process which is performed off-line. Within this trusted domain, we use SSH keys to allow remote logins directly to the antenna and management nodes.

Another possible attack vector is via the wireless interface. Should an attacker inject traffic to a node, the node could potentially begin routing packets from unauthorized users. To minimize this risk, nodes perform only the minimum forwarding required for experiments.

3 Deployment Logistics

Deploying a physically large testbed, especially with outdoor equipment, involves a number of challenges outside the traditional realm of computer science. There is a modest inherent engineering component, which is significantly compounded by the need for approval and cooperation from various outside parties. All of the WART nodes are located on University of Colorado property, meaning that we only needed to interact with a single (albeit large and bureaucratic) owner. We suspect that broadly similar issues would be likely to arise in working with another large organization, and possibly with multiple smaller ones.

Some of the more prominent logistical challenges encountered were:

- *Architectural Approval:* The aesthetic impact on campus buildings had to be approved by the campus architect.
- *Antenna Siting and RF Interference Approval:* A separate antenna committee had to be convinced that the proposed sites would not interfere with existing radio equipment.
- *Electrical Design and Installation:* The electrical requirements of the testbed equipment are extremely low – each node uses less power than a desk lamp. However, all construction projects involving new electrical connections are subject to the same approval process, regardless of the actual load. This means that an electrical design for each node had to be completed and signed off by a certified electrical engineer and installation of the electrical components had to be performed by licensed electricians. Both tasks had to be done by outside contractors, requiring an additional round of financial approvals before work could begin.
- *Environmental Health and Safety:* All construction projects must be audited for safety risks to both the workers and the campus in general. The primary concern was disturbing pre-existing asbestos building materials, although we also had to vouch for the microwave radiation levels.
- *Roof Integrity:* Because the equipment was to be mounted on the outside of buildings, both the attachment methods and cable connections had to be evaluated for

waterproofing, fire sealing, and structural impact. In the cases where new holes had to be made through the roof, the penetration and waterproofing had to be installed by campus roofing services.
- *Antenna Structure:* Local building codes and campus design rules establish standards for wind, snow, and ice tolerance. The university requirements were the more stringent in this case, requiring that equipment be designed for 120 mile per hour (53.6 m/s) wind load. Antenna mounting equipment, especially in the WiFi market, seldom meets those requirements. While commercial options do exist, we found it more cost-effective to design and construct our own.
- *Financial Approvals:* After our research group and department decided to allocate funds for the testbed, there were still a significant number of delays waiting for work orders and payments to be approved by other university entities. In particular, payments from the computer science department to facilities management, and from facilities management to outside contractors all required administrative approval before the payee could begin work.

3.1 Timeline

The testbed deployment process has required a total of two years. Most of that time has consisted of waiting for some necessary action by parties outside our department. Within that waiting, most of the time has been for administrative approvals, with actual design and construction requiring relatively little. Figure 2 shows our actual timeline; with more foresight it probably could have been compressed.

The architectural and RF approval steps are an unavoidable bottleneck, as they determine whether and where equipment can be installed. In our case, it required approximately nine months from the first informal proposals to a preliminary approval of the sites chosen. Once those decisions had been made, several of the remaining steps could likely have proceeded at once.

The obvious deployment tasks, namely physically installing the antenna node and management box, and running conduit and Ethernet cable between them, required on the order of one week per node.

3.2 Costs

Table 1 presents an approximate breakdown of the expense incurred *per node* in building this testbed. The dominant cost is not the research equipment itself but rather labor required for regulatory and university policy compliance. This includes both the electrical work mentioned earlier and the time spent by university employees on evaluation and project oversight.

4 Proof-of-Concept Experiments

As a proof-of-concept experiment for WART, we performed a full pairwise link quality test. In this test, each WART node takes a turn transmitting while the other nodes listen. During each turn, the transmitter and all receivers cycle through 17 pre-configured

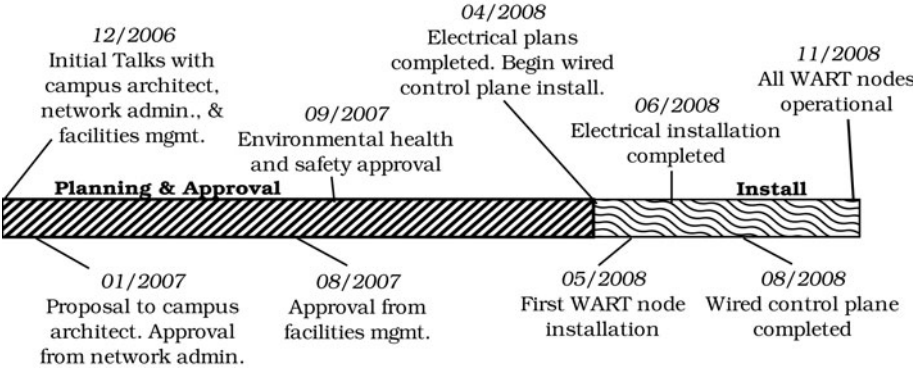


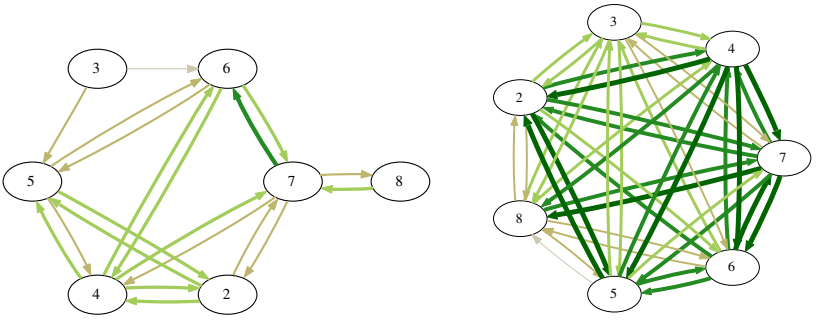
Fig. 2. Testbed deployment timeline: The entire process took 2 years, 16 months of which was devoted to planning and administrative approval while only 8 were required to install and configure the hardware.

antenna patterns, so that every combination of transmitter and receiver antenna patterns is tested. The patterns chosen point the main lobe in one of 16 directions about the azimuth plane (the 17th pattern is omnidirectional). Using the measured signal strength of received packets, we are able to determine (a) which links are possible between which nodes and (b) what the optimal "greedy" patterns are for each link. The results of this experiment are provided visually in Figure 3, which we believe makes a compelling case for the power of steerable directional antennas. When configured with omnidirectional patterns, which are comparable to the antennas used in many single-radio mesh networks, only a few links are even possible, and of those only a small number offer decent signal quality. With steering, however, we see a vast improvement: not only are all link-pairs able to pass traffic, but these links are typically of high quality (greater than -70 dBm).

Our present and future research utilizes WART to evaluate directional medium access control (MAC) protocols, with a particular emphasis on optimization for spatial re-use. We believe that the unique opportunity that WART provides for real-world evaluation of these protocols will lead to important results in this direction, and new insights into methods for improving wireless systems in general.

Table 1. Cost of labor and parts per WART node. The labor of research group members is not considered.

Description	**Cost**
Phased Array Antenna Node	$3,000
Management Box and Other Control Plane Equipment	$1,200
Installment Materials	$300
External Labor and Fees	$5,780



(a) Omnidirectional patterns (as in Figure 1b)

(b) Best (greedy) directional patterns

Fig. 3. Comparison of available links and link quality between seven testbed nodes using best-steered directional patterns and omnidirectional patterns. Stronger links are indicated with a wider arrow of a darker color. The best links are those with a link of greater than -60 dBm. The worst links plotted are barely above the noise-floor with greater than -95 dBm achieved RSS.

5 Related Work

There are too many wireless testbeds to discuss individually in the available space. For a more thorough treatment, please see our companion technical report [3]. Despite the large number, we are not aware of any existing testbeds that address the particular needs of WART. We observe that these testbeds fall into one of two categories: Wide area testbeds, which cover a significant outdoor environment but offer limited control over each node, and dense indoor testbeds with many nodes and excellent facilities for re-programming and control, but with very artificial RF environments.

The existing outdoor testbeds generally have more operational emphasis and less experimental control and management support than WART or the indoor testbeds. Most use stock 802.11 at the MAC and physical layers, although additional low-layer information can be collected. This may in part reflect their designers' research interests and may also reflect limitations resulting from the lack of a stable separate control network. Notable examples include Roofnet [5], the Rice/TFA mesh [6], and the Digital Gangetic Plains project [10].

In general, the indoor testbeds are physically smaller than the outdoor ones and benefit from a much more controlled environment. The problems of remote repair and establishing and maintaining a reliable communication infrastructure, which have been at the forefront of our design challenges, are largely non-issues. Many of the indoor testbeds have at least an order of magnitude more nodes than any of the outdoor ones: Both ORBIT and Emulab have over 400 nodes [12,15,7]. Much of the infrastructure developed for the indoor testbeds is oriented toward automating the process of configuring, controlling, and aggregating data from such a large collection of devices.

6 Conclusion

This paper has presented WART, a testbed that will facilitate future networking research by providing unique physical layer capabilities not seen in any other outdoor networking testbed. While the testbed covers an entire university campus, it is easy to manage and administer due to its wired control plane, which is remotely accessible from anywhere on the Internet.

The research motivation for building WART was to study the use of directional, steerable, and adaptive antennas. The prominent issues encountered in creating the testbed proved to be only indirectly related to that objective. The direct causes were *using commodity equipment*, *supporting low-level experimentation*, and *spanning a large geographical area*.

Commodity equipment: The research equipment (phased array antenna nodes) is comparatively affordable at $3,000 per node, while specialized test and measurement equipment could easily cost 10 to 20 times more. The consequences of using commodity hardware have been the need for significant calibration and testing and extensive software hacking to make the hardware operate in unintended ways.

Low-level experimentation: Many of the experiments we wish to conduct are low-level both in the sense of being at the physical and MAC layers of the OSI hierarchy, and in the sense of requiring "close to the metal" system implementation. This implies the need for easy reprogramming and crash recovery, high-volume data collection, and a flexible control interface. In practice, these in turn require a control connection that is separate from the experimental wireless system.

Large geographical area: It has been amply demonstrated that radio propagation in general, and directionality in particular, are very environmentally dependent [2]. Consequently, it was important that WART encompass a range of node densities and environmental features of interest. However, covering a large area implies *physical distance* and often *administrative diversity*, each of which contribute significant design challenges. Physical distance effectively precludes running dedicated cables from a central location to all of the nodes, which implies that power and network connectivity (if needed) must be supplied using resources available on site. It is this constraint which leads us to the "management box" design, with network support, power conversion, and power switching co-located with every measurement node.

Covering a larger area often implies involving more administrative domains. Our sites are all owned by the same university, but building at a campus-wide scale requires the involvement of many departments – administrative and academic – and the approval of several levels of hierarchy. The practical impact of this cannot be overstated. The approval processes – and the cascade of design decisions made in order to secure those approvals – account for at least half of the total time and cost for this project.

This testbed was developed to study particular physical layer technologies, but the design lessons are not specific to that objective. Most of the challenges encountered in designing this testbed – and the solutions developed – are likely to apply to other outdoor and wide-area testbeds. We have developed an infrastructure for deploying nodes at widely separate, minimally provisioned sites and connecting them into an easily-managed unified research system.

References

1. Aguayo, D., Bicket, J., Biswas, S., Judd, G., Morris, R.: Link-level measurements from an 802.11b mesh network. In: Proc. Sigcomm 2004. ACM, New York (August 2004)
2. Anderson, E., Phillips, C., Sicker, D., Grunwald, D.: Modeling environmental effects on directionality in wireless networks. In: 5th International Workshop on Wireless Network Measurements (WiNMee) (June 2009)
3. Anderson, E., Phillips, C., Yee, G., Sicker, D., Grunwald, D.: Challenges in deploying steerable wireless testbeds. Tech. Rep. CU-CS-1058-09, University of Colorado, Boulder (2009)
4. Anderson, E., Yee, G., Phillips, C., Grunwald, D., Sicker, D.: The impact of directional antenna models on simulation accuracy. In: 7th Intl. Symposium on Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks (WiOpt) (June 2009)
5. Bicket, J., Aguayo, D., Biswas, S., Morris, R.: Architecture and evaluation of an unplanned 802.11b mesh network. In: MobiCom 2005. ACM, New York (August 2005)
6. Broustis, I., Eriksson, J., Krishnamurthy, S.V., Faloutsos, M.: A blueprint for a manageable and affordable wireless testbed: Design, pitfalls, and lessons learned. In: TridentCom (2007)
7. Johnson, D., Stack, T., Fish, R., Flickinger, D.M., Stoller, L., Ricci, R., Lepreau, J.: Mobile emulab: A robotic wireless and sensor network testbed. In: Proceedings of 25th IEEE International Conference on Computer Communications, INFOCOM 2006, pp. 1–12 (April 2006)
8. Kohmura, N., Mitsuhashi, H., Watanabe, M., Bandai, M., Obana, S., Watanabe, T.: Unagi: a protocol testbed with practical smart antennas for ad hoc networks. SIGMOBILE Mob. Comput. Commun. Rev. 12(1), 59–61 (2008)
9. Mitsuhashi, H., Watanabe, M., Obana, S., Bandai, M., Watanabe, T.: A testbed with a practical smart antenna for directional mac protocols in ad hoc networks. In: AINAW 2007: Proceedings of the 21st International Conference on Advanced Information Networking and Applications Workshops, Washington, DC, USA, pp. 731–736. IEEE Computer Society, Los Alamitos (2007)
10. Raman, B., Chebrolu, K.: Experiences in using WiFi for rural internet in india. IEEE Communications Magazine (January 2007); Special Issue on New Directions In Networking Technologies In Emerging Economies
11. Ramanathan, R., Redi, J., Santivanez, C., Wiggins, D., Polit, S.: Ad hoc networking with directional antennas: a complete system solution. IEEE Journal on Selected Areas in Communications 23(3), 496–506 (2005)
12. Raychaudhuri, D., Seskar, I., Ott, M., Ganu, S., Ramachandran, K., Kremo, H., Siracusa, R., Liu, H., Singh, M.: Overview of the orbit radio grid testbed for evaluation of next-generation wireless network protocols. In: Proc. IEEE Wireless Communications and Networking Conference, vol. 3, pp. 1664–1669 (2005)
13. Singh, H., Singh, S.: Smart-aloha for multi-hop wireless networks. Mob. Netw. Appl. 10(5), 651–662 (2005)
14. Takai, M., Martin, J., Bagrodia, R.,, A.R.: Directional virtual carrier sensing for directional antennas in mobile ad hoc networks. In: MobiHoc 2002: Proceedings of the 3rd ACM International Symposium on Mobile ad hoc Networking & Computing, pp. 183–193. ACM, New York (2002)
15. White, B., Lepreau, J., Stoller, L., Ricci, R., Guruprasad, S., Newbold, M., Hibler, M., Barb, C., Joglekar, A.: An integrated experimental environment for distributed systems and networks. In: Proc. of the Fifth Symposium on Operating Systems Design and Implementation, Boston, MA, pp. 255–270. USENIX Association (December 2002)

TridentCom 2010

Practices Papers Session 4: Monitoring, QoS and Application Instrumentation in Large Scale Testbeds

ETOMIC Advanced Network Monitoring System for Future Internet Experimentation

István Csabai[1], Attila Fekete[1], Péter Hága[1], Béla Hullár[1], Gábor Kurucz[1], Sándor Laki[1], Péter Mátray[1], József Stéger[1], Gábor Vattay[1], Felix Espina[2], Santiago Garcia-Jimenez[2], Mikel Izal[2], Eduardo Magaña[2], Daniel Morató[2], Javier Aracil[3], Francisco Gómez[3], Ivan Gonzalez[3], Sergio López-Buedo[3], Victor Moreno[3], and Javier Ramos[3]

[1] Eötvös Loránd Univeristy, Budapest, Hungary
{csabai,fekete,haga,hullar,kurucz,laki,matray,steger,vattay}@etomic.org
[2] Universidad Pública de Navarra, Pamplona, Spain
{felix.espina,santiago.garcia,mikel.izal,eduardo.magana,daniel.morato}@etomic.org
[3] Universidad Autonóma de Madrid, Madrid, Spain
{javier.aracil,francisco.gomez,ivan.gonzalez,sergio.lopez-buedo,victor.moreno,javier.ramos}@etomic.org

Abstract. ETOMIC is a network traffic measurement platform with high precision GPS-synchronized monitoring nodes. The infrastructure is publicly available to the network research community, supporting advanced experimental techniques by providing high precision hardware equipments and a Central Management System. Researchers can deploy their own active measurement codes to perform experiments on the public Internet. Recently, the functionalities of the original system were significantly extended and new generation measurement nodes were deployed. The system now also includes well structured data repositories to archive and share raw and evaluated data. These features make ETOMIC as one of the experimental facilities that support the design, development and validation of novel experimental techniques for the future Internet. In this paper we focus on the improved capabilities of the management system, the recent extensions of the node architecture and the accompanying database solutions.

Keywords: ETOMIC, network measurement infrastructure, active measurements.

1 Introduction

Internet is existing since the late 60s and became a global network in the early 80s. By that time the general design and most of the basic protocols have been laid down. Since then its scale and complexity has become orders of magnitudes larger concerning both the number of nodes and the amount of traffic flowing through. Although smaller modifications have happened and the hardware behind the Internet has changed significantly, the basic structure has remained the

T. Magedanz et al. (Eds.): TridentCom 2010, LNICST 46, pp. 243–254, 2011.

same. Recently, several international efforts try to re-design the Internet from clean slate. It is expected that a more optimal design will solve the most pressing problems we currently face. To enable the design of a better system it is essential to understand the details of the network and traffic dynamics. For this purpose we need to create measurement tools and gather diverse measurement data.

The European Traffic Observatory Measurement InfrastruCture (ETOMIC)[1] was launched in 2004 [1,2,3] with the need to supply the network research community with various experimental data. It is targeted to provide the scientific community with an Internet measurement platform that is fully open and reconfigurable, extremely accurate and GPS-synchronized. The ETOMIC system has been designed to allow researchers to perform any kind of active network measurement. The users are provided with a web-based graphical user interface for the definition of the experiments to run. Researchers may either choose from a number of built-in scripts that cover the most popular measurement techniques, like traceroute or packet-pair experiments, or they can provide their own code for the experiments. To avoid conflicts in resource utilization each measurement has to be scheduled to exclusively reserve node resources for its execution. The node reservations are performed through the web-based user interface. The ETOMIC management kernel takes care of the software upload and experiment execution in a fully automated fashion.

After the successful duty of the measurement nodes since 2004 the renewal of the system components was necessary. In the OneLab project [4] we have extended the capabilities of the measurement hardware to match the current technologies and to incorporate the software evolution of the last years that are important from the perspective of network measurements. The ETOMIC infrastructure now provides two ways of collecting experimental data. One possibility is when the researcher reserves and configures the measurement nodes and sets the parameters of the experiment through the Central Management System. In this case, besides the original ETOMIC nodes, newly deployed enhanced measurement boxes can also be used for experimentation. To meet the requirements of high precision measurements the nodes are equipped with a DAG card (for the original nodes) or an ARGOS card (for the new generation nodes) to provide nanosecond-scale timestamping of network packets. Besides these nodes a third type of hardware component was also introduced, which is called Advance Probing Equipment (APE). APE is a low cost hardware solution developed to serve as a measurement agent for user applications: it provides a web service interface to conduct experiments. This approach enables autonomic software components to automatically collect relevant network data from the ETOMIC system they rely on for their operation. As a consequence of a development in the system kernel the nodes of the PlanetLab platform [5] can also be used as measurement nodes by the ETOMIC system. The goal of this integration was to enable the federated usage of the high precision ETOMIC nodes and the numerous PlanetLab nodes.

To make it easier to handle and archive the huge amount of data collected by the ETOMIC platform we have created data repositories. There are two different

[1] ETOMIC was awarded with the Best Testbed Award at TRIDENTCOM 2005.

interfaces for these data archives. The periodic measurements web interface can be used to poll automatically collected measurement data through pre-defined queries. Currently this interface provides one-way delay measurements, NTP and GPS measurements and several different types of traceroute experiments. As another approach, the Network Measurement Virtual Observatory (*nmVO*) [6] provides standard SQL database access to the user community. The nmVO provides a graphical user interface and a web service interface for accessing raw and evaluated measurement data.

The paper is organized in the following way: Section 2 explains the key features of the system architecture. In Section 3 and in Section 4 we introduce the management kernel that provides the central control of the system and the hardware architecture of the measurement nodes. Section 5 discusses the different data repositories and their interfaces. Finally, in Section 6 the conclusions close the paper.

2 System Architecture

The ETOMIC infrastructure is constituted of high precision measurement equipment modules hosted by European universities, research institutes and company laboratories. The clocks of the measurement nodes are synchronized via GPS signals, which allow not only packet round-trip time estimation, but also precise one-way delay measurements. The ETOMIC platform is very flexible, since researchers can develop and run any kind of active experiments.

A Central Management System (CMS) is in charge of system control, comprising not only the scheduling and execution of measurements experiments, but also system monitoring and configuration. The main software component of the CMS is the *management kernel* which is running on a dedicated server computer. The kernel is responsible for scheduling tasks, deployment of user software to the measurement nodes, node configuration, experiment execution and the collection of measurement results from the nodes. The CMS provides a web-based graphical user interface where the researchers can configure the system and reserve system resources for their measurements. An internal database is attached to the

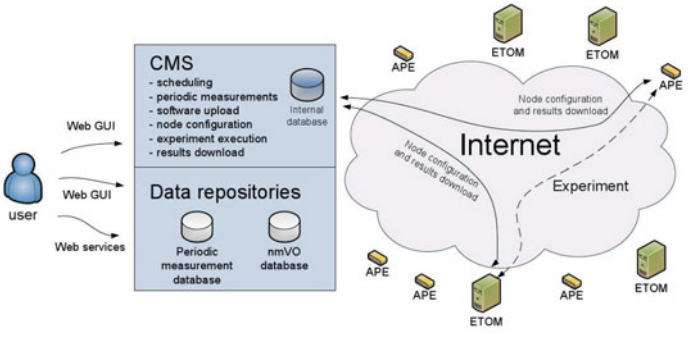


Fig. 1. ETOMIC system architecture

kernel where the system and user level management information are stored. The results of a finished measurement are also collected and stored in this database until the user downloads them. The system components are depicted in Figure 1.

As an important add-on for the original ETOMIC system the architecture has been extended with large capacity data repositories that are publicly available. The system provides several different interfaces for these repositories through which the users can reach the collected datasets. The interfaces allow the users to run intelligent queries in order to filter and process the raw data on the server side. In these repositories the results of traceroute, one-way delay, queuing delay tomography, available bandwidth, IP geolocalization, router interface clustering and NTP-GPS measurements are stored.

3 Management Kernel

The ETOMIC management kernel constitutes the core of the Central Management System (CMS). It is in charge of user management, experiment scheduling and keeping the corresponding results in the temporary data storage. In the following we briefly overview the tasks performed by the CMS.

Scheduling and calendar maintenance. In order to isolate the different measurements and to schedule experiments and maintenance tasks a calendar is used for each measurement node. The researcher is expected to book measurement nodes for a certain time interval and to upload the applications necessary for the measurement. The web interface is in charge of checking that the time line for the experiment does not collide with any other previously registered measurement. In case of successful resource reservation the CMS inserts the new experiment information into its internal database. Previously, the only way to create periodic experiments was to define their scheduling one-by-one. Due to a recent development in the kernel scheduler the CMS allows the definition of customized periodic experiments with an inter-experiment time interval and a repetition count. This feature enables to program for example daily or hourly experiments with the ease of defining only one experiment.

Experiment management. The management kernel is continuously checking the internal database for new measurement requests. When there are no scheduled experiments the measurement nodes are totally stateless for failsafe reasons. The communication is always initiated by the server via SSL based secure connection to command the nodes. Once a new experiment has been defined and the deadline for execution approaches, the management kernel performs the following tasks (that are internal to the system and transparent to the researchers).

- **Software upload and measurement node configuration.** The measurement applications and accompanying files are uploaded and the nodes are configured right before the experiment starting time. The measurement nodes are programmed with the starting and ending time and the parameter sets for each of the executables that are going to be run for the experiment.

- **Experiment execution.** During this task the management kernel is on standby until the end of the experiment. It only has to ensure that no multiple experiments are using the same measurement nodes in the same time. The aim is to completely isolate the measurement nodes so that the high-precision measurement hardware can be exclusively devoted to a single experiment.
- **Results download.** Once the experiment is finished, the management kernel downloads the resulting data files from each measurement node to the temporary data storage. Since the download time is highly dependent on the network connectivity and the size of the output data files we incorporated resume capability into the file transfer. In this way the data transfer can be pended and resumed according to other scheduled experiments in order to avoid interference between data transfer and the experiment execution. With these functionalities the CMS is able to manage even very large datasets. After successful data download all the nodes are set back into the initial state and the remaining user files are deleted.

Periodic measurements as maintenance tasks. Due to the upgrade of the management kernel not only the users are able to define periodic experiments but also the CMS itself can carry out periodic measurements. A new kind of low priority task has been introduced into the kernel scheduler. These periodic measurements are planned as management tasks into empty slots of the calendars of the measurement nodes and executed only if the nodes are idle. If a node has become reserved for that time slot by a user experiment then the management task is canceled and automatically re-scheduled for a later time period and the user's experiment is executed. For the end-users these measurements are not visible and they do not affect the availability of the nodes.

3.1 Web Interface to CMS

ETOMIC provides an interface for researchers and administrators which fulfills the different requirements they may have. The former require capabilities to define new experiments, to reserve the measurement nodes and to download results from the CMS. The latter require functionalities to manage users, measurement nodes, software and experiments. An internal database is used to store all the necessary information to run the experiments. The stored information includes the applied softwares, external data files, the experiment results, the experiment status and the measurement node status. The strong usability requirements imposed by the researchers and administrators represented a significant challenge for the interface design. Solutions based on providing console access (`ssh` or `telnet`) to the nodes during the experiment were not deemed adequate, in order not to burden the researcher with launching the software at each node and with scheduling the tasks during the experiment. Instead, users are provided with a graphical interface for setting up the experiment beforehand. Then, the management kernel is in charge of experiment execution. The interface is based

on the ubiquitous web service. The aim is to facilitate the definition of the experiments as much as possible. This targeted simplicity and ease of use posed more stringent challenges on the researchers interface.

User interface. The main user interface used by the researchers is described in depth in [2] and [3]. Users can find manuals, the programming API and example codes here. Next we highlight the main functionalities and explain only the new features developed since the release of the original ETOMIC infrastructure.

- **Adding a new program**, executable files or source code of the measurement.
- **Uploading data files** if it is needed in addition to the executable files.
- **Creating an experiment-bundle** to define the experiment with scheduling the start and end times of the measurement.
- **Booking ETOMIC time** to reserve the measurement nodes and the certain time frame for the experiment.
- **Obtaining results**, downloading the resulting data files from the temporary storage to the user's own computer.
- **Periodic experiments** can be defined with a repetition period and the system with automatically schedule as many experiments as requested and allowed by the user privileges. Note that these periodic experiments run with normal priority, their time and node reservation is ensured by the CMS in contrast to system level periodic measurement described in Section 3.
- **Sharing files:** the users can share their executable files, data files or results directly with other ETOMIC users. The users are also welcome to publish their results in the open repository or to store their raw experimental data in the Network Measurement Virtual Observatory (see Section 5.2).
- **Web service interface:** a REST web service allows to download experiment results from scripts that the researchers may want to develop.

Administrator interface. Different administrator types are in charge of the ETOMIC system. The CMS and the measurement nodes are centrally administrated, but there are local administrators per measurement node for the physical maintenance. The central administrative tasks include the user management, to open new accounts, change user privileges and maintain the internal databases. Finally, the ETOMIC superuser has privileges to access and modify low-level system information and to abort any experiment programmed or in progress. All these features are provided through a user-friendly web interface.

3.2 Integration of Planetlab's Nodes

PlanetLab [5] is a global platform for supporting the development of new network services. This platform is also used for network experiments. The PlanetLab nodes are accessed interactively via remote shell. This access method enables the CMS to use the PlanetLab nodes as its own nodes. Although the main hardware capabilities of the PlanetLab and ETOMIC nodes significantly differ, the

large number of PlanetLab nodes makes them very attractive to the user community. The capabilities of PlanetLab are not described in this paper, here we only note that the PlanetLab nodes are usually up-to-date server PCs without any hardware components specialized for network measurements. The slice based management of PlanetLab nodes allows multiple users to run experiments simultaneously in the same remote node at the same time, while the CMS takes care of the unique resource allocation. In spite of the basic differences of PlanetLab and ETOMIC the federated usage of the high precision ETOMIC nodes and the numerous PlanetLab nodes could lead to new ways of experimentation.

The software installed on ETOMIC nodes has been adapted to make the joint usage possible, using a slice of PlanetLab that is automatically renewed by the CMS. This makes the whole range of ETOMIC and PlanetLab remote nodes available through the ETOMIC web interface. The most important challenge for the integration was the synchronization of the clocks in nodes from both platforms as they use different reference signals with highly different precisions. Using the ETOMIC interface, a software that is transparent to the end user takes into account the time difference between the nodes in order to allow synchronized launch of experiments at all the nodes.

4 The Measurement Nodes

In this section the hardware and software components of the ETOMIC measurement nodes are presented. The nodes can be divided into two groups based on their hardware architecture. The ones that are built on server PC architecture are called *ETOM*. These nodes are accessible via the web-based graphical interface presented in Section 3.1. The ones that are based on a lightweight programmable board are called *APE*. The APE nodes are accessible via a web service interface. GPS receivers are connected to all types of measurement hardware to provide the precise time synchronization between the nodes and to provide the reference clock for the measurement cards.

Each of ETOMIC's hardware solutions is high-precision, due to the incorporated precision equipment that are specifically designed to transmit packet trains with strict timing, in the range of nanoseconds. Each measurement node is provided by two network interfaces: a standard network interface card for management purposes (maintenance, software upload, data download) and an additional precision card for the probe traffic.

4.1 ETOMs with DAG Card

The first version of the measurement nodes are based on Intel S875WP1E server PC architecture with Debian Linux operating system with enhanced kernel capabilities for low level network access without root privileges. For the network monitoring interface an Endace DAG 3.6GE is used, which is specifically designed for active measurements. Such cards do not use interrupts to signal packet arrivals to the kernel, and thus packets can be captured at gigabit speeds. Shared

memory is used as means to relay packets to the analysis program running in user space, in such a way that interrupts are avoided. Furthermore, the GPS reference signal is used to timestamp the incoming packets with high resolution. To program the DAG cards a C language API and some user space applications are provided, e.g. a packet pattern sender and a packet capturer.

The GPS reference is based on a Garmin GPS 35 HVS. It is a water-resistant GPS receiver that is used to synchronize the measurement nodes. Specifically, the GPS provides a pulse per second signal directly to the DAG card. The resulting accuracy is 60 nanoseconds in the packet timestamps and inter frame generation intervals. A signal converter has been designed in order to bridge the mismatch between GPS receiver and PC ports. The range of the transformed signal allows large distances between the GPS receiver itself and the PC which enables good sky visibility for the GPS receiver unit.

4.2 ETOMs with ARGOS Card

The new generation ETOMs are based on a HP ProLiant ML370 server PC architecture with Ubuntu Linux. The quad-core server processor provides sufficient computation power to carry out complex measurements and online data analysis. Two heavy-duty disks are responsible for the safe and fast data storage during the measurements. These nodes are equipped with ARGOS measurement cards that are based on the netFGPA platform of Digilent Inc. ARGOS is capable of timestamping IP packets with nanosecond precision. In contrast to the DAG cards, it can act as a standard network interface card. Practically, this solution improves the development of probing applications, since users can program the cards through the standard socket interface and query the packet timestamps with the `libpcap` library. A few libraries and kernel modules were modified to support `sk_buf` structures with nanosecond resolution, which in combination with the ARGOS driver, makes it possible to read data directly from either the socket interface or `libpcap`.

In order to provide precise time synchronization a U-Blox LEA-4T high performance, high precision timing GPS is used. The LEA-4T features Time Mode function that enables GPS timing with only one visible satellite and eliminates timing errors which otherwise results from positioning errors. The precise geographic location of the GPS device should be provided once in advance.

4.3 APE Lightweight Measurement Nodes

Besides the development of the ETOMs a novel low-cost measurement equipment was designed which is called Active Probing Equipment (APE). The main feature of the APE nodes is that they provide measurement services which can be remotely called by user applications in an online fashion, without time slot reservation. The APE is built on a development board with Blackfin processor. The board is manufactured by Analog Devices Inc. and has a number of different interfaces for hosting auxiliary hardware components that are responsible

Table 1. Available measurement nodes in the ETOMIC system

	ETOM w DAG	ETOM w ARGOS	APE	PlanetLab
platform	Intel server PC	HP server PC	Blackfin board	variable
timestamping accuracy	60 ns	10 ns	100 ns	$\sim 10\mu s$
time synchronization	yes	yes	yes	no
GPS receiver	Garmin 35HVS	U-Blox LEA-4T	U-Blox LEA-4T	–
number of deployed nodes	18	20	20	~ 300
user interface	web GUI	web GUI	web services	web GUI

for specific network measurements tasks. For instance, a flash memory card was integrated onto the board to enable the temporary storage of measurement data.

The APE nodes are installed with uClinux, a special Linux operating system developed for embedded systems. The devices can be instructed via a web service interface, which opens simple ways for researchers to develop their applications using the network measurement services of the APE nodes. The following APE services are already implemented, and can be called (similarly to remote procedure calls): ping, traceroute, an arbitrary packet pattern sender and a capture tool.

5 Accessing Measurement Data

For network measurements the collected raw data is traditionally stored in files in some standard (like traceroute dump, tcpdump) or custom formats. The files are then processed according to the research questions to be answered. Detailed analysis of complex networks requires large statistical samples. This requirement leads to substantial data size in case of measurements in high bandwidth networks, even if just a few parameters of the packets are recorded (like IP addresses, arrival time, protocol, size or delay). Practically, measurements can produce dozens of megabytes at each monitoring node that sums up to hundreds of megabytes or even terabytes in multi-node experiments. Keeping only the results of the data analysis and discarding the raw data itself is not a good way to solve the data handling issues. Since measurement data gathered today cannot be reproduced in the future, it is preferable to store the original datasets to allow further re-analysis (applying the various different statistical methods to be developed in the coming years), and to support the study of the long-term evolution of the network.

For these purposes we have created data repositories to store measurement data collected by means of the ETOMIC system. There are two different interfaces to reach raw and aggregated measurement data. The periodic measurements web interface and a web service interface can be used to poll data collected from automatic measurements in the system. The users can choose from pre-defined queries

by selecting a given type of measurement, a set of nodes on which the measurement was conducted and a time frame.

We also present another interface which is called Network Measurement Virtual Observatory (*nmVO*). The nmVO approach aims at providing efficient and flexible access to raw experimental data and server side analysis tools [6]. It is also capable of integrating various databases in a common framework that provides full SQL functionality. Researchers may reach the nmVO through a web GUI or a web service interface, where the users can automatically poll data through their own client programs.

5.1 Web Interface for Periodic Measurements

To collect long term traces we have created periodic measurement tasks in order to run specific measurements when the nodes are idle. All the data from the periodic measurements is publicly available through an easy-to-use web interface, where the user can choose from pre-defined queries. For each measurement the user can download filtered data specifying only some pairs of nodes and a period of time. Once the selection is done the user can choose between downloading the data in a compressed text format or viewing a plot for those data provided by the server. In the following we list the types of periodic measurements for which the collected data is available through the ETOMIC website's `Database / Open Repository` menu [1].

NTP-GPS: NTP and GPS synchronization information is collected several times a day from every node. These are measurements collect information from the GPS devices connected to each monitoring node and from the NTP daemon running on the nodes.

One-way delay: Every possible pair of ETOMIC node takes part in periodic one way delay measurements. These measurements are done using the high precision measurement cards and GPS synchronized timestamping.

Traceroute: Traceroute measurements between every pair of ETOMIC nodes are collected. The measurements are done several times a day using standard Ethernet cards.

Paris traceroute: `Paris traceroute` [7] measurements are also conducted between every pair of ETOMIC nodes. The probe packets are designed to always use the same path even if load balancing is being applied in the network. Three types of packets (ICMP, UDP and TCP) are used in order to evaluate different behaviors of the network depending on the transport protocol. The measurements are run several times a day using the Ethernet cards.

5.2 Network Measurement Virtual Observatory

The Network Measurement Virtual Observatory concept, presented in [6] is an approach to efficiently store and share research data. Beyond the simple data

collecting and archiving functions it aims at providing easy-to-use analysis tools via both human and machine readable interfaces. One of the main features of the nmVO is that it provides SQL access for the databases that are integrated under its framework, thus the users can edit and run their customized queries through either the web-based SQL interface or the web services interface. The main advantage of this solution is that the researchers can filter out the relevant information from the huge archives using server side processing. Hence, only the necessary datasets and results have to be downloaded from the server.

To sketch the nmVO principle through a possible application, consider a scenario in peer-to-peer overlay networks where management information is needed to optimize the routing between the peers. It would be unthinkable to use gzipped files for such real-time evaluation. On the contrary, the scenario is feasible if one turns to the nmVO to get the typical loss rate, the average delay on certain routes or the shortest path between the peers. This means that beyond the data itself, analysis tools are also needed to perform such data filtering and transformation queries efficiently. The recent efforts on enabling database engines to run complex user code and define new data types (e.g. MS SQL Server CLR integration) makes this task easier. Using these stored procedures we can move the typical filtering and pre-processing tasks - like getting slices or aggregates of data - to server side. Since end-users can call these functions remotely they can reach the evaluated results with less code and they have to fetch much smaller amount of data from the archive. The XML based web service technology allows running either simple queries or more complicated functions that are stored on the server side, where the data is.

The virtual observatory concept is adapted to the ETOMIC system. The majority of the experimental data collected in the system are inserted into the data repositories and can be reached through both the nmVO web graphical user interface and the web service interface. For the data insertion tasks a C++ and a Python language API has been developed. This API is integrated into the new APE nodes' software, so that all measurement data from the APE boxes are automatically copied into the nmVO data repository. In addition to the historical raw data collections and the evaluated results of periodic measurements also non-ETOMIC traceroute logs and topology data, one-way delay values, queuing delay tomography data, available bandwidth results, router interface clustering and IP geolocalization data can be found in the archive. The users can freely construct SQL queries to intelligently filter the raw data for their purposes. In addition to that, for some of the most common processing tasks the nmVO offers built-in server side functions, e.g. for histogram calculation or fitting the parameters of Weibull distributions.

The nmVO can be accessed through the ETOMIC website's `Database / CasJobs Query Interface` menu [1] and via web services (for which the WSDL can be requested from the server) for client applications.

All the user information, the structure of the archive, the history, queue limitation settings, etc. are stored in a separate management database. This makes it possible to search among the queries in the history.

6 Conclusion

In this paper we presented the enhanced ETOMIC network measurement infrastructure. We described the key components of the architecture and the new features of the Central Management System. The improved system kernel includes support for periodic measurements and the federated usage of the high precision ETOMIC nodes and the numerous PlanetLab nodes. Besides the kernel development novel hardware components have been developed and deployed. New lightweight measurement equipments have been installed that provide measurement services which can be remotely called by user applications via web services. The system now also includes well structured data repositories to archive and share the experimental data. Periodic measurement data can be polled with customizable pre-defined queries, while the nmVO framework gives full SQL access to its archive. The recent developments make ETOMIC an easy to use experimental facility with versatile features for network research.

Acknowledgements

The authors thank the support of the EU ICT OneLab2 Project (No.224263), the EU ICT MOMENT Project (No.215225) and the National Office for Research and Technology (NAP2005/KCKHA005).

References

1. The ETOMIC website: `http://www.etomic.org`
2. Morato, D., Magana, E., Izal, M., Aracil, J., Naranjo, F., Astiz, F., Alonso, U., Csabai, I., Haga, P., Simon, G., Steger, J., Vattay, G.: ETOMIC: A testbed for universal active and passive measurements. In: IEEE TRIDENTCOM 2005, Best Testbed Award, Trento, Italy, February 23-25, pp. 283–289 (2005)
3. Magana, E., Morato, D., Izal, M., Aracil, J., Naranjo, F., Astiz, F., Alonso, U., Csabai, I., Haga, P., Simon, G., Steger, J., Vattay, G.: The European Traffic Observatory Mea- surement Infrastructure (ETOMIC). In: IEEE IPOM 2004, Beijing, China, October 11-13, pp. 165–169 (2004)
4. OneLab - Future Internet Test Beds: `http://www.onelab.eu`
5. Peterson, L., Anderson, T., Culler, D., Roscoe, T.: A Blueprint for Introducing Disruptive Technology into the Internet Workshop on Hot Topics in Networks (October 2002), `http://www.planet-lab.org`
6. Mátray, P., Csabai, I., Hága, P., Stéger, J., Dobos, L., Vattay, G.: Building a Prototype for Network Measurement Virtual Observatory. In: ACM SIGMETRICS – MineNet 2007, San Diego, CA, USA, June 12 (2007)
7. Augustin, B., Cuvellier, X., Orgogozo, B., Viger, F., Friedman, T., Latapy, M., Magnien, C., Teixeira, R.: Avoiding traceroute anomalies with Paris traceroute. In: Internet Measurement Conference (October 2006)

Characterizing User Behavior and Network Load on a Large-Scale Wireless Mesh Network

Michele Vincenzi[1], Roberto Tomasi[1], David Tacconi[1],
Dzmitry Kliazovich[2], and Fabrizio Granelli[2]

[1] Futur3 srl,
via Antonio Abondi, 37, 38100 Trento, Italy
{m.vincenzi,r.tomasi,d.tacconi}@futur3.it
[2] DISI – University of Trento,
via Sommarive, 14, 38123 Trento, Italy
{kliazovich,granelli}@disi.unitn.it

Abstract. Wireless mesh networks represent a promising paradigm to provide a scalable infrastructure for Internet access in metropolitan areas. In this paper, a large-scale wireless mesh testbed deployed in three cities in the Trentino region is described and experimentation results obtained from the public use of the testbed are reported and analyzed. The large-scale of the deployment and high number of users ensure to have proper traces which can capture the trends in user traffic based on the applications used and realistic mobility patterns.

Keywords: mesh testbed, large-scale, wireless mesh networks.

1 Introduction

Wireless Mesh Networks (WMN) represent a hybrid solution between infrastructure and ad-hoc networking paradigms, where data forwarding is enabled by all the nodes in the network. Potentially, all the nodes act as hosts and as routers, forwarding packets generated by other nodes. Mesh networking offers several advantages: (i) it allows the combination of different wireless technologies, such as cellular networks, WiFi, WiMAX, etc.; (ii) WMNs can be incrementally deployed, in order to gradually extend connectivity and capacity, avoiding massive investments. Moreover, WMNs autonomously set up and maintain the connectivity: if a node fails, another route to the gateway is found through another path, improving robustness, resilience, preservation and providing self-healing properties.

WMNs provide a technological bridge between mobile ad hoc networks (MANETs) and traditional wireless local area networks (WLANs), such as the ones based on the IEEE 802.11 family of standards. A typical WMN consist of several nodes (routers and gateways) which exploit multi-hopping in order to build and maintain a wireless backhaul. WMNs enhance traditional star-shaped network architectures by providing increased robustness (e.g. no single points of failure are present and broken/congested links are encompassed), scalability and flexibility (without the need for deploying cables, connectivity may be provided only where and when

T. Magedanz et al. (Eds.): TridentCom 2010, LNICST 46, pp. 255–264, 2011.

needed/economically attractive), and incremental deployment. Moreover, WMNs can support heterogeneous transmission technologies. WMN currently represent a promising paradigm for cost-effective deployments in several metropolitan area scenarios, including community networks, digital divide affected areas, mobile internet infrastructure, etc. Such scenarios can fully capitalize on the scalability, incremental deployment and robustness of WMNs.

Nevertheless, as WMNs become a service infrastructure to deliver high end services, effective design is needed to provide the required levels of service to the plethora of applications demanded by the users. To this aim, some theoretical or mostly simulation studies are available, but only a few works address realistic user behaviour characterization (both in terms of mobility and connection, but also application preferences) and provide extended analyses of the network load on a WMN infrastructure.

In this framework, performance evaluation and testing using results obtained from real testbed experiments become a vital requirement on the way of wide deployment and public offering. Most of the mesh testbed studies available in the literature are performed in a small- or medium-scale testbeds deployed inside a single building or a campus and are mostly focused on data transfer performance of individual flows analyzing underlining protocol semantics. Examples of such testbeds include Roofnet [5], UCSB Meshnet [4], as well as mesh testbeds at Georgia Tech. [6] and Carleton University [7].

The main contribution of this paper is in the analysing of traffic traces derived from a large-scale wireless mesh network testbed deployed in three cities in the Trentino region (Trento, Rovereto and Riva del Garda). Being used by citizen and tourists in those cities, such WMN is able to capture the trends in user traffic based on the applications used and realistic mobility patterns – and thus provide useful models for further development and optimization of WMNs.

2 Wireless Mesh Testbed Setup and Components

2.1 Network Setup

Futur3 manages a wireless mesh testbed deployed in the province of Trento (Italy). It covers the main areas of the city of Trento, the city of Rovereto and the northern region of the lake Garda. At the core of each city, one internet connection point is deployed which forms the basis for a 5 GHz network deployment using HYPERLAN and HYPERLAN2 protocols. Such Internet connection points create a so-called first layer network. A second layer of the mesh network is built on 2.4 GHz Access Points (APs) proving WiFi access to the end users. These APs form an extension of the first layer infrastructure and operate according to the IEEE 802.11b/g standard.

The coverage of the second layer network APs overlap, thus creating redundancy for the multihop paths of the mesh network and guaranteeing improved connectivity.

Network Access Servers (NAS) are installed in each region for assigning IP addresses configured via DHCP functionality. An IP address assigned to a user will remain the same for the entire session duration. Routing is handled at the MAC layer and the hand-over between neighbouring APs is supported.

Furthermore, authentication to Futur3 network is provided according to the Italian anti-terrorism law: a user needs to register once by giving either his Italian valid

mobile phone number or a credit card as a unique identifier. Once a user is registered, he or she can use his account all over the network. In addition, users are able to select the preferred level of privacy by updating their personal profile, changing the visibility of personal data, position or deciding to be not visible at all.

At the current state (February 2010), Futur3 network is composed by more than three hundred Access Points (APs) covering around 40 Km of streets continuously and around 16000 users are registered.

2.2 Localization

Futur3 localization is performed at three different levels of accuracy:

1. *Zone-based Localization:* A user is associated, given his IP address, to the zone (i.e. the corresponding NAS) he is connected to. In such a way, it is possible to associate an approximate position to every user.
2. *WiFi server localization:* From the user MAC address it is possible to know the AP a user is associated to. No particular software is installed on the user device.
3. *WiFi client localization:* For those users who have installed a client-side software, it is possible to acquire all the beacons sensed in the surrounding by the user and apply a triangulation technique to define a precise location.

Client localization works in combination with a localization server. Each client sends his list of sensed beacons to the server which computes the corresponding position estimates and sends it back to the client. The localization algorithm works in two subsequent steps: the first one provides a coarse position estimation, while the second one adjusts it.

Three different algorithms are evaluated for the first step: a simple centroid, a weighted centroid and a reduced weighted centroid. Centroid formula is an average of each APs' coordinates, while weighted centroid formula computes a position by averaging APs' coordinates after weighting them with their sensed RSSI. In order to define the last algorithm, the weighted centroid algorithm was applied to a subset of beacons obtained by filtering the sensed beacons according to their RSSI and distance from the previous client's position. RSSI filtering avoids location error propagation caused by multipath losses on the wireless channel. In a similar way, the distance filtering is due to the testbed position: apparently, water presence amplifies lake-side APs coverage area, thus affecting the overall results.

Location estimation is performed by both APs belonging to the Futur3 network as well as private APs. This requires an a-priori knowledge of the APs' positions which can be found using multiple wardriving sessions and a subsequent data analysis made by wigle.net technology [8].

In order to validate performance and precision characteristics of the performed localization, the testbed area is logically divided into a number of regions selected by isolating border areas. In each area the average error metric is calculated between each position computed by the location algorithms and a real position recorded in the system. Location error per region is reported in Fig. 1. The reduced weighted centroid algorithm appears to be better than the others because it has the lowest average error. Despite a large error in region 2 which has been accepted, average errors are much lower in other regions such as 4, 6, 8 (where the other algorithms have large errors).

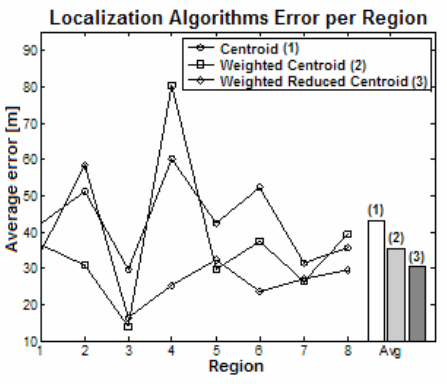


Fig. 1. Average error in meters computed in each region

Once a position of the user is estimated, the second step begins: the final latitude-longitude pair is selected from the set of available positions determined taking road topology of the area into account. At first, all positions within a range of 100 meters[1] from the coarse position are selected as candidate positions. Then, the final position is chosen by finding the position which minimizes both distances with the coarse position and with the previous valid one. Furthermore the selected position must be on the same road of the previous one; this constraint is ignored (and the second step repeated), either when the previous position is in proximity of street cross or if the distance from the coarse position is higher than 30 meters[2].

This client-server localization architecture doesn't charge clients with high computational load, enabling easy update of the positioning algorithms. Moreover, it reduces network overhead because the client doesn't have to query the database to fetch the available positions and the list of APs position used in the centroid algorithm.

3 The Measurements Campaign

3.1 Futur3 Network as a Data Collection Test-Bed

Futur3 network covers a wide area, with the purpose to offer both Internet connection and services to the largest number of people.

Typical users are residents of the area and university students, who connect during week days. A consistent fraction belongs also to the tourists coming to the area. Roughly 5-6% of the area population has joined the network, contributing to a growing penetration rate. This enables to have a suitable user base for performing a meaningful statistical analysis on the application and traffic traces.

[1] It is a value used to reduce the computational load because it avoids the server to process positions which are too distant from the previous valid one and therefore can't be selected.

[2] This represents the algorithm average error. When the distance between coarse position and position chosen accordingly to previous position is higher, it means that the user is in another street.

An additional advantage is that the network owner is Futur3; therefore it is possible to have simultaneous access to data about both users and network usage. On the user side, it is possible to perform accurate statistical analysis due to the knowledge of users' personal information such as age, sex and job [3], while on the network side it is possible to aggregate and process server activity patterns as well as APs load and links status. In detail, personal information about users is registered, their connection data in form of session time and traffic and AP usage in term of connection time and traffic; such data are periodically collected and aggregated in a data warehouse. Proper anonymization processes are used to guarantee privacy of the network users.

Moreover, Futur3 maintains a record of intranet activities related to its applications: users can interact with each other by exchanging messages and adding contacts. Analysing those data allows to study users behaviour and interaction in the social context. Furthermore, by using the provided positioning system, it is possible to study users' movement patterns.

The network covered area can be extended with on-the-fly installations, like in the case of the Blogfest event described in the following sub-section.

3.2 A Scenario of Interest: The BlogFest Event

Blogfest [9] is the second edition of an event hosted in Riva del Garda, a small city in northern Italy. The event is focused on the web community and its interaction within the Net; the aim is to allow people to speak about blog, social network and communities. It mainly consists of BarCamps organized and held by Italian bloggers all around the city. BarCamps are meetings held in different streets and squares within the old town, organized by bloggers on topics they usually write about, in which everyone can share his thoughts with the participants. BlogFest was held in October 2nd-4th, with more than 200 people attending 25 BarCamps. Moreover, there were a relax zone with radio entertainment and some stands of Internet companies and facilities such as food and kid areas.

In this context, the Futur3 WMN offered several services: (i) free Internet access, by improving the coverage of every area involved in Blogfest activities and most of the city area, (ii) introducing three beta applications based on the wireless positioning system. Those applications made Blogfest an interesting testbed, since they added the possibility to study users intranet interaction as well as their location and moving patterns. The applications provide information about other connected users, events and points of interest such as bars and restaurants which are nearby the user location, highlighting them on a map.

The data capture started the second day after the conference, when location services were deployed, at 11am. In order to be able to get access to Futur3's network and to use its applications, each user had to register to the network. 140 registrations were performed during the event: 20 the first day, 96 the second and 24 the third. The launch of the applications also explains the registration growth on the second day.

4 Experimental Data Analysis and Discussion

In this section an analysis of the available data is presented, with focus on users' behavior and network performance. The first sub-section introduces network usage and

load history. Then, users' behaviour is considered, focusing on their interaction with each other and with respect to Blogfest event. A relevant number of people were observed (259 single users connected during Blogfest's event).

4.1 Network Usage History

Futur3 network is a growing reality: starting from 1000 users of Jan.'09 there have been more than 3000 users joining the network with a constant trend in 10 months.

Analyzing in more detail October, it is possible to observe an average of 875 unique users, with a maximum of about 1000 along workdays and a minimum of 750 during weekends. This difference is due to the fact that many users are university students who live in the cities only during workdays.

The relevant number of people connecting every day allows performing a reasonable statistical analysis on the users' surfing behavior and habits. Traffic supported by the network is estimated in roughly 2500 GB per month and peer-to-peer traffic is not allowed. The average downloaded data in October is around 78250 MB, while the maximum is 92600 MB and the minimum is 51600. Lower loads are experienced nearby week ends, probably because of a lower number of people connecting.

Figure 2 shows APs' positions in the Riva del Garda area, with different markers representing the number of connections which have been registered in October. There are clearly hot areas, which mainly correspond to the old town.

However, such distribution is not directly correlated to the traffic load distribution, since there are some APs which had few connections but many downloaded data.

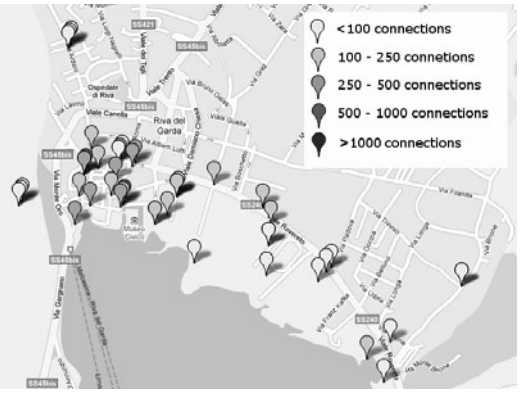


Fig. 2. Number of connections per AP in Riva del Garda

4.2 iPhone/iPod Application Usage

One of the most versatile applications was developed for iPod Touch and iPhone, which allows users to get access to all services and to surf the internet while moving. The application is interesting as iPhone can represent the reference for future mobile devices.

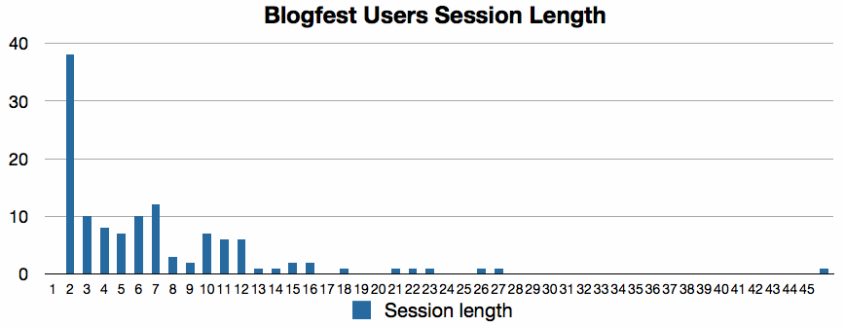


Fig. 3. Session length in minutes

Figure 3 describes the average session length in minutes. Several users had very short connections. Each user has had more than one session of different length; we measured around 5 sessions on average per user.

As far as movement is concerned, a mobility index is defined which estimates the level of movement of a user within one session. The index is defined as the ratio (covered distance)/(session length) measured in [meters/minutes]. Fig. 4 shows the indexes measured for each session. Sessions with mobility index lower than 60 are classified as very slow walk, or a session in which user didn't move while using the application. 72% of the overall sessions are characterized by such mobility index. However, 28% of the total sessions have a mobility index larger than 60, which means that users have used the provided application while moving around the area.
Table 1 underlines the interaction patterns among users, captured by the Futur3 application enabled users to add contacts and to chat among the WMN users.

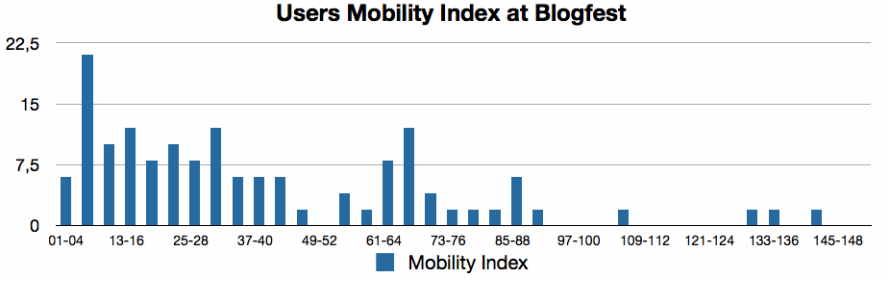


Fig. 4. Mobility index histogram

In the observation period, 35 new contacts were added by 14 over 31 overall users and 261 chat messages were registered. However, users didn't have long chat conversations. Most of the activities were done by users between 25 and 34 years old; those percentiles grow, by taking into account people younger than 24, to 80% for chat messages and to 78% for contact activities, as young people are keener to build social networks.

Table 1. Usage per age interval of features provided by the application

	<18	18-24	25-34	35-44	44<
Adding contacts	0%	21%	58%	21%	0%
Chatting	0%	13%	67%	20%	0%

4.3 User Distribution vs. BlogFest Activities

In this section, measurements are used to check the correlation between users' positions and Blogfest activities. Barcamps attendance is reported by highlighting on the map Barcamps' locations and indicating the users positioned there.

Figure 5.a shows a summary of users' participation at Barcamps held during saturday, Oct. 3rd in the afternoon. The purple area identifies both relaxing zone and a Barcamp. That area shows the largest number of people mainly due to the relaxing zone. The overall number of users do not match the number of Blogfest users because many of them have attended more than one event and have been in the relax zone.

Figures 5.b and 5.c show Barcamps attendance between 2pm and 3pm and between 5pm and 6pm, respectively. In those time intervals there were several active Barcamps. It is possible to observe a growing number of users connecting to the Internet while approaching evening. This is explained by taking into account the start of new Barcamps, which attracts more people.

Figure 5.d is a summary of Barcamps participation on Sunday morning. There were a few Barcamps active in the city, which causes most of the people to be in the purple area – i.e. the relaxing zone plus a Barcamp.

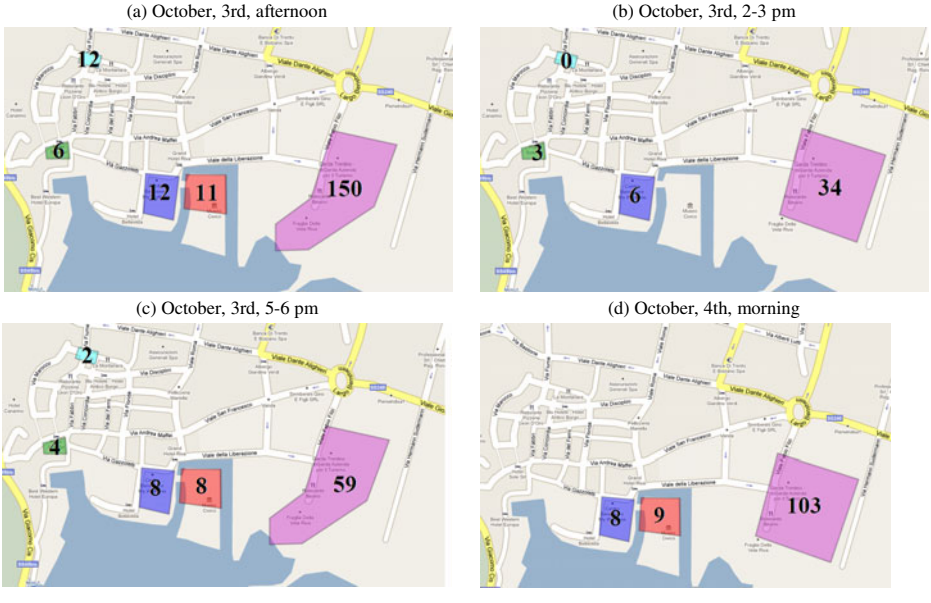


Fig. 5. Users' distribution at Barcamps and relax zone

4.4 Comment on the Results

Three different aspects, i.e. network usage, social behaviour and users' movement were analyzed, by taking advantage of a large WMN testbed and a special event, the Blogfest. The WMN has a sufficient dimension in term of connected users and traffic to allow conducting accurate statistical analyses. The possibility to estimate the position of the users identified areas highly loaded in term of connections and network traffic.

Basic social behaviour analysis is performed measuring the interaction among users allowed by intranet applications. Even tough most of the users were between 35 and 45 years old, younger users had most interaction in term of new contacts and chat traffic.

Moreover, a mobility index was defined, which allowed to discover that most of the users used the provided application while standing or for short movements, while 28% had relevant mobility.

During Blogfest events, it was possible to analyze the correlation between event and Internet connection (i.e. WMN resources usage). Knowledge of the users' location would be beneficial in the future for improving the coverage of the areas (in terms of transmission capacity, too). Based on the preliminary results obtained during the experiments, it seems reasonable to study methods for self-organization or self-resource allocation of the WMN possibly jointly with additional external information.

In addition, location-based traffic load information enabled to verify that some underutilized APs during regular days can become more crowded during a special event, but also that a WMN can be employed as a distributed locationing service to provide interesting insights to the events' organizers.

5 Conclusions and Future Work

The paper presented an analysis of measurements performed on a wireless mesh network deployed in three cities in Trentino. The presented test-bed is composed by a 350 APs network that provides access to around 16K users. Users can surf the web for free and receive also added-value applications to be used only as intranet services. Information and presented results are generated by analysing data gathered from users' normal behavior. In order to enable data information sharing within the scientific community, Futur3 is currently designing a testbed, working with the local university, in order to create a network subset with a similar behaviour.

The possibility of gathering data from a large testbed used by real users represents an important step in the investigation of large-scale access WMNs.

Results presented in the paper underline that most often the network usage is highly dependant on the APs' position and overall scenario (normal life or a special event). Moreover, the embedded locationing feature of the considered WMN enabled to check the percentage of moving users, which demonstrated to be relevant (28%).

Future works are aimed at using the gathered information in order to derive useful design guidelines for optimization and extension of the considered WMN.

References

1. Balanchandran, A., Voelker, G.M., Bahl, P., Rangan, P.V.: Characterizing user behavior and network performance in a public wireless LAN. In: Proceedings of the 2002 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems, Marina Del Rey, California, June 15-19 (2002)
2. Afanasyev, M., Chen, T., Voelker, G.M., Snoeren, A.C.: Analysis of a mixed-use urban wifi network: when metropolitan becomes Neapolitan. In: Proceedings of the 8th ACM SIGCOMM Conference on Internet Measurement, Vouliagmeni, Greece, October 20-22 (2008)
3. Mulder, S., Yaar, Z.: The user is always right: a pratical guide to creating and using personas for the web. New Riders Publishing, Indianapolis (2006)
4. Lundgren, H., Ramachandran, K., Belding-Royer, E., Almeroth, K., Benny, M., Hewatt, A., Touma, A., Jardosh, A.: Experiences from the design, deployment, and usage of the UCSB MeshNet testbed. IEEE Wireless Communications 13(2), 18–29 (2006)
5. Bicket, J., Aguayo, D., Biswas, S., Morris, R.: Architecture and evaluation of an unplanned 802.11b mesh network. In: ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom) (August 2005)
6. Akyildiz, I.F., Wang, X., Wang, W.: Wireless mesh networks: a survey. Computer Networks 47(4), 445–487 (2005)
7. Wireless mesh networking, Carleton University, `http://kunz-pc.sce.carleton.ca/MESH/index.htm`
8. Wireless Geographic Logging Engine: Making maps of wireless networks since 2001 (2001), `http://wigle.net/`
9. Blogfest (2009), `http://www.blogfest.it/`
10. HotSpot Management without the Pain, `http://www.coova.org`

Packet Tracking in PlanetLab Europe – A Use Case

Tanja Zseby[1], Christian Henke[2] , and Michael Kleis[1]

[1] Fraunhofer Institut FOKUS,
Kaiserin-Augusta-Alle 31, 10589 Berlin
{tanja.zseby,michael.kleis}@fokus.fraunhofer.de
[2] Technical University Berlin
Straße des 17. Juni 135, 10623 Berlin
c.henke@tu-berlin.de

Abstract. Making observations is fundamental for experimental research. Experimental facilities have to provide sophisticated and flexible tools to support scientific experiments by logging experiment results and monitoring environment conditions. Standardized measurements in experimental facilities can also provide consistent input for experiments with adaptive algorithms. In this paper we present a packet tracking architecture for PlanetLab Europe. Packet tracking correlates information from multiple observation points to observe the packet's path and experienced transmission quality. In our approach we use statistical sampling to control the measurement resource consumption and show how data selection processes at multiple observation points can be synchronized. As an example use case we show how the packet tracking architecture is used to support experiments on functional composition.

Keywords: packet tracking, multipoint measurement, hash-based packet selection, IPFIX, PlanetLab Europe.

1 Introduction

The current patchwork of protocols and the increasing demands on performance, flexibility and security in the Internet have led to a variety of programs on Future Internet research worldwide. They encourage, besides incremental solutions, new networking paradigms that follow a "clean state" approach, i.e. disruptive approaches that reconsider the IP protocol stack and try to find new networking designs. Functional Composition approaches revolve around the idea to functionally decompose the protocol stack and reorganize protocol functions in a composition framework in order to simplify the integration of new functionalities. Additional programs supplement theoretical research by large scale experimentation. Examples are the European Future Internet Research & Experimentation (FIRE) program and the US Global Environment for Network Innovations (GENI) program.

Experimental research is an essential building block of scientific work in order to proof a theory, investigate effects, and test new methods under real conditions before applying them in infrastructure and production environments.

One of the main enablers for thorough experimental research is measurements. As in other scientific disciplines, measurement and observation tools are needed to

T. Magedanz et al. (Eds.): TridentCom 2010, LNICST 46, pp. 265–274, 2011.

capture experiment outcome and to log experiment conditions. In [1] scientists from other disciplines strongly advise network researchers to take more care about measurements. They were shocked how limited the capabilities are to measure traffic in the Internet and how little effort is made in providing good observation tools. This advice comes from scientists from disciplines like biology, physics and astronomy who have a much harder time measuring their subjects of research hidden in human organisms, tiny atoms or far away in outer space.

In this paper we present an architecture for packet tracking that we implemented in PlanetLab Europe. We describe its benefits for the Future Internet experimenters in an example of a functional composition approach on application layer.

2 The Need for Measurements in Experimental Facilities

Scientists who perform experiments usually want to prove a theory, investigate a phenomenon or compare their own approaches to others. Observation tools in an experimental facility should be flexible in the sense that experimenters are able to zoom in or out to investigate traffic with different granularities. Furthermore, the tools should provide information about the accuracy of result data to provide comparability with results from others. Measurements in experimental facilities for Internet research can serve three different purposes:

- ***Experiment Supervision:*** Experiment supervision captures the results of the experiments, i.e. the input and output parameters under investigation, and provides them to the experimenter.
- ***Environment Supervision:*** Environment supervision captures further parameters not directly under investigation that may or may not be relevant for the experiment outcome. Especially parameters that cannot be controlled during the experiment but may influence the results need to be monitored.
- ***Measurement Service:*** Besides the capturing of results and conditions, the measurement tools can provide input to algorithms that require measurements for operation (routing, adaptation, learning). Providing a general measurement service allows researchers to use a common basis and eliminates the need to develop and deploy their own proprietary measurement solutions. Common input formats also support the comparability of results from different algorithms.

PlanetLab runs as an overlay over classical Internet connections where traffic from experiments interferes with many other flows. The underlying network, environmental conditions, physical effects, weather, etc. cannot be controlled by the experimenter. It is exactly such real unpredictable conditions, with potential undiscovered side effects, that make experimental research attractive. An ideal controllable environment would be unrealistic; simplified controllable settings would not differ much from a simulation. In uncontrollable environments a good documentation of experiment conditions is crucial. We may not be able to repeat all experiment conditions, but a good documentation helps to investigate observed side effects and analyze potential correlations if results differ from theory.

Our packet tracking architecture can substantially provide such measurement information as we will show based on the use case in Section 5. Path and delay information is relevant for a variety of experiments in the area of routing, overlay construction and optimization, congestion control and for any experiment with adaptive algorithms that require QoS values as input.

For measurements in federated testbeds it is also desired to send measurement results between different administrative domains. For this it is important that result data transfer uses protocols that support congestion control and encryption.

3 Related Work

PlanetLab is a global experimental platform for large scale experimental research. Slices that allocate resources on Planetab nodes are assigned to researchers who can use them for their experiments. The status of PlanetLab is reported by several programs and tools that view different aspects of the nodes and their connectivity.

Everstats [14] collects information from the slicestat [15] program running on each PlanetLab node which provides slice-level resource consumption information on each node. **CoMon** [16] provides the status of all PlanetLab nodes. It supports node-centric view statistics and a slice-centric view to see how a slice is consuming resources. **PlanetFlow** [20] is a flow measurement tool that logs the outbound network activity of all nodes. It is based on fprobe, which is a data collector using the Netflow format. For the ORBIT (wireless) testbed a framework is provided for control, measurement and resource management. For this the **ORBIT Management Framework** (**OMF**) uses the ORBIT Measurement Library (**OML**) to collect any type of measurement into a database. Everstat, CoMon and PlanetFlow trace activities and collect information on PlanetLab nodes. They provide user defined views but not user defined measurements. GIMS is still in the phase of specification. OMF already proposes measurements defined by applications and may be a good candidate for integrating packet tracking functions in wireless networks. To support measurements in GENI it is planned to establish a **GENI Instrumentation and Measurement Service (GIMS)**. The group has identified challenges with regard to data archiving, privacy issues, and the separation of measurement data from experiment data.

PlanetLab Europe is a European part of the worldwide experimental platform PlanetLab administered by the PlanetLab Europe Office in Paris. The European Project OneLab [2] works on the provisioning of highly sophisticated control and monitoring functions to support the needs of experimental-driven research in PlanetLab Europe. The project integrates further technologies like wireless testbeds and platforms for disruptive Future Internet research like autonomic communication, delay tolerant networks and pocket switched networks. The project is also working on solutions to federate with other experimental facilities.

The OneLab project has developed a measurement solution that integrates passive and active measurements. The **Advanced Network Monitoring Equipment (ANME)** is specified in [10]. For passive measurements it uses the Continuous Monitoring (CoMo) platform [11], [12] which allows several experimenters simultaneously to perform arbitrary traffic queries on the data streams on their slices. CoMo

supports user defined modules and resource control for measurement tasks. The mapping between PlanetLab slices and their traffic, and access to their relevant measurement data is controlled via a proxy described in [13]. The proxy provides a clear separation of experiments and ensures that experimenters can only access results of their own experiments. The deployment of the ANME and CoMo software in PlanetLab Europe is currently in progress. The service can be accessed at www.packet-tracking.de.

The presented packet tracking solution will be integrated as modules in the CoMo framework in order to use the CoMo and ANME control functions for experiment control, result data access and GPS-based time synchronization.

4 PlanetLab Europe Packet Tracking Architecture

4.1 Multipoint Packet Tracking Architecture

Following packets on their path requires a passive multipoint measurement architecture. This architecture consists of multiple measurement probes and a collector that correlates the probes' measurements. The measurement probes are located at observation points which can be any device that has a connection to the shared medium, such as router or network card. The probes export a packet ID and either the TTL or an arrival timestamp for each observed packet to the collector. Based on the packet ID the collector can correlate the observations and determine the packets' direction by the TTL or timestamp. Packet tracking can also be used for calculating one-way delay between the observation points, but this imposes additional requirements such as a common time base on the architecture. Passive delay measurements will be made available when the architecture is integrated in CoMo in the GPS-enabled ANME boxes.

In the following we present previous results on packet ID generation and hash-based packet selection as these are critical for proper packet correlation and sampling synchronization between measurement points.

4.2 Packet ID Correlation

In order to follow a packet's path through the network we need a unique identifier for each packet. The datagram identification field in the IP packet header was considered as ID in [3], [4] [5], but it is not sufficiently unique because of its limited 16bit size and different handling of the ID from OS (some OSs use an ID of 0). Another possibility is to generate a packet digest over packet header fields and part of the content. The packet ID generation should be done in a way that 1) the resulting ID is small to reduce measurement traffic; 2) the ID is fast to calculate to reduce resource consumption on the node; 3) the probability for collisions (getting the same packet ID for different packets) is low and 4) the digest is calculated over packet parts that do not change on the way.

Fields that are mutable but predictable could also be used for packet ID generation. Furthermore, it is advantageous (but not required) to use an operation that always leads to an ID with the same fixed length. This eases the handling, transmission and the estimation of the overhead caused by measurement export.

The amount of content used for ID generation influences the number of collisions (different packets that map to the same ID) that can occur. It is advantageous to consider fields that are highly variable between different successive packets; fields with low variability (e.g., version field) should only be included as long as there is no significant performance decrease. Investigations into this and comparisons of different packet ID generation methods can be found in [4], [5] and [6].

4.3 Coordinated Sampling by Hash-Based Packet Selection

Network traffic is highly dynamic and hard to predict. General challenges of passive measurements are unforeseen high data rates that can lead to an unexpected exhaustion of measurement resources. Data selection techniques, like sampling and filtering, aim at the reduction of measurement resources (processing, storage, transfer of data). This is achieved by selecting a finite subset of elements (the sample) and estimating the metric of interest from this subset. Especially random packet sampling techniques provide a solution to reduce the measurement traffic while still enabling an accuracy statement about the estimated characteristic. Nevertheless, if we apply random sampling, we randomly select a subset of the packets that traverse the observation point. This is a problem in multipoint measurements, because the random selection processes at the involved observation points would select different packets, making packet matching impossible. So if we want to apply random packet sampling at multiple observation points we have to somehow ensure that the same packets are selected at the involved observation points. We therefore use hash-based packet selection as proposed in [5] and [7] to emulate a random selection and synchronize selection processes at different observation points.

The packet selection is based on a hash function on invariant packet header fields and parts of the payload. If the same hash function and the same hash selection range is used at all involved observation points then packets selected at one observation point are also selected at the other observation points.

A problem with hash-based packet selection is that it is a deterministic function on the packet content which can introduce bias into the selection. In [6] we investigated several hash functions with respect to the achievable random properties and their suitability for multipoint measurements.

Due to the differing requirements for packet ID generation and packet selection [4], [5] and [6] recommend using two different functions for packet ID generation and packet selection. Nevertheless in [7] we could show that for small to medium sized measurement domains it is reasonable to use the same hash function.

Since PlanetLab Europe is a large federated testbed with distributed measurements, we decided to use a CRC32 function for packet ID generation and the BOB function as evaluated in [7] for hash-based packet selection.

4.4 IPFIX and PSAMP

In experimental research, measurement data provides the basis for the formulation of scientific results. It is crucial to agree on standard measurement methods and result data formats to be able to compare the outcome of experiments and to share data with others. Standardized formats simplify the development of tools (e.g., for data analysis) and allows the provisioning of reference data.

In our approach we utilize and exploit approaches for data export and data selection, which are developed with our contribution within the IETF working groups on IP Flow Information Export (IPFIX) and Packet Sampling (PSAMP). IPFIX is a protocol to export flow measurement results. It supports encryption of the transferred data and congestion control. Those features are essential for transferring results among different administrative domains e.g., in federated testbeds. IPFIX defines Information Elements (IE) [8] to report flow information. The PSAMP standard extends these Information Elements with additional fields for packet reporting and sampling. Our group in Fraunhofer FOKUS has developed an IPFIX Library which we used for exporting measurement results in our PlanetLab Europe packet tracking implementation. Since the PSAMP information model is required for the representation of packet IDs in multipoint packet tracking we extended our implementation to support PSAMP information elements.

We expect that in future standard routers will provide IPFIX data. Cisco Systems is working on IPFIX as successor to Cisco NetFlow. As a result of this, data from PlanetLab Europe (PLE) experiments become comparable to data recorded in other networks. The IPFIX group is also working on a file format to store packet and flow data. We contributed to this standardization effort [9] because we believe that this is an important step towards sharing data from experimental research among researchers and provides the right way towards re-usability of analysis tools. We plan to use the file format for storing packet tracking data in PlanetLab Europe.

5 Use Case: Functional Composition on Application Layer

Cooperative Service Provisioning (CSP) [18][19] is a service composition architecture that utilizes a decentralized approach for service path discovery in order to combine services at customer's premises. CSP supports functional composition on the service layer, which we will present as a use for packet tracking.

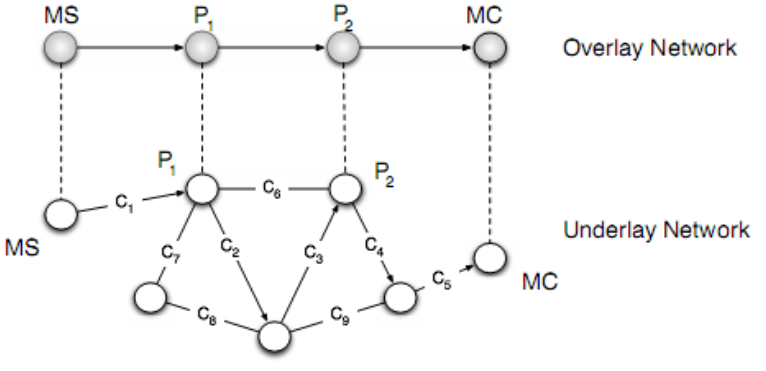


Fig. 1. Overlay Based Service Composition. A media server (MS) publishes a stream that the media client (MC) wants to receive. CSP dynamically creates a service path which satisfies MC's requirements by incorporating services offered by other peers.

In a collaborative P2P environment peers may offer services like caching and transcoding for other peers that do not have the resources to watch a media stream in the originally offered format, e.g., a mobile client peer that wants to watch a high definition video but only features QCIF. CSP organizes the peer nodes and their offered services in a structured distributed hash table (DHT) CAN [18] address space which enables a service path composition that satisfies the clients' requirements (see Figure 1). Because CSP is organized in a DHT it is also failure resilient. When a peer leaves the Overlay, another peer becomes responsible for its share in the CAN.

The research challenge here is the establishment of the function chain. In an environment where multiple nodes offer services that can be included (e.g., caching, transcoding, encryption, etc.) the algorithm needs to find the required functions (based on application demands) and combine them in a reasonable manner.

Since service composition for realtime media services and also for functional composition is QoS sensitive, it has to be ensured that the QoS constrains are not violated by the service composition process. As a consequence service composition resembles a (Multi-) Constraint Routing problem [17] where the construction of the function chain already becomes NP complete.

In previous work we investigated algorithms for the composition [18][19] and made a first simulation implementation on a few nodes. Since results are dependent on characteristics of the underlying network, we planned to test the algorithms in a realistic testbed. We decided to use PlanetLab Europe for the experiments because the traffic runs on real Internet connections and we are able to use nodes that are highly distributed around the globe. The ability to use many highly distributed machines is essential for the investigation of the composition algorithm that depends on the number of available functions and the QoS parameters between nodes.

In our experiments we investigate whether the function chain is correctly established and the packets traverse the network on the expected paths. The packet tracking service here is required for experiment supervision and providing path information. In future packet tracking can also be used to provide input data for overlay establishment and optimization (as a measurement service). It can also log environmental settings because adjacent traffic from other applications in the network will also be measured. In this way one can also examine the influence of different flows within the network.

6 Measurement Setup for Packet Tracking

Basically Planetlab nodes are end hosts which can directly communicate over the Internet. This means that none of the PlantLab nodes serves as a router and will forward packets. Therefore we created an overlay network on nodes in our slice in a way that at least one intermediate node does packet forwarding. We created virtual tap interfaces with their own addressing space upon the original (eth0) interfaces on nodes in our slice. These virtual tap interfaces are interconnected using socat tunnels. We are able to create multiple virtual tap interfaces, but forwarding between those

interfaces is not yet supported in PlanetLab, although planned. Until this feature is enabled we are using one intermediate machine in our facilities that connects with the tap interfaces on the PlanetLab machines, creating a star topology with our machine in the middle. Our machine is able to forward packets to the PlanetLab nodes and work as an intermediate router (cf. Figure 2). In addition, we deployed OpenIMP [21] measurement probes on all involved nodes (PlanetLab and our middle machine).

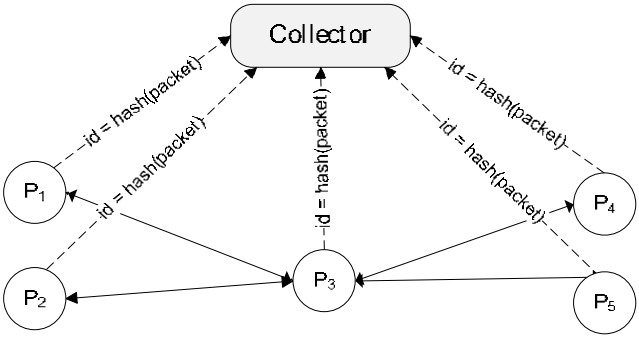


Fig. 2. Each PlanetLab node has a probe that applies hash-based selection, generates a packet ID for each selected packet and sends IPFIX results to a common collector. The nodes are interconnected using socat and a central node that forwards packets.

For performing accurate passive multipoint measurements it is necessary to separate measurement traffic from the measured traffic, i.e., in a packet tracking architecture the IPFIX measurement result traffic should not be observed. Therefore we export our measurement traffic over the original eth0 interface while the probe listens on the virtual interface. This setup is not recommended for delay measurements because the measurement traffic and real traffic is actually exported over the same (real) interface which can distort delay measurement. In a first demo scenario we used iperf to generate traffic between 5 nodes (KTH Stockholm, ELTE Hungary, Quantavis Pisa, Porto and University of Athen) and our node in Berlin in the middle.

7 Packet Tracking Visualization

Packet Tracking is very graspable; everyone can intuitively imagine how a packet traverses its path. We implemented a Java Application which uses OpenStreetMap and an animation framework to visualize the measurement data. The observed packet tracks are aggregated in user-defined intervals and displayed as animated light dots where each dot represents multiple packets. For the path visualization we used cubic splines so that the path from Point A to B does not cover paths from Point A to B to C, i.e. one can follow the packets path from ingress to egress. An example screenshot is shown in Figure 3.

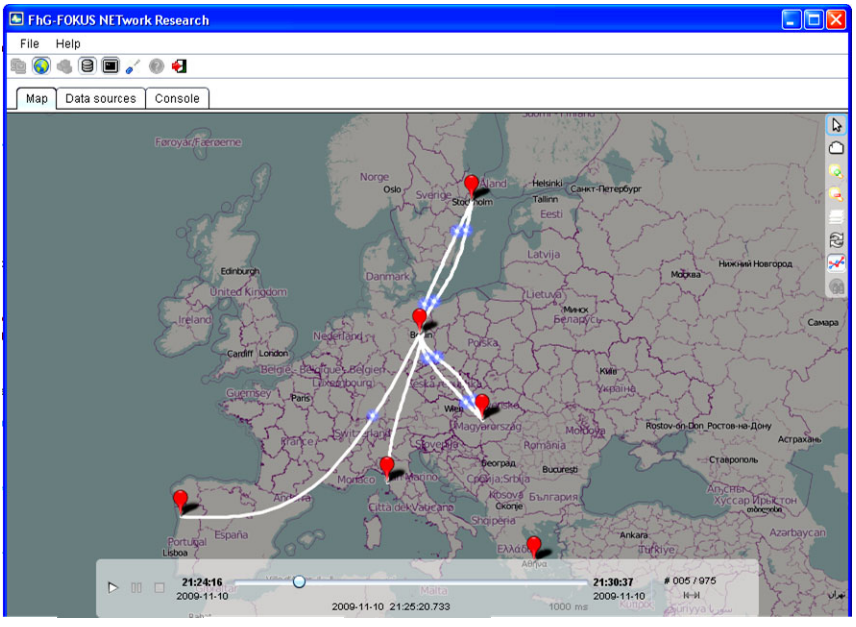


Fig. 3. Screenshot of Visualisation Tool for Packet Tracking in PlanetLab

8 Conclusion and Acknowledgements

We introduced a packet tracking architecture for PlanetLab Europe for experiment and environment supervision and as a general measurement service for algorithms that require such input. We showed how packet tracking with multiple observation points and coordinated data selection techniques can be realized. We presented a typical use case and showed how results can be visualized. We plan to integrate the service with the OneLab Advanced Network Monitoring Equipment (ANME) and to extend the architecture for passive delay measurements and result storage in IPFIX file format. We also plan to integrate packet tracking functions in the recently federated wireless networks of PlanetLab Europe.

This work was partly funded by the EU project OneLab2 under the FIRE program. We thank Joachim Kaeber for the PlanetLab Europe setup and Tacio Santos for implementing the visualization. Special thanks go to the PlanetLab Central and Europe support team Guthemberg Da Silva Silvestre, Anil Kumar Vengalil, Benjamin Quétier, Sapan Bhatia and Thom Haddow for quick responses and technical support.

References

1. Committee on Research Horizons in Networking, Computer Science and Telecommunications Board (CSTB). Looking Over the Fence at Networks: A Neighbor's View of Networking Research (2001). National Academy Press, Washington, D.C (2001)
2. OneLab, `http://www.onelab.eu/`

3. Graham, I.D., et al.: Nonintrusive and accurate measurement of unidirectional delay and delay variation on the Internet, INET (1998)
4. Zseby, T., Zander, S., Carle, G.: Evaluation of building blocks for passive one-way-delay measurements (April 2001)
5. Duffield, N.G., Grossglauser, M.: Trajectory sampling for direct traffic observation. IEEE/ACM Transaction on Networking 9(3), 280–292 (2001)
6. Henke, C., Schmoll, C., Zseby, T.: Empirical Evaluation of Hash Functions for Multipoint Measurements. ACM SIGCOMM Computer Communication Review (CCR) 38(3), 39–50 (2008)
7. Henke, C., Schmoll, C., Zseby, T.: Empirical Evaluation of Hash Functions for Packet ID Generation in Sampled Multipoint Measurements. In: Moon, S.B., Teixeira, R., Uhlig, S. (eds.) PAM 2009. LNCS, vol. 5448, pp. 197–206. Springer, Heidelberg (2009)
8. Boschi, E., Trammell, B., Mark, L., Zseby, T.: Exporting Type Information for IPFIX Information Elements, RC5610 (July 2009)
9. Trammell, B., Boschi, E., Mark, L., Zseby, T., Wagner, A.: An IPFIX-Based File Format. RFC5655 (October 2009)
10. Vattay, G., Fekete, A., Hága, P., Stéger, J., Aracil, J.: Giuseppe Iannaccone. Specifications for advanced monitoring boxes, OneLab Deliverable D4.2 (November 2008), http://www.onelab.eu/index.php/results/deliverables.html
11. Iannaccone, G., Niccolini, L., Duca, A.L., Canonico, R., Zseby, T., Witaszek, D., Kaeber, J.: Packet tracking architecture, OneLab Deliverable D5.1 (May 2009), http://www.onelab.eu/index.php/results/deliverables.html
12. CoMo Project, http://como.sourceforge.org
13. Iannaccone, G.: Fast Prototyping of Network Data Mining Applications. In: Proceedings of PAM (March 2006)
14. Everstat, http://everlab.cs.huji.ac.il/planetstats
15. Slicestat, http://codeen.cs.princeton.edu/slicestat/
16. CoMon, http://comon.cs.princeton.edu/
17. Korkmaz, T., Krunz, M.: Multi-Constrained Optimal Path Selection. In: Proceedings of IEEE INFOCOM, Anchorage, Alaska, USA (2001)
18. Kleis, M., Radier, B., Elmoumouhi, S., Carle, G., Salaun, M.: A decentralised service composition approach for peer-to-peer video delivery. Journal Peer-to-Peer Networking and Applications (2009)
19. Kleis, M.: CSP, Cooperative Service Provisioning using Peer-to-Peer Principles, PhD Thesis Chair for Network Architectures and Services Department of Computer Science, Technical University Munich (2009) ISBN 3-937201-05-X
20. PlanetFlow, http://www.cs.princeton.edu/~sapanb/planetflow2/
21. OpenIMP, http://www.ip-measurement.org/index.php?id=8&Itemid=8

A Zero-Nanosecond Time Synchronization Platform for Gigabit Ethernet Links

Carles Nicolau

Dept. of Information and Communication Technologies,
Universitat Pompeu Fabra, Barcelona, Spain
carles.nicolau@upf.edu

Abstract. Ethernet is becoming the dominant data transmission technology for service providers due to its simplicity and low cost. However, the need for Quality of Service (QoS) provisioning for some time-sensitive applications drives the definition of new functionalities in Ethernet. In this work we address QoS in Ethernet by introducing a time synchronization capability at MAC level while maintaining its asynchronous and distributed architecture. We present a reconfigurable logic platform based on a Field Programmable Gate Array (FPGA) in which we embed our custom timestamping unit (TSU). Leveraging the TSU, we have implemented a synchronization mechanism with which we achieved a best-case synchronization accuracy of zero nanoseconds. The effectiveness of the method is confirmed through several experiments.

Keywords: Ethernet, Quality of Service, Time Synchronization, FPGA, HW-timestamping.

1 Introduction

Ethernet technology is becoming the dominant access technology at local, metropolitan, and wide area networks. The low cost, high data rates and low complexity and maintenance offers an opportunity to service providers for using Ethernet on a very large scale and replacing their expensive legacy networks to more simple Layer 2 (of the OSI layered model) Ethernet networks. However, low cost and simplicity are only part of Ethernet's attraction. The challenge is that Ethernet is a 'best-effort' and asynchronous oriented technology, therefore industry bodies and equipment manufacturers are making considerable efforts to ensure Ethernet services become 'carrier-class'. High-precision time synchronization is a key enabler for offering such carrier-class QoS, a functionality that present native Ethernet lacks and that provides receivers with precise information about end-to-end delays, and that would benefit e.g. effective playout of time-sensitive data [1] or high-precision network analysis [2].

A few years ago the Ethernet Passive Optical Network (EPON) specification [3] introduced a rough synchronization capability in Ethernet based on timestamping for the purpose of coordinating the slotted access of the nodes. Synchronization accuracy and precision using timestamps are affected by the

T. Magedanz et al. (Eds.): TridentCom 2010, LNICST 46, pp. 275–284, 2011.

time variability of inserting the time information inside the message [4]. With a well-defined and characterized timestamp mechanism, we believe there is a chance to define a synchronization mechanism for native Ethernet to better support time-sensitive applications with more control of the QoS provided. Our objective is to improve Ethernet capabilities within an acceptable range while respecting its architecture simplicity principle. Hence, we address QoS by introducing a time synchronization capability but maintaining its asynchronous and distributed architecture.

With the former perspective in mind, we present a testbed for the design extension of native Gigabit Ethernet to address the QoS through time synchronization at Layer 2. The testbed is based on a FPGA which provides the resources for designing a custom and flexible hardware (HW) Timestamping Unit with the following improved key features from our previous work [5]: **1)** high-resolution and reliable HW timestamping, **2)** precise timing of the execution/processing times of the internal platform key blocks, with which we can optimally characterize the timestamping mechanism, and **3)** high-resolution calculation of message propagation times. Leveraging the three previous properties, we reuse the synchronization mechanism specified in EPON to achieve a perfect link synchronization between two nodes.

We start this work in Section 2 which pinpoints the actual solutions akin to our work. Section 3 is devoted to clarify the requirements of our platform, as well as to explain in detail its design and the different scenarios to obtain reliable timing and synchronization performance. In Section 4 we present the experimental results for characterizing the timestamping mechanism and the achieved synchronization accuracy. We draw conclusions in Section 5.

2 Related Work

Considering the requirement of maintaining the actual native Ethernet structure, i.e. without redefining the physical layer (PHY), other standards such as the IEEE Std. 1588 Precision Time Protocol (PTP) [6] and the Network Time Protocol (NTP) [7] are being developed. Both protocols distribute timing via UDP packets that carry timestamps generated by a master (server). PTP achieves up to six orders of magnitude better synchronization accuracy than NTP by relying on HW support for generating the timestamps. The hundred-ns accuracy target of PTP is consolidating it as THE standard for synchronization timing distribution over Ethernet in many different areas.

Since the last revision of the PTP standard in 2008, a new normative for transporting PTP over IEEE 802.3 Ethernet is considered (Annex F of [6]), a feature that is akin to our work. This new specification prospects a fully IEEE 1588 Layer 2 subsystem independent from upper layer services, and thus also reinforcing our initial beliefs on the success of a pure Layer 2 solution for Ethernet. This new PTP normative will benefit from the use of shorter messages (64 bytes) and the possibility to build PTP with small memory footprint and resources, even integrated in a single Ethernet chip. To our knowledge, we have

proof of one work [8] implementing this new *IEEE 1588 Layer 2* specification. The authors are targeting synchronization accuracies of tenths of ns between contiguous nodes.

3 Design

3.1 Platform Requirements

The major objective in the development of our platform is to provide a low-cost, programmable and flexible framework for experimenting with new Ethernet design extensions at low level. FPGA-based embedded platforms are the best tools for this purpose. However, one limitation of commercial Ethernet platforms is that their compliant MACs are black boxes, avoiding researchers to integrate and experiment with new MAC functionalities. Here we want to rapidly work around this problem and avoid (the hard task) of designing a compliant MAC from scratch. In this design we focus on the following high-level requirements: **1)** fast re-usability with other platforms, **2)** easy upgradeability with improved or new functionalities, **3)** compatibility with SW-based synchronization solutions [7], thus providing Ethernet with autonomy and independence from upper layers to perform fully Layer 2 functions. **4)** Keep the asynchronous and distributed Ethernet philosophy, avoiding synchronization approaches based on re-spinning the Layer 1 transceivers [9].

3.2 Embedded System

In Fig. 1 it is shown the reconfigurable platform, the Xilinx ML403 general purpose board which is based on a Virtex-4 FPGA [10]. The board mounts a tri-speed Ethernet PHY, several expansion headers (HDR) for external connectivity, 64MB of DDR SDRAM for massive storage and two XO's for clocking the PHY and the rest of the FPGA internal circuitry. The FPGA chip contains a PowerPC microprocessor (μP), an Ethernet MAC block (MAC), a digital frequency synthesizer (DFS) and reconfigurable logic with which we have implemented our Timestamp Unit (TSU).

The TSU is divided into four major blocks. The first one is the transmission block (tx block) which contains a finite state machine (tx FSM) that detects and manages the replacement of some synchronization frame fields. *tx block* operation is synchronous with the transmission clock signal (*tx_clk*) coming from the MAC ('- -' line).

The second block is the reception block (rx block) which has the same functionality as its transmission counterpart besides cyclic redundancy error checking of the incoming frames. The operation of *rx block* is synchronous with the reception clock signal coming from the on-board PHY chip ('· -' line).

The third block is the control adjustment unit (CAU) which contains a 32-bit counter (*localtime*) that is summing up clock ticks at the frequency of 269.96 MHz ('··' line), thus providing a timestamp granularity of 3.704 ns. We will use *localtime* counter to keep track of high-resolution time at MAC level and as the

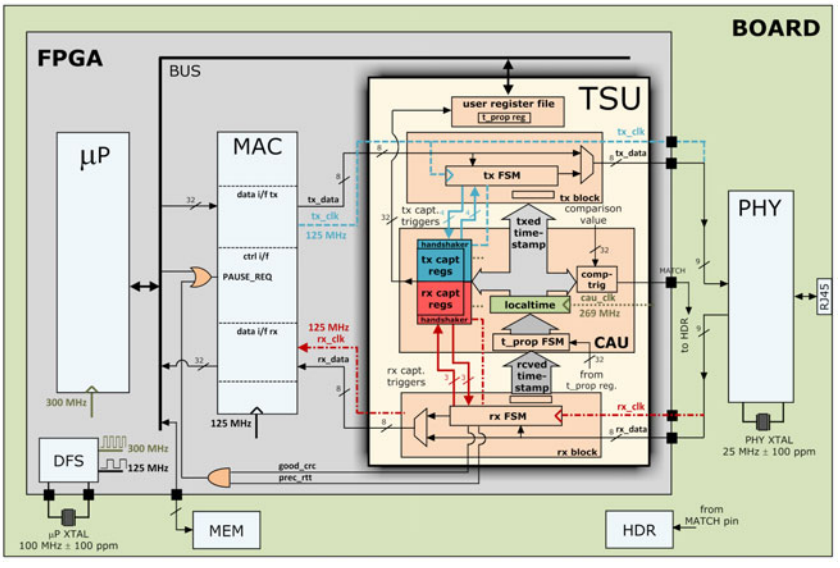


Fig. 1. Simplified block diagram of the reconfigurable platform

fine time measurement unit with which to compare the synchronization accuracy between nodes. As seen in Fig. 1, *localtime* register is interfaced to/from four regions:

1. to a capture register file within CAU, for storing different time freeze points (see Section 3.3) of the outbound and inbound frames (*tx capt regs* and *rx capt regs*, respectively). Each FSM triggers several timestamps when transmitting/receiving frames that are stored in *register stack*. The main problem of this approach is the interaction of the register stack with the three clock sources (*rx_clk*, *tx_clk* and *cau_clk*). This cross-clock domain scenario can hamper the reliability of the timestamps due to metastability errors [11]. In order to prevent inconsistent timestamps, we have introduced handshake operations (*handshaker*) between the capture registers and *localtime*. *handshakers* consist on a FSM and a flancter circuit [12].
2. to a comparator-trigger block (*comp-trig*) within CAU, for comparing *localtime* instantaneous values with a user-defined constant value (*comparison value*). When *localtime* equals *comparison value*, a 50 ns-wide pulse is raised on an external output (*MATCH*). This resource will provide us means to verify the synchronization accuracy between nodes (see Section 4.2).
3. to the transmission block (*tx block*) within the TSU, to provide the timestamps to be transmitted in the outbound synchronization protocol data unit (syncPDU). The timestamps are triggered at the 25^{th} byte of a current outbound frame, and inserted from 29^{th} to 33^{rd} fields (see Fig. 2).
4. from the reception block (*rx block*) within TSU, to read the incoming timestamps and syncPDUs. The time of arrival of a syncPDU is also taken at the

25^{th} incoming byte to keep the symmetry with the outbound path. *rx block* contains a state machine (*t_prop FSM*) that performs the propagation time correction with the content stored in *t_prop reg* register.

The last block is the *data register stack* register file which contains all the information of the last sent/received syncPDU, internal timing measurements and control/status registers.

As explained before, due to the MAC inaccessibility, we reuse the existent flow control frames (i.e. Pause frames) to generate the syncPDUs. When the TSU detects that an outbound frame is a Pause, it replaces on-the-fly 5 fields (17 octets) (see Fig. 2).

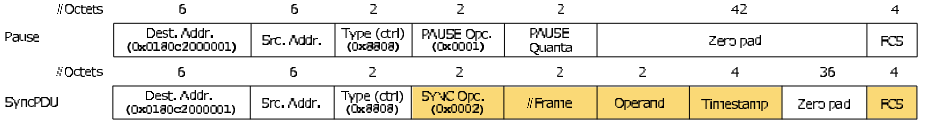


Fig. 2. Pause control frame (top). Synchronization protocol data unit (syncPDU) (bottom).

3.3 SyncPDU Timing

We aim at designing a timing model to measure with high resolution the processing and propagation times of a syncPDU from one node to the other for the purpose of characterizing the delay and jitter of the timestamping mechanism. In Fig. 3 it is shown the timing of the synchronization flow ('· -' line) at, and below the MAC level, which starts and finishes at the master side. The time freeze points of the synchronization flow are located at different strategic and symmetric physical points along the path. For a better follow up, we have assigned logical time variables to the HW registers with the purpose of storing intermediate transmission ($t_B(t_i)$-`tx_req`, $t_C(t_J)$-`st_tx`, $t_D(t_K)$-`txed_ts`, $t_E(t_L)$-`end_tx`) and reception ($t_M(t_F)$-`st_rx`, $t_N(t_G)$-`toa`, $t_O(t_H)$-`end_rx`) times.

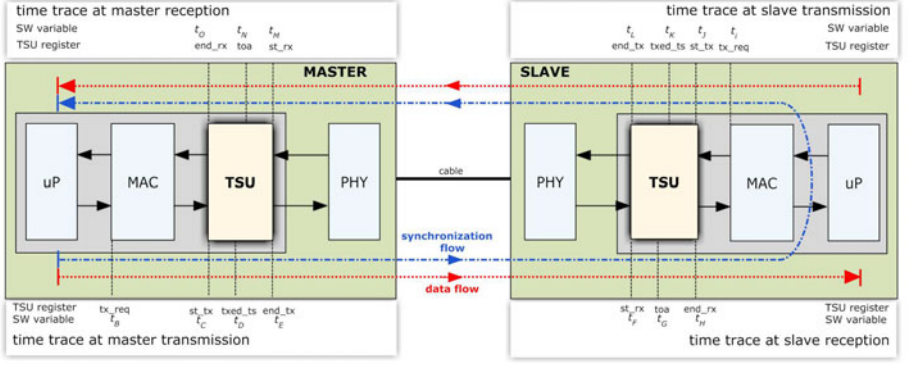


Fig. 3. Timing reference model for timestamping characterization

3.4 Synchronization Mechanism

As mentioned before we extrapolate and enhance EPON synchronization approach to synchronize two nodes in a direct link configuration. The goal using this synchronization mechanism is to measure and to adjust the clock offset between each pair of clocks in order to synchronize them perfectly. In Fig. 4 it is shown the timeline of the synchronization protocol operation at different block levels in each node. The inner columns show the instantaneous clock offset between the master and the slave and viceversa. The mid and outer columns denote which HW registers and logical variables are being written during the process.

The whole process is divided into two recursive phases, the *normal* operation phase and the *discovery* phase. In the former, the master is sending *GATE* type messages which are replied by the slave under *REPORT* type messages. At the normal phase the master is checking the progressive clock drift of the slave. If it detects a specific maximum clock drift, it enters into the discovery mode. If the clock drift keeps lower than a maximum threshold value, the master enters into *discovery* mode after a fixed and periodic interval of time (~ 14.3 sec).

The re-synchronization is performed in the *discovery* phase. As seen on the left-side in Fig. 4, the master calculates the Round Trip Time (RTT) upon the last received *REG_REQ* message according to eq. 1. This phase finishes after the master communicates the propagation delay (RTT/2) to the slave in a *REGISTER* message. The propagation delay will be used by the slave in the next normal operation phase to achieve the link-synchronization.

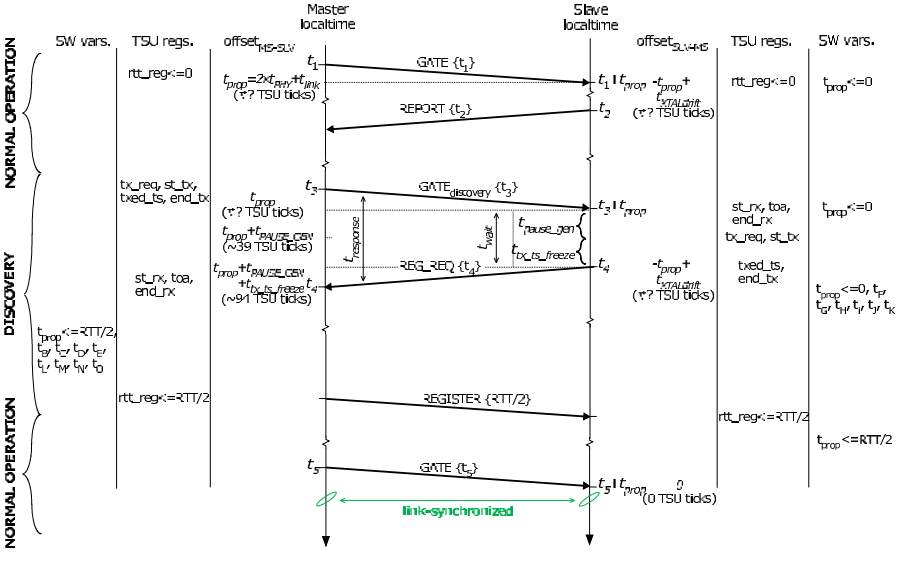


Fig. 4. Synchronization procedure adopted from EPON [3]

$$
\begin{aligned}
RTT &= t_{response} - t_{wait} \\
&= (t_N - t_D) - (t_{pause\_gen} + t_{tx\_ts\_freeze}) \\
&= (t_N - t_D) - (t_J - t_i) - (t_K - t_J) \\
&= (t_N - t_D) - (t_C - t_B) - (t_D - t_C) \qquad (1)
\end{aligned}
$$

4 Evaluation

4.1 Timestamp Variability

To characterize the timestamp variability of the timing model of Fig. 3, we observe the relative time differences between the time freeze points. In table 1 it is shown the statistics of the timing of the synchronization flow over 50k packet and under different load scenarios. The theoretical values of each time difference can be expressed in number of TSU clock ticks taking into account the CAU and the PHY interface frequencies, i.e. $\frac{f_{cau\_clk}[MHz]}{f_{tx/rx\_clk}[MHz]} = \frac{269.96}{125} = 2.15968$. For instance, a timestamp that is triggered at the 25^{th} byte of an outbound syncPDU is equivalent to $t_{ts\_tx\_freeze} \times 2.15968 \approx 54$ TSU clock ticks. The small differences between the experimental and the theoretical values observed

Table 1. Histogram of synchronization flow timing (Fig. 3) over 50k packet run test (expressed in number of TSU clock ticks (percentage over 50k samples))

		transmission			reception	
		t_{pause_gen} (t_C-t_B)	$t_{ts\_tx\_freeze}$ (t_D-$t_C \approx 54$)	t_{tx_rem} (t_E-$t_D \approx 104$)	$t_{ts\_rx\_freeze}$ (t_N-$t_M \approx 54$)	t_{rx_rem} (t_O-$t_N \approx 104$)
Master	no load	36 (16.1%)	57 (51.3%)	113 (100%)	56 (85.0%)	110 (10.2%)
		37 (10.6%)	58 (49.7%)		57 (15.0%)	112 (45.3%)
		38 (49.7%)				113 (44.5%)
		39 (25.6%)				
	w/64B data	37 (40.2%)	57 (55.3%)	109 (18.6%)	54 (30.8%)	112 (38.2%)
		38 (39.1%)	58 (45.7%)	113 (81.4%)	56 (69.2%)	113 (61.8%)
		39 (20.7%)				
	w/1500B data	36 (17.7%)	57 (55.5%)	109 (10.1%)	55 (23.5%)	112 (40.5%)
		37 (25.9%)	58 (44.5%)	112 (20.7%)	56 (76.5%)	113 (59.5%)
		38 (56.4%)		113 (69.2%)		
		t_{pause_gen} (t_J-t_i)	$t_{ts\_tx\_freeze}$ (t_K-$t_J \approx 54$)	t_{tx_rem} (t_L-$t_K \approx 104$)	$t_{ts\_rx\_freeze}$ (t_G-$t_F \approx 54$)	t_{rx_rem} (t_H-$t_G \approx 104$)
Slave	no load	36 (19.9%)	57 (54.5%)	113 (100%)	55 (78.4%)	112 (19.6%)
		37 (32.3%)	58 (45.5%)		56 (19.6%)	113 (75.4%)
		38 (47.8%)				
	w/64B data	36 (17.1%)	57 (55.3%)	109 (18.6%)	55 (80.4%)	112 (21.1%)
		37 (40.2%)	58 (45.7%)	113 (81.4%)	56 (19.6%)	113 (78.9%)
		38 (23.8%)				
		39 (18.9%)				
	w/1500B data	36 (23.4%)	57 (52.1%)	112 (9.8%)	55 (28.2%)	112 (30.3%)
		37 (35.8%)	58 (47.9%)	113 (90.2%)	56 (71.8%)	113 (69.7%)
		38 (40.8%)				

in the table stem from the additional clock ticks needed by the handshakers to perform reliable timestamp readings. We are interested on the timestamp variability as we will use them in eq. 1. The experimental data is well discretized, precise and confined, thus proving the reliability of the timestamping mechanism.

4.2 Link Synchronization Accuracy: Phase and Time

We have used two methods for the evaluation of the synchronization accuracy and precision between the two nodes at link level. The first method consists on comparing the phase error between TSUs just after a *localtime* adjustment. To observe this, we configure the comparator-trigger blocks inside the TSUs to raise a 50ns-wide pulse at periodic intervals of 225μs through *MATCH* output pin (see right-side of the Fig. 5). On the left side of the Fig. 5 it is shown a zoom in the time adjustment for the best and worst case synchronization accuracy. To obtain this screenshot, we have forced the slave node to re-synchronize every 5 milliseconds to better observe the variability of a re-synchronization event. Over the ∼200 overlapped signal traces, we have achieved best-case link synchronization accuracies of 0 ns between TSU's *localtime* counters, and a worst-case synchronization accuracy of 130 ns.

The second method consists on comparing the instantaneous *localtime* values at each node during the *normal operation* phase of the synchronization procedure (see Fig. 4) under three different load scenarios: no background data (no bg data), with minimum-length data packets (w/64B pckts) and maximum-length data packets (w/1500B pckts).

The plots in Figs. 6a,b,c refer to the synchronization accuracy at t_N and t_K time freeze points. Fig. 6a zooms in a 150 sec window to show the 'sawtooth' shape of the synchronization trace, which reveals the clock drift of the slave relative to the master (∼3.26μs/s). The sawtooth is not uniform, which means that the re-synchronization events are not executed exactly every 5 sec. This behavior owes to the fact that the μP handles multiple interruption requests from different HW blocks inside the platform and serve them undeterministically. Fig. 6b shows that the achieved synchronization accuracy is mostly confined within ±10 TSU clock ticks. The histograms in Fig. 6c better show the clock offset distribution in such region. For the three load scenarios, mean (μ) and standard deviation (σ) keep below 1 and 10 ns, respectively. The different σ values might be caused by the extra clock ticks needed by the HW mechanisms inside the MAC and the PHY to generate and process a packet.

The plots in Figs. 6d,e,f are equivalent to those of the first row but choosing different time freeze points (t_D and t_G). We observe several differences in the plot 6e compared with its top counterpart. Here the clock offset is bounded in the [20, 30] TSU ticks region, denoting an asymmetry when compared to t_N and t_K. The 20 cycles score is due to the round off error committed by the program code of the master when calculating the RTT. The bulk of the data in Fig. 6e and the zone with higher density of traces on the left plot of the Fig. 5 are equivalent

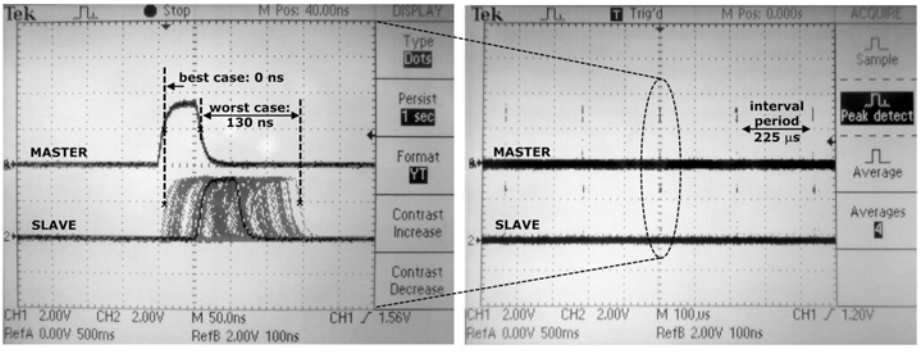


Fig. 5. Phase error of the synchronized timing signal (right). Re-synchronization variability (left).

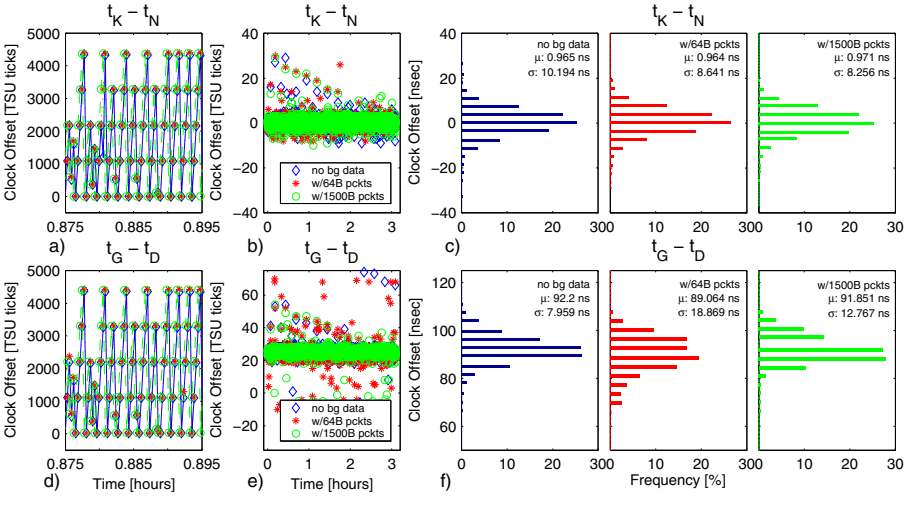


Fig. 6. a,d) Accumulated clock offset. b,e) Synchronization accuracy. c,f) Synchronization accuracy distribution.

([74, 111] ns). μ and σ statistics of histograms in Fig. 6f are well confined over 90 and 12 ns, respectively.

5 Conclusion

In this work we have presented an FPGA-based platform for developing and testing new Ethernet functionalities. We envision Ethernet as a technology capable to offer carrier-class QoS while maintaining its simplicity and low-cost if time synchronization is addressed. For this reason, we have implemented within the programmable resources of the FPGA an improved version of our custom hardware Timestamping Unit with several key functionalities: high-resolution and

error-free on-the-fly timestamping, high-resolution timing of inter-MAC events and frame propagation times. Leveraging these features, we have characterized the timestamping mechanism in terms of time variability and re-used the accurate timings in the EPON synchronization exchange mechanism to achieve best-case synchronization accuracies of zero nanoseconds at MAC level.

Acknowledgements

The author gratefully acknowledges key ideas on timing flow from Prof. Dolors Sala, implementation-related suggestions from Prof. Enrique Cantó and profitable comments from Prof. Johan Zuidweg.

References

1. Steinmetz, R.: Human Perception of Jitter and Media Synchronization. IEEE Journal on Selected Areas in Communications 14(1), 61–72 (1996)
2. Arlos, P.: On the Quality of Computer Network Measurements, Ph.D. Thesis, Blekinge Institute of Technology (2005)
3. IEEE Standard for Local and Metropolitan Area Carrier Sense Multiple Access with Collision Detection Access Method and Physical Layer Specifications, IEEE Std 802.3-2005 (2005)
4. Kopetz, H., Ochsenreiter, W.: Clock Synchronization in Distributed Real-Time Systems. IEEE Transactions of Computers C-36(8), 933–939 (1987)
5. Nicolau, C., Sala, D., Cantó, E.: Clock Duplicity for High-Precision Timestamping in Gigabit Ethernet. In: 19th Int. Conf. on Field Programmable Logic and Applications (FPL), Czech Republic, August 31–September 2 (2009)
6. IEEE 1588 Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems, IEC 61588:2009(E), pp. C1–274 (2009)
7. Mills, D.L.: Network Time Protocol (Version 3) Specification, Implementation and Analysis, RFC-1305 (March 1992)
8. Kutschera, C., et al.: Background IEEE 1588 Clock Synchronization over IEEE 802.3/Ethernet. In: IEEE Symp. on Precision Clock Synchronization (ISPCS 2008), September 22-26 , pp. 137–141 (2008)
9. Finn, N., Cisco Systems Inc.: Sync PDUs and the MAC Stack - Help needed from 802.3 to properly process IEEE 1588/P802.1AS sync PDUs. In: IEEE 802.3/IEEE 802.1 joint session, Denver, CO, USA (July 2008)
10. Xilinx, Inc.: ML40x Evaluation Platform User Guide, ug080 v2.5 (2006)
11. Stephenson, J., Altera Corp.: Don't Let Metastability Cause Problems in Your FPGA-Based Design (2009), http://www.pldesignline.com/220300400
12. Weinstein, R.: The "Flancter" (App. Note), Memec Design(2000), http://www.floobydust.com/flancter/Flancter_App_Note.Pdf

The DORII Project e-Infrastructure: Deployment, Applications, and Measurements

Davide Adami[1], Alexey Chepstov[2], Franco Davoli[1], Bastian Koller[2], Matteo Lanati[3], Ioannis Liabotis[4], Stefano Vignola[1], Anastasios Zafeiropoulos[4], and Sandro Zappatore[1]

[1] CNIT, University of Pisa/University of Genoa Research Units, Italy
[2] High Performance Computing Center Stuttgart (HLRS), University of Stuttgart, Germany
[3] EUCENTRE Pavia, Italy
[4] GRNET, Athens, Greece

Abstract. Remote Instrumentation Services go far beyond offering networked access to remote instrument resources. They are establishing as a way of fully integrating instruments (including laboratory equipment, large-scale experimental facilities, and sensor networks) in a Service Oriented Architecture, where users can view and operate them in the same fashion with computing and storage resources. The deployment of test beds for a large basis of scientific instrumentation and e-Science applications is mandatory to develop new functionalities to be embedded in the existing middleware to enable such integration, to test them on the field, and to promote their usage in scientific communities. The DORII (Deployment of Remote Instrumentation Infrastructure) project is a major effort in this direction. The paper presents the performance monitoring infrastructure that has been built in DORII and the results concerning a selected application in seismic engineering.

Keywords: Remote Instrumentation Services, SOA, e-Science.

1 Introduction

Almost all scientific areas and a good deal of technological developments use specialized instrumentation – laboratory equipment, measurement devices, large- and small-scale experimental facilities, sensor networks for data acquisition – other than computational and storage resources. The complex of activities that allow automated data processing and analysis, by exploiting distributed computational service like those offered by Grid architectures and cloud computing, and that can be referred to as *e-Science* (or, with a more precise term, Service-Oriented Science [1]) would greatly benefit from the full integration of such experimental instrumentation with the computational infrastructure into one powerful pool of resources that can be searched, selected, composed and configured, accessed, and controlled by their users.

The extension of the e-Infrastructure with this complex of activities, which can be termed *Remote Instrumentation Services (RIS)* [2], is not straightforward, and it requires addressing a number of issues in middleware and network architectural design, middleware development, and instrumentation and measurement related aspects. Recently, a number of European research projects, among others, have been

T. Magedanz et al. (Eds.): TridentCom 2010, LNICST 46, pp. 285–295, 2011.

dedicated to it, and a community of researchers in the field has been forming and actively investigating aspects in this field (see, e.g., [3], [4]).

At the same time, efforts have been dedicated to the deployment of the infrastructure and of test beds that allow user communities to become acquainted with the related technology, to perform experiments on-line, and to be involved in the development of new applications and in the extension of existing ones. This is one of the main objectives of the DORII (Deployment Of Remote Instrumentation Infrastructure) project [5], [6], funded by the European Commission in the 7th Framework Program.

The general architecture, the deployed applications and the test bed organization of DORII were described in [6]. In the present paper, we describe the currently deployed DORII e-Infrastructure, the performance monitoring tools' customization and deployment, and present the results of a selected application in earthquake engineering. The paper is organized as follows. Sections 2 and 3 describe the e-Infrastructure and the monitoring tools, respectively. Section 4 presents the deployment of the selected application, and Section 5 reports related experimental results. Section 6 contains the conclusions.

2 Deployment of the e-Infrastructure

One of the main requirements posed by applications of many strategic areas in science and technology (as the ones specified by ESFRI - European Strategy Forum on Research Infrastructure [7]) is to design a service-oriented IT architecture which should allow users manage, maintain and exploit diverse instrumentation and acquisition devices together with heterogeneous computation and storage facilities granted by the traditional Grid, as those set up by EGEE (Enabling Grids for E-sciencE) [8], DEISA (Distributed European Infrastructure for Supercomputing Applications) [9] and many other Grid projects. Unlike the traditional Grid, the e-Infrastructure should practically enable access to remote instrumentation in high-performance computing and storage environments, and allow users and their applications to get an easy and secure access to various remote instrumentation resources, supported by high-performance Grid computation and storage facilities. The e-Infrastructure does that by providing standardized services to access integrated instrumentation resources (including expensive experimental equipment, but also smaller network-connected sensors and mobile devices), in a unified way with the traditional Grid services (as provided, e.g., by gLite [10]).

The DORII e-Infrastructure is based both on the EGEE infrastructure and its middleware of choice gLite, and on specific middleware services built within the DORII project. The interaction of the users with the instruments is effected via the Instrument Element (IE), originally conceived in the GRIDCC [11] project, and then redesigned within DORII. The IE includes specific Instrument Managers (IMs). Information about the resources and services of the infrastructure is provided by the Berkeley Database Information Index (BDII). The Workload Management System (WMS) is the service responsible for the distribution and management of tasks across Grid resources, in such a way that applications are conveniently, efficiently and

effectively executed. The LCG Computing Element (LCG-CE) is responsible for submitting jobs to the underlying local cluster of Worker Nodes (WNs). Storage Elements (SEs) are responsible for data storage and management, while the LCG File catalogue (LFC) offers a hierarchical view of files to users, with a UNIX-like client interface. From the security perspective, the Virtual Organization Management Service (VOMS) is a full-fledged Attribute Authority, whose job is to assign attributes like group membership and role ownership to members of a Virtual Organization (VO).

At the time of writing of this paper the DORII e-Infrastructure consists of 9 sites offering computational and storage resources distributed among the partners of the project. Table 1 shows the sites that support the catch all DORII VO, where most of the DORII applications have been deployed:

Table 1. vo.dorii.eu Computational and Storage Resources

Country	Partner Name	Site Name	CPU Cores	Storage (TB)
Poland	PSNC	PSNC	1068	16
Spain	CSIC	IFCA-CSIC	372	107
		IFCA-I2G*	372*	107*
Italy	ELETTRA	ELETTRA	80	0.1
Greece	GRNET	HG-01-GRNET	64	4.78
		HG-02-IASA	118	3.14
		HG-03-AUTH	120	3.13
		HG-04-CTI-CEID	114	2.87
		HG-05-FORTH	120	2.33
		HG-06-EKT	228	7.76
		Totals	2284	147.11

Three scientific communities have deployed their instruments and the corresponding applications using them in the DORII infrastructure: i) environmental observation and monitoring; ii) earthquake engineering; iii) experimental science.

The e-Infrastructure addressed by DORII encompasses the following main environments:

- *Application environment*

This layer comprises a diverse set of applications, from those providing Remote Instrumentation Services to parallel applications performing simulation and modelling on the acquired data.

- *Middleware framework*

The framework consolidates traditional middleware solutions for Grid, enhancing and adapting to the requirements of the remote instrumentation. In the context of DORII, the middleware framework is comprised of the tools that are best practices for remote instrumentation (like the Instrument Element – IE [12]) and management of Grid resources (the Virtual Control Room – VCR [13]), application development

(g- Eclipse [14]) and workflow management (VLab [15]), interactivity and visualization (GLogin and GVid [16]), as well as parallel application support (Open MPI and PACX-MPI [17]).

- *Fabric layer*

This layer, comprising network-connected Grid resources, is extended by scientific instrumentation and remotely controlled devices.

- *Networking technologies*

This layer includes the networking solutions for connecting all the upper layers of the Grid architecture. TCP/IP is the common networking technology, along with specific access network protocols.

3 Monitoring the Network and the Infrastructure

The network monitoring infrastructure deployed for the DORII project consists of the following tools:

- Smokeping, for network latency measurement;
- Pathload, for the estimation of the available bandwidth along a network path;
- SNMP-based Web applications, for monitoring network interface utilization.

3.1 Smokeping

Smokeping [18] is a software tool that can be used to measure the network latency. More specifically, a Smokeping probe sends test packets out to the network and measures the amount of time they need to travel to a target host node and back. The RRDtool [19] is used to maintain a long-term data-store with latency measurement, and the presentation of the data on the web is done by means of a CGI with some AJAX capabilities for interactive graph exploration.

In the framework of the DORII project, Smokeping is used in master/slave mode: this way, Smokeping probes (slaves) are allowed to run remotely and to perform latency measurements from multiple locations to the target hosts.

As shown in Fig. 1, Smokeping has been deployed as follows:

- the Smokeping master is located at CNIT [20]; it maintains a configuration file with a specific section for each slave, and it stores and presents all monitoring data collected by the slaves.
- Remote probes (Smokeping slaves) have been installed at DORII partners' sites EUCENTRE, ELETTRA, GRNET, CSIC-IFCA, OGS and PSNC. Based on settings contained in the configuration file retrieved from the master (e.g., measurement utility, target host address, measurement length and period, etc.), each slave performs latency measurements and sends back the results to the Master server by using the HTTP protocol. In the DORII network infrastructure, the following targets have been identified: Computing Elements (CEs), Storage Elements (SEs), Instrument Elements (IEs), remote sites' Access Gateways (AGs).

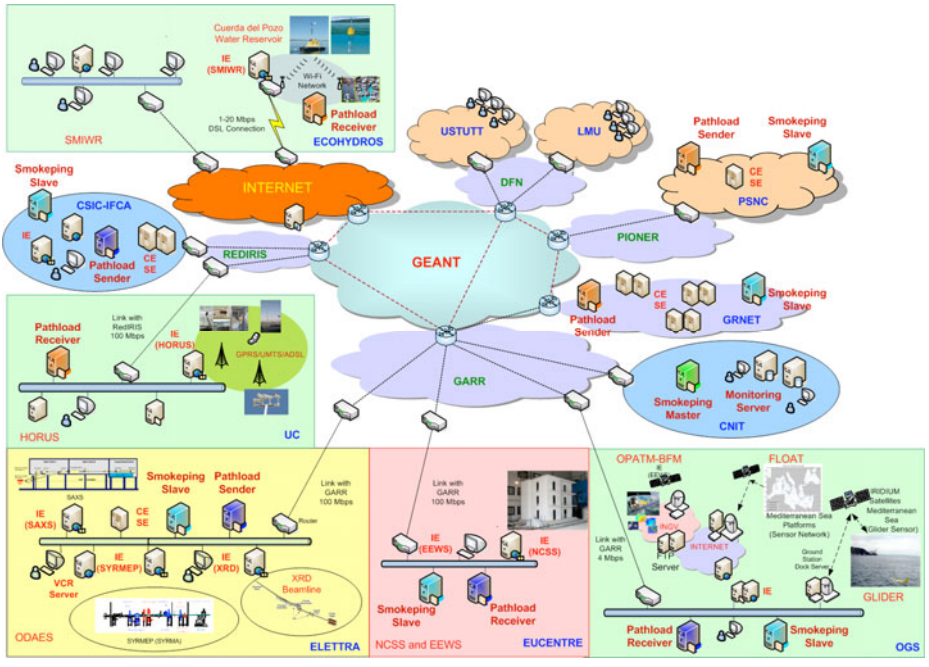


Fig. 1. Smokeping and Pathload deployment for the DORII project infrastructure

3.2 Pathload

Pathload [21] is a monitoring tool that estimates the available bandwidth of a network path. The basic idea behind Pathload is that the one-way delays of a periodic packet stream show an increasing trend when the stream rate is larger than the available bandwidth.

Pathload is based on a client-server architecture and consists of two main components:

- *pathload_snd* that listens on TCP port 55002 and acts as a traffic generator;
- *pathload_rcv* that starts a Pathload session and acts as a traffic receiver.

Pathload has been customized for the DORII project. In addition to the previous components, some scripts have been introduced to monitor the status of the sender and receiver processes and to automatically export the measurement data collected by the receiver to the management station located at CNIT via HTTP. The tool has been installed as follows (see Fig. 1):

- Pathload_sender: at each site where CEs and/or SEs of the DORII e-Infrastrcture are located (GRNET, PSNC, CSIC-IFCA);
- Pathload_receiver: at each site where IEs are deployed (EUCENTRE, OGS, ELETTRA, UC, etc.) and, therefore, DORII applications are running.

This way, the bandwidth available from the sites hosting CEs and SEs to the sites with applications (IEs and VCR) can be estimated.

3.3 SNMP-Based Network Monitoring

Various applications exist to collect and consolidate network usage information. At a basic level, such applications (also called managers) use the Simple Network Management Protocol (SNMP) to read statistics from each monitored device (router or host) where an SNMP agent is configured and running. A standard Management Information Base (MIB) collects counters of the number of datagrams and bytes sent and received on each interface of a device, and it also gives the number of packets discarded because of congestion. An SNMP application can periodically poll each device and convert the returned information into a view of usage across the whole network. SNMP can also help identify network interface failures or outage conditions. In the framework of the DORII, SNMP is required to be enabled on IEs, CEs, SEs and routers. Data are collected by an SNMP manager and interfaced with a Web server by using ad-hoc CGI programs.

3.4 Nagios

Nagios [22] is an open source monitoring system providing comprehensive and scalable monitoring of all mission-critical infrastructure components, including applications, services, operating system and system metrics or network protocols and infrastructure. Nagios is integrated in the monitoring framework of the DORII e-Infrastructure providing information on problems and failures related to the computational, storage and instrument resources of this infrastructure, and monitoring services such as the CE, SE, BDII, WMS, and IE.

4 Deployment of a Selected Application

4.1 The Application

The EEWS (Earthquake Early Warning System) aims at recording seismic data from sensors, possibly in real time, and at processing them in order to extract time history for ground velocity, ground acceleration and displacements. This is the starting point to calculate some interesting parameters widely adopted in the seismic community, such as the acceleration Fourier amplitude spectrum and the acceleration response spectrum (useful to evaluate a building's response to the force imposed by the earthquake). The rationale of this application is to provide scientists a unified environment mixing access to instruments and computational tools, speeding up the analysis carried out after a seismic event.

All the operations are performed remotely and on the grid, employing the DORII infrastructure. The VCR plays the role of the user interface: all the actions are carried out and all the resources are accessed through this web portal.

The IE is located at EUCENTRE (Pavia, Italy) and it hosts the IM devoted to access the server collecting data from sensors. This node is in Genoa, Italy, while the seismic sensors are spread over the Liguria Region. Measurements from each channel are time-stamped and saved locally on the IE in a separate file, then moved to a SE. Finally, a CE retrieves each file from the SE and performs the computation. Since the same computation is repeated for each input file, the job is parametric,

where the parameters are the file names. The JDL (Job Description Language) file characterizing the job is created using a VCR application, which is a Jython script that customizes a given template. The user is asked only to select the input folder on the right SE. The output is downloaded to the home folder on the VCR. An alternative approach is represented by the Workflow Manager, a graphical and friendly interface to specify the parameters.

4.2 The Instruments

A set of seismic sensors is connected to a central point over the UDP protocol, by means of wired or wireless links. Each device measures the ground velocity along the three Cartesian directions, so each station broadcasts at least three channels, plus state of health information. All the data are gathered by a central server, which manages replicated and out of order packets. The reconstructed stream is stored and made available as DAT service: the user can access the historical data series, specifying the starting point of information flow and the time window of interest. On the contrary, if a user or an application is focused on near-real-time access, the NAQ service is more suitable. Given that some parameters are tuned properly, it is possible to configure the service to forward the original packets from instruments to the application, minimising the delay. In fact this is the meaning of near-real-time acquisition. Seismic sensors send measurements to the central server as soon as possible, but the server has to store them, organizing in data structures called “bundles” and “packets”. These operations take some time, however limited to few seconds, depending on the distance between the server and the client.

Packets are uniquely identified by a sequence number and a time stamp; they include an odd number, from 1 to 255, of 17-byte bundles. This solution allows adapting the packet size to the network. Moreover, it is worth noting that data contained in a packet are homogeneous, for example the measurements of a particular channel or the status information of a station. The first bundle in a packet always acts as header, specifying some useful details such as station’s unique ID, time stamp, sampling frequency and sequence number. The following groups of bytes carry only data. State of health messages are obviously strings, while measurements are integer values, in compressed or uncompressed format.

4.3 The Instrument Manager

The information flow is provided to the IM by means of a TCP connection as a time series or a transparent serial stream. A transparent serial stream handles only uncompressed format and all packets have the same length, employing padding where necessary. On the contrary, a time series stream can deal with compressed data, too, and the packet size may vary. Both streams exist also in the buffered version, so it is possible to retrieve additional packets prior to the beginning of subscription, moving the starting point slightly to the past. Finally, one of the most important parameters is the Short Term Completion (STC) time. It represents the time interval, from 0 to 300 seconds, the server waits to fill the gaps in the stream in case of retransmitted packets. Our goal consists in reading seismic data in near-real-time, so we chose to subscribe a time series compressed stream (as it is more efficient) in its unbuffered version,

disabling STC. This means that measurements are not guaranteed to be in order because of errors and losses, so the gaps are filled using interpolation and delayed packets are discarded. The approach guarantees a continuous flow, fundamental feature for the subsequent computation, complying with the strict time constraint. All the operations are performed by a Java client library developed for the VCR architecture on the basis of the user's manual provided by the server manufacturer.

5 Experimental Results

The user task list previously described represents the starting point for the test bed set-up employed in this work. Moreover, network performance over the grid infrastructure is monitored during the entire life cycle of the application execution. In our experiments, we skipped the acquisition phase, since the server gathering sensors' data is not part of the infrastructure, unlike the IE sending the query. Moreover, the required network resources are very limited: as a matter of fact, a single station channel only needs few kbps. The size of the file containing the initial data set for the computation is about 115 MB and corresponds to measurement data coming from two channels and collected for a whole day. The archive is stored on a server at EUCENTRE that acts as IE and VCR; then, it is transferred to a SE at GRNET (se01.isabella.grnet.gr). The average throughput is about 7.5 Mb/s. The analysis is carried out by a parametric job, where the parameters correspond to different settings for the two seismic stations being monitored, so a single execution of the application produces two children nodes. The job is launched three times, and the target CEs are located at different sites: GRNET (ce01.athena.hellasgrid.gr), PSNC (ce.reef.man.poznan.pl) and IFCA-CSIC (egeece01.ifca.es).

Table 2. Upload time from se01.isabella.grnet.gr to each CE and processing time

CE	Upload Start Time	Upload Time [s]	Throughput [Mb/s]	Average Latency [ms]	Computation Start Time	Processing Time [s]
ce01.athena.hellasgrid.gr	27/11/2009, 12.59	2.0	478.4	-	27/11/2009, 12.59	26
ce01.athena.hellasgrid.gr	27/11/2009, 12.59	2.1	460.2	-	27/11/2009, 13.18	65
ce.reef.man.poznan.pl	27/11/2009, 13.07	351	2.6	59.2	27/11/2009, 13.07	16
ce.reef.man.poznan.pl	27/11/2009, 13.04	241	3.9	59.2	27/11/2009, 13.04	47
egeece01.ifca.es	27/11/2009, 13.46	95	9.8	94.3	27/11/2009, 13.48	22
egeece01.ifca.es	27/11/2009, 13.46	82	11.7	94.3	27/11/2009, 13.48	45

Table 2 reports the time necessary to upload the file from the SE to each CE, along with the processing time. The last column of the table reports the latency measured by using Smokeping. If the user chooses a local CE (ce01.athena.hellasgrid.gr) the latency is in the order of a few ms, and therefore it is not reported in the table, but

also in the other two cases the latency is very low, since it is less than 100 ms.

The job output consists of files whose size is comparable with the size of the input file. Finally, the output files are retrieved, by using the VCR at EUCENTRE. Table 3 reports the time necessary to download the output files stored by each CE from EUCENTRE.

Table 3. Download Time from each CE to EUCENTRE VCR

CE	Getting Output Start Time	Download Time [s]	Throughput [Mb/s]	Available Bandwidth [Mb/s]	Average Latency [ms]
ce01.athena.hellasgrid.gr	27/11/2009, 13.18	36	26.8	92	49.2
ce01.athena.hellasgrid.gr	27/11/2009, 13.19	32	30.2	92	49.2
.2ce.reef.man.poznan.pl	27/11/2009, 13.22	33	29.3	102	29.9
ce.reef.man.poznan.pl	27/11/2009, 13.23	89	10.9	102	29.9
egeece01.ifca.es	27/11/2009, 14.16	32	30.2	96	46.1
egeece01.ifca.es	27/11/2009, 14.17	32	30.2	96	46.1

Table 4. Total Time

CE	Total Time [s]	Elaboration Time [s]	Communication Time [s]
ce01.athena.hellasgrid.gr	64	26	38
ce01.athena.hellasgrid.gr	99	65	34
ce.reef.man.poznan.pl	400	16	384
ce.reef.man.poznan.pl	377	47	330
egeece01.ifca.es	149	22	127
egeece01.ifca.es	159	45	114

The last two columns contain the available bandwidth (estimated by Pathload) and the average latency (measured by Smokeping) from each CE to EUCENTRE VCR. It is relevant to highlight that the throughput is significantly less than the available bandwidth: this means that the communication protocols are not efficient enough to utilize the available bandwidth of the communication channel.

Finally, Table 4 reports the overall time for the execution of the application. As clearly shown in the table, the amount of time necessary for exchanging the data may significantly affect the performance of the application and represents (except in the second case) the major component of the overall execution time.

6 Conclusions

The DORII project has created and is managing a Virtual Organization for the deployment of applications in a number of different scientific fields. Applications that were traditionally executed locally have been ported to a Grid environment, where they can find and ask for computational and storage resources, and carry out the data processing operations they require efficiently and timely. More importantly, DORII middleware enables scientists to expose and access their instrumental resources on the Grid, by means of universal abstractions that apply to diverse e-Science domains. At

the same time, a ubiquitous monitoring service has been deployed, allowing the continuous evaluation of the distributed system performance and the discovery of possible problems and bottlenecks.

The architecture and the main features of the deployed test bed have been presented in the paper, together with an example in performance monitoring of a selected application. Scientists in the different disciplines involved are actively participating in DORII applications' deployment and experimental activity, with the goal of evaluating the new middleware functionalities and suggesting improvements. Future work will be aimed towards the development of additional functionalities and to providing input to standardization activities in Remote Instrumentation Services.

Acknowledgments

This work was supported by the European Commission under the DORII project (contract no. 213110).

References

1. Foster, I.: Service-oriented science. Science Mag. 308(5723), 814–817 (2005)
2. The RINGrid Consortium: Whitepaper on Remote Instrumentation. Computational Methods in Science and Technol. 15(1), 119–138 (2009)
3. Davoli, F., Meyer, N., Pugliese, R., Zappatore, S. (eds.): Grid-Enabled Remote Instrumentation. Springer, New York (2008)
4. Davoli, F., Meyer, N., Pugliese, R., Zappatore, S. (eds.): Remote Instrumentation and Virtual Laboratories. Springer, New York (to apppear, 2010)
5. DORII Project Home Page, http://www.dorii.eu
6. Adami, D., Cheptsov, A., Davoli, F., Liabotis, I., Pugliese, R., Zafeiropoulos, A.: The DORII project test bed: Distributed eScience applications at work. In: Proc. 1st Internat. Workshop on Pervasive Computing Systems and Infrastructures (PCSI), Washington, DC (April 2009)
7. ESFRI Home Page, http://cordis.europa.eu/esfri/
8. EGEE Project Home Page, http://www.eu-egee.org/
9. DEISA project Home Page, http://www.deisa.eu/
10. http://glite.web.cern.ch/glite/
11. GRIDCC Project Home Page, http://www.gridcc.org/
12. Frizziero, E., Gulmini, M., Lelli, F., Maron, G., Oh, A., Orlando, S., Petrucci, A., Squizzato, S., Traldi, S.: Instrument Element: a new Grid component that enables the control of remote instrumentation. In: Proc. 6th IEEE Internat. Symp. on Cluster Computing and the Grid Workshops (CCGRIDW 2006), Singapore (May 2006)
13. Ranon, R., De Marco, L., Senerchia, A., Gabrielli, S., Chittaro, L., Pugliese, R., Del Cano, L., Asnicar, F., Prica, M.: A web-based tool for collaborative access to scientific instruments in cyberinfrastructures. In: Davoli, F., Meyer, N., Pugliese, R., Zappatore, S. (eds.) Grid Enabled Remote Instrumentation, pp. 237–251. Springer, New York (2008)
14. Kornmayer, H., Stümpert, M., Knauer, M., Wolniewicz, P.: g-Eclipse – an integrated workbench tool for Grid application users, Grid operators and Grid application developers. In: Proc. Cracow Grid Workshop 2006, Cracow, Poland (October 2006)

15. Okoń, M., Kaliszan, D., Lawenda, M., Stokłosa, D., Rajtar, T., Meyer, N., Stroiński, M.: Virtual Laboratory as a remote and interactive access to the scientific instrumentation embedded in Grid environment. In: Proc. 2nd IEEE Internat. Conf. on e-Science and Grid Computing (e-Science 2006), Amsterdam, The Netherlands (December 2006)
16. Köckerbauer, T., Polak, M., Stütz, T., Uhl, A.: GVid - Video Coding and Encryption for Advanced Grid Visualization. In: Proc. 1st Austrian Grid Symp., Schloß Hagenberg, Austria (December 2005)
17. Keller, R., Liebing, M.: Using PACX-MPI in MetaComputing applications. In: Proc. ASIM 2005 - 18th Symp. on Simulation Technique (Symposium Simulationstechnik), Erlangen, Germany (September 2005)
18. Smokeping Home Page, `http://oss.oetiker.ch/smokeping/`
19. RRD Tool Home Page, `http://oss.oetiker.ch/rrdtool/index.en.html`
20. DORII Monitoring Platform Home Page, `http://monitor2.cnit.it`
21. Pathload Home Page, `http://www.cc.gatech.edu/fac/Constantinos.Dovrolis/bw-est/pathload.html`
22. NAGIOS Home Page, `http://www.nagios.org`

TridentCom 2010

Full Papers Session 3: Future Wireless Testbeds

Towards Maximizing Wireless Testbed Utilization Using Spectrum Slicing*

Angelos-Christos Anadiotis[1], Apostolos Apostolaras[2], Dimitris Syrivelis[2], Thanasis Korakis[2], Leandros Tassiulas[2], Luis Rodriguez[3], Ivan Seskar[4], and Maximilian Ott[5]

[1] School of Electrical and Computer Engineering
National Technical University of Athens
Athens, Greece
[2] Department of Computer and Communication Engineering
University of Thessaly
Volos, Greece
[3] Atheros Communications
[4] WINLAB, Rutgers University
Technology Center of New Jersey, USA
[5] National ICT Australia(NICTA)
Alexandria, NSW 1435, Australia

Abstract. As experimentation becomes one of the de-facto approaches for benchmarking, researchers are turning to testbeds to test, review and verify their work. As a result, several research laboratories build wireless testbeds, in order to offer their researchers a real environment to test their algorithms. As testbeds become more and more popular, the need for a managerial tool that will not only provide a unified way for defining and executing an experiment and collecting experimental results, but that will also serve as many users as possible maximizing the utilization of its resources, is growing. In this spirit, we propose a scheme that exploits wireless testbeds functionality by introducing *spectrum slicing* of the testbed resources. This scheme can be incorporated inside OMF, an already existing wireless testbeds managerial framework, which is widely used by many researchers.

1 Introduction

The theoretical analysis and the simulation of a new wireless protocol or technique can give us important information about its performance in terms of throughput, delay, power consumption, etc. However, in order to have analytically tractable models, several simplifications of the real world have to be made. While the simulations have the ability to incorporate more general models, we are

* The research leading to these results has received funding from the European Community's Seventh Framework Programme (FP7/2007-2013) under grant agreement n^{o}224263.

T. Magedanz et al. (Eds.): TridentCom 2010, LNICST 46, pp. 299–314, 2011.

still limited by the complexity of the simulation software and our limited knowledge of the wireless environment. Some specific limitations of the simulation approach in depicting a real wireless network include inaccurate respresentation of the wireless medium, simplification of synchronization issues that occur in wireless terminals and ignorance of several aspects such as the computational overhead.

Due to the above limitations, researchers have focused in the last few years on the studying of wireless schemes through implementing them on real platforms. Most of the implementation is done on open source platforms, such as software defined radios or open source wireless drivers. This new trend in wireless networks has triggered the birth and evolution of several wireless testbeds around the globe. Researchers may reserve a testbed for a specified time and execute their experiments there. But, how is that reservation made? Until now, the experimenter reserved the whole testbed (or a very large part of it if we are talking for a really big testbed such as ORBIT) even if he actually needed only a few nodes and frequency channels. This reservation policy prohibits other users from using the testbed at the same time, since the experiments may interfere with each other. Moreover, most of the times the reservation is made after an oral agreement between the potential users.

An answer to these issues would be the dynamic, on-demand partition of the testbed to smaller parts, based on the available resources and the experimenters demands. So, we need to build a managerial mechanism that will be able to both handle multiple requests from the testbed users and partition the testbed efficiently by creating virtual slices and assigning them to the respective users. We intend to build such a mechanism using spectrum slicing techniques.

Currently, one of the most used testbeds is ORBIT [11] in WINLAB [6]. ORBIT consists of 400 nodes, available to the registered users. It has a very well organized management system which allows users to book the testbed at available time slots. Although ORBIT's reservation framework is very useful since it allows a large amount of users to remotely access the testbed, it has a significant drawback: It does not allow for efficient use of the testbed resources. In most of the experiments, only a small amount of nodes are being used, while the rest are staying idle. Usually, a researcher reserves the whole testbed (400 nodes) for a couple of hours and he only uses no more than 10 nodes, leaving the rest 390 nodes idle. With slicing, these nodes could serve the needs of other users.

ORBIT's example shows the need to develop a tool that will maximize the utility of a wireless testbed. In this paper, we are proposing a scheme based on spectrum slicing, which takes advantage of the large availability of a particular resource -that is spectrum- and, through that, increases the whole testbed's availability to experimenters. Of course slicing can refer to other resources too, such as power (adjust the power that each slice will transmit to create a "safe" area for each user), network cards (a node that has many network cards could assign subgroups of them to different experimenters) and nodes (many users could use the same node using virtualization techniques), however in this paper we

focus on spectrum slicing. This scheme is developed as a part of a more generic managerial framework that is being designed in the concept of OneLab2 [4]. OneLab2 intends to federate heterogeneous testbeds located in different places under a unified system. As we are illustrating in later sections, our new managerial mechanism allocates a particular group of channels to a group of nodes that is assigned to one user. In this way, we optimize the resources usage of the testbed by allowing multiple users to operate on the testbed simultaneously, without interfeering with each other.

The challenge in wireless testbeds slicing is the isolation of experiments, as there are inter-dependencies among the resources. In contrast to a wired interface where all we need to do is to manage the sharing of a specified resource on a single node, sharing a wireless interface may also affect the sharing of interfaces on other nodes. What correlates them are things like spectrum, location and power which are also correlated. Power and location for instance, are two factors that could affect each other.

The remainder of the paper is structured as follows: In Section 2 we give related and previous work made in the domain of wireless testbeds resource allocation. In Section 3 we give a short description of the OMF framework that we used for developping our scheme. In Section 4 we present our spectrum slicing scheme from a top-down approach. In Section 5 we present our testbed, an example usage scenario for our scheme and statistics which give us useful feedback on the improvement of testbed utilization. Finally, in Section 6 we give the conclusions that he have reached through our work in this field and in Section 7 our future plans.

2 Related Work

Several work has been made on efficient resource allocation on wireless testbeds. However, most of this work is focused on virtualization techniques, which implies more complex implementation and operating system dependence. Next, we are giving two representative examples of such systems:

Emulab. Emulab is a network testbed, giving researchers a wide range of environments in which to develop, debug, and evaluate their systems. In Emulab, there has been developed a system which virtualizes hosts, routers and networks, while retaining near total application transparency. This system is based on FreeBSD Jails, which provides filesystem and network namespace isolation and some degree of superuser privilege restriction.[13]

Mirage. Mirage is a resource allocation system, which was designed for sensor networks testbeds and it is based on an auction scheme. The experimenters are bidders, who argue for resources, using a virtual currency issued by the central system. So, if a user uses the testbed in a way that matches the system's criteria, then he has more credits to claim resources for a next experiment.[10]

NITLab. In NITLab [5], we have implemented a spectrum slicing scheme, which however had some significant drawbacks and we decided to change it to this one

we are describing here. Specifically, we had focused on the new framework of Linux wireless drivers, provided by cfg80211 [1]. This packet, which is meant to replace Wireless Extensions [7,15], can support Central Regulatory Domain Agent (CRDA) [2] which controls the channels to be set on the system, based on the regulations of each country. By making some changes on this, we managed to succeed spectrum slicing on our testbed. However, this scheme limited us in terms of the available drivers that could be used with it and the Linux kernel versions that could enhance CRDA. Moreover, this scheme's implementation was tricky and very much system dependable. [8,9]

Our work here is independent from the related works described above and can be used in cooperation to them, as it schedules resource utilization from a higher level. Furthermore, we are moving our implementation on to a more abstract level, that is the one of the management framework, in order to set it platform independent, since OMF is intended to cover more platforms that just Linux. An analysis on virtualization schemes can be also found in [14], however in this paper, we are actually implementing the spectrum slicing scheme, which as shown in Section 5 has a very good performance on our testbed, while with its extensions that we are planning (see Section 7), we are expecting to scale for even more large and complex testbeds.

3 Wireless Testbed Managerial Framework

We are using cOntrol and Management Framework (OMF) [3] for managerial framework. Currently OMF is deployed on several testbeds around the globe, including ORBIT. Using a ruby-like experiment definition language, the experimenter writes an abstract description of the experiment, stating which nodes to use and what for, uses traffic generators, sinkers and other utilities which are being constantly updated and integrated inside OMF. Providing full transparency to users, OMF is responsible for loading their images to the testbed nodes that they have asked for, for configuring the nodes based on the experiment description and for gathering the results. Currently OMF consists of three basic components: *Gridservices*, *Nodehandler* and *Nodeagent*. Next, we give a short description of each one of them:

Gridservices. Gridservices consist of a set of web services, which are responsible for both executing system actions, such as turning a node on or off, rebooting nodes, loading images, etc. and getting information about the testbed as they have access to two databases: one for the testbed and its configuration and one for the scheduler where we keep information critical for slicing. Gridservices are residing on the testbed server.

Nodehandler. Nodehandler does the actual testbed management using Gridservices and other operating system applications. The user interacts with the Nodehandler to load an image to the nodes and to execute an experiment. Based on the experiment definition, which Nodehandler is responsible to interpret, this component is sending the respective commands to the nodes in order to configure

them and trigger the applications needed for the experiment. Like Gridservices, Nodehandler runs on the testbed server too.

Nodeagent. Contrarily to Gridservices and Nodehandler, Nodeagent runs on the client-side of the testbed; that is the nodes. Previously we said that Nodehandler is responsible for sending commands to the nodes, based on the experiment definition. Here comes Nodeagent, which is responsible for receiving these commands them, understanding them and then trigger the respective applications. These applications could refer to the node configuration, a traffic generator, a traffic sink, etc.

We had to extend all the three components above to integrate spectrum slicing support inside OMF. In the next section, we will show our basic idea for achieving spectrum slicing, the dilemmas and the decisions we had to make when implementing our scheme in OMF.

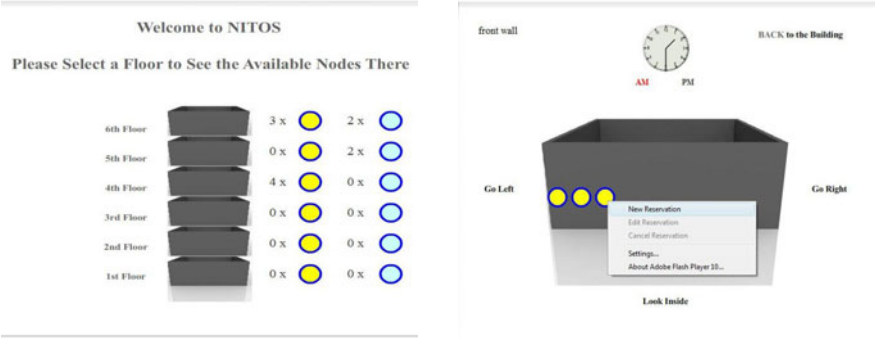
4 Scheduling Experiments on Wireless Testbeds

Currently OMF does not include any scheduling algorithms that would synchronize the experiments execution. Its implementation does not include any permissions checking for access to the testbed resources. However, in a public, multiuser environment, we need a system that will be able to assign resources only to the users that have the right to use them, while offering the experimenters a way to declare the resources that they need for their experiments. In our work, resources are divided in two categories: *nodes* and *spectrum*. So, we are providing a tool which is used by the experimenters to reserve nodes and spectrum for some time (which should not exceed some limit). Using spectrum slicing, our tool makes the testbed available to users who would like to use different resources at the same time.

4.1 Spectrum Slicing

By slicing, we mean the partitioning of the testbed based on some criteria. With spectrum slicing, we aim to partition the testbed into smaller, virtual, testbeds which are using different spectrum and, hence, they do not interfere with each other. The spectrum that each virtual testbed will use could be either defined by the experimenter at scheduling or dynamically assigned, if the experimenter does not care about the channel that he uses (for example he could ask for any channel of 802.11g modulation).

Spectrum slicing can be combined with any other resource allocation scheme, since it does not require any "negotiations" with other system resources. Furthermore, spectrum is always associated with a wireless testbed experiment and, hence, there will always be a chance to slice the testbed based on the wireless channels each experiment needs.



(a) Testbed deployment overview. (b) Selection of particular testbed node.

Fig. 1. NITOS Scheduler Node Selection

4.2 Allocating Resources - Slices

Slices are created dynamically, upon the user reservation procedure. As we have already mentioned, we discriminate resources in two categories: nodes and spectrum. Resource allocation can be made statically or dynamically. Currently, we have developed a static scheme, but we working on extending it based on Topology and Link Quality Assessment Protocol (TLQAP) [12]. In this scheme, the experimenter selects the nodes and the channels he would like to use during reservation, while at the same time, he also declares the time slots that he will be using those resources. Next, we are illustrating the basic idea of our resource allocation scheme, based on spectrum slicing. Finally, we are making a brief report on how dynamic resource allocation would be succeeded by extending our already existing tools.

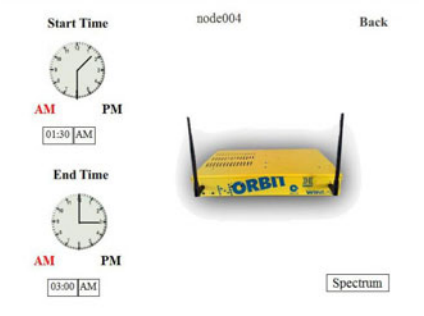
Let us consider a testbed with OMF as its management system. As we have already mentioned, OMF does not include a scheduler, hence we need to develop one as a separate component of our system. In NITLab, we have developed a scheduler, whose User Interface is available to public, through our web site. This User Interface is responsible for guiding the user through the reservation process and is designed in such a manner that the experimenter may have a very specific view of the testbed topology. Providing outside and inside view of our six-floor building, we aim to give the experimenters the best perspective of the nodes that they are reserving for their experiments.

Now consider an experimenter who would like to use the testbed. Assuming that he has already registered, he may log in to the scheduler's web site and gain access to its User Interface. From there, he first chooses the date that he would like to run his experiments. Then, the actual scheduling process begins, with the experimenter seeing our testbed building with an indication beside each floor on the number of each node type that reside on that floor, as shown in Figure 1(a).

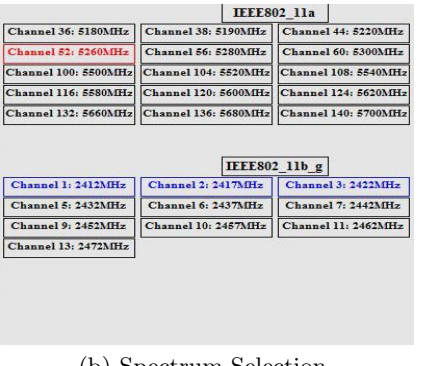
Based on these data, the user can choose a floor and guide around it from both an outside and inside view. Having an exact view of the position of each node, he makes his choice by selecting to reserve one, as shown in Figure 1(b). The clock that appears on this frame can be clicked by the user on the time he would like to check the nodes status. By clicking there, the frame is automatically refreshed and the nodes are colored according to their status, while at the same time, the new user loses permission to request a new reservation on that node at that time. So, at this phase, the demand for reservation overlap prevention is satisfied; the experimenter chooses a node available at the time he needs it and proceeds to the next step.

At this point, the user has selected his node and he is about to reserve it for some time. For this end, we give him two clocks, one for choosing the start time for his experiment and one for the end time (see Figure 2(a)). Giving the user another clock here may seem to have a security gap since the user may try (either willingly or not) to "trick" the scheduler. However, this cannot happen. First of all, the clock of the previous step is used for checking and not for reserving, as such a thing would not be very practical for the user. Then we need to perform the same check for availability of the current node here too. So, when the time duration that the experimenter chooses at this step, includes time of another user on this node, the scheduler does not allow him to move on to the next step and, hence, reserve the node.

Guided by the scheduler, the experimenter has successfully chosen a node and some time to use it. The last thing he has to do is to choose the spectrum he would like to use; that is a group of channels that will be reserved for him during his time (see Figure 2(b)). Again, the scheduler does not allow the experimenter to choose a channel that is reserved by another one during that time. This is the final step; the experimenter submits his choice and the system reserves the node and the spectrum for him. After that, he goes to the first step, getting the picture of the whole bulding to continue with his reservations.



(a) Time reservation of particular node.



(b) Spectrum Selection.

Fig. 2. NITOS Scheduler Resource Reservation

The scheduler identifies the user and lets him edit or delete his reservations at any time. It also keeps track of the last choices that he made on reservation time and spectrum, providing them to him as default choices for the current session. Finally, the scheduler provides the experimenter with the option to check out all his reservations, grouped by the reservation time. So, at any time, he may login and checkout the nodes and the channels he has reserved for some time.

4.3 Implementation

The implementation of our spectrum slicing scheme is done on two levels: (a) the user interface which guides the user through the reservation process and does not allow him to reserve an already reserved resource and (b) the OMF components, where we have added new and extended old ones to succeed the monitoring and control of the slices that are created at reservation. Next, we are examining in more depth the implementation details of each one of these two levels.

4.3.1 User Interface

The scheduler's user interface is designed to be available through a web site, so that any users may have access to it. Its goal is to allow the experimenters reserve the resources they need (in terms of nodes and spectrum) in an efficient way for the testbed usage. So, we need to reassure two things: on the first hand an easy to use environment and, on the other hand, an application that does not allow their choices to mess with other experimenters ones. Next, we are giving the reservation procedure procedure giving all the details of what happens underneath.

First the user has to log in and choose a date for his experiment. After that, we create session for that user where we hold the details of his account. From this point on, the scheduler knows who that user is and, based on that and the date, it manages permissions to resources that the user might need to access on the next steps. The main scheduling application is now deployed. This application consists of a flash animation which uses multiple PHP scripts and XML files to give the experimenter the information he needs, as we explain next.

The scheduler gives the user a perspective of the testbed topology. On our testbed, NITOS, which is located on a six floor building, the scheduler shows the number of each node type, residing on each floor. The topology view is loaded dynamically by using XML files. The scheduler application reads the respective configuration file and loads the topology that the experimenter will be able to use during his session. We have reached the choice of XML files because we are aiming to develop a tool that would easily support any other similar wireless testbed, without having to do any major changes on the code. Moreover, the testbeds themselves are not static and it would not be convenient for the administrators to change the code each time a node falls down, or a new one is added to the testbed.

The user clicks on a floor and gets its perspective. Our user interface provides full inside and outside view of the floor. Along with this view, scheduler provides

the user a clock where he clicks on the time he would like to check for reservations. Each click triggers a PHP script, which checks a database that resides on the testbed web server (for greater speed) and colors the nodes respectively. So, if the node is free at that time, it is colored with its native color and the user can make a new reservation. If the node is reserved by another user, it is colored red and the current user cannot do anything on it. Finally, if the node is reserved by this user, it is colored purple and the user can edit his reservation. What we actually do here is to grant permissions on each node (resource) based on the user that claims it.

The next step, is using similar tools with the experimenter clicking on start and end time for his experiment and the scheduler checking if those are available. At each step, the scheduler does not allow the user to move on without making a right choice. When this step has finished, the experimenter has selected a node and a time duration for his experiment.

Moving on to the final step, the experimenter has to declare the set of channels that he needs to perform his experiments. Again, using an XML file, scheduler loads all channels available, based on the laws of each country. After that, it checks the database to see which of these channels are reserved by other users and which ones are reserved by this user at a previous step. Keeping the same template as before, we mark with red the frequencies that cannot be chosen, with purple the user's previous choices on this node (in case the user had selected to edit his reservation) and with blue the previous user's choices on this session, so that he does not have to select the same channels again and again for each node he reserves.

After that step, the user's choices are committed to the scheduler's database. This database contains information about the testbed topology, the available spectrum and, of course, all users reservations. Using PHP scripts and XML configuration files, this database can be automatically updated through the web site by the scheduler's administrators. Furthermore, the scheduler's web site can provide information to each user of the reservations he has made until now, so that he may see older preferences which fitted, check the exact reservation details when the time has come to execute the experiment, or anything else.

Finally, since the scheduler's user interface resides on the web server, while the testbed has another server, we have setup a secure communication channel, which is used by the scheduler to inform the testbed server's cron daemon to schedule necessary tasks for each experiment. Such tasks are unlocking the users accounts when the reservation starts and locking them when it ends and setting up firewall rules that prevent the users from trying to access nodes that are not assigned to them, by using applications others than OMF (for example secure shell).

4.3.2 OMF Components

Until now we have described the part of the scheduler which is focused to the experimenter and his choices at reservation. This, however, is not always enough. Mistakes can be made some times willingly, some times not; in any case, we need to ensure that the experimenters will stick on their choices and, even if they try, the system will not allow them to use any resources that they have not reserved.

In order to do that, we have chosen to extend OMF, which is a very popular managerial framework for wireless tesbeds. In Section 3, we gave a short description on OMF, how it is structured and the role of its components. Here, we give a detailed description of the additions and the extensions we had to make inside this framework to integrate spectrum slicing support in it.

Before anything else, we need a way for OMF and the scheduler's database to communicate. For this purpose, we have added one more service group to Gridservices named scheduler and we have added one more service to the inventory service group. Next, we are showing what these services are responsible for. First of all, the inventory service group is developed inside OMF and provides a set of webservices that provide general information about the testbed (such as node names, IP addresses, etc). This information is stored in a database residing on the testbed server and the inventory service group reads this database to return the proper response. Our addition here is a service which gets a node location (that is its coordinates) based on its IP address. Note here that the node location is a piece of information that is the same on both the scheduler's and the testbed's database and, thus, we can use it to do the matching. We have added this service, because when an experiment is executed, OMF does not know a node's location; only its IP address.

Now that scheduler knows the exact location of the node, it can use the scheduler service group to get any information needed from the scheduler's database. Namely, the services provided by this group provide functionality to get a node reservations based on its coordinates, the spectrum that this reservation contains and the user that owns it. Furthermore, it provides services that can do the matching between a channel or a frequency number and the respective spectrum identification number as it is stored in the database. All this information will be used by Nodeagent, which decides whether to allow the user use the channel or not.

So, Nodeagent is responsible for deciding whether the resources declared in the experiment should be allocated to the experimenter. In order to decide, the Nodeagent has to ask the scheduler's database if the specified resources have been reserved by the experimenter. So, when the experiment sets the wireless card channel, this information is passed to the Nodeagent, which now knows the channel along with its own IP address. All he needs is the user identification to check with the scheduler's database if this channel (and, of course, node) should be allocated to that user.

However, this is not straightforward, since the user usually logs into the node as root (keep in mind that the experiment loads his own image to the nodes, so he has full privileges on them). So, we need to track where did he use the username that he also used for registering. The scheduler is designed in such a manner that, when a user registers to the system, then an account with the same username and password is automatically created to the testbed's server. The user uses this account to both access the user interface and the testbed server (using secure shell connection). This can solve our problem, since we can say for sure

that the user that is running the experiment is logged into the console with the same username that he has made his reservation.

This information, though, relies on the testbed server, while the Nodeagent runs on the client side; that is the nodes. We need to pass that information from the server to the clients. This is done by the Nodehandler, the OMF service that is running on the server side and is responsible for controlling the experiment execution. Using its built-in message passing mechanism, Nodehandler tells the Nodeagent the username of the experimenter and now the last one has almost everything he needs to do the matching, except the date. The system should not rely on the experimenter to keep the clock of his clients coordinated with the testbed. This is why, Nodehandler sends, along with the username, the date at that time and the Nodeagent adjusts its clock to match the server's.

At this point, Nodeagent has all the information needed to check with the scheduler if the requested resources should be allocated to the experimenter. Using the web services we described above, the Nodeagent checks if there is a reservation at that time for that user and if the spectrum reserved at this reservation matches the channel that the experimenter has requested to assign to the network card through his experiment.

If all data match, then the Nodeagent lets the experiment execution move on. Otherwise, it notifies the Nodehandler that a resource violation has taken place and stops its execution (without assigning the channel to the node's network card). When the Nodehandler receives that message, the execution is terminated immediately and an ERROR message is thrown back to the experimenter describing the resource violation.

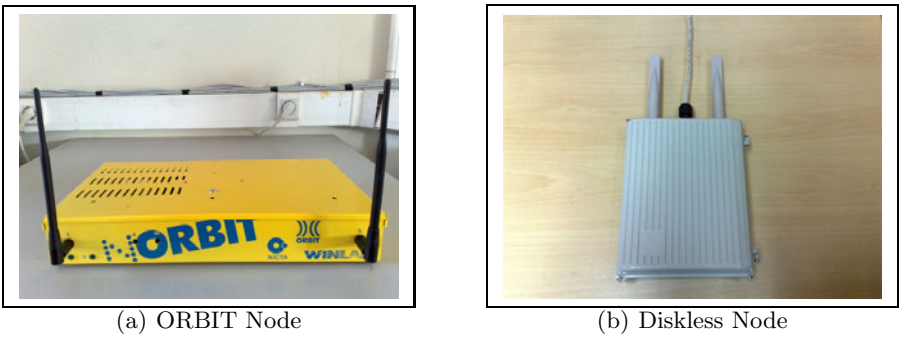
5 Slicing in Action - Usage Statistics

5.1 Testbed Description

The testbed that we used for design and deployment of our scheme is consisted of 10 ORBIT-like nodes, as depicted in Figure 3(a) and 5 Diskless nodes, as shown in Figure 3(b). An ORBIT-line node consists of a 1GHz VIA C3 processor, 512MB of RAM, 40GB of hard disk, two ethernet ports and two miniPCI slots which are used to host two Atheros WiFi cards. Our diskless nodes consist of a 500 MHz AMD Geode LX800 CPU, 256MB of RAM, a 1GB Flash Memory Card, two ethernet LAN ports and two Atheros wireless cards.

All the nodes are connected through wired Ethernet with the testbed's server - *console*. In console we have all the required testbed services running. These services are both network services, such as Dynamic Host Configuration Protocol (DHCP) server which gives IP address to the nodes, Domain Name System (DNS) server which gives names to the nodes, Network File System (NFS) server for experiment results and slicing support as we are going to see next, and testbed services which are combined to the functionality of OMF.

We also maintain a web server where we keep the web interface of our system's scheduler. On this server, we also keep some scripts mandatory for remotely booking the nodes and a MySQL server for keeping records of the testbed status



(a) ORBIT Node (b) Diskless Node

Fig. 3. NITOS Nodes

at each slot. Finally, we have set up a secure communication line, using Secure Shell (SSH) and a RSA key between the web server and console so that the scripts on the web server trigger the respective scripts on console.

After the user books some nodes at a specific time on the testbed, he logs into console at that time and from there, he can start using the testbed. The image is loaded on each node from console through the wired Ethernet interface. More information about our testbed's architecture can be found at our web site.

Although we have built this scheme on our testbed, it could also be applied to any wireless testbed which is using OMF as its management framework.

5.2 Experiment Execution Scenario

Here, we give an experiment execution scenario on our testbed, in order to illustrate the scheduler's usage and importance. In this scenario, we have Bob and Alice to be our experimenters, who have made their reservations and want to use our testbed, NITOS.

In Figure 4 we are showing a snapshot of NITOS execution on October 10, 2010 at 11:05 AM. At that time, there are two users whose experiments are about to be loaded: Bob and Alice. As we can see, both our users have reserved resources through NITOS Web Server and these reservations are kept into the Scheduler DB. Bob's reservation begins at 10:00:00 AM, ends at 12:00:00 AM and includes channels 7 and 8. Similarly, Alice's reservation begins at 10:30:00 AM, ends at 11:30:00 AM and includes channels 2 and 3.

Now, based on these reservations, Bob and Alice are trying to execute an experiment, so they log into NITOS server. At this point, we should mention, that since we allow experimenters to log into our testbed server, we need to be very careful with security. This is why, we alter our firewall rules, which are tailor made on each user, so that he would not be able to access any of the testbed resources he has not already reserved (e.g. in our case, Bob cannot use a node that belongs to Alice or to neither of two).

After they have logged into the server, Bob and Alice ask OMF to execute their experiments. In his experiment description, Bob is setting the channel

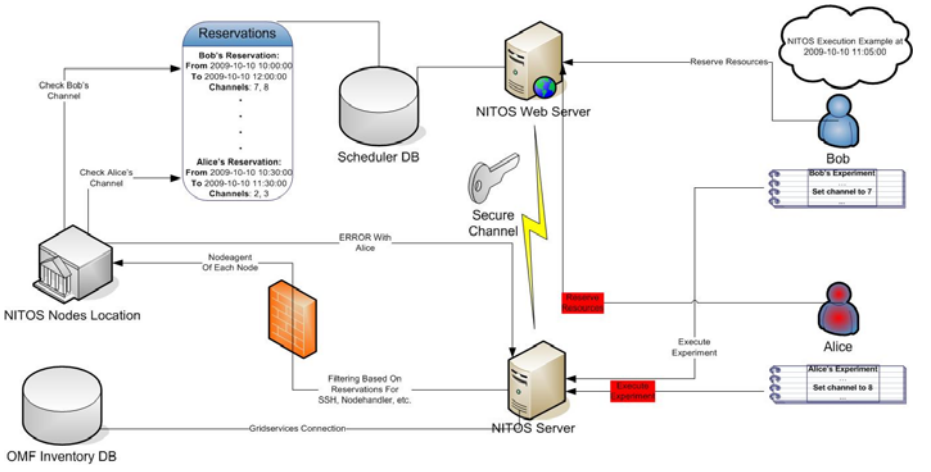


Fig. 4. Bob and Alice Experiments Execution

of his nodes to 7 and Alice to 8. Based on each experiment, the Nodehandler notifies accordingly the Nodeagent that is running on the nodes, which rely on a University building. When the Nodeagent is asked to set the channel of Bob's nodes, it checks Bob's reservation on Scheduler DB to see if channel 7 is included. Indeed, channel 7 is included in his reservation and the experiment execution continues normally. Now, when the Nodeagent is asked to set the channel of Alice's nodes, it checks Alice's reservation on Scheduler DB and sees that Alice has reserved channels 2 and 3 and not channel 8. In this case, the Nodeagent does not set the channel, terminates this session and sends an ERROR code back to the Nodehandler which indicates that there has been a channel allocation problem with Alice's experiment. The Nodehandler terminates the experiment execution and notifies Alice of the problem.

5.3 Usage Statistics

To outline slicing benefits, we have monitored testbed usage for a period of 5 months. We have added logging support to our scheduler as follows: The user is firstly prompted to enter the number of nodes that are needed by his/her experiment without being able to see which testbed nodes are occupied. If the system has enough resources to satisfy the request, the user may continue with the standard allocation procedure. Otherwise, the scheduler informs the user that the required amount of nodes cannot be allocated. With this approach we were able to log the allocation requests which were denied. Note that in the standard scheduler interface, the user is provided with enough visual information to determine whether the required number of nodes is available or not and avoids issuing requests which would be denied.

During these 5 months, the testbed use was approximately 500 hours. During these hours a total of 1008 requests were issued. To determine overall testbed

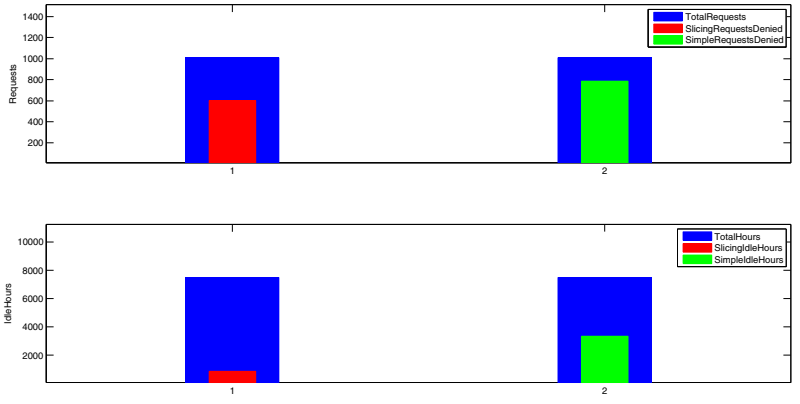


Fig. 5. Slicing performance for a 5-month period, compared to the simple allocation policy simulator results for exactly the same workload

utilization, we logged for each hour during which the testbed was in use and allocation requests were denied, the number of nodes that were not occupied. More specifically, we regard as the testbed idle time unit the idle hour of a single node. If for example 5 nodes are idle during a testbed usage hour this amounts to 5 idle hours. Since we have 15 nodes, the available usage time units of the logging period were 7500 ($TestbedUsageHours * NumberOfNodes$). To compare slicing with the simple allocation scheme that cannot allocate wireless frequencies, we developed a simulator. We should note here, that our testbed topology, in terms of physical wireless range, forms 2 independent neighborhoods. The first neighborhood has 7 nodes and the second 8. Therefore,the simple node allocation scheme can assign each node neighborhood independently and host up to two testbed users concurrently. Our simulator implements this policy, replays the allocation requests that have been logged for the 5-month period, determines the number of denied requests and testbed utilization time of the simple allocation scheme. In Figure 5 we depict the number of denied requests along with the testbed total idle time that we logged for slicing and simulated for simple allocation scheme.

6 Conclusions

As wireless networking research emerges, the respective testbed infrastructures and management systems should employ more sophisticated approaches to distribute available resources. While many of the management concepts that have been introduced for wired testbeds, have been extended and reused by wireless testbed management frameworks, the latter face an additional important challenge: the distribution and management of the wireless bandwidth in terms of frequency channels, which along with the node topology and connectivity range

can become a very complicated task. In this work we attempted to address this issues.

7 Future Work

In the domain of scheduling experiments on wireless testbeds, there still is much work that has to be done. Since we are expecting a growth to the testbeds usage by the researchers in the near future, we should put our efforts in developing a scheduling scheme, which will be able to allocate resources to experimenters efficiently, while, in the same time, it will be providing a transparent mechanism for executing the experiments.

First of all, we are working on integrating TLQAP in our scheduler. The scheduler can use TLQAP to extract information about the status of the testbed resources at any time needed. With this information, it is in place to match the experimenters demands on power, spectrum, location, etc. with the resources and schedule the experiment whenever they are available, without having the experimenter himself to check for their availability at each slot.

Furthermore, we are working on implementing other slicing schemes, which will depend on the transmission power control, the sharing of wireless network cards and the sharing of nodes themselves. For instance, we may adjust the power that a node will transmit based on the experiment characteristics and, thus, create a smaller neighborhood where the experiment will take place, while the rest of the testbed will stay available to other users. With network cards sharing, we plan to let two users make use of a node which has two wireless cards on, by assigning one card to each user. Finally, nodes sharing will give us the power to run multiple experiments on different images on the same for multiple users. We are expecting that this last scheme, in combination with spectrum and power slicing and wireless cards sharing, will give us a large scale improvement to the testbed utilization.

Wireless testbeds federation is another step that we are planning to take. The additions made for spectrum slicing in OMF services, provide us a very good tool for this end. We are thinking federation in two aspects: that of experiment execution and of experiment scheduling. Our work focuses on satisfying both these aspects using web services, which can be used as the tool to remotely invoke resource, get information, etc. However, with federation, we have other issues arising too, such as security, since we are dealing with resource allocation through a public network (Internet).

References

1. cfg80211 - Linux Wireless, `http://wireless.kernel.org/en/developers/Documentation/cfg80211`
2. CRDA - Linux Wireless, `http://wireless.kernel.org/en/developers/Regulatory/CRDA`
3. OMF Developer Portal, `http://omf.mytestbed.net`

4. OneLab2, http://www.onelab.eu/
5. UTH NITLab, http://nitlab.inf.uth.gr
6. WINLAB - Rutgers University, http://winlab.rutgers.edu
7. Wireless-Extensions - Linux Wireless, http://wireless.kernel.org/en/developers/Documentation/Wireless-Extensions
8. Anadiotis, A.-C., Apostolaras, A., Syrivelis, D., Korakis, T., Tassiulas, L., Rodriguez, L.R., Seskar, I., Ott, M.: A Demonstration of a Slicing Scheme for Efficient Use of Testbed's Resources. Demo - Mobicom (September 2009)
9. Anadiotis, A.-C., Apostolaras, A., Syrivelis, D., Korakis, T., Tassiulas, L., Rodriguez, L.R., Ott, M.: A New Slicing Scheme for Efficient Use of Wireless Testbeds. In: Proceedings of the 4th ACM International Workshop on Experimental Evaluation and Characterization (2009)
10. Chun, B.N., Buonadona, P., AuYoung, A., Ng, C., Parkes, D.C., Schneidman, J., Snoeren, A.C., Vehdat, A.: Mirage: A Microeconomic Resource Allocation System for Sensornet Testbeds. In: Proceedings of the 2nd IEEE workshop on Embedded Networked Sensors (2005)
11. Raychaudhuri, D., Seskar, I., Ott, M., Ganu, S., Ramachandran, K., Kremo, H., Siracusa, R., Liu, H., Singh, M.: Overview of the ORBIT Radio Grid Testbed for Evaluation of Next-Generation Wireless Network Protocols
12. Syrivelis, D., Anadiotis, A.-C., Apostolaras, A., Korakis, T., Tassiulas, L.: Tlqap: A topology and link quality assessment protocol for efficient node allocation on wireless testbeds. In: Proceedings of the 4th ACM International Workshop on Experimental Evaluation and Characterization (2009)
13. Hibler, M., Ricci, R., Stoller, L., Duerig, J., Guruprasad, S., Stack, T., Webb, K., Lepreau, J.: Large-scale Virtualization in the Emulab Network Testbed. In: USENIX 2008 Annual Technical Conference on Annual Technical Conference (2008)
14. Mahindra, R., Bhanage, G.D., Hadjichristofi, G., Seskar, I., Raychaudhuri, D., Zhang, Y.Y.: Space Versus Time Separation For Wireless Virtualization On An Indoor Grid. In: Next Generation Internet Networks (2008)
15. Tourrilhes, J.: Wireless Extensions for Linux, http://www.hpl.hp.com/personal/Jean_Tourrilhes/Linux/Linux.Wireless.Extensions.html

Measurement Architectures for Network Experiments with Disconnected Mobile Nodes

Jolyon White, Guillaume Jourjon, Thierry Rakatoarivelo, and Maximilian Ott

NICTA*
Australian Technology Park
Eveleigh, NSW, Australia
firstname.lastname@nicta.com.au

Abstract. Networking researchers using testbeds containing mobile nodes face the problem of measurement collection from partially disconnected nodes. We solve this problem efficiently by adding a proxy server to the Orbit Measurement Library (OML) to transparently buffer measurements on disconnected nodes, and we give results showing our solution in action. We then add a flexible filtering and feedback mechanism on the server that enables a tailored hierarchy of measurement collection servers throughout the network, live context-based steering of experiment behaviour, and live context-based control of the measurement collection process itself.

Keywords: measurement, testbeds, mobile, OML, disconnected measurement.

1 Introduction

Distributed networking experiments require distributed measurement collection systems. Approaches to remote network measurement collection range from ad-hoc methods used in academia through to large, commercial systems deployed by network operators. Ad-hoc methods are typically sub-optimal, error-prone, and time consuming, but available measurement and monitoring frameworks [11,12] tend to be prohibitively complex for use in many research projects. A measurement framework for network research should be simple to use and administer, but must be flexible enough to match the heterogeneous, dynamic needs and environments that usually characterize it.

Mobile networking research is a good example. Indeed, for static testbeds, a simple client/server measurement collection architecture is adequate [5], as long as the rate of measurement output does not influence the studied phenomena, and does not overload the collection server. If a testbed network includes mobile nodes, some nodes may not always be connected to the network. In that

* NICTA is funded by the Australian Government as represented by the Department of Broadband, Communications and the Digital Economy and the Australian Research Council through the ICT Centre of Excellence program.

T. Magedanz et al. (Eds.): TridentCom 2010, LNICST 46, pp. 315–330, 2011.

case, what should happen to the measurements that the disconnected nodes are generating?

On the other hand, in an experiment where all nodes are always connected, the rate of generation of measurement data by even a single node may congest either the network, the measurement collection server, or the client applications generating the measurements. This can result in lost measurements; worse still, it can lead to the measurement collection activity influencing the behaviour of the network under observation, and with it, the results of the experiment.

These two different problems can both be solved by making a single but important change to the architecture: namely, the addition of a proxy server on the experiment node, effectively a queue, to act as an intermediary between the client applications and the measurement collection server. Once the measurement architecture contains such proxy servers on the experimental nodes themselves, a further innovation of the architecture becomes apparent, that to our knowledge has not been attempted before. We extend the proxy server to implement a measurement database instead of just a queue, which allows us to perform measurement-based feedback to the experiment applications themselves in what we term *distributed, context-based experiment steering*. This leads to a flexible hierarchy of measurement servers for future advanced testbed networks.

In this paper, we consider the architecture of measurement collection frameworks in detail:

- We describe the two problems of mobility (Section 3.1) and measurement bandwidth constraints (Section 3.2).
- We show how both of these problems can be solved by introducing a proxy server to buffer measurements on the local node before sending them to the central measurement server (Section 3.3).
- We give some quantitative measurements to demonstrate the benefits of this approach (Section 3.4).
- We discuss extensions to the proxy server to allow measurement-based experiment steering (Section 4).
- We compare and contrast our architecture to existing measurement frameworks (Section 5).

The measurement architectures described in this paper are embodied in OML2, the second generation Orbit Measurement Library, which we have developed and made freely available at [3]. OML is a generic measurement framework capable of instrumenting the entire software stack, and is not just limited to network-specific measurements.

2 Background

To set the scene for this paper, we begin with a description of the testbed network environments that we are considering, and a simple, naïve client/server architecture for measurements that we use as our starting point.

2.1 Testbed Architectures

Fig. 1 shows a general testbed architecture. The *experiment nodes* participating in the experiment use one or more *Experiment Networks* (EN) to perform the networking tasks that comprise the experiment itself. Meanwhile, the *control nodes* communicate with the experiment nodes using a separate *Control Network* (CN) to perform tasks such as imaging the nodes with an operating system at the start of the experiment, bringing the experiment nodes up, starting the applications that participate in the experiment on the experiment nodes, logging status and error information, and ensuring orderly shutdown of the experiment once it is complete.[1]

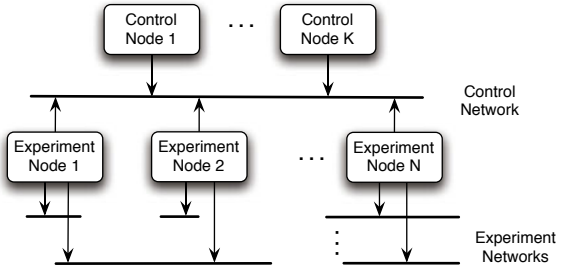


Fig. 1. Generic testbed network architecture

The architecture in Fig. 1 can contain heterogeneous experiment nodes, each node connected to different experiment networks. In practice, there will be variations in the hardware capabilities and available interfaces on each experiment node. With mobile nodes, the connectivity may even change mid-experiment.

The separate control network minimizes the impact of control tasks on the behaviour of the experiment tasks, so that the underlying protocols, algorithms, and applications can be studied in as much isolation as possible. This improves quality of results and repeatability. However, sometimes we do not have the luxury of a separate control network. The node hardware might not support enough interfaces of the right type, or the separate infrastructure required for a control network might not be available for some nodes. Mobile nodes often have these properties.

We have drawn the control and experiment networks as single network segments, but this is just for simplicity: the actual network topology of each network could be more complex. Also note that infrastructure nodes such as routers could also be participating in the experiment and generating measurements.

2.2 Experiment Node Architecture

Each experiment node runs a number of applications and services (i.e., daemons) that execute the tasks required to run the experiment itself, as shown in Fig. 2.

[1] We use OMF, a control framework that we have developed, to perform these tasks on our own testbed networks [16].

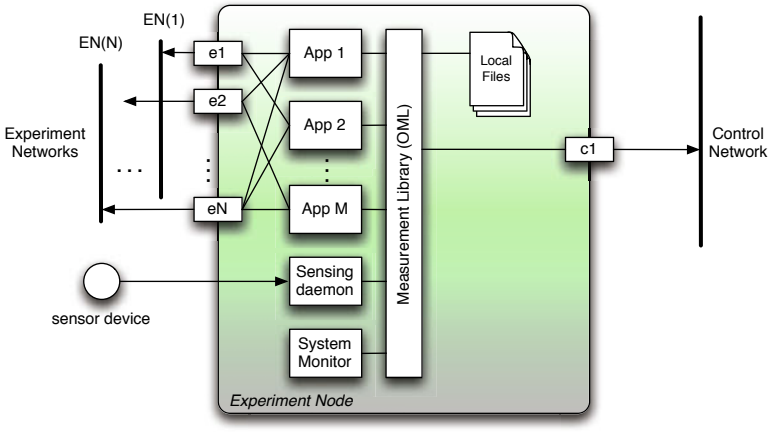


Fig. 2. Architecture of an experiment node, showing some applications, a system monitor daemon generating measurements from information provided by the operating system, and a sensor daemon generating measurements from an input sensor device

These applications and services can communicate with other experiment nodes using the interfaces $e1$–eN. They can also access and monitor operating system information and local devices, such as GPS receivers, temperature sensors, and pressure sensors.

The applications and services perform measurements of the system under study, measuring quantities such as:

- network characteristics and impairments (e.g., bandwidth, packet loss rate);
- local context information (e.g., RAM or CPU usage); and
- device-generated data (e.g., GPS coordinates, temperature, pressure),

for example. They use functions provided by the OML measurement library to send their measurements either to a file on the local filesystem, or to a measurement server on the control network via $c1$. OML is flexible enough that the applications can send measurement data to multiple measurement servers if desired. Fig. 2 shows the general case. The node may have only one experiment network interface ($N = 1$) and it may have to send and receive control and measurement data over the experiment interface if no separate control interface is available.

The experiment nodes may have a wide variety of hardware, operating systems, attached peripheral devices, and networking interfaces. Thus, the measurement architecture must be portable, flexible, and efficient enough to cope with such a wide range of platforms.

2.3 Client/Server Measurement Architecture

The simplest distributed measurement architecture, which is our starting point, is a client/server architecture. OML operates in this fashion in its most basic configuration. Fig. 3 depicts the data path from a single experiment application to the server in OML.

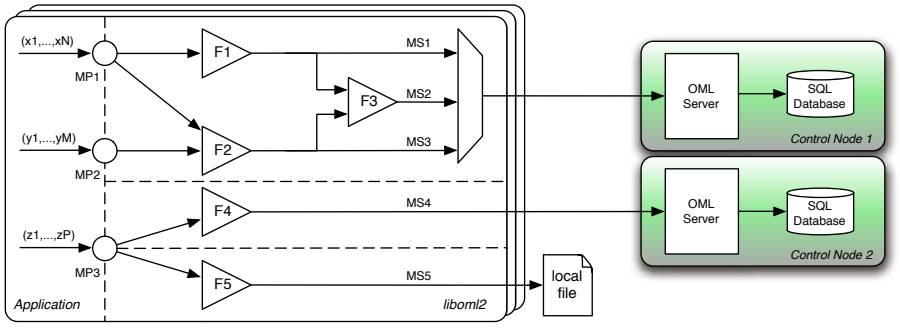


Fig. 3. Measurement data path in OML. The application illustrated defines three measurement points, and the user has configured the library to generate five measurement streams.

The application defines a number of *measurement points* (MP) into which it injects a stream of typed measurement tuples. Each MP is an interface to the client library, `liboml2`. The client library creates a number of *measurement streams* (MS), based on the run-time configuration specified by the user in an XML file, to match the needs of the experiment. Each MS filters the MP inputs in a configuration defined by the XML file. OML supports built-in and user-defined filters. Fig. 3 shows that an MP can be a source of data for multiple MS's (MP1 participates in streams MS1, MS2, and MS3), and that filter outputs can be combined to form new streams (filters F1 and F2 are inputs to F3, which generates stream MS2). Measurement streams can be sent to an OML server or a local file (also configurable via the XML file) and different measurement streams can be sent to different destinations, including potentially multiple OML servers. The filter outputs are also typed tuples.

Currently OML supports integer, floating point, and string data, and we have plans to add support for more data types such as blobs. OML uses a one-way protocol initiated by the client. Both text and binary versions of the protocol are available, and we are currently evaluating adopting IPFIX [6].

The server collects data from each experiment and sends it to a storage backend. Because the measurement streams consist of sequences of typed tuples, they are well suited to be stored in tables in a relational database; currently the concrete backends supported by OML are SQL databases. OML imposes very little structure on the measurements collected to remain as flexible as possible and support an evolving research context. The SQL database allows us to offload the problem of devising our own measurement storage format, and provides easy and efficient result querying. OML currently supports SQLite directly, but some OML users have added support for PostgreSQL. We plan to extend OML to directly support multiple database backends in the future.

There is a table in the database for each measurement stream in each client application; the same application can be running as part of the same experiment on multiple nodes, in which case all of their measurement outputs will be stored in the same table.

3 Measurement in Dynamic Networks

We now describe two scenarios where the assumptions underlying the client/server architecture are broken, and we further show how our proxy-based architectural enhancement addresses these problems. These examples are informed by the previous experiments of users evaluating their own research prototypes on wireless testbeds, such as the ORBIT or NICTA testbeds [17]. Thus, they represent real problems that users had to overcome to advance their research agendas.

3.1 Mobile Nodes

Sometimes when users perform experiments involving mobile nodes, they are explicitly interested in studying the behaviour of networking technologies and algorithms when the mobile nodes move outside the testbed network's normal wireless coverage. For example, smart phones typically have multiple radio interfaces, such as WiFi, 3G, and WiMAX. We may be interested in the behaviour of a distributed algorithm that preferentially favours a low-cost radio interface (WiFi) when available, but falls back on a more expensive interface (3G, WiMAX) if no other networks are available in the mobile handset's vicinity [14]. They may even go out of range of all wireless networks for a period.

Such experiments could be done with real mobile handsets, or they could be done with mid-range hardware emulating the mobile handsets. In either case, this configuration causes two problems for measurement collection.

The first problem is that if measurements are sent during the experiment, then the measurement traffic must often be sent over one of the experiment network interfaces, which may interfere with the experiment itself and taint subsequent measurements. Depending on the testbed configuration, the measurement server may not even be reachable from any of the experiment networks, and this situation could even extend beyond the duration of the experiment. This leads to the second problem: what should the mobile node do with the measurements that it generates while it is out of range of the control network?

We have two options: either drop measurements while the control network is not reachable, or buffer them until the mobile node reconnects to the control network. Discounting the first option as undesirable, we must buffer.

3.2 Throughput-Constrained Measurement

Even in networks with a static topology, we still sometimes need to buffer measurement data on the local node. If the experiment involves high traffic rates on high bandwidth interfaces, then the rate of generation of measurements can also be very large, and the datapath to the measurement server can become congested, risking either loss of measurement data or changes in the behaviour of the experiment applications due to delays in the measurement datapath.

One of the primary aims of the filtering facility of `liboml2` is to allow reduction of the measurement data that needs to be transmitted to the server, for instance, using averaging. However, in some circumstances the experimenter

might want to observe effects that the filtering would discard. In that case another solution is required: buffering measurements on the experiment node.

3.3 Proxy Servers

Recalling that we want our measurement framework to be as convenient as possible for researchers to use, we want to ensure that buffering measurements does not force complicated modifications to the client applications. Our solution is to create a separate *proxy server* on the experiment node. The proxy server acts as a FIFO queue, but allows the experimenter to gate the FIFO output.

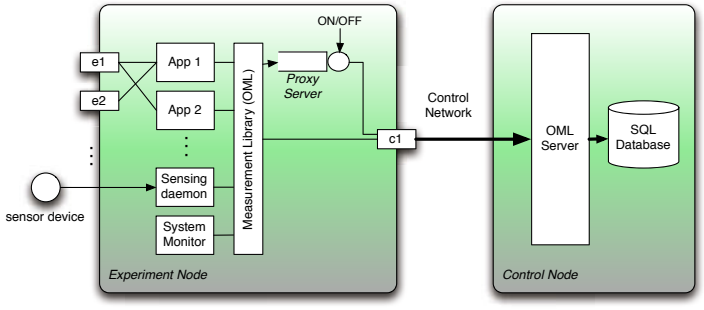


Fig. 4. Measurement architecture with proxy servers

Figure 4 shows the proxy server architecture. The proxy server presents an interface to the client applications that is identical to the regular OML server: it supports TCP or UDP socket connections using the same protocol as the server. It is thus transparent to the client applications, which do not need to be modified or re-compiled. The measurement server protocol (TCP/UDP), address, and port number are run-time configuration parameters, specified on the client application's command line or through an XML configuration file.

In Fig. 4, the proxy server is shown running on the same node as the experiment applications, but it can be hosted on a separate node if the situation requires. However, for the two use-cases we presented in the preceding sections, running the proxy server on the experiment node is exactly what we want, because it makes the measurement collection independent of the network. In both of those cases, the proxy server is configured to buffer all measurement data in memory until the end of the experiment. At the end of the experiment, the experimenter instructs the proxy server to turn 'ON' the output stream, whereupon the proxy server connects to the upstream full OML server and transmits the stored measurements to it.

The proxy server implementation is simple, as it does not have to process any data on its input stream. Furthermore, since it is a one-way stream, the proxy server can simply store the raw octets from the experiment applications in memory, and then replay them out to the OML server verbatim. It also has the option to write the measurement data to file to provide a backup and limit its memory usage.

3.4 Results

We now provide some experimental data that demonstrate the management of the disconnection. This experiment was originally presented at the 4^{th} GENI Engineering Conference [4]. In this experiment, two mobile nodes exchange UDP traffic over a WiFi ad-hoc network. One node is stationary, and the other moves along a short circuit as illustrated in Fig. 5. Both nodes run an OML-enhanced version of `iperf` [2]. In addition, the roaming node also runs a GPS application collecting location information. The UDP traffic and GPS measurements are collected using the OML framework.

Fig. 5. Path of the roaming node from GPS data (aerial photo from Google Maps [8])

To demonstrate the proposed proxy scheme, the UDP traffic measurements are collected via OML over the WiFi network, which will become unavailable as the roaming node moves away from the static one. However, to allow real-time visualization during a live demonstration, a second permanent WiMax network is used to continuously collect GPS information. The duplication of the GPS measurement stream is completely transparent to the application.

We have run this experiment numerous times, with a typical result shown in Fig. 7. In this figure, the x-axis represents time; the OML server automatically time stamps all samples received throughout the experiment. The distance was computed based on the GPS localisation, and the bandwidth was computed using fixed windows of one second on the receiver side.

In Fig. 7, we can observe the correlation between the distance and the achieved bandwidth. This can be explained by the fact that during this time the two nodes are disconnected. During this disconnection period, all the measurements are stored by the proxy. Once the roaming node gets closer to the static node and to the OML server, the proxy is set to resume sending the buffered measurements to

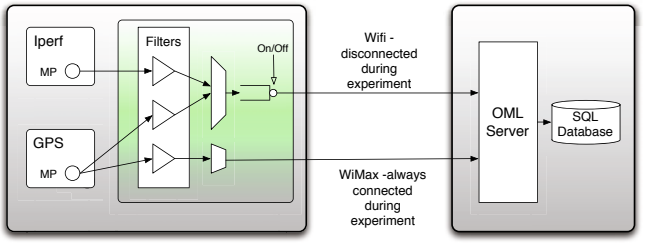


Fig. 6. OML Internal Configuration of the two Nodes and Server

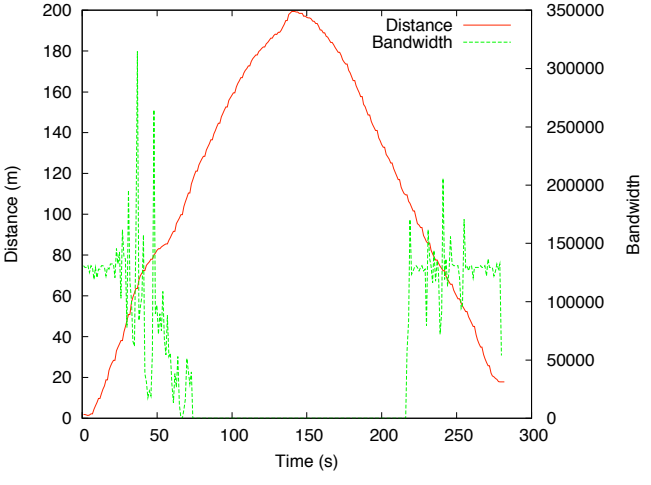


Fig. 7. Packet Loss Rate and Distance in function of the time

the OML server. Another advantage of OML in this experiment is the automatic time stamping, which allows the time evolution of different quantities to be put in perspective.

4 Further Extensions

Once the measurement architecture contains a processing element beyond the client applications on the experiment node, it is natural to ask what further sorts of processing could be done on the node itself. This line of thought takes the measurement architecture away from a basic client/server architecture. In the following sections, we describe two architectural enhancements we have developed as a result of discussions with users of OML whose needs were not met by the standard existing OML facilities.

4.1 Hierarchical Measurement Collection

In a measurement application that generates large volumes of data, it may be either too expensive or impracticable to store every sample collected. This may

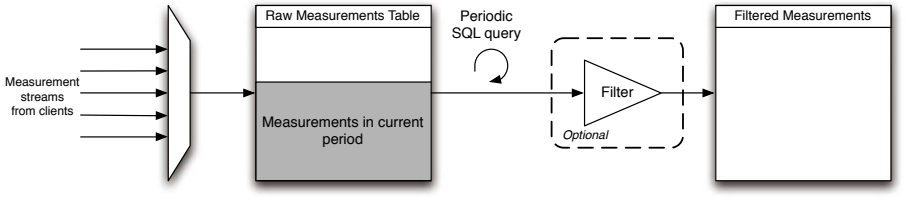


Fig. 8. Server architecture for hierarchical measurement collection

not be a problem if the utility of the collected samples decreases over time. For instance, in a server load-monitoring application, high resolution measurements for the last hour might be interesting and useful, but the same for a period six months ago might be useless. An average over a coarser timescale might suffice for historical records of that age or older, so that full-rate, high resolution data does not need to be stored in its entirety.

The basic architecture described in the previous section does not permit this type of volume-thinning. To support it, we augment the server architecture as shown in Fig. 8. The server includes a query mechanism that periodically executes an SQL query on any measurement table. The results of the query are appended to another table. The query can, e.g., compute an average of numeric quantities stored in the table, then cull the rows that were averaged, to prevent the table size from increasing beyond a set bound. This gives a compromise between the storage requirements and availability of high resolution data, and is similar to stream databases [7] and round-robin databases [18].

We can extend this idea in three ways. First, we can compose a hierarchy of measurement timescales to suit the requirements of various different users of the collected data. For instance, we could store high resolution measurements for the last ten minutes, medium resolution for the last hour, and low resolution for the last six months.

Secondly, the destination table for the periodic query results does not have to be hosted on the same machine. We can instead transmit the aggregate measurements to another OML server, using the same measurement protocol that the client applications use. The hierarchy of timescales is then reflected in the hierarchy of collection servers. This flexibility can be put to several uses, such as server load management or multi-site redundant storage, but we will describe what we think are some of the most interesting ones in the following subsection.

Thirdly, if SQL does not provide enough expressive power to compute the desired summary metric for a particular measurement table, we can augment the server with a configurable filtering mechanism, identical to the one available to the client applications in OML. This is the filter block shown in Fig. 8.

We can use the measurement stream architecture in the client application to implement very flexible measurement collection configurations. For instance, we can create two streams from the same MP and send one stream to a local high-resolution server on the experiment node and the other to a lower-resolution server elsewhere on the network.

4.2 Context-Driven Experiment Steering

We now have an architecture with an OML server that can do periodic computations on the received data, and send results of the queries to an upstream server. The local OML server can be running on the experiment node itself. We could also use this mechanism to periodically check for particular events that might be reflected in the measurement data. If we add a feedback mechanism, then we can use this event detection facility to modify the course of an experiment while it is running. We call this capability *context-driven experiment steering*.

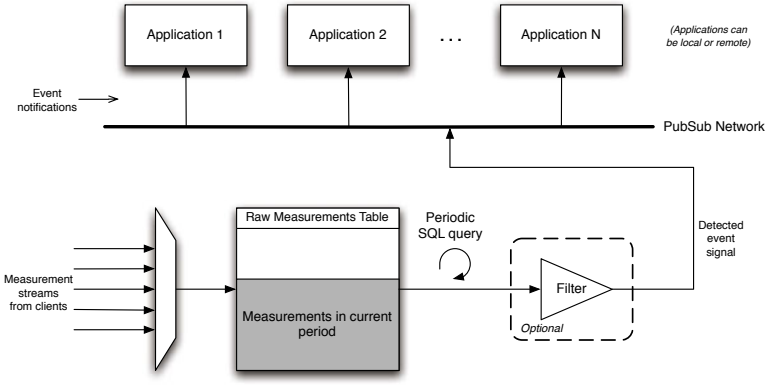


Fig. 9. Context-driven experiment steering. Measurements are used to detect conditions that trigger pre-defined behaviour in the client applications, using a publisher/subscriber notification mechanism.

Fig. 9 illustrates this architecture. The feedback loop can be contained entirely within one node if the OML server is running on the same node as the applications. More generally, the feedback mechanism is distributed, so that a remote OML server in the measurement hierarchy can give steering feedback to one or many nodes participating in the experiment.

One example of such an application would be to detect when the quality of service to a node becomes too degraded, and remove the node from the experiment. Another approach might be to only start some of the experiment applications after a condition has been met, for example, in a peer-to-peer download experiment, only starting the peers once the seeder has downloaded enough of the file from a central server. The idea of trip lines, where a mobile node crossing from one geographic region to another causes some action to be performed, is a third example of experiment context that can be implemented using our measurement architecture [9].

We are considering a publisher-subscriber framework to implement the feedback mechanism, such as Dbus or XMPP. The client applications must subscribe to and listen for particular events, and the server must have a mechanism for specifying what events are published and how they are detected. This could use a combination of SQL queries and filtering, with the final stage of the filter being

a predicate function. The management framework can also play a part in the feedback mechanism, e.g., for starting and stopping experiment applications.

This extension opens up a range of new possibilities for experiment design and measurement applications.

4.3 Context-Driven Measurement

If we have a feedback mechanism that detects events in the measured environment, why not then allow detected events to influence the measurement process itself? This is the third extension. We reflect the feedback mechanism back onto the OML server, so that we can tailor the measurement strategy to the current conditions. For instance, suppose we are only interested in low-resolution measurements of a particular quantity most of the time, but when an alarm occurs, we want to start recording high-resolution measurements. In this case we can add a second query/filter path to Fig. 8, and switch between them based on a feedback signal, as in Fig. 10.

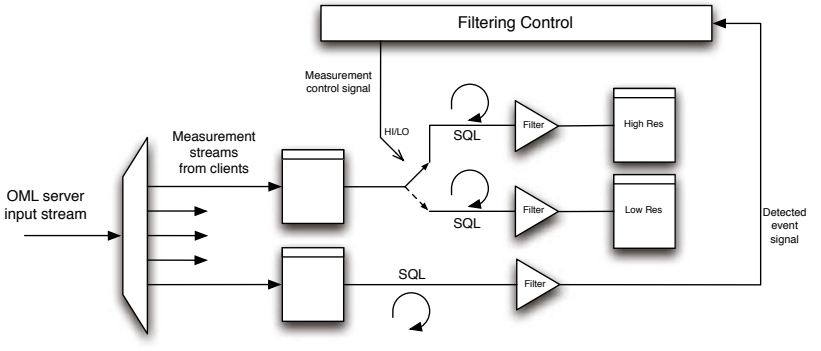


Fig. 10. Context-driven measurement. A measured event feedback signal is used to influence the measurement capture process itself.

With this third extension, we have outlined our architecture for distributed measurement collection. We now go on to compare our architecture against other work in the field.

5 Related Work

There are various existing measurement frameworks, some of them open source and some of them proprietary. Some of them are geared towards network monitoring for system administration, whereas others are more useful in research contexts.

CoMo (*Co*ntinuous *Mo*nitoring) [11,12] is a network measurement system based around measurement of packet flows. It has core processes that are linked in stages, namely packet capture, export, storage, and query. These processes capture, filter, measure, and store properties of packets and packet traces. The

core processes are linked by user-defined modules that are used to customize the measurement system and implement filtering functions. The query process provides an interface for distributed users to run queries on the captured packet traces.

The core processes are designed for speed and efficiency and are in charge of data movement operations. One of the overriding goals in CoMo is to make querying as efficient as possible[11], because CoMo can operate on very large data sets ($\sim$ 1 TB). CoMo modules essentially pre-compute the answers to queries, speculatively. Queries identify traffic with specific properties, such as finding flows that match certain criteria. As the CoMo system itself, including captured packet storage, can be distributed across the network, CoMo introduces the notion of "distributed indices" to speed up the process of finding the locations of packet traces of interest to a query.

CoMo is a highly tailored tool designed for efficient packet trace capture and analysis. OML, by contrast, is a generic framework that can instrument the whole software stack, and take input from any sensor with a software interface. One of CoMo's great strengths is its query architecture, and OML does not include a comparable mechanism, relying instead on its SQL database storage substrate to provide the experimenter with a query interface to her data.

One could also compare OML to network adminstration monitoring tools such as SNMP (covered by numerous IETF RFC's, starting with RFC 1155, 1157, for instance). SNMP has a high overhead compared to OML. Monitoring in SNMP is based around OID's—object identifiers—that identify measurement items of interest and are essentially numeric and not human-readable. A central management information base (MIB) must be maintained to map OID's to human-readable strings. This is at odds with the needs of research, which is by nature dynamic and often not centralized enough to permit the maintenance of an MIB, which also adds unnecessary cost. In OML, a user wanting to measure a new quantity simply defines a new measurement point in the relevant client application and configures the filters for his experiment run to filter it into the database. There is no central organization needed. From our survey, there do not appear to be suitable open source implementations that could be easily adapted to the needs of research.

Of all the measurement architectures we surveyed, MINER [1,5] appears to be the closest to our architecture. MINER is not available as open source software, but [5] describes its architecture. MINER is Java-based and comprises a measurement architecture as well as elements of what we refer to as the management framework. A *client library* provides an API for defining and running experiments, which consist of invocations of *tools*. A *core* component is the server component of the infrastructure and the mediator between the client library and the measured network. A *tool proxy* component acts as a mediator between the core component and the MINER tools. A tool proxy executes a scenario request on a network node, starts the requested MINER tools, and then grooms the measurement results back to the core.

The MINER tools are Java components that may provide measurement results directly, or may be wrappers around external libraries or applications that do the actual measurements. MINER tools can be defined by the user.

Our management framework (OMF [16]) is decoupled from the measurement aspect of experimentation. This makes each component more generic and flexible. The MINER approach of providing a wrapper interface for existing tools is a great idea. The main method to instrument existing applications with OML is to directly modify their source code (e.g., `iperf` in Section 3.4). When these sources are not available, it is easy to develop a short program to process the application's outputs and collect the resulting measurements using OML.

Emulab [19] is a large network emulator based on a set of computers that can be configured into various topologies through emulated network links. Many experimenters currently use Emulab testbeds to evaluate their research schemes. It allows them to monitor and capture network traffic (packet headers or full payload) on links and LANs within their experimental topologies. The capture points, equivalent to OML measurement points, are either on the resource that emulates a link, or on end-point resources. In both cases, the captured data are stored as a local file on that resource. To analyse the experimental results, the user has to retrieve the resulting file from all the used resources at the end of an experiment. This simple scheme is limited to the measurement of only network traffic, and does not allow the monitoring of any experiment's contextual variables (e.g., node location) or application integrated data (e.g., download/upload statistics for a peer-to-peer application).

PlanetLab [15] is a global research platform based on more than 1000 distributed computers, which are hosted by independent organisations. It is the primary large-scale testbed used for experimental overlay and service oriented systems (e.g., distributed storage, peer-to-peer content distribution). Multiple services are currently deployed on Planetlab, which provide users with measurements of their experimental slices and the whole testbed, such as CoMon [13], or PlanetFlow [10]. CoMon provides different statistics (e.g., memory, disk usage) at a node or a slice granularity. However, it does not support collection of application or experiment generated measurements. CoMon uses a client/server design like the basic OML architecture. The processed measurements are made available to the entire experimenter community via a distributed content delivery system. PlanetFlow also uses a client/server scheme. On each node, a client entity captures all outgoing packet headers, then aggregates and classifies them into flows. This process is akin to the OML client filtering. However PlanetFlow does not provide any other client-side flow processing. These flow measurements are centrally collected in a MySQL database accessible via a Web interface.

6 Conclusions

In this article we presented extensions to the versatile measurement library OML. The new features that we presented allow the experimenter to extend the range of possible measurements. In particular, we detailed the transparent integration

of a proxy server on the experiment node allowing measurements in a disconnected environment. This solution has been made possible by the addition of a measurement proxy server on the mobile node within the existing measurement framework. The first goal of this proxy is to buffer the measurement stream without losing any information. We identified two main fields of application for this measurement feature, a disconnected experiment and a shared control and experiment network. We demonstrate the benefit of this new feature in the context of a simple disconnected experiment during the 4^{th} GEC and presented the results in this article. Finally we extended this architecture with hierarchical measurement collection and server-side filtering, which allows us greater control over the measurement collection process, and with a feedback mechanism that allows us to steer both experiments and the measurement process itself while the experiment is running, based on the current measured context.

Acknowledgements

This work was achieved in the context of the Onelab2 projects funded by the E.U. 7^{th} Framework Program, and the GENI (Global Environment for Network Innovations) initiative funded by the U.S. National Science Foundation.

References

1. MINER: The Measurement Infrastructure for Network Research, `http://miner.salzburgresearch.at/index.php`
2. NLANR/DAST: Iperf - the TCP/UDP bandwidth measurement tool, `http://dast.nlanr.net/Projects/Iperf/`
3. OML: The OMF Measurement Library, `http://omf.mytestbed.net/projects/show/oml`
4. The 4th GENI engineering conference (March 2009)
5. Brandauer, C., Fichtel, T.: MINER – a measurement infrastructure for network research. In: International Conference on Testbeds and Research Infrastructures for the Development of Networks & Communities, pp. 1–9. IEEE Computer Society, Los Alamitos (2009)
6. Claise, B.: Specification of the IP Flow Information Export (IPFIX) Protocol for the Exchange of IP Traffic Flow Information. RFC 5101 (Proposed Standard) (January 2008)
7. Franklin, M.J., Krishnamurthy, S., Conway, N., Li, A., Russakovsky, A., Thombre, N.: Continuous analytics: Rethinking query processing in a network-effect world. In: Fourth Biennial Conference on Innovative Data Systems Research (CIDR 2009), Asilomar, CA (January 2009)
8. Google. Google maps, `http://maps.google.com/`
9. Hoh, B., Gruteser, M., Herring, R., Ban, J., Work, D., Herrera, J.-C., Bayen, A., Annavaram, M., Jaconbson, Q.: Virtual trip lines for distributed privacy-preserving traffic monitoring. In: ACM MobiSys (2008)
10. Huang, M., Bavier, A., Peterson, L.: PlanetFlow: Maintaining Accountability for Network Services. Operating Systems Review 40(1) (2006)

11. Ianaccone, G., Diot, C., McAuley, D., Moore, A., Pratt, I., Rizzo, L.: The CoMo white paper. Technical Report IRC-TR-04-17, Intel Research Cambridge (September 2004)
12. iANNACCONE, G.: CoMo: An open infrastructure for network monitoring—research agenda. Technical report, Intel Research Cambridge (February 2005)
13. Park, K., Pai, V.S.: CoMon: A Mostly-Scalable Monitoring System for Planet-Lab. ACM SIGOPS Operating Systems Review (2006)
14. Petander, H.: Energy aware network selection using traffic estimation. In: Proc. of MITCN 2009 Workshop in ACM Mobicom Conference (September 2009)
15. PlanetLab Consortium. Planetlab: An open platform for developing, deploying, and accessing planetary-scale services, `http://www.planet-lab.org/`
16. Rakotoarivelo, T., Ott, M., Seskar, I., Jourjon, G.: OMF: a control and management framework for networking testbeds. In: SOSP Workshop on Real Overlays and Distributed Systems (ROADS 2009), Big Sky, USA, p. 6 (October 2009)
17. Raychaudhuri, D., et al.: Overview of the ORBIT Radio Grid Testbed for Evaluation of Next-Generation Wireless Network Protocols. In: Proc. IEEE Wireless Communications and Networking Conference, WCNC (2005)
18. Oetiker, T.: RRDtool, `http://oss.oetiker.ch/rrdtool/`
19. White, B., Lepreau, J., Stoller, L., Ricci, R., Guruprasad, S., Newbold, M., Hibler, M., Barb, C., Joglekar, A.: An integrated experimental environment for distributed systems and networks. SIGOPS Oper. Syst. Rev. 36(SI), 255–270 (2002)

AiroLAB: Leveraging on Virtualization to Introduce Controlled Experimentation in Operational Multi-hop Wireless Networks

Roberto Doriguzzi Corin, Roberto Riggio, Daniele Miorandi, and Elio Salvadori

CREATE-NET, via alla Cascata 56/D, IT - 38100, Povo, Trento, Italy

Abstract. Network Virtualization represents one of the most promising approach to unlock innovation in current network technologies. This paper presents *AiroLAB*, a wireless network virtualization solution aimed at providing Wireless Internet Service Providers with an effective tool to support experimental testing of novel network–level solutions on *production* networks. In the paper, the design choices which lie at the basis of *AiroLAB* are presented and discussed together with a prototype implementation. The outcomes of measurement activities performed on a small–scale testbed demonstrate the strength of the proposed framework in preserving guaranteed performance for production traffic while allowing several experimental instances to run on a network in operation.

Keywords: network virtualization, multi–hop wireless networks, embedded devices, resource constrained environment.

1 Introduction

One of the most promising approaches to enable innovation in today's network is Network Virtualization (NV) [1,2]. In general terms, NV refers to the possibility of pooling together low–level hardware and software resources belonging to a networked system into a single administrative entity. In such a way network resources could be effectively shared in a transparent way among different logical network instances. NV differs from "conventional" virtualization techniques (aimed mostly at virtualizing computing and storage resources) in that it explicitly addresses low–level resources (e.g., bandwidth), well below the IP waist. From an academic standpoint, NV can be seen as a tool for evaluating novel Internet architectures ("clean–slate approaches") in large–scale realistic environments. Similarly, from a business perspective, NV can change the functional role of Internet Service Providers (ISPs) by decoupling the provisioning of the physical infrastructure from the provisioning of communication/computing resources. This could set the basis for the introduction of new business models and stakeholders in the Internet ecosystems (i.e. Infrastructure Providers, Virtual Network Providers and Service Providers). Finally, NV can enable a smooth and controlled introduction of novel services in an operational network by providing means to isolate them from already deployed applications, thereby unlocking innovation in telecommunication networks.

T. Magedanz et al. (Eds.): TridentCom 2010, LNICST 46, pp. 331–344, 2011.

While several NV architectures and solutions have been proposed in recent years, most (if not all) of them were designed and developed for wired networks, characterized by virtually unlimited processing/storage power and link bandwidth (PlanetLab [3], VINI [4], etc). On the other hand, a particularly challenging, yet interesting, domain for NV is that of wireless multi–hop networks [5,6] with a particular emphasis on Wireless Mesh Networks (WMNs). WMNs are a cost–effective access networking paradigm, which represents an interesting solution in scenarios where a fixed wired infrastructure is either not feasible (e.g., in the case of mobile nodes such as vehicular ad hoc networks) or not economically attractive (e.g., access to Internet in developing countries). However, despite these expectations, very few studies have been performed, so far, on virtualization in resource–constrained environments in general, and multi–hop wireless networks in particular. Furthermore, the available literature on the theme focuses mainly on comparing how different wireless medium virtualization techniques affect the overall network slices performance in term of isolation and stability [7,8].

The aim of this paper is to introduce *AiroLAB*, a novel network virtualization framework specifically tailored to multi–hop wireless networks [9]. AiroLAB was designed to provide Wireless Internet service providers (WISPs) with an effective virtualization solution. In *AiroLAB*, an innovative mechanism to assess wireless link capacity and realize soft–performance isolation among virtual networks instances allows production traffic to share part of the available network resources with a variable number of network slices, where novel solutions, such as new routing protocols, services or network operation tools, can be experimentally tested in a severely controlled yet realistic environment with no impact on the operational traffic. Compared to [9], which is mainly focused on the description of the objectives and constraints that have driven the design of our Network Virtualization framework, in this paper we provide a more accurate analysis of the experimental results obtained in a small–scale wireless testbed.

The paper is organized as follows: Sec. 2 provides an overview of the main challenges behind Network Virtualization, emphasizing how *AiroLAB* differentiates from solutions proposed in literature. Section 3 describes the *AiroLAB* architecture and protocols. Section 4 presents the results of experimental tests carried out with a prototypical implementation of *AiroLAB* while Sec. 5 concludes the paper.

2 Network Virtualization: An Overview

Network Virtualization research main challenge refers to the definition of appropriate and efficient algorithms, architectures and protocols to effectively share a common physical network infrastructure, splitting it into several logical instances (generally referred to as "slices") composed of virtual links and virtual network nodes [10]. Network nodes should be fully programmable to allow the instantiation of several network instances, each one potentially based on a different architecture. Several projects worldwide are working on the various aspects

underpinning NV: GENI in USA [11], 4WARD [12], FEDERICA [13] in Europe and AKARI in Japan [14].

An effective NV solution shall satisfy several requirements [15]: scalability (performance should not depend on the number of slices), isolation (among slices), flexibility (fully programmable network elements), efficiency (limited overhead due to virtualization), manageability (it should work in multi-domain scenarios) and heterogeneity (in term of underlying technologies, end-users,...). When network virtualization solutions are used for running concurrent research experiments in dedicated testbeds, two further requirements are lack of inter–experiments interference and experiments repeatability.

Most of the research groups proposing NV mechanisms in wireless networks have been focusing in large–scale testbed scenarios, i.e. ORBIT [16] or GENI [11], where several experiments could run concurrently on the same physical infrastructure. Due to their strong focus on creating a stable testing environment where experiments should be repeatable and should not affect each other when running on concurrent slices, most of the works have focused on evaluating wireless virtualization techniques to realize performance isolation. This has been obtained by exploring both dimensions of (i) virtualization of the wireless medium (through multiplexing techniques like SDM, FDM, CDM or TDM [17]) and (ii) virtualization of the network node. Studies regarding the feasibility of each of these approaches have been already provided in literature with an analysis of their pros and cons [7,8,18,19].

Compared to those works, the scenario addressed in this paper is a sort of intermediate one between such purely research approaches applied to *dedicated testbed* network infrastructures, and a conservative approach (pursued so far by most of the operators) where novel services or recent protocols are tested on a small-scale testbed separated from the main production network. In fact, *AiroLAB* investigates techniques and architectures to provide a NV framework specifically tailored for *production networks*, where operational traffic is fully guaranteed over a "privileged" slice, while all novel (experimental) services and protocols under test are run on "background" (lower–priority) slices. Our aim is somehow similar to the ones pursued by solutions such as Cabernet [20,21] and generalized in [22] on carrier-class networking equipment; however, given the specific technological domain under consideration, we are providing an emphasis toward the possibility for a Wireless ISP (WISP) to perform experimental activities in a controlled fashion directly on its production network.

3 The Architectural Framework of *AiroLAB*

Multi–hop wireless networks are usually built using commodity components and are characterized by rather limited computing capabilities. Such a scenario calls for a radically new approach to network virtualization, where the trade–off between flexibility and scalability shall, due to pragmatical consideration, drift towards the latter. Before analyzing the intricacies behind the proposed NV architecture, a simplified network scenario is described in this Section to emphasize objectives and constraints that have driven the *AiroLAB*'s design.

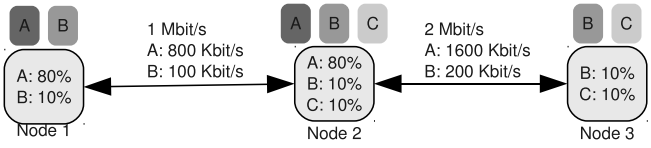


Fig. 1. Simplified deployment scenario

Fig. 1 sketches a simplified setup where a network, composed of three nodes organized in a string topology, is running three distinct *slices*: one production slice (A), and two experimental slices (B and C). In this scenario, links are symmetric and their capacity is assumed to be time–invariant. Moreover, mesh routers are equipped with a single radio interface.

In this simplified scenario, the production slice A is assigned 80% of the resources in the network, while the two experimental slices equally share the remaining 20% of resources. It is worth noticing that, with our architecture, we do not aim at supporting hundreds or even tens of concurrent slices, instead we foresee a scenario where 5 to 10 slices share the overall network resources. Such limitation is mandated by the computing and storage constraints that characterized currently used wireless multi–hop networking devices.

Traffic shaping is performed at each node in order to limit the amount of network resources used by each *sliver*. In this simplified setup the resources that each sliver can exploit are upper bounded by a fixed threshold derived from the relative performance goal given during the planning phase. As a result, slice A "sees" an 800 Kb/s bidirectional link between node 1 and node 2, while the available bandwidth between node 2 and node 3 is 1600 Kb/s. In this setup some bandwidth is voluntary left unused. However scenarios where a sliver can have full access to all the available bandwidth are also supported.

3.1 Assessing Wireless Link Capacity

Estimating the capacity of a wireless link is not trivial due to the use of a shared medium; in fact interference coming from external sources, changes in the propagation characteristics or interference from the same signal traveling along different paths make the link's total capacity fluctuate over time. Even if we limit our attention on communications realized using the IEEE 802.11 facility of standards, an ideal estimator of the link capacity from an Access Point toward a generic Station should take into account both the the data frame SNR (measured at the receiving station) and the ACK frame SNR (measured at the access point). Such a level of precision is difficult to achieve without introducing additional signaling and/or modifying the standard IEEE 802.11 MAC operations.

In this work we decided to use an indirect way of assessing a link's total capacity based on the rate adaption functionalities already available in current IEEE 802.11 devices. Rate adaptation algorithm aims at dynamically selecting

the transmission rate in order to achieve optimal performance under varying operating conditions. Rate adaptation is left unspecified by the IEEE 802.11 standard, as a result of the years a considerable number of solutions have been proposed by both the academic and the industrial worlds.

Our work builds on top of currently available rate–control algorithms for IEEE 802.11–based wireless networks. In particular we exploit the Minstrel [23] algorithm. The minstrel rate–control algorithm aims at selecting the transmission rate that maximizes the throughput. In order to so, the algorithm collects statistics of all the packets that have been transmitted. This data is then exploited to compute the probability of a successful transmission P_{ab} between a pair of nodes, a and b, for each available data–rate. In order to cope with environmental changes, minstrel uses an Exponential Weighted Moving Average (EWMA) based approach. EWMA has a smoothing effect, so that new results have a larger influence on the selected rate. Finally, if D_{tx} is the time spent for a single transmission, and B is the packet length, the empirical throughput T_{ab} that is used by *AiroLAB* as an estimation of the wireless link capacity is computed as follows:

$$T_{ab} = \frac{P_{ab}B}{D_{tx}}, \tag{1}$$

3.2 Providing Soft–Performance Isolation

AiroLAB provides soft–performance isolation between slivers by leveraging on the Hierarchical Token Bucket (HTB) traffic shaping facilities provided by the Linux kernels 2.6.x. HTB organizes traffic classes in a tree structure; each class is assigned an average rate (*rate*) and a maximum rate (*ceil*). Three class types exist: root, inner and leaf. A root class corresponds to a physical link; its bandwidth is the one currently available for transmission. Leaf classes, placed at the bottom of the hierarchy, correspond to a given type of traffic (e.g., TCP-controlled or VoIP etc.). Two internal token buckets are maintained for each class. Classes which have not exceeded their rate can unconditionally transmit; classes which have exceeded their allowed rate but not their upper limit (ceil) can transmit only borrowing unused bandwidth, if available, from other classes. In order to borrow bandwidth, a request is propagated upwards in the tree. A request that would exceed the ceil limit is terminated. A request that would satisfy the allowed rate is accepted. A request that would not satisfy the allowed

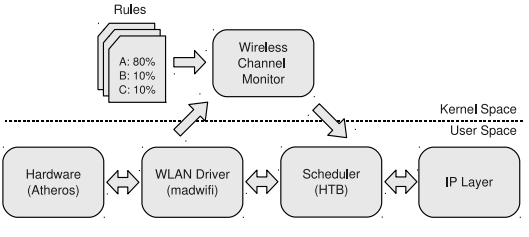


Fig. 2. *AiroLAB* wireless channel monitor architecture

rate constraint but the ceil one is propagated upwards until the procedure is completed.

Due to the stochastic nature of the wireless links capacity, an HTB scheduler alone is not able to deliver performance fairness among competing traffic flows in wireless networks. In order to address such an issues we devised and implemented a wireless channel monitor which exploits the channel statistics computed by the wireless driver in order to properly distribute the available bandwidth among the slivers running in a node. Figure 2, sketches the the architecture of the *AiroLAB* wireless channel monitor. The overall link capacity T_{ab} is assigned to the HTB's root class, while each sliver is associated to a leaf class in the HTB hierarchy. Available bandwidth is distributed among the slivers according to a set of input *policies*. The wireless channel monitor is implemented in the form of a software process running within each wireless router and periodically updates the HTB's configuration in order to reflect the actual channel capacity. HTB's configuration is also updated if either a new slice is deployed over the network or if the *policies* have changed.

3.3 A Description of the Node–Level Architecture

The *AiroLAB* framework design, whose architecture is sketched in Fig. 3, has been centered around two open source tools based on GNU/Linux operating

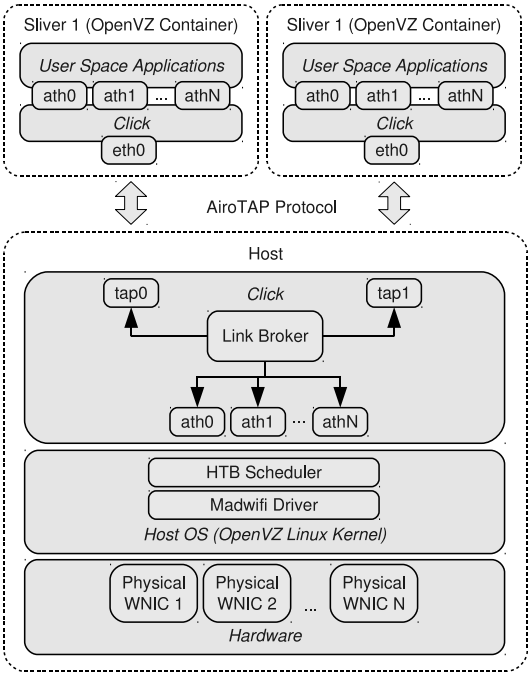


Fig. 3. *AiroLAB* node–level architecture

system: OpenVZ [24] and Click [25]. OpenVZ consists of a modified Linux kernel tree that supports virtualization, isolation and resource management and a set of user–level tools that allows the installation, configuration and maintenance of the virtual environments (also known as *containers)*. Container–based virtualization solutions are typically characterized by reduced overhead and thus better performance. They also provide good performance isolation (in terms of CPU cycles, memory consumption, and storage), because processes running within a container do not significantly differ from processes running in the hosting system. Thus, it is possible to apply existing resources sharing techniques, such as HTB for traffic scheduling. The major drawback of container-based virtualization solutions is that, since a single kernel is used for every sliver, kernel modifications are not allowed.

Within OpenVZ, each Virtual Environment (VE) performs and executes exactly like a stand–alone host; a container can be rebooted independently and can have root access, users, IP addresses, memory, processes, files, applications, system libraries and configuration files. Moreover, OpenVZ provides a resource management system that controls the amount of resources available for the environments. The controlled resources include parameters such as CPU power, disk space, and set of memory-related parameters. Furthermore, unlike alternative container-based solutions such as Linux-VServer [26], OpenVZ provides full virtualization of the networking subsystem allowing each virtual environment to create its own internal routing or firewall setups.

Due to the limitations imposed by the use of OpenVZ, namely the impossibility to run customized kernel images in different slivers, we decided to implement our virtualization stack in user–space using the Click modular router [25]. Our approach is not meant to replace OpenVZ, but rather to extend it in order to support flexible virtualization of the wireless resources. In fact, albeit characterized by an higher overhead in comparison to pure kernel–level implementation, Click–based solutions are highly customizable allowing us to circumvent the flexibility limitations of typical container based solutions [27]. Table 1 summarizes the trade–offs involved in the most relevant virtualization techniques currently available, namely containers, hypervisor, and hosted VMM. *AiroLAB* belongs to the second columns (Containers w/ Click) in that on the one hand container–based virtualization is used to achieve performances and scalability, and, on the

Table 1. Taxonomy of network virtualization techniques and relevant features [27]

	Containers	**Containers w/ Click**	**Hypervisors**	**Hosted VMM**
Scalability	*Good*	*Good*	*n.a.*	*n.a.*
Fault/Security Isolation	*n.a.*	*n.a.*	*Good*	*Good*
Performance Isolation	*Good*	*Good*	*Good*	*Good*
Flexibility	*Poor*	*Good*	*Good*	*Good*
Code Re–Usability	*n.a.*	*Poor*	*Poor*	*Good*
Efficiency	*Good*	*Good*	*Good*	*n.a.*

other hand, user–space wireless network virtualization delivers high flexibility in terms of packet processing capabilities.

Click is used both within each sliver (*guest click*) and at the host operating system level (*host click*). More specifically, the Click instance running within a sliver provides the guest environment with a set of virtual interfaces (*ath0, ath1, . . . , athN*) implemented as Linux TAP devices. A TAP device operates at layer 2 of the traditional ISO/OSI networking stack and simulates an Ethernet device. User-space process, running within a sliver, can exploit the virtual interfaces to implement their routing strategy. Communication over the virtual interfaces can be done using three different frame formats:

- 802.3 headers (Ethernet). Used to expose a standard Ethernet interface.
- 802.11 headers (WiFi). Used to expose a wireless interface complaint with the IEEE 802.11 protocol. In this case the user–space applications must properly encapsulate their traffic in 802.11 frames.
- Radiotap. Used to expose a raw wireless interface. In this case the user–space applications must properly encapsulate their traffic using the radiotap [28] header format. The radiotap header format is a mechanism to supply additional information about 802.11 frames, from the driver to user–space applications, and from a user–space application to the driver for transmission.

In either situation, outgoing traffic is encapsulated by the *guest click* process and sent to the *host click* process through the virtual interface *eth0* provided by the OpenVZ Container. Please note that, if the user-space application is already using the radiotap header, no additional encapsulation is performed by the *guest click* process and the frame is delivered unchanged to the host operating system. The *host click* process receives the incoming frame and dispatches it to the suitable device according to a set of policies maintained by the *Link Broker*.

The *Link Broker* is a software module that can expose different connectivity graphs to the various slivers without requiring that the nodes must be physically separated (i.e., out of radio range). Connectivity graphs are defined on a per-slice basis allowing us to define a different topology for each slice. This is particularly useful to test novel routing strategies on a subset of the nodes. Moreover, if wireless routers are equipped with multiple radio interfaces, it is possible to create multiple slices (whose cardinality equals the number of radio interfaces) operating on orthogonal frequency bands, implementing therefore an FDM wireless network virtualization solution. Hybrid solutions where only a subset of the slivers operates on orthogonal frequencies are also supported. Albeit network connectivity graphs are defined at deployment time, they can change during the network operations in order to create connectivity scenarios that simulates different operating conditions (i.e. link failures/outages).

3.4 An Example of Network–Level Configuration

A possible use case of *AiroLAB* is sketched in Figure 4, where a production slice exploiting a legacy version of a routing protocol is running in parallel with

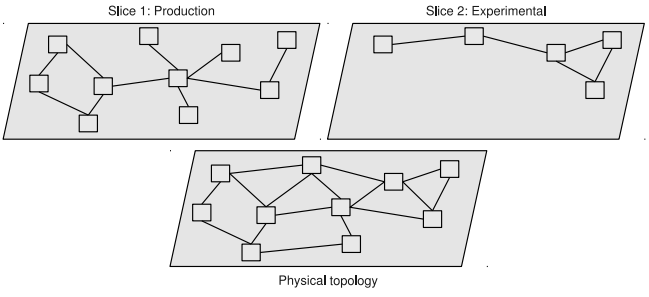


Fig. 4. Network–level configuration: an example with one production slice and one experimental slice sharing a common physical substrate

an experimental slice where novel routing strategies are being tested. In this scenario the *Link Broker* is used to expose two different connectivity graphs to the production and the experimental slices. On the other hand, the *Wireless Channel Monitor* is used to redistribute the available link bandwidth among the competing slices, 80% to the production slices and 20% to the experimental slices in this cases. Please note that, a minimum bandwidth, e.g. 1 Mb/s, can also be allocated to the production slice.

4 Results of the Experimental Activities

The main objective of the experimental measurements described in this Section is to prove the effectiveness of the *AiroLAB* framework in preventing traffic on a privileged slice being affected by traffic from other (lower–priority) slices, therefore guaranteeing a peaceful coexistence between operational and experimental traffic in a production network.

The wireless routers employed in the experimental set-up are built exploiting the PCEngines ALIX 2C2 (500MHz x86 CPU, 256MB of RAM) processor board. Operating system and application are stored on a 1 GB Compact Flash. Connectivity is provided by 2 Ethernet channels, 2 miniPCI slots and one serial port. PCEngines ALIX boards are equipped with two Mikrotik R52 WiFi IEEE 802.11a/b/g cards based on the Atheros AR2412 chipset. OpenWRT [29] has been selected as Operating system for our testbed, even though its original kernel has been replaced with a kernel provided by OpenVZ. The software configuration of the wireless routers is summarized in Table 2.

Table 2. AiroLAB wireless routers setup

Operating System	OpenWRT trunk (release 14748)
Linux kernel	OpenVZ 2.6.18-028stab056
Wireless drivers	MadWiFi trunk (release 2568)
Virtualization tools	vzctl-3.0.23, vzquota-3.0.12

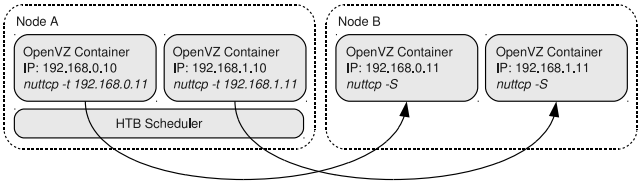


Fig. 5. Representation of the packet scheduling process for the case with two slivers

Several experimental scenarios have been set up to demonstrate *AiroLAB* performance isolation capabilities. The HTB configuration exploited during the measurements campaigns creates a traffic class for each sliver. For each traffic class, we specify the minimum (*rate*) and the maximum (*ceil*) throughput. Figure 5 shows the node setup for the case of two slivers. The performance metrics considered in each experimental scenario are throughput and delay. The results have been obtained by averaging the samples obtained as *nuttcp* benchmarks over 300 seconds with an averaging interval of 10 seconds.

In the first scenario, we use two wireless nodes, each one running two concurrent slivers sharing the same wireless interface. The privileged slice (#1) has higher transmission priority and a minimum guaranteed outbound bandwidth set to 10 Mb/s, and it provides an offered load of 10 Mb/s. The second slice (#2) has no minimum guaranteed outbound bandwidth, and it generates traffic off and on periodically with varying loads. The graph plotted in Fig. 6 shows the throughput and delay distribution per slice. As expected, when the wireless link is not saturated (from 0 to 140 secs), *AiroLAB* correctly limits the impact of slice (#2) on the privileged slice, by guaranteeing a stable throughput and averaged delays always below 10 msec. Of course, as soon as the offered load on Slice #2 leads to link saturation (from 140 to 160 secs), the throughput of Slice #1 is slightly affected as well as its averaged delay which increases up to 20 msec. However, compared to a similar test presented in [18], *AiroLAB* doesn't show any "drop to zero" effect when the second slice starts to carry some traffic, thus

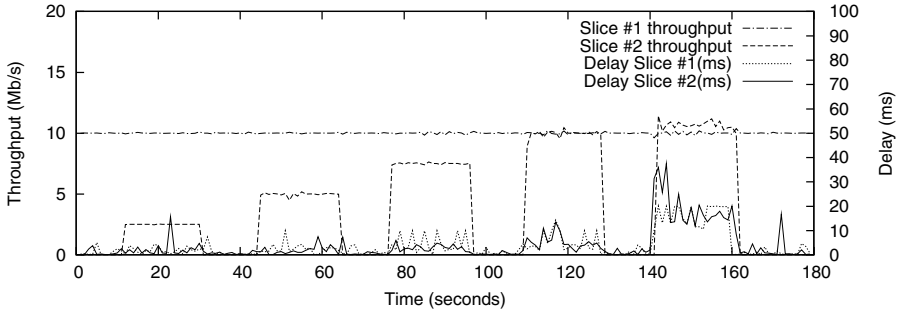


Fig. 6. Analysis of the cross–coupling effect among a privileged slice (#1) and an experimental one (#2) in term of throughput and delay

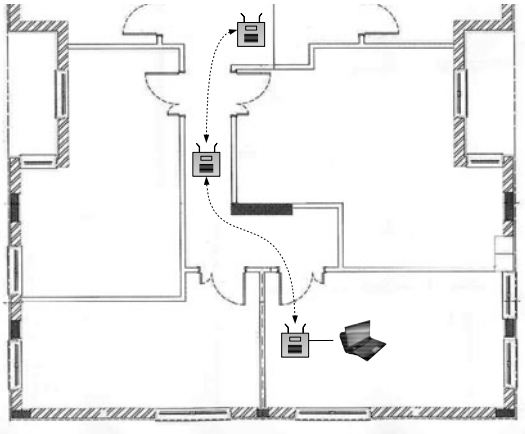
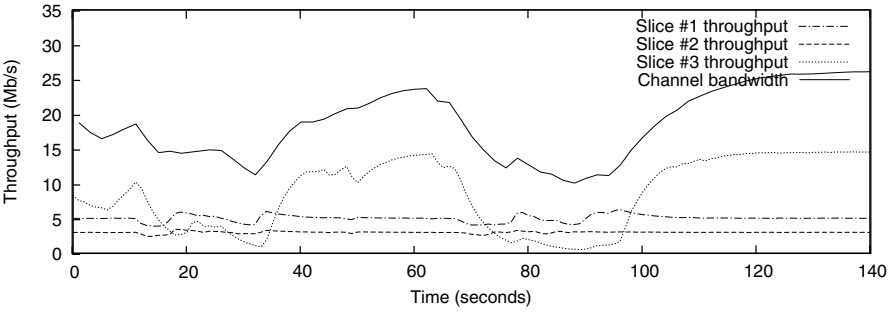


Fig. 7. The testing setup involved 2 nodes deployed in a typical office environment

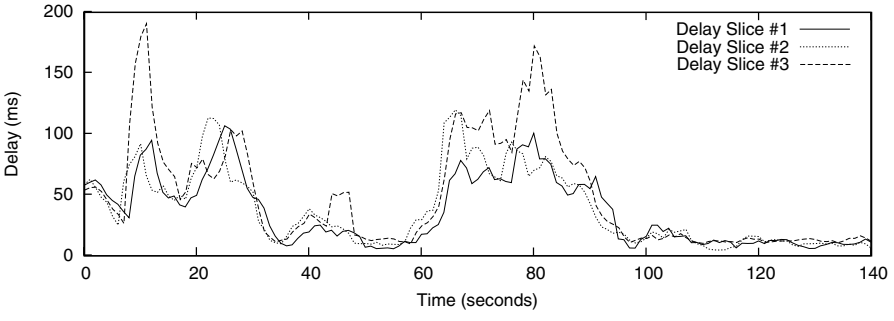
showing a more stable environment. It is worth noticing that no CPU reservation policies provided by OpenVZ have been used on the "priviliged" slice: such operation would have further limited the impact on the average delay for this slice, mainly due to some buffer processing delay on each physical node.

In the second scenario, we want to test the ability of the proposed architecture to effectively preserve production traffic in challenging conditions. To this purpose, the experimental setting sketched in Fig. 7 has been set up. We considered two wireless nodes, each one running three slivers sharing the same wireless interface. The experimental setup includes a wireless node connected to a PC lying on a desk in Office 1. Changes in link quality are emulated by moving the second node from Office 1 to another room. A continuous UDP flow is generated among the two nodes; its rate is such that the wireless link is *always* saturated.

In this scenario, there are two privileged slices (#1 and #2) with higher transmission priority and a minimum guaranteed outbound bandwidth set to 5 and 3 Mb/s respectively, while the remaining slice (#3) has no guaranteed bandwidth (one can suppose a WISP having slice #1 for production traffic and the remaining slices #2 and #3 for, respectively, testing a novel video–streaming service and for network management and monitoring).Moreover, Slice #1 and #2 provide an offered load of 5 and 3 Mb/s, while Slice #3 has no upper bounds on the maximum throughput it can inject in the wireless link. The results plotted in Fig. 8 show the throughput and delay distribution per–slice in different conditions of available wireless link capacity. As expected, *AiroLAB* guarantees that the throughputs of Slice #1 and #2 are only slightly affected by wireless link conditions to detriment of Slice #3. Results related to throughput measurements are summarized in Tab. 3. The impact on the average delay per slice is higher mainly due to the saturation conditions of the experimental scenario. However, it is worth noticing that an average delay lower than 150 msec is tolerable for video–streaming–based services; while network-management traffic can tolerate even higher delays [30]. As per the previous test scenario, the impact



(a) Throughput



(b) Delay

Fig. 8. Relative performance for the three slices in the second scenario

Table 3. Throughput statistics

	Slice #1	**Slice #2**	**Slice #3**
Average	5	3	8,85
Minimum	3,85	2,42	0,54
Maximum	6,27	3,46	14,6
Std Deviation	0,47	0,14	5,1
Confidence interval (95%)	±0,08	±0,02	±0,82

on the average delay for the slice running the production traffic could have been further minimized through an appropriate configuration of the CPU reservation via OpenVZ.

5 Conclusions

In this paper, we have presented *AiroLAB*, a novel virtualization framework specifically tailored to multi–hop wireless networks. *AiroLAB* has been designed with the explicit goal to empower WISP with an effective tool to allow production traffic to safely share part of the available network resources with a

variable number of network slices where novel solutions, such as new routing protocols, services or network operation tools, can be experimentally tested in a severely controlled yet realistic environment. The architecture and protocols at the hearth of *AiroLAB* have been presented, discussed and compared with existing solutions. A first prototypical implementation of *AiroLAB* capable of supporting performance isolation between concurrent slices is described, together with experimental measurements obtained in a small–scale wireless testbed.

Despite the encouraging results presented in the paper, the proposed framework requires further development before reaching the stability level needed to enable its wide adoption. Among the possible research directions to enhance the current architecture, we believe that the ability of leveraging, by means of appropriate FDM approaches, the presence of multiple wireless interfaces (such that, for example, different slices could associate to dedicated radio channels), could definitely improve the efficiency and scalability of *AiroLAB* in realistic scenarios. Moreover, integration with currently available frameworks for automatic NV resources allocations, such as OMF [31], shall be cosidered for future evolution of the framework.

References

1. Feldmann, A., Kind, M., Maennel, O., Schaffrath, G., Wehrle, C.: Network Virtualization An Enabler for Overcoming Ossification. ERCIM News 77, 21–22 (2009)
2. Papadimitriou, P., Maennel, O., Greenhalgh, A., Feldmann, A., Mathy, L.: Implementing Network Virtualization for a Future Internet. In: Proc. of 20th ITC Specialist Seminar on Network Virtualization, Hoi An, Vietnam (2009)
3. Planet Lab project, `http://www.planet-lab.org`
4. VINI project, `http://www.vini-veritas.net`
5. Bruno, R., Conti, M., Gregori, E.: Mesh Networks: Commodity Multihop Ad Hoc Networks. IEEE Communications Magazine 43(3), 123–131 (2005)
6. Akyildiz, I., Wang, X., Wang, W.: Wireless mesh networks: a survey. Elsevier Computer Networks 47(4), 445–487 (2005)
7. Smith, G., Chaturvedi, A., Mishra, A., Banerjee, S.: 'Wireless Virtualization on Commodity 802.11 Hardware. In: Proc. of ACM WinTECH, Montreal, Quebec, Canada (2007)
8. Mahindra, R., Bhanage, G., Hadjichristo, G., Seskar, I., Raychaudhuri, D., Zhang, Y.: Space Versus Time Separation for wireless virtualization On an Indoor Grid. In: Proc. of EURO NGI, Krakow, Poland (2008)
9. Doriguzzi Corin, R., Riggio, R., Miorandi, D., Salvadori, E.: AiroLAB: A Framework Toward Effective Virtualization of Multi-hop Wireless Networks. International Journal of Communication Networks and Distributed Systems (2010) (accepted for publication)
10. Evans, J., Raychaudhuri, D., Paul, S.: Technical Document on Overview Wireless, Mobile and Sensor Networks. The GENI Project Office, Tech. Rep. GDD- 06-14 (2006)
11. GENI project, `http://www.geni.net`
12. 4WARD project, `http://www.4ward-project.eu`
13. FEDERICA project, `http://www.fp7-federica.eu`
14. AKARI project, `http://akari-project.nict.go.jp`

15. Chowdhury, N.M.M.K., Boutaba, R.: Network Virtualization: State of the Art and Research Challenges. IEEE Communications Magazine (July 2009)
16. Orbit Lab, http://www.orbit-lab.org/
17. Paul, S., Seshan, S.: Technical Document on Wireless Virtualization. The GENI Project Office, Tech. Rep. GDD-06-17 (2006)
18. Singhal, S., Hadjichristo, G., Seskar, I., Raychaudhuri, D.: Evaluation of UML based wireless network virtualization. In: Proc. of EURO NGI, Krakow, Poland (2008)
19. Bhanage, G., Seskar, I., Zhang, Y., Raychaudhuri, D.: Evaluation of OpenVZ for wireless testbed virtualization. WINLAB Rutgers University, Tech. Rep. 331 (2008)
20. Feamster, N., Gao, L., Rexford, J.: How to lease the Internet in your spare time. ACM SIGCOMM Computer Communications Review, 61–64 (January 2007)
21. Zhu, Y., Zhang-Shen, R., Rangarajan, S., Rexford, J.: Cabernet: Connectivity architecture for better network services. In: Proc. of Workshop on Rearchitecting the Internet (2008)
22. Schaffrath, G., Werle, C., Papadimitriou, P., Feldmann, A., Bless, R., Greenhalgh, A., Wundsam, A., Kind, M., Maennel, O., Mathy, L.: Network Virtualization Architecture: Proposal and Initial Prototype. In: Proc. of ACM SIGCOMM Workshop on Virtualized Infastructure Systems and Architectures, Madrid, Spain (2009)
23. Minstrel, http://linuxwireless.org/
24. OpenVZ, http://openvz.org/
25. The click modular router project, http://read.cs.ucla.edu/click/
26. Linux-VServer, http://Linux-VServer.org/
27. Nakao, A., Ozaki, R., Nishida, Y.: Corelab: An emerging network testbed employing hosted virtual machine monitor. In: Proc. of ACM ROADS, Madrid, Spain (2008)
28. Linux Radiotap, http://www.radiotap.org/
29. OpenWRT Linux Distribution, http://openwrt.org/
30. Szigeti, T., Hattingh, C.: End-to-End QoS Network Design: Quality of Service in LANs, WANs, and VPNs. Cisco Press (2004)
31. OMF, Control and Management Framework, http://omf.mytestbed.net

TridentCom 2010

Full Papers Session 4: Management, Provisioning and Tools for Future Network Testbeds

Experimental Validation and Assessment of Multi-domain and Multi-layer Path Computation

Sebastian Gunreben[1], Ramon Casellas[2], Ricardo Martinez[2], Raul Muñoz[2], and Joachim Scharf[1]

[1] Universität Stuttgart - IKR, Pfaffenwaldring 47, 70569 Stuttgart, Germany
gunreben@ikr.uni-stuttgart.de
[2] Optical Networking Area, Centre Tecnològic de Telecomunicacions de Catalunya, CTTC, Av. Canal Olímpic s/n, 08860 Castelldefels (Barcelona), Spain

Abstract. Within the framework of the BONE European Network of Excellence, we setup a multi-domain multi-layer testbed covering three different networks at two distinct locations in Europe. The testbed includes two Ethernet switched client networks, which are interconnected by a wavelength switched server network. Each of these networks is operated by a GMPLS control plane and implements a path computation entity, following either the IETF PCE proposal or the DRAGON NARB. Since the communication protocols of IETF PCE and DRAGON NARB are incompatible, we propose and develop an application layer gateway, enabling inter-domain path calculation.
In this paper, our contributions are three-fold: First, we provide a comparison of both communication protocols. Second, we present the architecture and working principles of the designed NARB/PCE Gateway, specifying the available features and constraints of our implementation. Third, we validate, for the first time, the PCE/NARB connectivity while evaluating the performance of a path computation request in terms of request response time in the multi-domain and multi-layer testbed.

Keywords: multi-domain, multi-layer, PCE, path computation.

1 Introduction

The ITU-T defined the term next generation network (NGN) in [9]. One of their major objectives is the separation of the transport network (transport stratum) and the service platform (service stratum). The transport stratum enables a variety of transport network technologies, e. g., connection (wavelength) or packet switched networks (Ethernet [7], Internet Protocol). Networks operating and controlling multiple technologies in parallel refer to multi-layer networks. Therein, the underlying transport technology is in general abstracted by a virtual topology. Any layer operating on top of this virtual topology may request additional connectivity from the transport technology. In general, these transport networks with different underlying connection-oriented technologies

T. Magedanz et al. (Eds.): Tridentcom 2010, LNICST 46, pp. 347–362, 2010.

required separate controlling entities. The GMPLS control plane framework [13] is one proposal for a unified integrated control plane, i. e., operating various technologies on different layers. It is responsible for the establishment, management and release of end-to-end connections, such as optical connections or lightpaths, commonly referred to as lambda switched capable Label Switched Paths (LSP). Such a control plane includes the functionalities of: a) routing and topology information dissemination, b) path computation and c) LSP signalling. Including several layers within one single control plane instance increases the number of traffic engineering links (TE-links) within one routing domain. Any path computation, which operates on these TE-links, becomes complex and requires powerful processing engines.

Additionally, as the ITU-T proclaims the NGN as the implementation of a global information infrastructure [9, 8], it demands inter-operable networks throughout different network operators and network technologies. Consequently, paths traversing multiple domains refer to a multi-domain scenario, and any path computation involves the controlling entities within each of these domains. These computation tasks may add additional constraints to the path computation, which further increase the complexity.

Summarizing, the path computation task becomes complex and extensive in multi-domain and multi-layer networks. Consequently, the IETF proposed the Path Computation Element (PCE, [5]), i. e., an explicit entity to perform path computation within these transport networks. They defined the requirements, the architecture [5] and a communication protocol [14] for the path computation function. In parallel to the IETF, the US DRAGON project [11] also proposed a Network Aware Resource Broker (NARB, [15]), which serves the same purpose, but provides a different communication protocol.

In this paper, we present the experiences of interoperating PCE and NARB devices within a multi-domain, multi-layer network infrastructure. For the first time, we present a successful path computation covering three different domains (located in Spain and Germany) and two different transport layers (an Ethernet switched network and a Wavelength Switched Optical Network, WSON). Our contribution is three-fold. For the interoperation of PCE and NARB, we first provide a comparison of both communication protocols; second, we present the architecture and working principles of the designed NARB/PCE Gateway, specifying the available features and constraints of our implementation. Third, we evaluate the performance of a path computation request in terms of request response time in the multi-domain and multi-layer testbed.

The next section introduces and classifies the related work in the field of multi-layer, multi-domain path computaion. Section 3 introduces the implementations of the two path computation elements used in our scenario. Section 4 presents the architecture and functionality of our PCE/NARB Gateway and compares the PCE and the corresponding NARB protocol. Section 5 introduces our testbed with three different domains on two locations in Europe. We evaluate the performance of the whole system in the result section 6. Section 7 summarizes our contribution.

2 Related Work

We classify the related work in multi-layer and multi-domain studies and focus on the applied methods. Multi-layer path computation strategies were the subject of numerical simulation studies of Cugini et al. in [4] and Gunreben/Rambach in [6]. Multi-domain path computation was subject of the prototypical implementations of Bianzino et al. [2] and Casellas et al. in [3]. This paper provides both, prototypical implementation in a multi-layer and multi-domain testbed.

3 Path Computation Elements

This section introduces the architecture and functional details of both path computation elements applied in our testbed. Firstly, we describe the IETF PCE implementation of CTTC. Next, we present the proprietary PCE solution from the DRAGON project. Thereby, both sub-sections include a general introduction, a description of the architecture and a brief summary of the application program interface (API). Finally, we compare that part of the APIs, which is relevant for our scenario.

3.1 IETF PCE Implementation

The IETF proposes in [5] and [14] the architecture and the API/communication protocol for path computation elements. This section gives an introduction and a brief summary of both. A working implementation of a PCE in CTTC's ADRENALINE® testbed shows the applicability of this IETF solution.

Introduction. In general, the IETF PCE proposal covers generic PCE-based implementation building blocks, such as composite, external, and multiple PCE path computation approaches. The retained architecture in this work involves an external PCE, available for multiple PCE path computation tasks.

Further, the aforementioned normative documents discuss architectural considerations including centralized and distributed computation, synchronization, PCE discovery and load balancing. The PCE provides the following additional functions: detection of PCE aliveness; communication between Path Computation Clients (PCC) and the PCE; PCE-PCE communication; Traffic Engineering Database (TED) synchronization; monitoring; policy and confidentiality enforcement.

Architecture. CTTC implemented the PCE according to the IETF standard. Fig. 1 shows the architecture of the implementation. It consists of three major blocks, (a) the management and processing of Path Computation Element Communication Protocol (PCEP) messages from PCC and PCE peers, (b) the management and update processing of the traffic-engineering database (TED) and (c) the path calculation itself. Multiple asynchronous processes (threads) realize the functionality of these blocks.

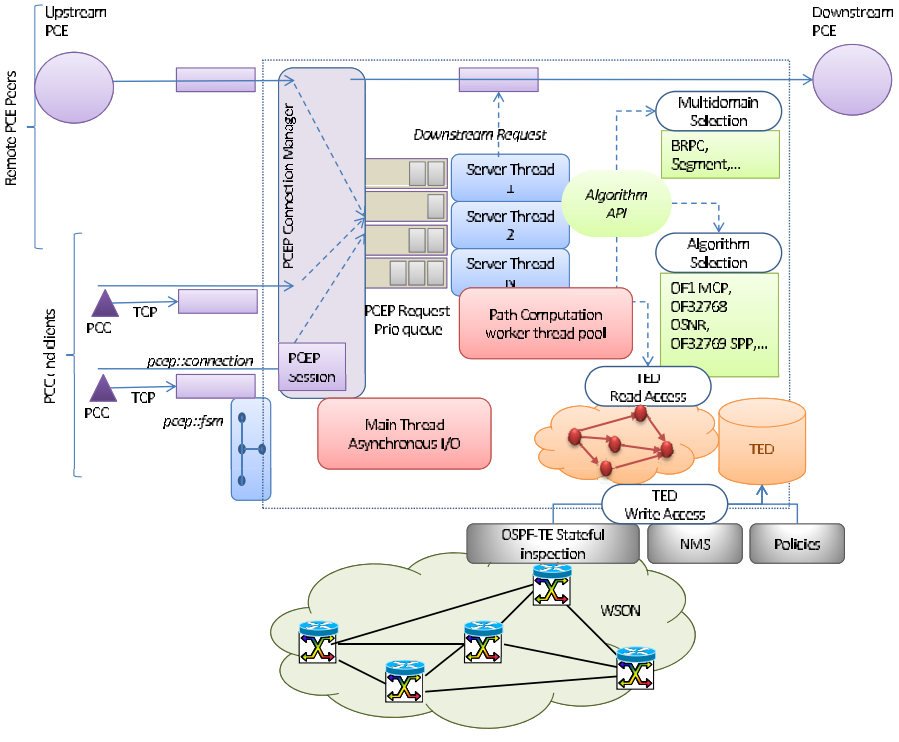


Fig. 1. Functional Architecture for CTTC PCE

The PCE main thread is responsible for managing PCEP connections. It queues incoming path computation requests in a priority-based queue. A pool of threads (with a configurable number of worker threads) serves these requests from the queue. In case of a multi-domain scenario, the PCE requires, besides the local topology, a downstream response from a peer PCE. This dependence blocks this thread until it receives the computation answer from the downstream PCE, controlled by a timeout mechanism. Thereby, a timeout mechanism avoids infinite blocking. In case of timeout, the PCE replies with a NO PATH object.

The PCE implementation provides an interface to allow different algorithms for path computation as plug-ins. This leverages granular, per-request algorithm selection (using Objective Function - OF - codes to identify specific algorithms), based on client preference and pre-defined policies. The common interface of these path computation algorithms involves: a) access to the abstracted TED in form of a directed graph; b) the possibility of requesting path computations from an external (e. g., downstream) PCE and c) access to dynamic / static information covering network reachability and pre-configured PCE domain chains.

In our scenario, the TED includes the optical network topology including link and node GMPLS TE attributes along with specific optical networks extensions such as the number of available optical signal regenerators and optical signal to

noise ratio (OSNR) figures. One or more dedicated threads update this TED. Thereby, the TED synchronization mechanism is non-intrusive. The PCE monitors the OSPF-TE [10] traffic and constructs the database by means of stateful inspection of the OSPF-TE link state updates. This approach passively reuses the OSPF-TE dissemination mechanism and does not require the creation of an additional listener adjacency.

PCEP API. The main interface to access path computation services from a PCE is the PCEP protocol [14]. PCEP allows PCCs to request path computations by means of a message oriented protocol over a TCP connection [12]. The PCEP protocol defines seven basic messages, which we classify in three different categories: session management, path computation and exception handling. The evaluation in section 3.3 studies the detailed messages and compares them to the NARB equivalents.

A special feature of PCEP is the session management. After an initial session setup, the lifetime of this session follows one of two different modes. In the *persistent mode*, the lifetime can span several requests. In the *non-persistent mode* the session ends after a single request. Additionally, the path request message (PCReq) may include one or several requests, allowing for synchronized and dependent path computation.

3.2 DRAGON NARB

Besides the IETF initiative, the US project DRAGON (Dynamic Resource Allocation via GMPLS Optical Networks, [11]) proposed and implemented a device with similar functionality than that of a PCE. The project designed and implemented a Network Aware Resource Broker (NARB, [15]), which performs path computation in multi-layer and multi-domain networks. Comparable to PCEP, NARB also provides an API for remote path computation requests. The next two sections introduce the NARB architecture as well as the API protocol.

Introduction. The NARB represents a path computation element within an Autonomous System (AS). It consists of two entities, the NARB itself and the Resource Computation Element (RCE). NARB provides higher-level functions like topology abstraction for an inter-domain scenario or inter-domain path computation. In an inter-domain scenario, the NARB elements of each domain interconnect and exchange static reachability information. For security and competitive reasons, the exchanged topology information may only provide a subset or abstracted view of the real topology. A RCE enables path computation and provides a raw database of the topology.

Architecture. The NARB provides inter-domain as well as intra-domain path computation functionality. It obtains the intra-domain topology information from listening to the local OSPF-TE routing instance and the inter-domain topology information from listening to the inter-domain OSPF instance. Besides, the NARB software provides the following interfaces:

– Internal interface to RCE for path calculation
– API interface for path computation requests
– External interface to peer NARB of other domains

For a detailed view on the NARB architecture, we refer the reader to the documents published by the DRAGON consortium [11, 15]. The next section introduces the application programmable interface for inter-active usage of the NARB functionality.

NARB API. We refer to [15] for a complete feature list of the NARB API. This section covers the major functions exploited within this study. The NARB API clients connect to the NARB API server on a dedicated port. Thereby, it allows several simultaneous connections. The NARB API message structure shows a header and a message body. The header includes, among others, the message type and options. The body includes type-length-value (TLV) encoded data.

The message type distinguishes seven different messages, which belong to two different categories. The first class includes messages related to path request and reply. Resource reservation related messages make up the second class. While [15] provides details on all these messages, we do not use all types of messages in our scenario. Especially the reservation messages are out of scope of this paper. The following four message types are relevant for our study.

– *Client LSP Query Request* indicates a path computation request from a client.
– *Peer LSP Query Request* indicates a recursive path computation request from NARB to its peer NARB in a multi-domain environment.
– *LSP Query Reply* with an Explicit Route Object (ERO) indicates a successful path computation task, and the reply includes the explicit route object required for the signalling process.
– *LSP Query Reply* with ERROR indicates an unsuccessful path computation task. The message includes the reason for this error, e. g., no source, destination, no route or internal errors.

3.3 Comparison of PCEP and NARB API

In this section, we compare NARB API and PCEP. However, our comparison is neither exhaustive nor complete as we restrict ourselves to the basic functionality for path computation in a multi-domain and multi-layer scenario. Table 1 compares both protocols with respect to the path computation scenario. The table classifies the protocol features in three different classes: session management, path computation and exceptions. The left column gives the functionality. The second and the third column depict the realization of this functionality in PCEP and NARB API, respectively.

Both, PCEP as well as the NARB API rely on TCP. On top of TCP, PCEP implements a session management including Open, Close and KeepAlive messages. The Open messages enable the negotiation on different parameters (deadtime, keepalive timer) during the session initiation phase. The KeepAlive messages provide mechanisms to check if the PCE is still alive and operable. Besides

Table 1. Comparison of PCEP and NARB API

Functionality	PCEP	NARB API
Session management		
Open Message	Session negotiation	-
KeepAlive Message	Session liveness monitoring	-
Close Message	Session termination	-
Identifier	Session identifier	Universal client id, sequence number
Path computation		
Computation request	May include several requests per message	One request per message
Constraint request	support of end-points constraint support of bandwidth constraint	
	-	Requires technology constraints (encoding, switching type)
	Can exclude objects (XRO)	-
Computation reply	successful reply includes ERO	
	NO PATH object to indicate no path	Error message indicating no path found
	May include several replies per message	One reply per message
	Multiple alternative paths per reply	One reply per message
Exceptions		
Error messages	several error classes	one error class
Notification	exceptional signalling for unforeseen events	-

this, they acknowledge the Open message during session establishment. The Close message terminates a session. The NARB API does not provide any session management at all. After the TCP session is established, the client may pass the NARB API message directly to the NARB server. With respect to the response time of the initial request, the reduced session overhead leads to a reduced response time.

Path requests also provide similarities as well as differences. PCEP allows adding multiple path requests within a single path request message. In contrast to this, the NARB API expects only one request per message. Both protocols allow specifying constraints for path computation. While PCEP allows indicating optional or mandatory constraints, the NARB API requires certain parameters in any case, i. e., encoding type, switching type, and bandwidth. The only exception in the NARB API is the quality of the hops. The client may specify if hops may be strict or loose and if this constraint is mandatory.

The path computation reply messages are very similar. In case of successful path computation, the reply messages of both protocols contain standard compliant explicit route objects (ERO, [1]). The difference is again the number of

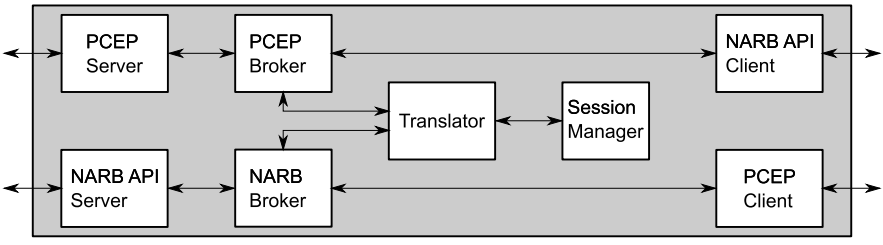


Fig. 2. NARB/PCE Gateway architecture

possible paths per reply message. While PCEP allows multiple responses, each containing one or more paths and attributes, the NARB API only allows one.

In addition, the error handling is different in case that no path is found. While PCEP replies with a NO PATH object in a message with optional TLVs, the NARB API message only indicates the reason for the unavailability of a path. Both protocols implement the errors unknown path source/destination and no path could be computed.

At any time an error may occur. PCEP as well as the NARB API provide mechanisms to indicate such errors. The classification of the NARB API errors includes mainly two error classes: unavailable paths and NARB internal errors. Additionally, PCEP includes classes for policy violations, malformed messages and non-conforming PCEP requests. Summarizing, the error classes of both protocols are incompatible, the commonality may be a most generic error.

4 PCE/NARB Gateway

The previous section highlighted the major differences of both protocols. IETF PCEP and the DRAGON NARB API are in general incompatible as they show different message formats and protocol states. Nevertheless, on a functional level they are compatible. Any interoperation requires an application layer gateway, which implements both interfaces and performs a translation of messages. It implements at the interface to the NARB the NARB API while on the interface to the PCE it implements the corresponding PCEP interface. On both sides, server instances of both protocols reside. As a path computation request from a NARB translates to a path computation request to the PCE, the gateway also implements the client functionality for NARB API and PCEP.

We implemented the NARB/PCE Gateway using the Java programming language for platform independence. The next three sections introduce the architecture and working principle of the gateway and specify the available features and constraints of our implementation.

4.1 Architecture

Fig. 2 depicts the gateway architecture with the according building blocks. The following paragraphs introduce the functionality of each block.

The PCEP and NARB API server are responsible for the TCP connection management on both sides, i. e., they accept connections and create sockets. On a successful connection, they forward the socket information to the corresponding broker instance.

The broker represents the instance, which associates a NARB connection with a PCE session. It reads messages from the socket and interprets these messages. It finally forwards these messages to the translator block.

The translator block is responsible to convert a PCEP message to an equivalent NARB API message and vice versa. Thereby, it translates the common feature-set of a message with respect to the other protocol. The translator is able to translate messages required for a path computation request including session and error handling. Besides, the translator employs the functionality of the session manager.

NARB API and PCEP apply different identifiers for a session (cf. Tab. 1). However, an entity needs to associate replies from one protocol to the requests of the other protocol and vice versa. The session manager implements this functionality. It maintains a list of identifiers for both, NARB API and PCEP requests. Besides, the list contains the information on the corresponding broker instance.

The client instances of each protocol forward the translated messages to the peer NARB/PCE, after necessary connection setup. The next section introduces the working principle in detail.

4.2 Working Principle

Fig. 3 illustrates the working principle of the gateway. The starting point is a path computation request using PCEP. Therefore, the PCEP client connects to the PCEP server instance at the gateway and establishes a TCP connection. After successful connection establishment, the PCEP server passes the information on the socket to the PCEP broker. The PCEP client and the PCEP broker perform the handshake mechanism using the PCEP open sequence (two times Open and KeepAlive, [14]).

After a successful handshake, the PCEP client sends a request to the gateway. The PCEP broker receives the request and translates the request in an internal data structure. It passes the internal data structure to the translator entity. The translator maps the internal data representation into an equivalent representation of a NARB API message. Thereby, it applies the session manager to record the identifiers used in both protocols and PCEP broker instances. The PCEP broker receives back the translated message from the translator. It forwards the NARB request to the NARB client instance, which connects to the peer NARB server and forwards the request to it.

After processing the request on NARB side, the NARB client may receive a response message. It parses the response message and checks with the Session

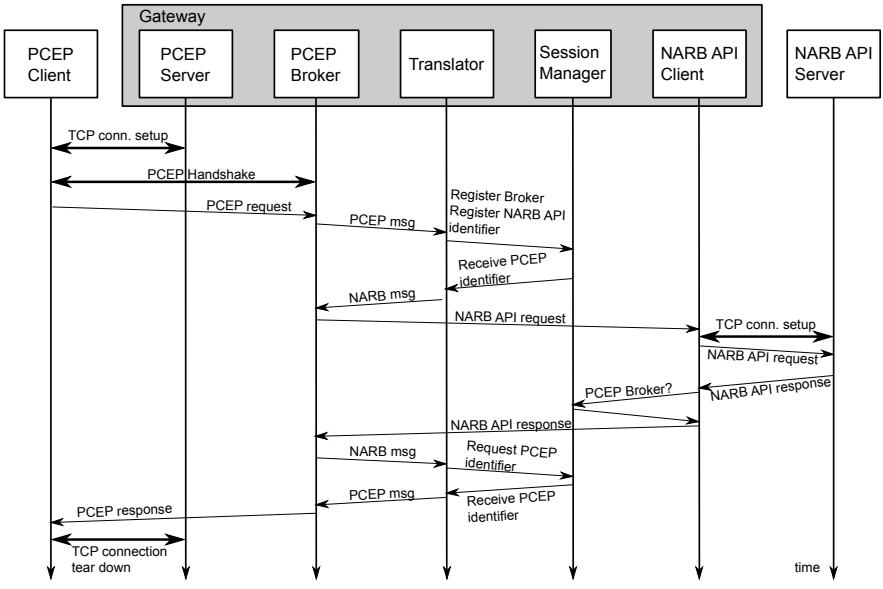


Fig. 3. NARB/PCE Gateway working principle

Manager for the corresponding PCEP broker. With this information, the NARB client forwards the message to the PCEP broker. The latter one uses again the translator to receive a PCEP response message. The Translator performs the translation from NARB API response to PCEP response and requests the session manager for the corresponding sequence number on PCEP side. The PCEP broker passes the complete NARB API message to the requesting PCE client.

The same procedure applies for NARB API requests, which the gateway translates to PCEP messages. The message sequence chart is the same, except the open sequence, which occurs after translating, when forwarding to the peer PCE server.

4.3 Evaluation

This section presents a brief evaluation of the gateway implementation. On a system perspective, the gateway is a multi-threaded program. It operates one main thread for network input/output and maintains for each peer an additional thread. Within this thread, the gateway receives and sends messages and performs the translation tasks as described before. This thread is non-blocking, i. e., the thread does not require to wait for an answer from a downstream peer for further processing. Incoming messages are processed in a first-come first-service discipline per peer NARB/PCE.

This single thread per peer NARB/PCE remains open until a dedicated request tears down this session, i. e., the gateway implements the persistent mode

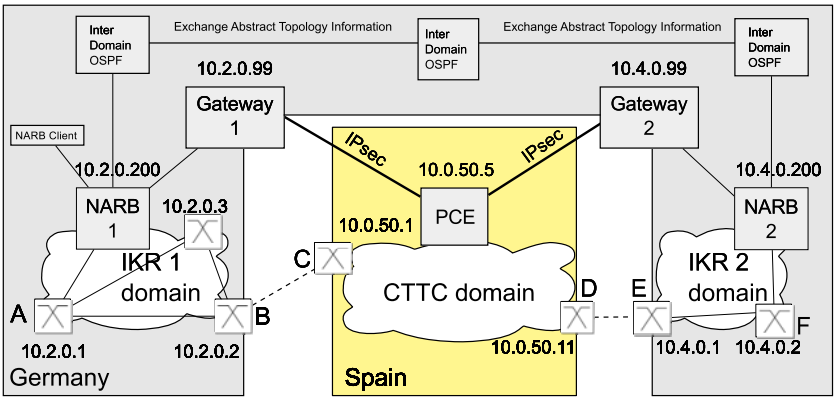


Fig. 4. Multi-domain, multi-layer testbed

of PCEP. For simplicity, the gateway announces a value of 0 for both keepalive and deadtimer, i. e., it keeps the connection open until arrival of a close request.

For multi-domain path computation, the PCE requires a METRIC object [14] in the path computation reply message. This object includes the value of a metric classifying the replied path, e. g., length in number of hops. As the NARB does not support such METRIC Object, the gateway adds this object to the reply from the NARB and forwards it to the PCE.

5 Multi-layer Multi-domain Testbed

This section presents our multi-layer and multi-domain testbed to evaluate both, the inter-PCE connectivity using the PCE/NARB gateway and the performance of a path computation request. The first section introduces the testbed setup while the second section evaluates a sample for a path computation request.

5.1 Introduction

The testbed setup (cf. Fig. 4) interconnects three domains, the IKR 1 and IKR 2 and the CTTC domain. The testbeds of IKR 1 and IKR 2 reside at the University of Stuttgart, Germany. Both testbeds implement an Ethernet based data plane and the DRAGON/GMPLS based control plane [11]. Thereby, the data plane is connection-orientated using Ethernet VLAN tagging [7]. Each domain represents an individual OSPF-TE area and realizes the path computation task with a NARB element in each network.

The ADRENALINE® testbed resides at the premises of CTTC in Castelldefels (Barcelona) in Spain and implements a Wavelength Switched Optical Network (WSON). The optical layer is GMPLS-controlled and shows 14 network nodes with an emulated optical hardware. The path computation function has

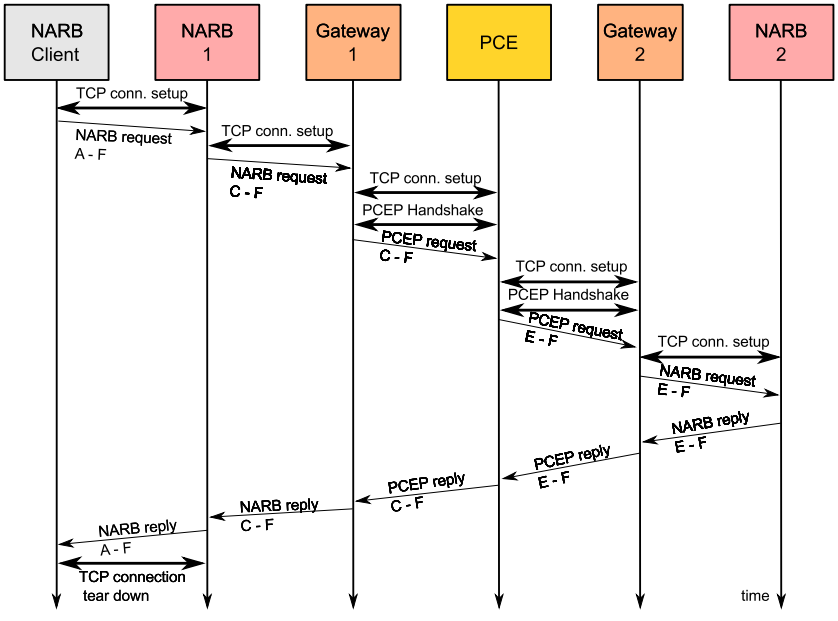


Fig. 5. Path computation request in our inter-domain testbed

been centralized in a single Path Computation Element within the domain, which spans a single OSPF-TE area.

Both, NARB and PCE receive intra-domain topology information through passive listening to the OSPF-TE [10] protocol. Between these networks, they exchange inter-domain topology information, which corresponds to the abstract/summarized topology as introduced in section 3.2. The abstract topology information includes only the border nodes of each network hiding the nodes and TE attributes within the network. For simplicity, we emulate the OSPF adjacency to the ADRENALINE® instance to the CTTC domain by a third OSPF instance at the IKR testbed. This additional OSPF adjacency announces the abstract topology of the CTTC domain deployed using the ADRENALINE® testbed (nodes C and D) to the OSPF instances of networks IKR 1 and 2.

As both show non-compatible protocols, a PCE/NARB gateway resides between both (Gateway 1 and 2). An IPsec tunnel via the public Internet realizes the interconnection between the NARB/PCE gateway and the PCE of the ADRENALINE® testbed. For NARB 1 and NARB 2, the gateway acts like a peer NARB. For the PCE, the gateways act like peer PCEs.

5.2 Multi-domain Multi-layer Request

This section presents a sample path computation request for inter-domain path computation. Fig. 5 depicts the message sequence chart of a typical request

covering all three domains and the intermediate devices. The message sequence chart assumes unestablished TCP and PCEP connections between all entities. The request asks for a suitable path from node A to node F of Fig. 4. The request does not include any other constraints than the endpoints.

The NARB client from the DRAGON suite performs the path computation request. After the initial TCP handshake, the NARB client sends the request to NARB 1 using the NARB API. NARB 1 receives the request and splits it in two parts. While NARB 1 handles itself the local part, i. e., the path computation from node A to node B, it forwards the remote part to the downstream NARB, i. e., Gateway 1. Therefore, NARB 1 establishes a TCP connection first. The gateway performs a translation of the NARB message, which results in a PCEP request for the downstream PCE. Besides, the gateway records the identifiers of the received NARB message and the PCEP message to later identify the response to the correct request. Before forwarding the request to the remote PCE, the gateway performs the TCP and the PCEP handshake. The PCE receives the request and splits the request in a local part (path from C to D) and a remote part. After the TCP and PCEP handshake, it forwards the remote part to the peer PCE, i. e., Gateway 2. Again the gateway translates the request in a NARB conform way, records the identifiers and forwards the request after the TCP handshake to NARB 2. Now, the request of NARB 2 only includes local nodes E and F.

After computation, NARB 2 responds with a suitable path to the peer NARB, i. e., Gateway 2. After passing the gateway, the PCE receives the response and joins the results from the local path computation and the results from the downstream PCE, i. e., NARB 2, and forwards the result to the requesting entity, i. e., Gateway 1. After passing the gateway, NARB 1 receives the responses of PCE and NARB 2, joins them and forwards the outcome to the requesting NARB client. After reception of the response, the NARB client terminates the TCP connection, as there is no further request. The other TCP and PCEP connections remain open, as there is no indication to close them.

If there is any error on this path, the response in upstream direction includes an error message, which is translated in an appropriate format. As the error reasons do not overlap, in most cases a general error of no path found occurs in the response.

6 Performance Evaluation

For a quantitative performance analysis, we evaluated the time from sending a request until receiving the response. For this evaluation, we analyzed two different scenarios, the PCEP non-persistent and persistent mode.

In PCEP non-persistent node, for every request, the PCE establishes a new TCP and PCEP connection to NARB 2. The PCE closes this PCEP connection after receiving the response from NARB 2. For a new request, the whole setup procedure is performed anew. The connection from NARB 1 to the PCE remains persistent.

No.	Time	Source	Destination	Protocol	Info
1	0.000000	10.0.50.5	10.2.0.99	PCEP	OPEN MESSAGE
2	0.000280	10.2.0.99	10.0.50.5	PCEP	OPEN MESSAGE
3	0.053381	10.0.50.5	10.2.0.99	PCEP	KEEPALIVE MESSAGE
4	0.053782	10.2.0.99	10.0.50.5	PCEP	KEEPALIVE MESSAGE
5	0.116968	10.2.0.99	10.0.50.5	PCEP	PATH COMPUTATION REQUEST MESSAGE
6	0.171188	10.0.50.5	10.4.0.99	PCEP	OPEN MESSAGE
7	0.225655	10.4.0.99	10.0.50.5	PCEP	OPEN MESSAGE
8	0.225715	10.0.50.5	10.4.0.99	PCEP	KEEPALIVE MESSAGE
9	0.279501	10.4.0.99	10.0.50.5	PCEP	KEEPALIVE MESSAGE
10	0.279558	10.0.50.5	10.4.0.99	PCEP	PATH COMPUTATION REQUEST MESSAGE
11	0.337972	10.4.0.99	10.0.50.5	PCEP	PATH COMPUTATION REPLY MESSAGE
12	0.338104	10.0.50.5	10.4.0.99	PCEP	CLOSE MESSAGE
13	0.338509	10.0.50.5	10.2.0.99	PCEP	PATH COMPUTATION REPLY MESSAGE

Fig. 6. Trace of PCEP setup and response/request messages at PCE

In PCEP persistent mode, the PCE also keeps the PCEP connection to NARB 2 open and reuses this connection for further requests. Only the first request performs the setup procedure, while the PCE forwards subsequent requests immediately. The mechanism to keep the PCEP connection open corresponds to the KeepAlive messages and deadtimer agreed in the open sequence.

6.1 Non-persistent Mode

In non-persistent mode, the PCE requires a TCP/PCEP setup between PCE and NARB 2 for each request (cf. Fig. 5). Thereby, the round trip time (RTT) between the two instances is the main driver for duration of the setup. In our scenario, we measure a RTT of about 52 ms between the IPsec router at IKR and the IPsec router at CTTC. We also observe some jitter as the IPsec tunnels pass several thousands of kilometres through the Internet.

Fig. 6 illustrates the PCEP messages at the PCE including the setup and request/response messages. The current implementation of the PCE (IP: 10.0.50.5) sends as direct consequence of the TCP connection establishment a PCEP Open message to Gateway 1 (IP: 10.2.0.99) (line 1). Gateway 1 also sends a PCEP Open message after successful TCP connection establishment. Due to the three-way handshake of TCP and the different times of connection establishment, the PCE receives this message from Gateway 1 more or less immediately after sending its own OPEN (line 2). At the beginning, the TCP window size allows only one outstanding unacknowledged packet. Therefore, both entities need to wait on the TCP acknowledgment before being able to send a further PCEP message. After reception of the TCP ACK which takes one RTT, the PCE sends a KeepAlive (line 3) to Gateway 1 and receives one (line 4). On reception of the KeepAlive by the PCE, Gateway 1 sends the path computation request. The PCE receives this one RTT after sending the KeepAlive (line 5). Subsequently, it triggers the establishment of a TCP and PCEP connection to Gateway 2 (10.4.0.99). In principle, the messages of the open sequence are exchanged asynchronously, i. e., there is no predefined order of messages. However, in our case,

messages of the PCE trigger the according actions at Gateway 2 (lines 6-9). After sending the path computation request (line 10) and receiving the reply message (line 11), the PCE closes the PCEP connection to Gateway 2 (line 12). Finally, the PCE sends the reply to Gateway 1 (line 13).

For a subsequent request, the TCP as well as the PCEP connection between Gateway 1 and PCE is already established. When measuring the response time of such a subsequent multi-domain path request, the above mentioned jitter introduces some slight variations. On average, we obtain a response time of 273 ms. This overall response time includes 3 RTTs for TCP and PCEP connection setup between PCE and Gateway 2 and 2 RTTs for exchange of path requests and replies between Gateway 1, PCE and Gateway 2. The round trip times sum up to a value of about 260 ms. Thus the accumulated processing of all entities is in the order of 13 ms or even below.

6.2 Persistent Mode

The persistent operation only requires the TCP/PCEP handshakes for the very first request. All subsequent requests use established TCP and PCEP connections. Consequently, a trace analogue to Fig. 6 would not show lines 1-4, 6-9 and 12. Thus, we save 3 RTTs in contrast to the overall response time of the non-persistent mode. The measured average response time of about 117 ms reflects this consideration very well.

7 Conclusion

The contribution of this paper was three-fold. First, we studied both PCEP and DRAGON NARB architectures and provided a comprehensive comparison on their functionality and communication protocols. For the interconnection of both, we introduced an application layer gateway, which is able to translate one protocol to the other and vice versa. Second, we setup a multi-domain multi-layer testbed, which includes connection-oriented Ethernet as well as optical networks at two distant locations in Europe. For path computation, the domains exchange reachability information. We showed that multi-layer multi-domain path computation is feasible together with IETF PCE, DRAGON NARB and gateway implementations. Third, we evaluated the scenario and sampled the response time for the path computation for two different operations: persistent and non-persistent.

Our next steps are the improvement of the PCE implementation to serve requests in parallel as well as the enhancement of the gateway to support a larger number of IETF PCE functions, e. g., multiple requests per PCEP message. Besides, we are currently working towards a queuing theoretical model of the whole system.

Acknowledgement

The studies of this paper were partly carried out with the support of the BONE-project ("Building the Future Optical Network in Europe"), a Network of Excellence by the European Commission through the 7th ICT-Framework Programme and the Spanish national project MICINN DORADO TEC2009-07995. The authors would also thank Kristijan Kijurina for implementing the gateway.

References

1. Awduche, D., Berger, L., Gan, D., Li, T., Srinivasan, V., Swallow, G.: RSVP-TE: Extensions to RSVP for LSP Tunnels. RFC 3209, IETF (December 2001)
2. Bianzino, A.P., Rougier, J.-L., Secci, S., Casellas, R., Martinez, R., Munoz, R., Djarallah, N.B., Douville, R., Pouyllau, H.: Testbed implementation of control plane extensions for inter-carrier gmpls lsp provisioning. In: International Conference on Testbeds and Research Infrastructures for the Development of Networks & Communities, pp. 1–6 (2009)
3. Casellas, R., Martinez, R., Munoz, R., Gunreben, S.: Enhanced Backwards Recursive Path Computation for Multi-area Wavelength Switched Optical Networks Under Wavelength Continuity Constraint. To Appear in the IEEE/OSA Journal of Optical Communications and Networking, A180–A193 (July 2009)
4. Cugini, F., Giorgetti, A., Andriolli, N., Paolucci, F., Valcarenghi, L., Castoldi, P.: Multiple path computation element (pce) cooperation for multi-layer traffic engineering. In: Optical Fiber Communication Conference (March 2007)
5. Farrel, A., Vasseur, J.-P., Ash, J.: A Path Computation Element (PCE)-Based Architecture. RFC 4655, IETF (August 2006)
6. Gunreben, S., Rambach, F.: Assessment and Performance Evaluation of PCE-based Inter-Layer Traffic Engineering. In: Proceedings of the IFIP Working Conference on Optical Network Design and Modelling (ONDM 2008) (March 2008)
7. IEEE Computer Society. 802.3: IEEE Standard for Local and Metropolitan Area Networks–Carrier sense multiple access with collision detection (CSMA/CD) access method and physical layer specifications (2005)
8. ITU. Rec. Y.100: General overview of the Global Information Infrastructure standards development (1998)
9. ITU. Rec. Y.2011: General principles and general reference model for Next Generation Networks (October 2004)
10. Katz, D., Kompella, K., Yeung, D.: Traffic Engineering (TE) Extensions to OSPF Version 2. RFC 3630, IETF (September 2003)
11. Lehman, T., Sobieski, J., Jabbari, B.: Dragon: a framework for service provisioning in heterogeneous grid networks. IEEE Communications Magazine 44(3), 84–90 (2006)
12. Postel, J.: Transmission Control Protocol. RFC 793, IETF (September 1981)
13. Generalized Multi-Protocol Label Switching (GMPLS) Signaling Functional Description. RFC 3471, IETF (January 2003)
14. Path Computation Element (PCE) Communication Protocol (PCEP). RFC 5440, IETF (March 2009)
15. Yang, X.: Network Aware Resource Broker (NARB) Design and User Manual. Technical report, University of Southern California (USC), Information Sciences Institute (ISI) (June 2006)

Virtualized Application Networking Infrastructure

H. Bannazadeh, A. Leon-Garcia, K. Redmond, G. Tam, A. Khan, M. Ma, S. Dani, and P. Chow

Electrical and Computer Engineering Department
University of Toronto
10 King's College Road, Toronto, ON M5S 3G4 Canada
hadi.bannazadeh@utoronto.ca

Abstract. In this paper, we present a new platform for experimenting with networked systems and distributed applications called Virtualized Application Networking Infrastructure (VANI). This infrastructure is designed as a converged communications and computing infrastructure that would facilitate operation of an open applications marketplace. VANI enables introduction of new network architectures that require in-network (hardware-accelerated) content processing and storage. We describe the VANI architecture and the resources it provides. VANI has two main planes; control and management plane, and applications plane. VANI resources are virtualized and made available to the researchers and application providers through a service-oriented control and management plane. The current VANI resources are processing, storage, networking and various software-based resources. VANI also includes a new reprogrammable hardware resource that enables experimenting with hardware-based or hardware-accelerated networking algorithms and protocols. We present performance evaluations of this reprogrammable hardware resource, and the VANI virtual networking mechanism. The results show that by using the reprogrammable hardware resource, researchers can evaluate high performance and high throughput networking algorithms as easily as evaluating software-based networking algorithms.

Keywords: Networking Testbed, Network Architecture, Service-Oriented Architecture.

1 Introduction

In the past few years, the idea of clean slate network design has been circulated in the networking community and there have been several proposals for introducing new network architectures and protocols [1,2,3]. One of the major obstacles in introducing new network architectures was and still is experimentation with proposed network architectures in a large scale environment and possibly with massive numbers of end users. To address this problem, there have been several initiatives to build large scale testbeds for networking research.

GENI [4] is one of these initiatives that tries to create a testbed by federating different testbeds such as PlanetLab [5,6] and Emulab [7] on top of a research dedicated network. GENI is still in the design and development phase, but currently it follows a slice-based architecture [8], and different testbeds would be able to connect to each other through

T. Magedanz et al. (Eds.): TridentCom 2010, LNICST 46, pp. 363–382, 2011.

GENI wrappers. The exact communication protocol between the GENI wrapper and the testbed is left to each testbed's control plane and currently there are a few major control planes that are trying to federate using the wrappers.

Probably among these testbeds PlanetLab [5] is the most developed. PlanetLab provides edge hosts on Internet and implements a slice-based architecture using the Linux vServer [9] technology. PlanetLab, however, does not have a clear solution for experimentation with new layer three protocols, and it's not clear how it would facilitate building high scale new routers that would need hardware-based acceleration.

In Canada, there is a research dedicated optical network called CANARIE [10] that provides light paths connecting universities and research centers across Canada. CANARIE has sponsored design and development of a User Controlled Light Path [11] (UCLP) software that enables researchers to configure CANARIE network elements through Web Services (WS) interfaces on-demand.

Another major initiative is FEDERICA [12] in Europe that is under development through federation of several research network platforms in Europe such as i2CAT in Spain and HEAnet in Ireland. FEDERICA uses WS-based UCLP software for creating on-demand virtual networks atop of involving test platforms.

Another project for experimentation with lower layer protocols and networking algorithms is NetFPGA [13]. NetFPGA is a PCI card with a Field Programmable Gate Array (FPGA), and four Gigabit Ethernet interfaces that could be used for developing networking components such as a layer three router or a hardware accelerator.

In this paper, we present a new testbed for networking experiments and networked systems. This testbed is different than the above mentioned projects in several aspects. It benefits from a novel architecture for control and management functions capable of managing various hardware-based and software-based resources. It also allows experimenting with new network architectures that require in-network content processing and storage capabilities. Moreover, it includes a new high performance and high throughput hardware resource that makes experimentation with hardware-based or hardware-accelerated networking algorithms and protocols as easy as experimentation with software-based protocols.

Our vision in designing this testbed was to develop a converged computing and communications infrastructure to support an open applications marketplace. We investigated architectural aspects of this application-oriented network and presented a proposal in [14]. We also investigated autonomic management issues and proposed an approach using virtual networks in [15].

The essential aspects to enabling the above application-oriented environment are: 1. Service-oriented application creation; 2. Infrastructure as a Services methods for configuring and scaling resources to support applications; 3. Virtualization of physical resources.

Based on this view of an application-oriented network, we began the development of a testbed that would allow university researchers and application providers to develop new networked systems and networking architectures. This testbed, Virtualized Application Networking Infrastructure (VANI), allows the creation of virtual networks of computing and communications resources. A VANI node consists of resources such as processing, storage, networking, and programmable hardware. A service-oriented

control and management plane allows VANI nodes to be interconnected into virtual networks to support applications operating in the applications plane.

In the rest of this paper, we describe the main requirements in VANI design, and its architecture and main components. Also, we explain how our design would satisfy the requirements. Moreover, we present the performance evaluations on the developed resources for this infrastructure including a virtualized reprogrammable hardware resource that enables hardware-based experimentation of networking algorithms and protocols.

2 VANI Design Requirements

Virtualized Application Networking Infrastructure (VANI) is a testbed that allows university researchers and application providers to utilize its internal resources to rapidly create and deploy networked systems, and to even experiment with new layer three protocols. Although the underlying concepts of the VANI testbed comes from our view on Application-Oriented Network [14], but networked systems running in VANI environment could follow any architecture in any networking layer. The only limitation that the researchers are facing in VANI is that their experiments should run on top of Ethernet as their layer two. Next, we describe the main requirements in designing VANI.

The VANI design follows some basic requirements (figure 1) The first requirement for VANI testbed is that it should allow experimentation for future network architectures that might not fit into the traditional layer three definitions. Currently networks are primarily responsible for delivering raw data but in future it would be possible for future network architectures to shift-up the network tasks to new functionalities that might be required by emerging applications. Among these functionalities could be the task of content-delivery in addition to data-delivery (such as the network architecture discussed in [14]) that would imply having content processing and storage functions in the infrastructure.

The second main requirement was to allow researchers to experiment with new layer three protocols (as in the traditional definition of L3) instead of the current Internet

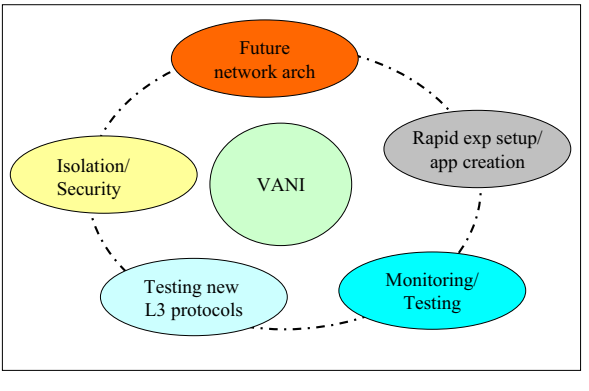


Fig. 1. VANI design requirements

Protocol. To do so, we designed the testbed assuming that everything above layer two could be redesigned and experimented with, and we chose Ethernet protocol as the basis of our layer two design.

Another main requirement in the testbed is to be able to setup experiments or create new applications rapidly using already developed and ready to use components that could be accessed through open interfaces. These components could be the virtualized resources such as processing, low-latency hardware processing, and accelerator nodes, or software components such as event processors that are used in many experiments for data gathering and analysis. This requirement could be satisfied through the use of the SOA technologies and standards that could allow flexible and dynamic composition of reusable service components.

The fourth main requirement was to provide an isolated and secure environment for researchers to carry on their experiments and develop their networked applications. This requirement has to be satisfied at different levels such as traffic separation, bandwidth allocations, storage access, secure access to the physical resources, and isolation between different physical resources. The fifth main requirement was the monitoring and debugging mechanisms. In our design, we envisioned powerful complex event processing components that could be customized to gather and analyze test and debugging data for each experiment separately as well as for the testbed itself.

2.1 VANI Architecture

Based on these main requirements, we designed a two plane architecture for our platform: control and management plane (VANI-CMP) and applications plane (VANI-AP).

VANI-CMP is responsible for virtualizing physical resources and allocating them to the researchers and application providers. On the other hand, researchers deploy their applications and experiments in the VANI applications plane (VANI-AP). Applications operating in the applications plane can have their own architecture inside an applications plane slice that is created by VANI-CMP.

For example, an experiment/application could be a new layer three protocol that covers OSI layer three and four functions, could replace TCP/IP layer, or could be

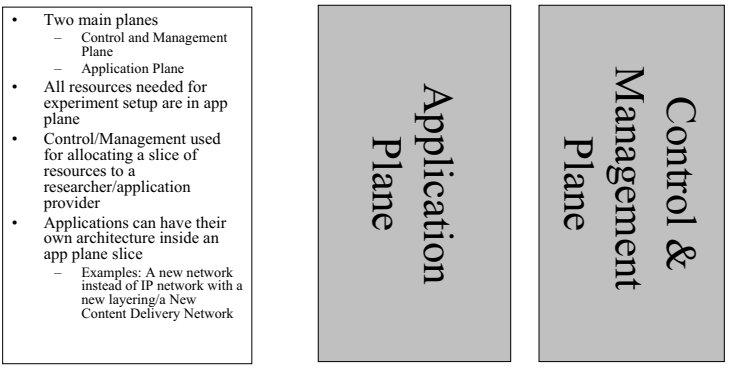


Fig. 2. VANI architecture

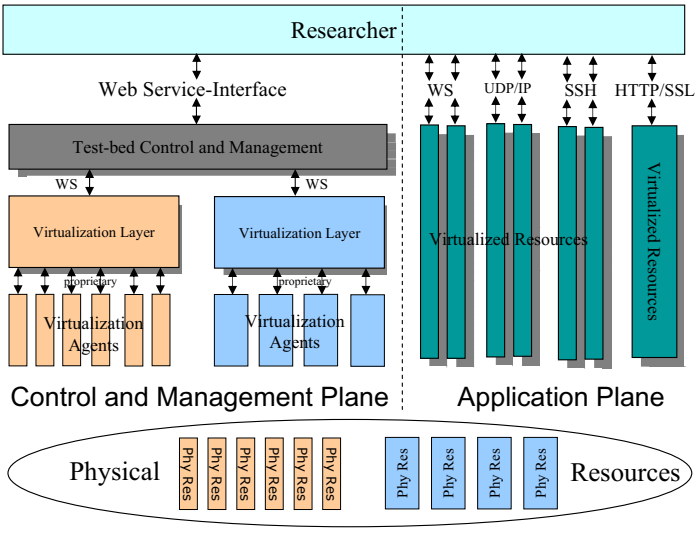


Fig. 3. Researcher interaction with VANI planes

a new content delivery network. Figure 2 shows this architecture including its two planes.

All virtualized resources and service components that can be used by researchers for creating an application reside in the applications plane. Researchers can ask for these resources through the testbed control and management plane and then they can directly connect to the virtualized resource in the applications plane through any resource specific protocol such as HTTP, UDP/IP, or ssh.

For example, a user can ask for uploading or downloading of a file to the storage service through the control plane, and then if permitted by the control plane, it has to directly contact the storage file service using HTTP/TLS connection and download or upload its files.

VANI control and management plane (VANI-CMP) is responsible for allocating testbeds resources to the researchers. Researchers ask VANI-CMP for a resource using VANI-CMP's Web Service interface. WS interface is chosen due its universal acceptance for SOA, and the abundance of available tools for orchestrating and creating new applications using independent Web Services.

After receiving the requests for resources from a researcher, VANI-CMP authenticates the researcher and authorizes its request and then sends the request to the resource virtualization layer. The resource virtualization layer is the layer which abstracts a physical resource and offers it as a service to the control and management layer. If the allocation is successful, VANI-CMP records the allocation, and replies back to the researcher with a successful return result.

VANI-CMP also programs and releases the resource whenever an authorized researcher wants to do so. Figure 3 depicts the logical view of the VANI testbed and how a researcher interacts with VANI planes.

2.2 Current Physical Resources in VANI (VANIv1 Resources)

Currently, several physical resources have been virtualized and made available to VANI users. In [16], the design and development details of these resources have been presented, and here, we briefly overview these resources and type of functionalities that they can offer to researchers.

In VANI all physical resources are virtualized. Through virtualization, we separate applications from their underlying physical resources. To do so, we developed a virtualization layer and virtualization agents for each physical resource as shown in figure 4. The task of the virtualization layer is to coordinate the system wide virtualization of a resource and to expose the resource as a service component with Web Service interface to the rest of the system, and the agents task is to launch or destroy the virtual resources on top of each physical resource.

The first physical resource that we have virtualized is the reprogrammable hardware resource. To develop this resource we have used BEE2 boards [17]. Each BEE2 board has four high-end Xilinx Field Programmable Gate Arrays (FPGA) each connected to four 10GE interfaces. We have virtualized all four FPGAs in a BEE2 board so that a researcher could ask for one or more FPGAs and program it as s/he likes.

Researchers can ask for an FPGA through the control plane and then program it, configure it, or release it. They also have access to the libraries for controlling the 10 GE interfaces and some other commonly used hardware blocks such as DDR2 memory modules. After programming an FPGA, a researcher can directly connect to the FPGA through the 10GE interfaces according to whatever protocol designed for that FPGA. For example, a researcher can use one FPGA or all four FPGAs to develop a layer three router with 4x10GE ports or 16x10GE ports, or a content-based routers that routes packets based on the packets payload rather than their headers. We present the performance evaluation results for this hardware resource in the performance evaluation section of this paper.

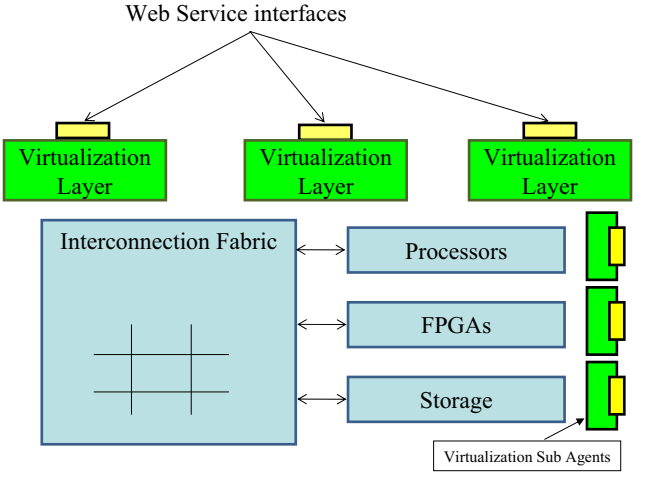


Fig. 4. Virtualizing physical resources in VANI

Another physical resource in the VANI testbed is the processing resource. The processing service is developed based on Linux vServer [9] technology. Linux vServer is an OS-level virtualization software that creates a virtual processing node on top of a Linux kernel. Researchers are able to get a processing resource through VANI-CMP, and release it whenever they wish to do so. Once a virtual processing node is allocated, the researcher can directly ssh to the node. Researchers are also able to program the virtual processing node with a specific image, create an image of their own, and save it on the storage service, and share it with others or program other virtual nodes with that image.

We have also virtualized the internal fabric of the testbed for creating virtual networks. The internal fabric consists of a set of high capacity Ethernet switches that are able to isolate traffic between different applications and experiments by creating separate virtual LANs. Moreover, it allows different experiments to intercommunicate by creating shared virtual LANs that all have access to. This resource, together with the processing resource, enable VANI to guarantee the bandwidth for an experiment. Later in the bandwidth guarantee section, we will discuss this feature in more detail.

The gateway and bridge resource is another developed resource that enables communication between different VANI nodes. If one of the resources in VANI needs to be accessible from the Internet or from a resource in another VANI node, it can ask for a public address through the gateway service and get an address for duration that the external access is needed. The researcher can release the public address when it is no longer needed.

Th bridge service is used for experiment involving new layer three protocols on top of Ethernet network. Using the bridge service, a researcher can send and receive layer two Ethernet frames to any other VANI node, and hence, would be able to develop and test new layer three protocols over a wide area network. This functionality would only be available if the VANI nodes are connected using a wide area Ethernet network. We will discuss this case later in more detail.

Another physical resource developed for VANI is the storage resource. Storage resource is implemented on a set of distributed file servers that emulates one big storage server. Researchers are able to connect to the storage service through VANI-CMP and then directly connect to a file server for uploading and downloading files. All the direct communications to the file servers for uploading and downloading files are done over a secure HTTP/TLS connection. Researchers can use this service to store images for programming other resources such as processing resource, and reprogrammable hardware resource, and they can also share file with other researchers through this service.

2.3 Example: Requesting a Resource in VANI

Figure 5 shows a sample message exchange scenario between a researcher, the VANI control and management plane and physical resources inside a VANI node. A researcher starts requesting for a resource by invoking the getResource operation of the VANI-CMP WS interfaces. In that request, the researcher includes the type of resource, the duration and number of required resources.

VANI-CMP authenticates and authorizes the request and forwards the request to the resource. All resources in the testbed expose their operations to VANI-CMP through a

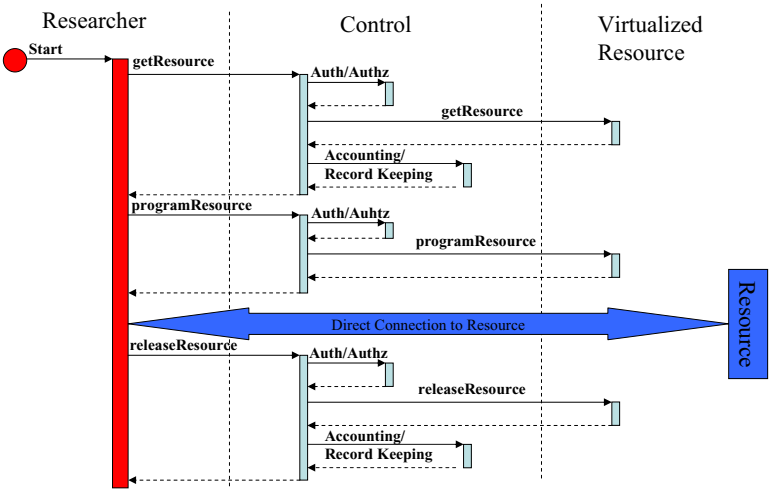


Fig. 5. A sample interaction between a researcher and VANI

generic WSDL interface. This makes it possible to easily extend the types of resources and services in the testbed without changing the control and management software.

The resource responds back to the control plane request with a success result, and a Universally Unique IDentifier (UUID) for the resource. The control plane stores this returned UUID and passes it to the researcher. The researcher can program the resource identified by returned UUID, and release it at a later time.

In the next section, we delve into the control and management design and we describe its main functionalities in detail.

3 VANI Control and Management Plane (VANI-CMP)

VANI-CMP is responsible for performing Authentication Authorization Accounting (AAA) operations and allocates resources to the researchers and application providers. In addition, it performs user management functions, and stores and manages the testbed configuration data. It also has a registry for all services and resources that can be used by researchers for creating a new application or experiment setup. Researchers can register new types of resources in this registry, and make them available for use by other researchers.

VANI-CMP is designed based on service-oriented design concepts and developed using SOA technologies. VANI-CMP is developed in Business Processes Execution Language (BPEL) [18] and deployed on an Enterprise Service Bus (ESB) [19]. Similar to other virtualized resources and services in the testbed, all internal components and functions of VANI-CMP have also been developed as independent service components, and are accessed through Web Services interfaces.

The use of ESB and Web Services enables VANI-CMP to be easily extended in functionality and accessed through other types of interfaces in the future. This design choice

also enables independent development, testing, and redeployment of internal functions of VANI-CMP such as AAA operation, configuration management, etc. Moreover, the use of BPEL language for VANI-CMP enables a high level description of the VANI control and management operations. This enables rapid and easy modifications of the control and management logic.

In the next subsections, we examine each of the functionalities of the control and management plane and we describe the design steps and interfaces of each of the modules.

3.1 User Management

Three concepts are used to manage users in VANI: application plans, service levels, and plan administrator levels. Application plans are used to show different experiments and to organize resources and resource usage in each experiment. When booking a resource, the researcher must specify which plan (experiment) the resource is being booked on. Any researcher belongs to a service level which governs what control operations s/he is allowed to call and also how much of each resource s/he is allowed to book. Custom service levels may be designed for specific users in order to maintain flexibility. Lastly, plan administrator levels are used to govern access to certain resources. Resource users will be granted specific levels of access defining their ability to release, program, save, etc.

3.2 Authentication Authorization Accounting

The control software is responsible for handling authentication of users. All operations in the control plane require users to provide credentials. Currently, credentials are in the form of a user name and password combination however the implementation allows this to be easily changed. On every call to the control software, the user is authenticated and a check is made to ensure that the user has the rights to execute the requested operation. In addition to authentication, the control software is responsible for authorizing access to resources. Every access to a resource consists of two checks, ensuring the resource belongs to the user, and the user has the rights to manipulate the resource as requested.

In order to prevent outsiders from directly accessing resources and bypassing the control plane, all requests to resources require credentials known only to the control plane. This credential is generated when resources are initialized.

The control software keeps a record every time a resource is booked or released. This keeps an account of which resource was used by which user (on which plan) and for how long as well as all resources currently in use. Resources are identified by a UUID generated by the resource and passed back through the control plane.

3.3 Resource Allocation

Resources are booked through the control plane whether the user is a researcher or an application provider building a resource on top of another. Users provide their credentials and specify which resource they wish to book (on which VANI node) and the plan

```
<xsd:element name="getRequestGenericContents">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="internalIP" type="xsd:string"></xsd:element>
        <xsd:element name="uuid" type="xsd:string"></xsd:element>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
```

Fig. 6. A sample schema for generic XML content in a getRequest response message

to which the resource will belong. The control plane ensures the user is allowed to book the resource and determines the location (WSDL address) of the resource in the network. A getResource request is then made to the resource. The resource does not know who is requesting the resource as this information is hidden by the control software. If successful, the resource will return a UUID identifying the resource as well as any other relevant data which is then passed back to the user. The UUID is used by the control plane for accounting purposes.

3.4 Generic Resources/Registration

New resources can be made available dynamically in the control plane through a registration operation. The new resource must consist of a unique name, a service name, a port name, one or more WSDL addresses, and optionally a JNLP address for the resources GUI. The service and port name are used to create an end point reference which is assigned to the partner link when the resource is to be accessed. The resource may have multiple WSDL addresses if there are different instances of the resource on different VANI nodes. The control software will select the appropriate address depending on which node the user is attempting to access. Lastly, a JNLP address may be included which allows resource creators to design and deploy their own GUI using Java web start technology [20].

In order for resource creators to dynamically add new resources to the control plane, it is necessary to use a generic WSDL interface for all resources. The main objective with the generic interface is to provide a template that makes creating resources easy while providing flexibility. This is accomplished by providing a number of operations, messages that are common between many resources such as get, release, and program. To maintain flexibility, each operation contains an optional XML string which can be used to customize data that is passed in and out (figure 6). Furthermore a generic operation is included in the WSDL which can be used to include operations not already included in the template.

4 Security in VANI

One of the basic requirements in VANI design was to make sure the experiments are done in a secure and isolated environment from the other applications and experiments. To create this secure environment we have to consider security issues in various parts of the system architecture.

The first part is to secure the communications between the researchers and VANI-CMP. In VANI all communications between these two entities are encrypted using secure SSL connections and WS-security specification. To do so, each researcher has to share his/her public key with VANI (and vice versa). On top of that VANI-CMP authenticates the researchers and application providers using the credentials provided in all transactions, and then, authorizes the researcher's access level to the resource.

The second part is the communications between the resources and VANI-CMP. These communications have also been encrypted. Moreover, credentials only known to the resource and VANI-CMP are included in all communications from VANI-CMP to the resources.

All internal traffic within one experiment is separated from other experiments using tagged Ethernet VLANs. By proper configuration of the testbed internal fabric resource, we are able to isolate these tagged VLANs from each other. This case is discussed in more detail in the bandwidth guarantee section.

Communications inside the applications plane, internal to one experiment, or coming to and from that experiment could be encrypted or not depending on the experiment, and therefore it is outside of the scope of the VANI design. This allows researchers to freely design and develop new encryption and decryption algorithms in different layers inside their application plane slice.

5 Bandwidth Guarantee in VANI

In order to make sure that one experiment cannot undermine another experiment's capability to send and receive traffic, we need to have a bandwidth guarantee mechanism in place. Likewise, for communications between different VANI nodes, there should be a rate guarantee in place so that a distributed experiment could have a guaranteed access to the available bandwidth.

Since all communication in VANI is carried over the VLAN tagged Ethernet frames, an Ethernet rate limiting mechanism in processing nodes has been developed. By doing so, we limit the rate in which each virtual processing node sends and receives traffic from/to another virtual processing nodes inside a VANI node.

Also the gateway and bridge service controls the rate in which an experiment sends/receives traffic to/from the VANI wide area network. The wide area network that is used to connect the VANI nodes would be a research-dedicated network like CANARIE [10] that can guarantee the aggregated traffic to/from the VANI nodes. If the wide are network was able to provide dynamic and on-demand bandwidth allocation, VANI would be able to use this functionality whenever an experiment asks for sending/receiving traffic to/from the wide area network. VANI nodes could also be connected to the public Internet network, however, bandwidth could not be guaranteed for the experiments in this case.

To request a bandwidth guarantee in VANI, a researcher can specify the bandwidth requirements of a virtual processing node in the resource get request. Likewise, a bandwidth requirement can be specified when access to the VANI wide are network is requested. The virtualization layer in VANI control and management plane makes sure that the specified requirements are met when allocating virtual resources to the experiment.

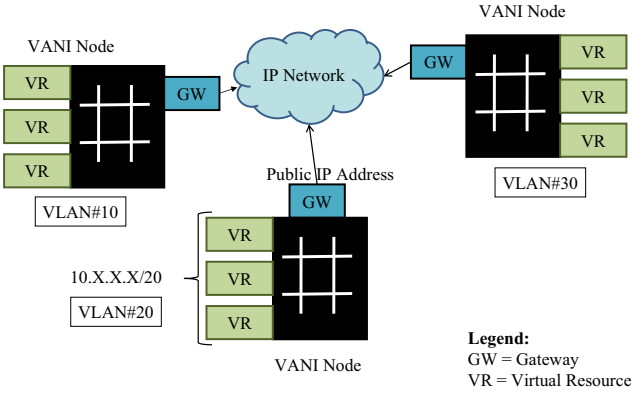


Fig. 7. Connecting VANI nodes in IP layer

5.1 Interconnecting VANI Nodes in IP Layer

Figure 7 shows how we can set up an experiment or create a distributed application across a wide area IP network. In this setting, all resources inside an experiment in a VANI node get a local IP address in the range of 10.X.X.X. All resource could send traffic to the wide are network using the NAT functionality implemented in the gateway service (shown as GW in figure 7). It is possible to put multiple gateways in place and direct outgoing traffic to different gateways to avoid bottlenecks in the system.

On the other hand, if a resource needs to be accessible from the wide area network, the researcher can ask the gateway service for a public address/name, and the gateway service redirects all traffic to that public address to the resource's internal IP address/VLAN.

5.2 Interconnecting VANI Nodes in Ethernet Layer

Figure 8 shows an Ethernet connected VANI. Ethernet connected VANIs use the bridge service instead of the gateway service to interconnect. Inside a VANI node, all resources

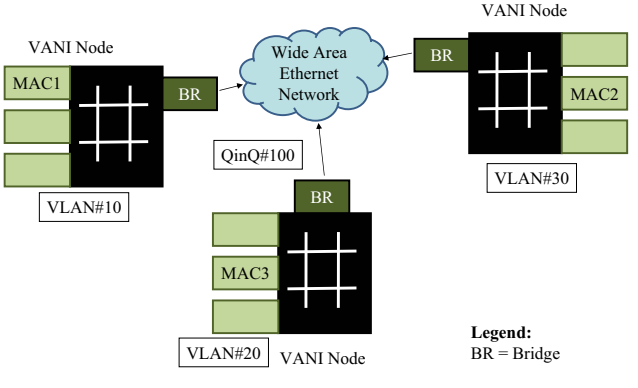


Fig. 8. Connecting VANI nodes in Ethernet layer

in an experiment communicate using a specific VLAN which is unique to the VANI node. If an experiment needs to operate across multiple VANI nodes (for instance, to test a new layer three protocol), the VANI wide area network has to be able to transfer Ethernet frames. In this case, a unique Q-in-Q tag [21] would be assigned to the experiment. The bridge service would be used to re-frame the internal tagged Ethernet frames to the wide are Q-in-Q frames and the destination bridge would do the reverse operation, and deliver the Ethernet frames to the destination MAC/VLAN in the destination VANI node.

Since Q-in-Q tagged Ethernet frames might not be available in a wide area network, we are able to define public MACs that can be used for redirecting traffic to an internal MAC/VLAN by the bridge service. This functionality would enable any other Ethernet-based experiment to send Ethernet frames to a resource in another experiment through the bridge service.

5.3 Experimentation with L3 Protocols

Figure 9 shows how the testbed could be used to test a new layer three protocol in a a large scale and distributed environment using proxy nodes. In this setting, the new L3 protocol is tunneled within IP payload to a resource inside a VANI node, and then that resource strips off the IP header and feed the new L3 packet over the VANI wide are Ethernet network.

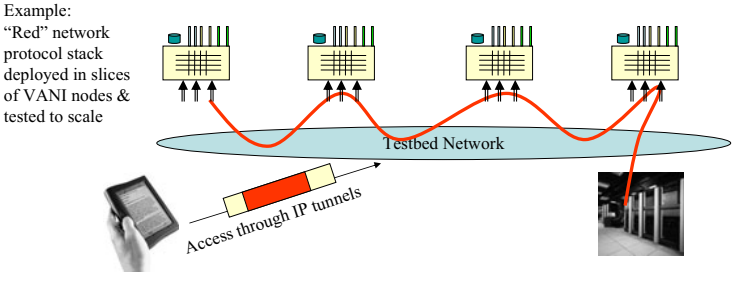


Fig. 9. Large scale experimentation with new L3 protocols

6 SW-Based Resources in VANI

One of the main contributions in our testbed control and management plane is that we could encapsulate any software or hardware resource in our testbed as a service. To do so, the resource can be virtualized, and abstracted as a service component that follows a generic resource WSDL template. Then it can be registered into the control plane and made available to other researchers. Details on how this task can be accomplished have been discussed in the control and management plane section in this paper.

Examples of such resources as a service are any hardware function or resource that could be reused in different applications and experiments such as hardware accelerators for encryption, decryption, content conversion, and content compression/decompression.

Also other reconfigurable hardware modules such as NetFPGA could be virtualized and offered to the researchers on an on-demand basis.

Other types of processing nodes could also be offered to the researchers as a resource. For example, Amazon Elastic Computing Cloud (EC2) nodes [22], GENI virtual processing nodes, VMWare-based virtualized processing nodes [23], or Graphics Processing Units (GPUs) could be controlled and managed by VANI-CMP.

Moreover, software services such as database service, BPEL orchestrator engine and Complex Event Processing (CEP) engine, could be developed and/or deployed on top of current virtual resources and made available to the researchers through VANI-CMP. Currently, we have developed and deployed several software-based resources as service components in VANI including a database service, BPEL orchestrator engine, and a sensor service.

7 Federation with GENI

GENI is an initiative to create a large scale experiment through federation between different testbeds. Federation in GENI is done using GENI wrappers. A GENI wrapper is developed for each testbed and testbeds could connect to each other through them. In VANI, we developed a wrapper for control and management plane, and through that we invoke GENI wrapper operations to get a node on any GENI testbed. We tested our wrapper with PlanetLab GENI wrapper and managed to obtain a PlanetLab processing node through our VANI-CMP.

In VANI, researchers are able to get a PlanetLab processing resources using VANI generic resource template. Since PlanetLab does not support storage service, and also does not support other VANI requirements such as processing and bandwidth requirements, access to PlanetLab processing resources would not support these functionalities. Figure 10, shows the structure of interconnection between VANI and PlanetLab

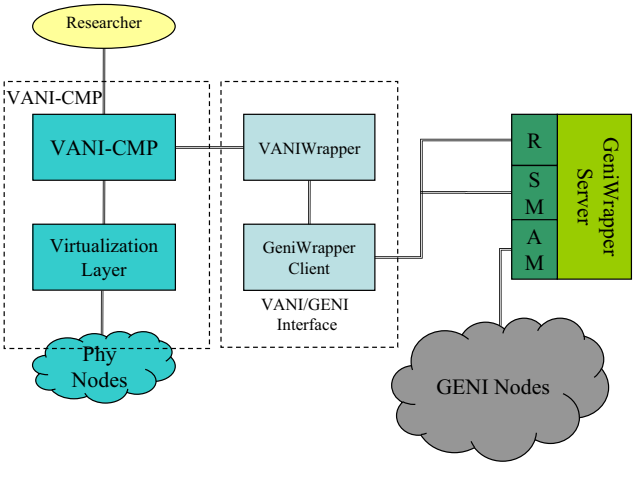


Fig. 10. Connecting VANI to GENI

through the GENI wrappers. Currently, we are in the development phase of offering VANI resources to GENI researchers through the VANI wrapper.

8 A VANI Node

A VANI node is composed of the resources described in this paper, their corresponding virtualization software, control and management software, and the storage service. A VANI node can be totally deployed on a computer cluster composed of normal computing blades, and manageable Ethernet networking elements. The basic resources in a VANI node are the processing resource, the storage service, and the fabric service for the network virtualization that are deployed on a computer cluster.

All other resources and the control and management software are deployed on these basic services. In addition, all other software-based resources, and the virtualization layer for resources like reconfigurable hardware resource, and the VANI wrapper for connecting to GENI testbeds are also deployed on these basic resources.

The only elements that cannot be found in a normal computer cluster are the reconfigurable hardware resources, the gateway and bridge services, and required 10GE Ethernet switches. These resources are also co-located with the computing cluster to provide the WAN connectivity and to enable running experimentation with the reconfigurable hardware resource.

9 Performance Evaluations

Up to now, we presented the VANI architecture and we discussed different aspects of its design. To find if the currently developed resources can meet VANI design requirements, we performed several experiments on those resources. In this section, we present performance measurements on two key physical resources that have been virtualized and offered to the researchers in VANI. The first one is the reprogrammable hardware resource, and the next one is the processing resource. Our main focus in this part would be to see if we could guarantee the promised quality of service to the researchers that use these resources in their experiment.

9.1 Reprogrammable Hardware Resource

By introducing a virtualized and reprogrammable hardware resource in VANI, we enable researchers to test new networking algorithms and protocols using high performance and high throughput hardware resources. To do so, we virtualized BEE2 boards developed in the University of California at Berkeley. A BEE2 board consists of one controlling FPGA, and four high capacity Xilinx Vertex-II FPGAs (figure 11) that can be programmed by users. Each FPGA has four 10GE interfaces, and 4 GB of memory.

In VANI, a researcher can get a set of FPGAs on a BEE2 board, and can ask for on-board inter-chip communication channels which can carry up to 5 GigaBytes per second (GBps). The detailed design of BEE2 virtualization system and introducing it as a resource in VANI can be found in [16]. Here, we present the performance measurements on this resource. The parameters of interest are the programming time of the

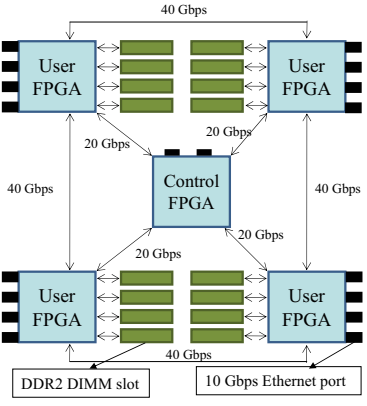


Fig. 11. Reprogrammable Hardware (BEE2 Board)

FPGAs through the virtualization software as well as the speed with FPGAs can send and receive data.

The first parameter is the time in which a researcher can program an FPGA through the testbed control plane. Also, we would like to know how this time would change if four researchers want to program all four FPGAs concurrently. To do so, we developed a bitstream that initializes all 10GE interfaces on the FPGAs and starts sending a burst of UDP/IP packets on one of its 10GE interfaces, and we programmed FPGAs through VAN-CMP using the generated bistream for several times. Table 1 shows the average maximum programming time that programming one, two, three, and four FPGAs take. As can be seen, it only takes 30 seconds on average to program an FPGA in the case where all four FPGAs are programmed concurrently, and this time is around 11 seconds if only one FPGA is programmed at a time.

This fast programming time allows a researcher to get an FPGA with four 10GE interfaces in less than a minute, and to run an experiment and return the FPGA back to the VANI resource pool as soon as it's not required.

The next experiment that we performed is to measure the speed with which the FPGAs can send and receive traffic. To do so, we developed a traffic generator using Verilog hardware description language, and we started sending traffic from one 10GE interface to another 10GE interface on the same FPGA, and we recorded the maximum bandwidth that we could receive in the hardware resource. We also compared this with the traffic statistics gathered by the Ethernet switch connected to the FPGA. We repeated this experiment several times and were able to send and receive Ethernet frames to the rate of 1GBps, which is equal to 8Gbps. The reason that we could not send more traffic is the 8/10 bit encoding mechanism for 10GE-CX4 interfaces, and 8Gbps is the

Table 1. Average maximum FPGA programming time

FPGAs	1	2	3	4
Programming Time (s)	11	17	24	30

maximum achievable traffic rate per port on a BEE2 board. In our measurements, this rate did not change if all ports started sending and receiving traffic at the same time since separate internal modules are controlling each port. This experiment shows that one FPGA alone can send and receive 32Gbps traffic. If a researcher get all four FPGAs on a BEE2 Board it is possible to send/receive traffic in the rate of 4x32=128Gbps.

We have used this reprogrammable resource in developing the high capacity gateway and bridge service for VANI, and we have developed a bandwidth control mechanism on this resource that controls and guarantees the rate at which one experiment could send and receive traffic to/from a wide are network. In the future, we will present our design for the gateway and bridge service, and we will present our performance measurements for this service as well.

9.2 Processing Service and Network Virtualization

Another main physical resource that we have virtualized is the processing service that uses Linux vServer software. There have been studies on processing virtualization techniques [24], and also specifically on Linux vServer [9]. Linux vServer performance evaluations show that this virtualization module has a very low overhead on overall system performance.

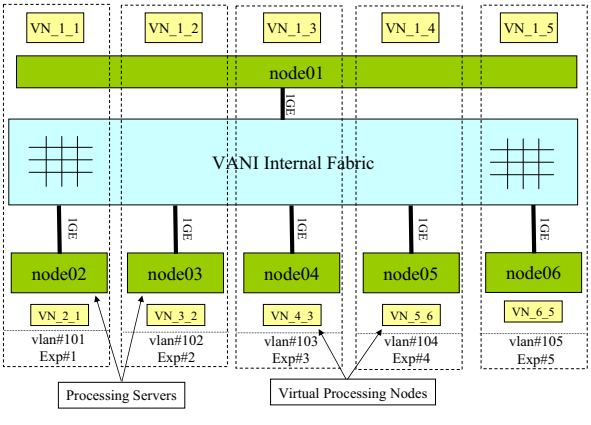


Fig. 12. Traffic measurement experiment topology

However, since we are also doing network virtualization in addition to the processing virtualization, we conducted two more experiments that were necessary to show that virtual processing nodes can have guaranteed access to the VANI network.

In our experiment, we virtualized cluster blades with dual Xen 1530 CPUs and 2GB of RAM and one 1GE interface. The Linux kernel version that we used was 2.6.16, and we used vServer 2.3.2. patch. The developed virtualization layer allows up to ten virtual nodes on a physical node. For this experiment, we initialized and launched 5 virtual nodes on a node named node01. We also launched 5 other virtual processing nodes on five separate servers with same capabilities described for node01. These nodes are

Table 2. UDP/TCP traffic measurements in MBytes per second (MBps)

node01 from/to	UDP	UDP (rl)	TCP	TCP(rl)
node02 (12.50MBps)	24.5/24.3	12.4/12.4	15∼35/24.7	12.3/12.3
node03 (18.75MBps)	24.5/24.3	18.8/18.8	15∼35/24.3	18.4/18.4
node04 (25.00MBps)	24.5/24.3	25.3/25.3	15∼35/24.1	24.8/24.6
node05 (31.25Mbps)	24.5/24.3	31.7/31.6	15∼35/22.1	31.3/31.1
node06 (31.25Mbps)	24.5/24.3	31.7/31.6	15∼35/23.2	31.3/31.1

named node02 to node06. Each of the virtual nodes in node01 belongs to an experiment that includes one other virtual node running on one of the other nodes. The topology and VLAN tags for experiments are shown in figure 12.

In this experiment, we measured the UDP and TCP traffic rate that each virtual node in an experiment could send and receive in different cases. The first case is to find out the maximum achievable rate when no limit is placed on the traffic rate and only one experiment is active. This rate is 122MB per second (MBps) for both UDP and TCP traffic which is equal to 976Mbit per second (Mbps). Table 2 show the achievable rate in different cases when all experiments are active and send as fast as they can. Since all experiments running on node01 try to send and receive on one 1Gbps Ethernet link concurrently, they get a different share of this available traffic in different cases.

In table 2, we show the maximum traffic rate in MBps between a virtual node on node01 and its corresponding virtual node on node02 to node06. The UDP and TCP columns show the maximum rate when all virtual nodes in all experiments send and receive UDP or TCP traffic, concurrently, without any rate limit mechanism in place. As it can be seen, because of the massive packet loss in this case, TCP cannot achieve a stable rate, and its rate changes from 15 to 35 MBps. These measurements prove the need for a rate limiting mechanism when different experiments want to run on a shared virtualized infrastructure.

The columns with (rl) show measurements when we limit the send and receive rate in experiments to (12.5), (18.75), (25), (31.25), and (31.25) MBps respectively, totaling to 118.75 MBps (950 Mbps). As can be seen, using the rate limit functionality we could achieve the bandwidth guarantee requirements (with maximum 1% deviation from the target rate) in a VANI node. Another case that we have studied is the case where all virtual nodes in one experiment start sending traffic to one virtual node concurrently. This would result in congestion on the shared link that is serving the destination virtual node. To solve this problem, we have developed a novel traffic control mechanism that we will present in a separate paper in future.

10 Conclusion and Future Work

Virtualized Application Networking Infrastructure (VANI) is a converged communications and computing network that facilitates the realization of an open applications marketplace using a service-oriented control and management plane capable of managing hardware-based and software-based resources.

The architecture of VANI is designed to allow rapid application creation and experiment setup using service-oriented approaches. VANI utilizes virtualized commodity physical resources such as processing, storage, and networking resources. It also includes reprogrammable hardware resources used for development and deployment of high scale and high throughput networking algorithms and protocols.

VANI is designed to enable experimentation with architectures and applications that provide responsiveness and quality of service by having processing, storage, and hardware acceleration resources in all its nodes. Example applications that are video streaming applications, new content delivery networks, as well as power-aware and green networking architectures. In addition, applications that require high performance computing and networking can benefit from VANI's reprogrammable hardware resource. This resource can be reprogrammed in a short time to run hardware-based networking algorithms and protocols, and can send and receive traffic rates up to 128Gbps. Currently, we are working on development of a novel green networked system on VANI. We are also in the process of designing novel functionalities into the VANI control and management plane to automate application creation and deployment.

References

1. Bellovin, S.M., Clark, D.D., Perrig, A., Song, D.: A Clean-Slate Design for the Next-Generation Secure Internet (2005), http://sparrow.ece.cmu.edu/group/pub/bellovin_clark_perrig_song_nextGenInternet.pdf
2. Stanford University Clean Slate Design For Internet: An Interdisciplinary Research Program, http://cleanslate.stanford.edu
3. 100x100 project, http://100x100network.org
4. GENI System Overview (September 2008), http://www.geni.net
5. Peterson, L.: PlanetLab: A Blueprint for Introducing Disruptive Technology into the Internet (January 2004), http://www.planet-lab.org
6. PlanetLab GENI Control Framework Overview (January 2009), http://www.geni.net
7. Hibler, M., Ricci, R., Stoller, L., Duerig, J., Guruprasad, S., Stack, T., Webb, K., Lepreau, J.: Large-scale Virtualization in the Emulab Network Testbed. In: Proceedings of the 2008 USENIX Annual Technical Conference, pp. 113–128 (June 2008)
8. GENI Control Framework Requirements (January 2009), http://www.geni.net
9. Fiuczynski, M.E., Pötzl, H.: Linux-VServer, Resource Efficient OS-Level Virtualization (June 2007), http://ols.108.redhat.com/2007/Reprints/potzl-Reprint.pdf
10. CANARIE Inc. CANARIE: Canadian Network for the Advancement of Research, Industry and Education, http://www.canarie.ca
11. Grasa, E., et al.: UCLPv2: A Network Virtualization Framework Built on Web Services. IEEE Communications Magazine 46(3), 126–134 (2008)
12. Szegedi, P., Figuerola, S., Campanella, M., Maglaris, V., Cervello-Pastor, C.: With evolution for revolution: managing FEDERICA for future Internet research. IEEE Communications Magazine 47(7), 34–39 (2009)
13. Gibb, G., Lockwood, J.W., Naous, J., Hartke, P., McKeown, N.: NetFPGA: An Open Platform for Teaching How to Build Gigabit-Rate Network Switches and Routers. Trans. on Education 51(3), 364–369 (2008)

14. Bannazadeh, H., Leon-Garcia, A.: On the Emergence of an Application-Oriented Network Architecture. In: Proc. of IEEE Int. Conf. on Service-Oriented Computing and Applications, SOCA 2007, Newport Beach, California, pp. 47–54 (June 2007)
15. Farha, R., Leon-Garcia, A.: Blueprint for an Autonomic Service Architecture. In: International Conference on Autonomic and Autonomous Systems, ICAS 2006, Silicon Valley, CA (July 2006)
16. Redmond, K., Bannazadeh, H., Leon-Garcia, A., Chow, P.: Development of a Virtualized Application Networking Infrastructure Node. In: Proceedings of the 3rd IEEE Workshop on Enabling the Future Service-Oriented Internet, Honolulu, Hawaii (December 2009)
17. Chang, C., Wawrzynek, J., Brodersen, R.W.: BEE2: a high-end reconfigurable computing system. IEEE Design and Test of Computers 22(2), 114–125 (2005)
18. Mathew, B., Sarang, P., Juric, M.: Business Process Execution Language for Web Services BPEL and BPEL4WS. Packt Publishing, Birmingham (2006)
19. Sun Microsystems Inc.: OpenESB: The Open Enterprise Service Bus, `http://open-esb.dev.java.net`
20. Sun Microsystems Inc.: Java Web Start Technologies, `http://java.sun.com/javase/technologies/desktop/javawebstart`
21. IEEE 802.1ad-2005, Virtual Bridged Local Area Networks Amendment 4: Provider Bridges (2006), `http://standards.ieee.org`
22. Murty, J.: Programming Amazon Web Services: S3, EC2, SQS, FPS, and SimpleDB. O'Reilly Media Inc., California (2008)
23. Inc VMWare. VMware: A Virtual Computing Environment (2001), `http://www.vmware.com`
24. Padala, P., Zhu, X., Wang, Z., Singhal, S., Shin, K.G.: Performance Evaluation of Virtualization Technologies for Server Consolidation (2007), `http://www.hpl.hp.com/techreports/2007/HPL-2007-59R1.html`

The Network Testbed Mapping Problem*

Rick McGeer[1], David G. Andersen[2], and Stephen Schwab[3]

[1] Hewlett-Packard Laboratories, Palo Alto, CA
rick.mcgeer@hp.com
[2] Department of Computer Science, Carnegie-Mellon University, Pittsburgh, PA
dga@cs.cmu.edu
[3] Cobham Analytic Solutions, Columbia, MD
Stephen.Schwab@cobham.com

Abstract. The *Network Testbed Mapping Problem* is the problem of mapping an emulated network into a test cluster such as Emulab or DETER. In this paper, we demonstrate that the Network Testbed Mapping Problem is $\mathcal{NP}$-complete when there is constrained bandwidth between cluster switches. We demonstrate that the problem is trivial when bandwidth is unconstrained, and note that a number of new proposals for data center networking have removed this barrier. Finally, we consider new heuristics in the bandwidth-limited case.

Keywords: network emulation, network embedding, graph partitioning.

1 Introduction

The *Network Testbed Mapping Problem* is the fundamental combinatiorial problem faced by network test environments such as Emulab [10] and DETER [1]. This is the problem of embedding instances of an emulated or virtual test network, which consist of a collection of nodes, links, and LANs, onto a physical cluster using virtual LANS so that the inter-node bandwidth requirements of the test network is satisfied. This problem is thought to be difficult, since inter-node bandwidth in a typical cluster often depends on the relative placement of the nodes within the cluster. Nodes which are placed on the same switch have sufficient bandwidth with adequately provisioned switches; however, since switch-switch bandwidth is typically much less than the aggregate bandwidth each switch allocates to its attached nodes, nodes attached to different switches in the data center have limited inter-node bandwidth.

* This research partially supported by the Defense Advanced Reseach Projects Agency under DARPA Subcontract 09-2080 for DARPA National Cyber Range Phase I - Prime Contract HR0011- 09-C-0039. Approved for Public Release, Distribution Unlimited. The views, opinions, and/or findings contained in this article are those of the authors and should not be interpreted as representing the official views or policies, either expressed or implied, of the Defense Advanced Research Projects Agency or the Department of Defense.

T. Magedanz et al. (Eds.): TridentCom 2010, LNICST 46, pp. 383–398, 2011.

For this reason, network testbed mapping is typically solved by heuristic methods. For example, Emulab and DETER use simulated annealing, a probabilistic hill-climbing algorithm originally used for VLSI placement in the 1980's. Simulated annealing, described below, works by starting from some solution and attempting to find a better, lower cost, solution. In this, it is similar to a large number of perturbative or progressive-improvement solutions. It differs from many in that it permits "hill-climbing" perturbations which temporarily lead to a worse solution but ultimately yield a superior solution. The intuition is that pure improvement perturbative methods such as the Greedy method find only local optima; permitting occasional "upward" moves which find a poorer solution offers the prospect of finding a better global solution than one can find from pure improvement methods.

The attraction of perturbative methods in general is that a perturbative algorithm works on *any* combinatiorial optimization problem. All one needs is a metric to measure the quality of a solution, an initial solution, and an ablity to move from one solution to another; given these three elements, the algorithm iterates repeatedly and eventually produces a better solution.

For example, in the case of the network testbed mapping problem, the initial solution is an assignment of nodes to switches; the metric is the deficit in actual vs. desired inter-node bandwidth; and a move is the exchange of pairs of nodes or the reassignment of a single node to a switch.

Given how simple perturbative methods are to use, one might wonder why they are not universally used. The answer is that the generality of perturbative methods means that they can only give fairly weak guarantees of solution quality. Despite the best efforts of many theorists, simulated annealing only gives stochastic assurances of quality, and the characteristics of the solution space on which simulated annealing works well is largely conjectural. Sorkin [14] persuasively argued that simulated annealing was likely to do well on self-similar solution spaces, but one cannot *a priori* determine which problems or instances have self-similar solution spaces.

Furthermore, it is easy to demonstrate hard problems for which simulated annealing will not do well. Consider, for example, an instance of SAT[6] with relatively few satisfying assignments. Starting from a random assignment, the odds that simulated annealing's random exploration of the solution space will find a satisfying assignment in reasonable time is vanishing; and in the case of an unsatisfiable instance, simulated annealing will prove nothing until it has explored the entire space.

Simulated annealing has proven to be succesful in testbeds at the scale of Emulab and DETER[12]. However, recent data suggests that it may not scale well in significantly larger settings [8][3]. Other interesting approaches have been proposed in the recent literature, notably [11], who exploited the fact that in many network graphs isomorphic subgraphs are detected. [16] broke the ground of observing that modification of the underlying substrate network can make the problem more tractable. In this paper, we show that some classes of underlying network substrate make this problem easy.

Given that the next generation of testbeds is projected to handle emulated test networks more than an order of magnitude larger than Emulab or DETER, this problem is worth revisiting. In particular, we would like to investigate the following questions:

1. When *is Network Testbed Mapping $\mathcal{NP}$-hard*? It is widely believed to be hard, but to date the reductions appearing the literature have been sketches in papers devoted to heuristic development. While this is quite common, it doesn't tell us precisely *why* NTM is hard, which often illustrates where and when it can be solved.
2. Though Network Testbed Mapping is $\mathcal{NP}$-hard, are there any circumstances under which it is easy? Can we design and engineer our range cluster networks to make it easy?
3. Though Network Testbed Mapping is $\mathcal{NP}$-hard, is it amenable to non-perturbative methods which can give stronger guarantees of the relative quality of solutions as compared to simulated annealing? For example, some variants of VLSI placement can be solved by the *Linear Programming with Randomized Rounding*[15] technique, which can give strong probabilistic guarantees. While approximation and probabilistic techniques typically are not conserved across polynmial reductions (or else every problem in $\mathcal{NP}$ would be easy to approximate, a strong and manifestly false result), often the nature of a reduction may provide a heuristic guide to good approximation and probabilistic techniques.

In this paper we consider these questions.

2 The Network Testbed Mapping Problem

The formal network testbed mapping problem is a simplification of the actual mapping problems solved within real testbeds. We assume, e.g., that all nodes are homogenous which means we can freely assign nodes to any switch. We ignore details of the interconnect fabric between the physical switches, assuming that each switch has a dedicated connection of some bandwidth to every other switch. These two assumptions simplify the problem to enable us to prove theorems about it, without assuming the problem away.

For example, one can easily capture heterogeneity of nodes by making the capacity (maximum port count) of each switch a vector, as opposed to scalar, quantity. The independence of inter-switch bandwidth is certainly a simplifying assumption, but for the purposes of complexity results it is unnecessary to introduce a more sophisticated model in the problem description – if this formulation of the problem is hard, then the general problem will be as well. Furthermore, the simplified problem generalizes very naturally to the general problem, as will be seen below.

Problem 1. The Network Testbed Mapping Problem. Given: *A network of switches, $s_1, ..., s_n$ with (port) capacities $C_1, ..., C_n$ and interswitch bandwidth capacities $B_{11}, ...B_{1n}, B_{21}, ..., B_{nn}$, and a test network of nodes $N_1, ..., N_m$ with*

internode bandwidth requirements $b_{11}...b_{mm}$. Question: *is there an injective assignment* $\mathcal{A} : N \rightarrow s$ *such that:*

$$|\mathcal{A}(u) = i| \leq C_i \ \forall i, 1 \leq i \leq n \tag{1}$$

and:

$$\sum_{\mathcal{A}(u)=i, \mathcal{A}(v)=j} b_{uv} \leq B_{ij} \ \forall i, j \tag{2}$$

(Where the summation is taken over all $\mathcal{A}(u), \mathcal{A}(v)$ *satisfying the equalities.) We say any mapping that satisfies (1) and (2) is* feasible.

The *capacity* of a switch is the number of edge nodes to which it can connect; the *bandwidth capacity* of a pair of switches is the total bandwidth between them. The two conditions on the problem are therefore that the assignment function not assign too many nodes to any switch, and the total assigned bandwidth between any pair of switches doesn't exceed the available bandwidth between that pair of switches. (We note that this formulation of the problem identifies only one edge link per edge node. In practice, a real test cluster would support n edge links per edge node. If we assume that the physical test cluster wiring connects all links from an edge node to a single switch, then the problem is unchanged. The formulation generalizes easily by permitting $\mathcal{A}(u)$ to be a fixed sized set for each u, where $|\mathcal{A}(u)|$ is the number of outgoing connections from u. The bandwidth constraint 2 becomes slightly messier, since there are now multiple paths between u and v. In practice, one cannot discuss this sensibly without some knowledge of the routing discipline used in the multiple path case. If routing between terminals is deterministic, as it usually is, then one can treat each NIC of each node as a separate node).

Remark 1. *The definition of network testbed mapping can extend to the conventional tree-of-switches network in a data center. Each switch in a tree-of-switches network has an upward bandwidth capacity* U_i*, a downward bandwidth capacity* D_i*, a set of descendant switches* c_i*, and a set of switches* r_i *for which it is the least common ancestor. The second constraint equation (2) in NTM is replaced by a pair of constraints, where the upward capacity must exceed the total bandwidth exiting the tree rooted at this switch. Formally, let*

$$u_i = \sum_{\mathcal{A}(u) \in c_i, \mathcal{A}(v) \notin c_i} b_{uv}$$

Then: $U_i \geq u_i$*. Similarly, the downward capacity must exceed the total bandwidth exiting the sub-tree + the total bandwidth passing through this switch as the root switch. Formally, let*

$$d_i = \sum_{\mathcal{A}(u) \in r_i, \mathcal{A}(v) \in r_i} b_{uv}$$

then

$$D_i \geq u_i + d_i$$

where u_i *is defined as above.*

Observation 1. *NTM is in $\mathcal{NP}$*

Proof. Given an assignment $\mathcal{A}$, conditions (1) and (2) are checked in linear time straightforwardly.

We will now demonstrate that NTM is $\mathcal{NP}$-hard, by reducing the known $\mathcal{NP}$-hard problem Minimum Graph Bisection.

Problem 2. *Minimum Graph Bisection.* Given: *an unweighted, directed graph $G(V, E)$, integer K.* Question: *is there a partition of V into sets V_1, V_2, $||V_1| - |V_2|| \leq 1$, such that:*

$$|(v_1, v_2) \in E | s.t. v_1 \in V_1, v_2 \in V_2 \leq K$$

See, for example, http://tracer.lcc.uma.es/problems/bisect/bisect.htm.

Theorem 1. *Network Testbed Mapping is $\mathcal{NP}$-complete.*

Proof. We reduce Mimimum Graph Bisection. Given an instance $G(V, E)$, integer K of Minimum Graph Bisection, derive an instance of Network Testbed Mapping as follows. We define two switches, s_1 and s_2, with capacities $C_1 = \lfloor(|V|/2)\rfloor, C_2 = \lceil(|V|/2)\rceil$, and interswitch bandwidth capacities $B_{12} = B_{21} = K$. Our test network of nodes is $N_1, ..., N_{|V|}$ (one node per graph node), with $b_{uv} = 1$ if (u, v) is an edge in E, $b_{uv} = 0$ otherwise.

Plainly, the derivation of an instance of Network Testbed Mapping is linear. Let $\mathcal{A}$ be the assignment which solves the Network Testbed Mapping instance. Let $V_1 = \{u|\mathcal{A}(u) = 1\}, V_2 = \{v|\mathcal{A}(v) = 2\}$. V_1 and V_2 are a bisection of V, and V_1 and V_2 are derived in linear time from $\mathcal{A}$. Further,

$$\sum_{\mathcal{A}(u)=1, \mathcal{A}(v)=2} b_{uv} \leq K$$

since $\mathcal{A}$ is a solution to the Network Testbed Assignment problem, we have:

$$|(u, v)|u \in V_1, v \in V_2 \leq K$$

so (V_1, V_2) is a solution to the Minimum Graph Bisection instance

This suffices to show that the Network Testbed Mapping is $\mathcal{NP}$-complete. In the next section, we will consider special cases where the problem is easy, and following that we will consider approximation and heuristic algorithms for the general case.

3 Polynomial-Time Special Cases

The reduction in the previous section permitted switches of arbitrary capacity and outgoing bandwidth. In practice, of course, switches have finite capacity and interswitch bandwidth is largely a function of the network topology. Further,

for many network topologies, such as the common tree-of-switches fabric we discussed earlier, interswitch bandwidth is not independent but competitive.

Some network topologies, however, permit easy solution of the Network Testbed Assignment problem. We characterize this set here.

In this discussion, we will assume network fabrics constructed from homogeneous switches with a specific bandwidth capacity per port, and the same port bandwidth is used on the network interface cards (NICs) of the various nodes. This simplification, and appropriate choice of units permits us to take B_{ij} as a non-negative integer and $0 \leq b_{uv} \leq 1$, and to take a node as equivalent to a unit of bandwidth. This has no substantive effect on the results below, but does permit us to state the results clearly and without introducing spurious constants.

Definition 1. *A network fabric is said to be* bandwidth-unconstrained *if and only if:*

1. *Inter-switch bandwidth capacities are independent; assigning bandwidth between one pair of switches doesn't affect available bandwidth between a different pair*
2. *For each pair of switches i, j:*

$$B_{ij} \geq \max(C_i, C_j)$$

where B_{ij} and C_i are taken from the definition of the Network Testbed Mapping problem.

Several scalable network fabrics have been proposed in the literature recently, notably the data center networks proposed by Al-Fares et al. [2] and by Scott et al. [13]. These conditions make network testbed mapping trivial.

Theorem 2. *Consider* any *Network Testbed Mapping problem where the fabric is bandwidth unconstrained, in particular $B_{ij} \geq \max(C_i, C_j)$ for all i, j.* Every *assignment $\mathcal{A} : N \to S$ such that $|\mathcal{A}(u) = i| \leq C_i \ \forall i$ is feasible.*

Proof. The constraint $|\mathcal{A}(u) = i| \leq C_i$ merely says that we can't assign more nodes to a switch than it can take, so consider *any* assignment $\mathcal{A}$ that meets this constraint. We must show for any pair of switches i, j, that:

$$\sum_{\mathcal{A}(u)=i, \mathcal{A}(v)=j} b_{uv} \leq B_{ij}$$

But this is trivial. Under our notational convention, $0 \leq b_{uv} \leq 1$ for all u, v, so:

$$\sum_{\mathcal{A}(u)=i, \mathcal{A}(v)=j} b_{uv} \leq |\mathcal{A}(u)=i, \mathcal{A}(v) = j|$$

and the constraint $|\mathcal{A}(u) = i| \leq C_i \ \forall i$ implies $|\mathcal{A}(u)=i, \mathcal{A}(v)=j| \leq \max(C_i, C_j)$, so:

$$\sum_{\mathcal{A}(u)=i, \mathcal{A}(v)=j} b_{uv} \leq |\mathcal{A}(u) = i, \mathcal{A}(v) = j| \leq \max(C_i, C_j)$$

and the fact that the bandwidth is unconstrained ensures:

$$\max(C_i, C_j) \leq B_{ij}$$

This theorem strongly motivates the use of scalable network architect ures as building blocks for network testbeds.

4 Heuristic Approaches

The premise of Theorem 2 is in fact stronger than necessary. It is simply a premise that can be stated entirely in terms of cluster topology, independent of the details of the test network to be embedded. One can achieve the same objective by significantly underpromising bandwidth to embedded nodes. In particular, if

$$\max_{u,v} b_{uv} \leq \min_{i,j} \frac{B_{ij}}{\max(C_i, C_j)} \tag{3}$$

then any assignment is feasible.

The proof is quite similar to the proof of Theorem 2. In fact Theorem 2 is the special case of (3) where:

$$\max_{u,v} b_{uv} \leq 1 \leq \min_{i,j} \frac{B_{ij}}{\max(C_i, C_j)}$$

Since by convention

$$\max_{u,v} b_{uv} \leq 1$$

and the premise of Theorem 2 is

$$1 \leq \min_{i,j} \frac{B_{ij}}{\max(C_i, C_j)}$$

These two inequalities are sufficient to maintain the general inequality needed for the theorem, but not necessary. Indeed, one can ensure the general invariant by a variety of means. The general invariant is a restriction of bandwidth demands of the test network relative to the bandwidth capacity of the cluster. For example, the DieCast system of Gupta et al.[7] articially enhances the relative capacity of a switch infrastructure to a test network by slowing down the test network's system clocks.

5 Network Testbed Mapping on a Leaf DAG

An interesting topology for network testbed mapping is a variant of a tree called a *leaf Directed Acyclic Graph*, or *leaf DAG*. A leaf DAG is a multi-rooted directed acyclic graph where internal nodes have a single parent, but leaves are permitted to have multiple parents. This specific class of topologies is often chosen for data center networks, because it permits alternate paths from the leaves while

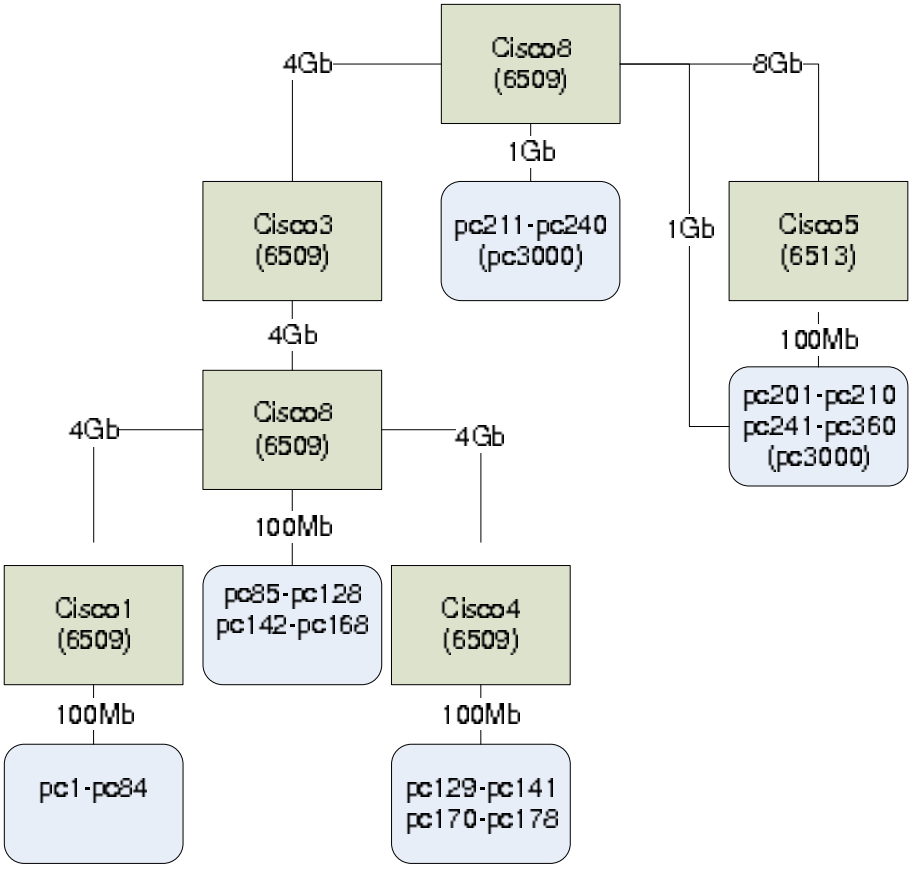


Fig. 1. Emulab Topology

retaining the autoconfiguration properties of standard Ethernet networks. The Emulab topology, for example, is a leaf DAG [4]. It is reproduced in Figure 1. The switches in this figure form a tree of depth three; Cisco8 is the root switch, with Cisco3 and Cisco5 as its (switch) children. Cisco1 and Cisco4 are children of Cisco3. Each switch has leaves attached. The network is a leaf DAG because pc201-210 and pc241-360 are multihomed, attached to both Cisco8 and Cisco5.

The reduction given above suggests a strong similarity between the network testbed mapping problem and the graph partitioning problem. The latter problem has been studied extensively, particularly in the field of VLSI placement. The first procedure to be suggested was the Kernighan-Lin procedure [9]. This procedure bipartitioned a graph to minimize the weight of the edges crossing between the two partitions. It is shown in Figure 2. This procedure requires the computation of the *gain* to be obtained by interchanging a pair of nodes v_1, v_2 where $v_1 \in V_1$ and $v_2 \in V_2$. The gain is simply the change in the sum of the

weights of the edges crossing the partition. An invariant of this procedure is that each node can be moved at most once, and hence we must differentiate between those nodes that have been moved and those that have not. Hence there are four sets of nodes in this procedure; V_1, the set of unmoved nodes on the left side of the partition, U, the nodes on the left side of the partition that were moved from the right, and the corresponding sets V_2 and V on the right side of the partition. So the gain from exchanging v_1 and v_2 is the weight of the edges that crossed the partition before the exchange minus the edges that crossed the partition after the exchange. The sum before exchange is just:

$$\sum_{v \in V_2 \bigcup V} w(v_1, v) + \sum_{u \in V_1 \bigcup U} w(v_2, u) - w(v_1, v_2)$$

with the subtraction of $w(v_1, v_2)$ to eliminate double-counting. Similarly, the sum after exchange is just:

$$\sum_{v \in V_2 \bigcup V} w(v_2, v) + \sum_{u \in V_1 \bigcup U} w(v_1, u) - w(v_1, v_2)$$

The total gain, G_{v_1,v_2}, is just the difference:

$$G_{v_1,v_2} = \sum_{v \in V_2 \bigcup V} (w(v_1, v) - w(v_2, v)) + \sum_{u \in V_1 \bigcup U} (w(v_2, u) - w(v_1, u)) \qquad (4)$$

The Kernighan-Lin procedure has been well-studied. Its cost is dominated by the need to recompute the potential gain at each step. Assuming fixed, low degree for each node in the graph, computing the gain for each pair is a constant-time operation. Since there are $O(n^2)$ pairs, where n is the number of nodes in the graph, each iteration takes time $O(n^2)$. Each iteration reduces the size of V_1 and V_2 by one node each; there are $O(n/2)$ nodes initially in each partition, the algorithm is $O(n^3)$.

The algorithm in Figure 2 may be iterated so long as cost improves; in this case the global runtime is $O(cn^3)$, where c is the initial partition cost.

In 1982, Fidducia and Mattheyses improved the performance of the Kernighan-Lin procedure to linear time[5]. Fidducia and Mattheyses made two key innovations to improve the performance of Kernighan-Lin. First, Fiduccia-Mattheyses does not *exchange* nodes as Kernighan-Lin does, but simply moves individual nodes from one side to another. The sides alternated to keep the partition balanced. Second, the gain for moving each node is precomputed, and nodes are stored in an array *indexed by the gain for moving the node*. This permits efficient selection of the best node to move. The valid indexes of the array are $-dw, ..., dw$, where d is the maximum degree of a node and w is the maximum edge weight. In the case of VLSI design (specifically, VLSI placement), which inspired the Fiduccia-Mattheyses procedure, typically $w = 1$ and d was guaranteed small due to physical considerations.

When nodes are moved, only the gain of neighbors need to be updated, and the neighbors themselves need to be moved to the appropriate list. Assuming

```
KernighanLin(G, V1, V2):
    U = V = ∅
    foreach pair v1, v2, v1 ∈ V1, v2 ∈ V2
        gain(v1, v2) = G_{v1,v2}
    bestCost = Σ_{v1∈V1, v2∈V2} w(v1, v2)
    bestPartition = V1, V2
    currentCost = bestCost
    while V1 ≠ ∅ and V2 ≠ ∅
        choose v1 ∈ V1, v2 ∈ V2 such that gain(v1, v2) is maximized
        V1 = V1 − v1, V2 = V2 − v2
        U = U + {v2}, V = V + {v1}
        currentCost = currentCost - gain(v1, v2)
        if currentCost < bestCost
            bestPartition = (V1 ⋃ U, V2 ⋃ V)
            bestCost = currentCost
        foreach pair v1, v2, v1 ∈ V1, v2 ∈ V2
            gain(v1, v2) = G_{v1,v2}
    return bestPartition
```

Fig. 2. Kernighan-Lin Algorithm

bounded degree, gain updates can be computed in constant time. Careful attention to data structures permits the move operation to be done in constant time. The procedure is shown in Figure 3.

As with Kernighan-Lin, the Fiduccia-Mattheyses procedure can be iterated while it continues to improve cost. The total complexity for the original procedure is bilinear in the graph size and the initial cost.

We can adapt the Fidducia-Mattheyses procedure to the network testbed mapping problem. Two significant modifications must be made to the algorithm:

1. The original procedure assumed that both the degree of each node and the weight of each edge were bounded by small constants. The former assumption holds in the network testbed mapping problem (if we treat a LAN as a single large node); the latter does not. In most network testbed mapping instances, internode bandwidth can be specified to any value up to one gigabit a second. Even if we make the simplifying assumption that bandwidth is only specified in units of a megabit a second, that still gives us a `gain` array on the order of several thousand entries, almost all empty, and this size cannot be neglected in considering the complexity of the algorithm
2. The Fidducia-Mattheyses procedure assumed that each node must reside on only one side of the partition. For leaf-DAGs, this isn't the case; in particular, referring to Figure 1, we see that we can assign up to 130 nodes to both sides of the root partition (represented by the switch Cisco8).

In order to cope with these two changes to the underlying use case of the procedure, we offer two modifications to the Fidducia-Mattheyses procedure.

```
FidduciaMattheyses(G, V1, V2):
    U = V = ∅
    foreach node v ∈ G:
        g = gain from moving v across the partition
        add v to gain[g]
    currentCost = Σ_{u∈V1, v∈V2} w(u, v)
    bestCost = currentCost
    bestPartition = (V1, V2)
    while V2 ≠ ∅ and V1 ≠ ∅:
        choose the highest gain node u from V1 and move it into V
        update the gains of each neighbor v of u
        choose the highest gain node s from V2 and move it into U
        update the gains of each neighbor t of s
        currentCost = currentCost - (gain(u) + gain(s))
        if currentCost < bestCost:
            bestCost = currentCost
            bestPartition = (V1 ⋃ U, V2 ⋃ V)
    return bestCost, bestPartition
```

Fig. 3. Fidducia-Mattheyses Procedure

1. We replace the array of gain by an exponential trie. This data structure, fundamentally a tree indexed by each digit of the magnitude of the gain, gives a guarantee of $O(\log dw)$ to find the node of maximal gain and $O(\log dw)$ for trie updates, giving an algorithm with a total complexity of $O(n \log dw)$ and space $O(n + \log dw)$. The conventional Fidducia-Mattheyses procedure has a time complexity of $O(ndw)$ and a space complexity of $O(n + dw)$. This suffices to make the Fidducia-Mattheyses procedure efficient when dw is large.
2. In addition to moving nodes, we permit a further operation: *clone*. The *clone* operation assigns a node to both sides of the partition, exactly as required for a leaf DAG. When a node is assigned to both sides of a partition, none of its edges cross the partition; essentially, for purposes of subsequent partition calculations, the node has been deleted from the graph. This is done in a preprocessing step, by deleting the cloned nodes from the initial partitions V_1, V_2. A further parameter, C, gives the total number of cloned nodes.

   Cloning is incorporated directly into the Fidducia-Mattheyses algorithm. The *clone gain* of a node is set equal to its weighted degree. Nodes of highest weighted degree are eliminated successively from the partition V_1, V_2, and the *clone gain* recomputed after each deletion. Further, other nodes are moved from side to side during this process to ensure that the invariant – $|V_1 \bigcup U| = |V_2 \bigcup V|$ is maintained. At each deletion, C is decremented. When it reaches zero, the preprocessing step halts, and the conventional Fidducia-Mattheyses procedure resumes.

The revised Fidducia-Mattheyses procedure is shown in Figure 4. The first revision, to accomodate a large range of gain values, is left opaque here for

```
FidduciaMattheysesWithCloning(G, V1, V2, C):
    U = V = ∅
    sortedNodes = v ∈ G sorted in decreasing order by total connections
    place top C nodes in sortedNodes into both U and V and delete from V1, V2, G
    foreach node v ∈ G:
        g = gain from moving v across the partition
        add v to gain[g]
    while |V1| > |V2| − 1:
        choose node v from V1 with highest gain, move to V
        recompute gains of neighbors
    while |V2| > |V1| − 1:
        choose node u from V2 with highest gain, move to U
        recompute gains of neighbors
    (S, T) = FidduciaMattheyases(G, V1, V2)
    return (S ⋃ U, T ⋃ V)
```

Fig. 4. Fidducia-Mattheyses Procedure with Cloning

reasons of brevity and clarity. The cloning operation appears as a preprocessor step.

Given the partitioning procedure, the assignment problem on a leaf DAG is straightforward; one simply partitions at the root, and recursively partitions at each subsequent level.

6 Experiments

A number of preliminary experiments were run using the revised Fidducia-Mattheyses procedure, without cloning, in Python 2.6. Great care should be taken in interpreting these results. Runtimes are quite short, especially in comparison to the Simulated Annealing procedures in the literature. However, it should be noted that the revised Fidducia-Mattheyses procedure does only a subset of the actions of the standard mapping procedures. In particular, it does not consider heterogeneity, nor multihoming. We view this procedure less as a replacement for the standard SA procedures, than as a preprocessor to help the SA procedure start from a known, fairly good solution.

In the first set of experiments, a set of structured graphs were constructed, with known partition properties. For these graphs, the weight of each node was set to one, and edges were assigned as follows for nodes $v_0, ..., v_n$

$$\begin{aligned} w(v_i, v_{i+1}) &= 10, i = 1 \bmod 2, i < n/2 \\ w(v_i, v_{i+2}) &= 50, i = 1 \bmod 2, i < n/2 - 1 \\ w(v_i, v_{i+3}) &= 50, i = 1 \bmod 2, i < n/2 - 2 \\ w(v_i, v_{i+1}) &= 50, i = 1 \bmod 2, n > i \geq n/2 \\ w(v_i, v_{i+2}) &= 10, i = 1 \bmod 2, n - 1 > i \geq n/2 \\ w(v_i, v_{i+3}) &= 10, i = 1 \bmod 2, n - 2 > i \geq n/2 \end{aligned}$$

The intent in this case was to construct a set of nodes with a known best partition. In particular, the outdegree of each node in this graph is four, and the best partition is 120.

We obtained the following results on the structured graph experiment, for a single partition. In all experiments, the maximum capacity of each side of the partition was set to 60% of the total graph weight.

The implementation was done in Python 2.6 under Windows Vista. All times for all experiments were measured on an HP Elitebook 2530p laptop. Times reported are user time, measured with the builtin os.times() function in Python. In all experiments, we report the number of nodes in the graph (the column MaxNum), the number of iterations of the Fidducia-Mattheyses procedure required for convergence, the total time, the initial partition weight, the final (post F-M procedure weight), and the total improvement.

MaxNum	Iterations	Time (ms)	Initial Weight	Final Weight	% Improvement
100	4	0	1480	60	95.95
200	4	0	2670	120	95.51
300	6	0	4080	110	97.30
400	5	0	5870	110	98.13
500	6	15.6	6050	170	97.19
600	5	0	8220	150	98.18
700	7	0	10260	170	98.34
800	7	0	11390	150	98.68
900	6	15.6	13490	470	96.52
1000	7	15.6	15820	160	98.99

Fig. 5. Results for the Structured Graph Experiment

These results are somewhat promising, but the graphs involved are highly structured and artificial. To see how the procedure performs on less structured graphs, we ran two more sets of experiments. For a second set of experiments, we ran on a sequence of random graphs, where each node was given a random number of connections, exponentially distributed with a mean of five connections per node. Weights were randomly given, again with an exponential distribution with a mean of 50. Results are given in figure 6.

These results still show improvement over a random distribution, but they are not nearly as dramatic as the structured graph results.

For a third set of experiments, we constructed random graphs using a normal distribution of connections (mean 5, standard deviation 1) with connection weights normally distributed (mean 50, standard deviation 10). The results are shown in Figure 7.

MaxNum	Iterations	Time (ms)	Initial Weight	Final Weight	% Improvement
100	3	0	10495	3282	68.73
200	5	15.600	23290	8934	61.64
300	3	0	32534	13959	57.09
400	6	15.600	45520	19429	57.32
500	4	31.200	55782	23103	58.58
600	14	78.001	70979	30760	56.66
700	6	15.600	76513	32958	56.92
800	5	0	87914	36076	58.96
900	4	46.8	105298	44836	57.42
1000	5	62.4	116133	50541	56.48

Fig. 6. Results for the Exponential Random Graph Experiment

MaxNum	Iterations	Time (ms)	Initial Weight	Final Weight	% Improvement
100	7	31.2	11243	5579	50.38
200	11	15.6	22190	10830	51.19
300	5	15.6	32758	16833	48.61
400	6	0	44414	22254	49.89
500	4	15.6	55008	28793	47.66
600	6	15.6	69878	34143	51.14
700	5	15.6	76520	39243	48.72
800	6	62.4	86433	44697	48.29
900	7	31.2	101490	52529	48.24
1000	8	31.2	111873	55381	50.50

Fig. 7. Results for the Normal Random Graph Experiment

Results are somewhat promising, and the relatively low runtimes even for large graphs indicates that this technique has some promise, perhaps as a preprocessor to the standard simulated annealing technique.

	Structured Graph			Random Exponential			Random Normal		
MaxNum	Random	Fidducia	Imprvmnt	Random	Fidducia	Imprvmnt	Random	Fidducia	Imprvmnt
100	24100	3840	84.07	104752	85942	17.96	112117	95622	14.71
200	67390	7560	88.78	295521	259936	12.04	306664	281520	8.20
300	112620	13560	87.96	540842	400591	25.93	533112	408881	23.30
400	166990	20600	87.66	879114	783680	10.86	804577	753200	6.39
500	225380	38200	83.05	1152150	845132	26.65	1129426	906998	19.69
600	271800	30960	88.61	1312668	1061238	19.15	1311184	1083681	17.35
700	318740	60200	81.11	1573438	1359245	13.61	1536886	1388424	9.66
800	409090	43720	89.31	2096036	1928859	7.98	1933719	1832996	5.21
900	495840	142360	71.29	2617836	2082327	20.46	2439612	2024681	17.01
1000	556160	77210	86.12	2693960	2032458	24.56	2669548	2251410	15.66

Fig. 8. Total Bandwidth Experiments for All Graph Classes

As a final experiment, we continued a full partition recursively of the network, until we reached 12 nodes in a partition – a number arbitrarily set as the notional capacity of a switch. We then summed the total edge weight of connections which crossed a partition, effectively the total interswitch bandwidth of the fabric. We measured this for both a random assignment of nodes to switches in the tree, and then a partition using Fidducia-Mattheyses. The results are presented in Figure 8.

The results are reflective of the individual partition experiments: the Fidducia-Mattheyses procedure is effective on structured graphs, and less so on random graphs.

7 Future Work

While a formal proof of the complexity of the network testbed mapping problem is important, future work remains in characterizing variants of the problem. In particular, the complexity bounds on solving generalizations of this problem for federations of testbed clusters will ultimately be relevant to the scalability of algorithms or heuristics for automatically mapping test networks onto large, constrained federations.

Acknowledgments

The authors gratefully acknowledge the assistance of the TridentCom general and program committees, and in particular many thoughtful comments on early versions of this paper from Rob Ricci.

References

1. The deter testbed: Overview, `http://www.isi.edu/deter/docs/testbed.overview.htm`
2. al Fares, M., Loukissas, A., Vahdat, A.: A scalable, commodity datacenter architecture. In: Proc. SIGCOMM (2008)
3. Duerig, J., Ricci, R., Byers, J., Lepreau, J.: Automatic ip address assignment on network topologies. Technical Report Flux Technical Note FTN-2006-02, University of Utah (2006)
4. Emulab. Emulab topology diagram, `http://www.emulab.net/doc/topo.pdf`
5. Fidducia, C.M., Mattheyses, R.M.: A linear-time heuristic for improving network partitions. In: Proc. 19th Design Automation Conference, pp. 175–181 (1982)
6. Garey, M.R., Johnson, D.S.: Computers and Intractability: A Guide to the Theory of NP-Completeness. W. H. Freeman & Co. Ltd., New York (January 1979)
7. Gupta, D., Vishwanath, K.V., Vahdat, A.: Diecast: testing distributed systems with an accurate scale model. In: NSDI 2008: Proceedings of the 5th USENIX Symposium on Networked Systems Design and Implementation, pp. 407–422. USENIX Association, Berkeley (2008)

8. Hibler, M., Ricci, R., Stoller, L., Duerig, J., Guruprasad, S., Stack, T., Webb, K., Lepreau, J.: Large-scale virtualization in the emulab network testbed. In: Usenix (2008)
9. Kernighan, B.W., Lin, S.: An efficient heuristic procedure for partitioning graphs. Bell Systems Technical Journal 49, 291–307 (1970)
10. Lepreau, J.: Emulab network emulation testbed, `http://www.emulab.net`
11. Lischka, J., Karl, H.: A virtual network mapping algorithm based on subgraph isomorphism detection. In: ACM SIGCOMM Workshop on Virtualized Infrastructure Systems and Architectures (VISA) (August 2009)
12. Ricci, R., Alfeld, C., Lepreau, J.: A solver for the network testbed mapping problem. Computer Communications Review (2003)
13. Scott, M., Moore, A., Crowcroft, J.: Addressing the scalability of ethernet with moose. In: First Workshop on Data Center - Converged and Virtual Ethernet Switching (DC-CAVES), ITC 21 (2009)
14. Sorkin, G.B.: Efficient simulated annealing on fractal energy landscapes. Algorithmica 6, 367–418 (1991)
15. Srinivasan, A.: Approximation algorithms via randomized rounding: A survey. Series in Advanced Topics in Mathematics, pp. 9–71. Polish Scientific Publishers PWN (1999)
16. Yu, M., Yi, Y., Rexford, J., Chiang, M.: Rethinking virtualnetwork embedding: Substrate support for path splitting and migration. ACM SIGCOMM Computer Communications Review 38(2) (April 2008)

TridentCom 2010

Practices Papers Session 5: Federated and Large Scale Testbeds

Managing Distributed Applications Using Gush

Jeannie Albrecht and Danny Yuxing Huang

Williams College, Williamstown, MA

Abstract. Deploying and controlling experiments running on a distributed set of resources is a challenging task. Software developers often spend a significant amount of time dealing with the complexities associated with resource configuration and management in these environments. Experiment control systems are designed to automate the process, and to ultimately help developers cope with the common problems that arise during the design, implementation, and evaluation of distributed systems. However, many of the existing control systems were designed with specific computing environments in mind, and thus do not provide support for heterogeneous resources in different testbeds. In this paper, we explore the functionality of Gush, an experiment control system, and discuss how it supports execution on three of the four GENI control frameworks.

1 Introduction

As network technologies continue to evolve, the need for computing testbeds that allow for experimentation in a variety of environments also continues to rise. In recent years, there has been significant growth in the number of experimental facilities dedicated to this purpose around the world, including GENI in the U.S. [1], FIRE in Europe [2], AKARI in Japan [3], and CNGI in China [4]. These testbeds play a crucial role in the development of the next generation Internet architecture by giving researchers a way to test the performance of new protocols and services in realistic network settings using diverse resources.

While these new testbeds offer many benefits to developers with respect to experimentation capabilities, they also introduce new complexities associated with managing computations running on hundreds of computing devices worldwide. For example, consider the task of running an experiment on one of these testbeds, which involves first installing the required software and then starting the computation on a distributed set of resources. When running a computation on a single resource, it is trivial to download any needed software and start an execution. However, ensuring that hundreds of devices are configured correctly with the required software is a cumbersome task that is further complicated by the heterogeneity—in terms of both hardware and software—of the resources hosting the experiment. Similarly, starting a computation requires synchronizing the beginning of the execution across a distributed set of resources, which is especially difficult in wide-area settings due to the unpredictable changes in network connectivity among the resources involved in the experiment [5].

In addition to configuration and deployment, there are many other challenges involved with keeping an experiment running in distributed environments, such as failure detection and recovery. In experiments that only involve a single resource, monitoring

T. Magedanz et al. (Eds.): TridentCom 2010, LNICST 46, pp. 401–411, 2011.

an execution and reacting to failures typically consists of watching a small set of processes and addressing any problems that arise. In experiments involving distributed applications, monitoring an execution consists of watching hundreds of processes running on resources around the world. If an error or failure is detected among these processes, recovering from the problem may require stopping all processes and restarting them again. The difficulties associated with these tasks are frustrating to developers, who end up spending the majority of their time managing executions and coping with failures, rather than developing new optimizations for increased application performance.

Experiment control frameworks are often used to alleviate the burdens associated with installing and executing software in distributed environments. They are designed to simplify the tasks associated with software configuration, resource management, failure detection, and failure recovery. However, many of these frameworks are designed with a single execution environment in mind, and thus are not adaptable to other environments with different resources. This limitation reduces the overall usefulness of the framework, and restricts its use to a single deployment platform. Further, as computers become more ubiquitous and the diversity of network-capable computing devices continues to grow, there is an increasing need for extensible management frameworks that support execution and experimentation in a variety of environments.

In response to the limitations of other application management frameworks, we developed Gush [6], an experiment control system that aims to support software configuration and execution on many different types of resources. Gush leverages prior work with Plush [7,8], and is being developed as part of the GENI project. Gush accomplishes experiment control through an easily-adapted application specification language and resource management system. The resource management system abstracts away the resource-specific details and exports a simple, generic API for adding and removing resources from an application's *resource pool*. The Gush *resource matcher* then uses the resources in the resource pool and the application's requirements as defined in the application specification to create a *resource matching*.

In this paper, we summarize the basic operation of Gush, with an emphasis on PlanetLab [9] resource management and experiment configuration in Section 2. In Section 3, we examine how the Gush resource matcher interacts with different types of resources in GENI (in addition to PlanetLab) to construct valid matchings and run experiments in two other GENI control frameworks: ORCA [10] and ProtoGENI [11]. We then discuss related work in Section 4. The fourth GENI control framework, ORBIT [12], focuses on wireless resources, which are currently not supported by Gush. In Section 5 we discuss how Gush can be extended to support execution in this environment and other wireless environments as well, and make general conclusions.

2 PlanetLab Application Management with Gush

Before discussing how Gush provides support for various types of resources, we first summarize the basic operation of Gush. The purpose of this section is to provide a high-level overview of how Gush works in a typical usage scenario. The design and implementation of Gush greatly leverages our previous work with Plush; however, this paper focuses on GENI-specific extensions to Gush. A detailed discussion of the design

and implementation of Plush can be found in [7]. It is important to note that Gush and Plush were both initially designed for the PlanetLab control framework, and the performance and operation of both systems on PlanetLab is largely the same. Thus, this section uses experimentation on PlanetLab as a motivating example. In the next section we discuss how Gush also supports other GENI control frameworks.

Gush is an experiment control framework that aims to simplify the tasks associated with the development, deployment, and maintenance of software running on a distributed set of resources. The two main components in the Gush architecture are the controller and the clients. The Gush controller process is responsible for managing the Gush client processes running on the distributed resources. The controller process is often run on the desktop computer of the Gush user (*i.e.*, the software developer).

The main role of the controller is to receive and respond to input provided by the user and guide the flow of execution on the clients. The clients are lightweight processes that run on specified ports on each resource involved in an experiment. When starting an execution, the controller initiates a separate TCP connection to each client process creating a communication fabric. For the remainder of the execution, the controller sends messages to the clients via the fabric instructing them to run commands and start processes on behalf of the user. The clients also periodically send the controller updates regarding their individual status or in response to failures. Using these status updates, the Gush controller can construct a single, global view of the progress of an experiment.

To manage an experiment using Gush, the user must provide the Gush controller with two pieces of information: an application specification and a resource directory. We discuss each of these components in detail in the following subsections.

2.1 Describing an Experiment in Gush

The *application specification* is an XML file that describes the flow of control for the experiment. In Gush, these are described using a set of "building block" abstractions that describe the required software packages, processes, and desired resources. The blocks can be arbitrarily combined to support a range of experiments in different environments.

Figure 1 shows a sample application specification for a very basic experiment. Starting at the top of the XML, we define the software required for our experiment. The software definitions specify where to obtain the required software, the file transfer method as indicated by the "type" attribute for the package element, and the installation method as indicated by the "type" attribute of the software element. In this particular example, the file transfer method is "web" which means that a web fetching utility such as wget or curl is used by the Gush clients to retrieve the software package from the specified URL. The installation method is "tar." This implies that the software.tar package has been bundled using the tar utility, and installing the package involves running tar with the appropriate extraction arguments.

Moving to the next main section in the XML, we define our experiment's *component*. This is essentially a high-level description of our desired resources. Each component is given a unique name, which is used by the component blocks later to identify which set of resources should be used. Next, the "rspec" element defines "num_hosts," which is the number of resources required in the component. The "software" element within the component specification refers to the "SimpleSoftwareTarball" software package that

```
<?xml version="1.0" encoding="utf-8"?>
<gush>
  <project name="simple">
    <software name="SimpleSoftwareTarball" type="tar">
      <package name="Package" type="web">
        <path>http://sysnet.cs.williams.edu/~jeannie/software.tar</path>
      </package>
    </software>
    <component name="GENIMachines">
      <rspec><num_hosts>20</num_hosts></rspec>
      <software name="SimpleSoftwareTarball" />
      <resources>
        <resource type="planetlab" group="williams_gush"/>
        <resource type="gpeni" group="gpeni_gush"/>
        <resource type="max" group="maxpl_gush"/>
      </resources>
    </component>
    <experiment name="simple">
      <execution>
        <component_block name="compBlock1">
          <component name="GENIMachines" />
          <process_block name="procBlock1">
            <process name="catProc">
              <path>cat</path>
              <cmdline><arg>software.txt</arg></cmdline>
            </process>
          </process_block>
        </component_block>
      </execution>
    </experiment>
  </project>
</gush>
```

Fig. 1. Gush application specification that is used to manage an experiment on 20 GENI resources. This trivial example simply runs "cat software.txt" on each resource.

was previously defined. Lastly, the "resources" element specifies which resource group the Gush controller will use to create a resource matching. In this case we are interested in 20 hosts from one of three GENI resource aggregates (all part of the PlanetLab control framework): PlanetLab [9], GpENI [13], and MANFRED (or MAX) [14]. Specifically, we want to use hosts assigned to the williams_gush, gpeni_gush, or maxpl_gush slices.

Finally, we define the experiment's execution using XML, and specify which component we want to use. Our simple example contains one component block that maps to our previously defined component comprised of 20 PlanetLab, GpENI, and MAX machines, and one process block consisting of a single process. This process runs the Unix command "cat software.txt" on each of our machines and then exits. (Note that software.txt is contained in software.tar.) More complicated executions are described in a similar fashion. Examples can be found on the Gush website [6].

2.2 Constructing a Resource Pool

In addition to the application specification, the user must also provide Gush with a resource directory. The resource directory is used to define resource pools in Gush, which are simply groupings of resources that are available to the user and are capable

of hosting an experiment. The simplest way to define a resource directory in Gush is by creating another XML file (typically called directory.xml) that lists available resources. This file is read by the Gush controller at startup, and internally Gush creates a *Node* object for each specified resource. A Node in Gush contains a username for logging into the resource, a fully qualified hostname, the port on which the Gush client will run, and a group name. The purpose of the group name is to give users the ability to classify resources into different categories based on application-specific requirements. We discuss how this name is used when creating a matching in the next subsection.

The resource file also contains a special section for defining hosts in the PlanetLab control framework. Rather than specifically defining which PlanetLab hosts a user has access to, the directory file instead lists which *slices* are available to the user. In addition to slice names, the user specifies their login to all available aggregate managers (which internally run their own version of the PlanetLab Central server) as well as a mapping ("port_map") from slice names to port numbers. At startup, the Gush controller uses this login information to contact each manager (PLC) directly via XML-RPC using the API specified by the Slice-based Facility Architecture (also called SFA or geniwrapper [15]). Each PLC server returns a list of hostnames that have been assigned to each available slice. The Gush controller uses this information to create a Node object for each host available to the user. The username for these hosts is the associated slice name, and the port is determined by the port_map. The group name is set as the slice name.

Note that for all PlanetLab aggregate managers, Gush assumes that the experimenter has a slice *a priori*. Since the current SFA does not provide APIs for creating slices, Gush also does not have the ability to create slices. The experimenter must register with the aggregate managers and upload their public key before using Gush. Once a slice has been created, Gush does have the ability to add and remove nodes from the slice. When a node is added to a PlanetLab slice, a Linux vserver [16] is created on the node, and the experimenter's key is eventually copied out to the vserver. However, the SFA does not provide the ability for Gush to receive any notification as to when the vserver is available for use.

Figure 2 shows a resource directory file that could be used in conjunction with the application specification in Figure 1. The first group of resources are identified as standard SSH resources, which means that they can be accessed using the standard SSH protocol with the username specified in the "user" attribute. The port specified indicates the port on which the Gush client process will run. The "group" attribute is optionally used to categorize resources. In our example, we only have two resources in the local group. Note that these resources have nothing to do with GENI. The next three groups of resources are all GENI resources. Gush uses XML-RPC to contact each aggregate manager's PLC database using the email addresses provided in the "user" tags to obtain information about the resources assigned to the williams_gush, gpeni_gush, and maxpl_gush slices respectively. The ports specified in the "port_map" tags indicate the ports on which the Gush clients will run for these slices. (Note that this syntax is likely to change, since the SFA APIs are changing rapidly.)

To gain an appreciation for the flexibility of the resource management abstractions in Gush, consider our example experiment from before as shown in Figure 1. Recall that in this example we are running "cat software.txt" on 20 PlanetLab, GpENI, and

```
<?xml version="1.0" encoding="UTF-8"?>
<gush>
    <resource_manager type="ssh">
        <node hostname="sysnet1.williams.edu:15400" user="jeannie" group="local" />
        <node hostname="sysnet2.williams.edu:15410" user="jeannie" group="local" />
    </resource_manager>
    <resource_manager type="planetlab">
        <user>jeannie@cs.williams.edu</user>
        <port_map slice="williams_gush" port="15415"/>
    </resource_manager>
    <resource_manager type="gpeni">
        <user>jeannie@cs.williams.edu</user>
        <port_map slice="gpeni_gush" port="15416"/>
    </resource_manager>
    <resource_manager type="max">
        <user>jeannie@cs.williams.edu</user>
        <port_map slice="maxpl_gush" port="15417"/>
    </resource_manager>
</gush>
```

Fig. 2. Gush resource directory file specifying different types of resources.

MAX hosts. Now suppose we want to change our application to instead run on the 2 cluster resources defined in the "local" group in our resource directory. To change our target resources, the only modification required to the application specification is in the component definition. Thus, if we change the value of num_hosts in the component definition in Figure 1 to 2 instead of 20, and also change the resource element to

```
<resource type="ssh" group="local"/>,
```

our experiment will run on our local cluster instead.

In addition to the resources defined in a Gush resource directory file, resources can also be added and removed by aggregate managers at any point during an experiment's execution. This is typically accomplished using an XML-RPC interface provided by the Gush controller. Managers that create virtual resources dynamically based on an experiment's needs, for example, contact the Gush controller with information about new available resources, and Gush adds these resources to the user's resource pool. If these resources become unavailable, the external service calls Gush again, and Gush subsequently removes the resources from the resource pool. This is especially useful for lease-based control frameworks, such as ORCA.

2.3 Creating a Matching

After the Gush controller parses the user's application specification and resource directory file, the controller's *resource matcher* is responsible for finding a valid matching—a subset of resources that satisfy the application's demands—for the experiment being managed. The matcher starts with the user's global resource pool consisting of all available resources, and then filters out the resources that are not in the group specified in the component definition. In our initial example, this includes all hosts not assigned to our three slices. Using the remaining resources in the resource pool, the matcher randomly picks 20 (as specified by num_hosts in our example) Node objects and inserts them into

a *resource matching*. The Gush controller then connects to the Gush clients running on the resources, and begins installing and configuring the required software.

If any failures occur during configuration or execution, the controller may choose to requery the resource matcher to find replacement resources. In the case of failure, the matcher sets the "failed" flag in the Node that caused the failure, removes it from the matching, and inserts another randomly chosen resource from the resource pool. This process is repeated for each failure throughout the duration of the experiment's execution. Note that resources that are marked as failed are never chosen to be part of a matching unless they are explicitly un-failed by the user.

3 Supporting ORCA and ProtoGENI

The preceding section discusses how Gush resources are internally maintained and organized into resource pools. It also describes how the Gush resource matcher uses these resource pools and the component definition section of the application specification to create matchings for the experiments being run. These basic abstractions for managing resources, in addition to our extensible XML-based specification language, are the keys to providing an adaptable framework that supports execution in a variety of environments and across multiple control frameworks. The previous section describes how Gush interacts with resources in the PlanetLab control framework. In this section, we take a closer look at how Gush interacts with slice controllers and aggregate managers in two other GENI control frameworks to select and use sets of resources for hosting experiments on different testbeds.

3.1 ORCA Control Framework

In addition to resources in the PlanetLab control framework, Gush also supports execution on virtual machines (VMs) that are dynamically created by the ORCA control framework [10]. A thorough description of the Gush XML-RPC interface and implementation details are described in [7]; we provide a brief summary here only to help the reader appreciate the flexibility of the Gush resource management framework. ORCA provides users with the ability to create clusters of Xen virtual machines [17] or Linux vservers [16] *on demand*. It is important to note that the VMs do not exist before Gush starts the experiment. Instead, the VMs are created dynamically at startup, and the Gush XML-RPC interface is used to add the new VMs to the Gush resource pool. The XML-RPC interface is also used to remove VMs from the Gush resource pool when the experiment ends or when a resource lease expires.

Suppose we want to run our sample application (Figure 1) on a cluster of VMs created by ORCA instead of 20 PlanetLab machines. Using Gush, we do not have to modify our application and execution elements in any way. We only have to add an <orca> element into the component's rspec definition. The Gush resource management framework abstracts away the low-level details associated with contacting ORCA and configuring the new VMs with the Gush client process. Figure 3 shows the resulting XML. The XML tags that are required in the new <orca> component correspond to supported VM attributes in ORCA, including OS type, memory, bandwidth, CPU share, and requested lease length.

```
<component name="VMGroup1">
  <rspec>
    <num_hosts>20</num_hosts>
    <orca>
      <num_hosts>20</num_hosts>
      <type>1</type>
      <memory>784</memory>
      <bandwidth>300</bandwidth>
      <cpu>75</cpu>
      <lease_length>12000</lease_length>
      <server>http://geni.renci.org/orca:8080</server>
    </orca>
  </rspec>
  <resources>
    <resource type="ssh" group="orca"/>
  </resources>
</component>
```

Fig. 3. Gush component definition that includes an ORCA VM cluster description

3.2 ProtoGENI Control Framework

In the spectrum of architectural design choices between PlanetLab and ORCA, ProtoGENI [11] lies somewhere in the middle. Like PlanetLab, ProtoGENI requires users to register in advance and upload a public key. Unlike PlanetLab, ProtoGENI also requires experimenters to specify a network topology for the nodes involved in the experiment. Gush currently does not provide any support for creating these files, so when using ProtoGENI, any necessary topology files must be created before using Gush. In addition, Gush contacts the ProtoGENI XML-RPC server using SSL; thus, experimenters should download an SSL certificate from the ProtoGENI website and save it locally before starting Gush.

After creating an account on the ProtoGENI website, specifying a network topology, and downloading an SSL certificate, experimenters may begin using Gush to deploy experiments on ProtoGENI. The first step involves configuring the resource directory with the information necessary to communicate with the ProtoGENI XML-RPC server. An example is shown in Figure 4. (Note that Gush currently uses the Emulab API [18] in ProtoGENI, which is why the resource manager's type is "emulab." We are in the process of upgrading Gush to use the new ProtoGENI API.) The pertinent information in this case includes our login name, port number, project ID, experiment ID, and network topology (NS) file. This information is sent to the ProtoGENI server to create and swap in an experiment (if it does not already exist) using the specified topology. Like ORCA, ProtoGENI creates virtual machines dynamically when an experiment is created. Unlike ORCA, the ProtoGENI server has not yet been extended to provide any callbacks to Gush to indicate when the resources are actually ready for use (although the API does provide a blocking RPC call that does not return until all resources are ready). Thus, if a researcher uses Gush to create an ProtoGENI experiment, they must wait some small period of time (less than 5 minutes on average) for the virtual machines to be created and the topology to be instantiated before running any experiments.

At this point, the execution of Gush proceeds in the same way as before: Gush contacts the ProtoGENI server to obtain a list of available resources, and then uses SSH

```
<?xml version="1.0" encoding="UTF-8"?>
<gush>
    <resource_manager type="emulab">
        <user>jeannie</user>
        <port>15420</port>
        <EmulabProjectID>Gush</EmulabProjectID>
        <EmulabExperimentID>gush</EmulabExperimentID>
        <EmulabNSFile>nsfile.ns</EmulabNSFile>
    </resource_manager>
</gush>
```

Fig. 4. Gush directory file that includes ProtoGENI (Emulab) resources

to login to these resources and configure them accordingly. Upon the completion of an experiment, Gush uses the ProtoGENI XML-RPC interface to swap the experiment out, which shuts down all virtual machines associated with the experiment.

4 Related Work

With respect to related work, it is worthwhile to point out the key differences between this paper and [7]. The goal of this paper is to highlight how Gush manages resources in a variety of environments, while the focus of [7] is on the design and implementation of the experiment controller architecture. In addition, this paper discusses several extensions that were added to Gush specifically for GENI that are not part of Plush.

In the context of application management, Gush has similar goals as PlanetLab's appmanager [19] and HP's SmartFrog framework [20]. appmanager is designed exclusively for long-running services on PlanetLab, and thus cannot easily be extended to support different types of resources or control frameworks. SmartFrog is a Java toolkit that can be used to manage a variety of applications on resources that run Java. Unlike Gush, SmartFrog is not process-oriented, however, and thus only supports execution via Java classes and remote method invocation.

In GENI, every project has a set of testbed-specific tools for resource configuration and experiment management. The majority of these tools are not designed for extensibility, and thus do not support execution across testbeds or control frameworks.

5 Future Work and Conclusion

Gush is an application management framework that helps developers deploy and maintain software running on distributed sets of resources. Through a generic resource management interface and an extensible application specification, Gush supports execution on several different types of resources, including PlanetLab machines, ORCA virtual machines, and ProtoGENI hosts. However, some of the fundamental assumptions that Gush makes about resources—such as unlimited energy, always-on network connectivity, and process-oriented execution—do not support an emerging class of wireless and sensor-based resources. Moving forward, we hope to relax these assumptions to allow Gush to perform experiments on a wider variety of resources. For example, to support sensors or configurable routers, Gush may not connect directly to the resources,

but instead connect to a client proxy connected to the resource that can interpret Gush commands and perform the corresponding action on the sensor or router. Maintaining persistent TCP connections may also prove to be futile in wireless sensor networks comprised of mobile devices. The overall challenge in these environments is to provide environment-specific support from a common experiment controller framework. We are currently in the process of exploring ways to use Gush in some of these environments, including the ViSE [21] and DOME [22] testbeds. In doing so, our goal is to ultimately provide an adaptive experiment control framework for all GENI users.

References

1. GENI, http://www.geni.net
2. FIRE, http://cordis.europa.eu/fp7/ict/fire/
3. AKARI, http://akari-project.nict.go.jp/eng/conceptdesign.htm
4. CNGI, http://www.cernet2.edu.cn/en/bg.htm
5. Albrecht, J., Tuttle, C., Snoeren, A.C., Vahdat, A.: Loose Synchronization for Large-Scale Networked Systems. In: Proceedings of the USENIX Annual Technical Conference (USENIX), pp. 301–314 (2006)
6. Gush, http://gush.cs.williams.edu
7. Albrecht, J., Braud, R., Dao, D., Topilski, N., Tuttle, C., Snoeren, A.C., Vahdat, A.: Remote Control: Distributed Application Configuration, Management, and Visualization with Plush. In: Proceedings of the USENIX Large Installation System Administration Conference (LISA), pp. 183–201 (2007)
8. Albrecht, J., Tuttle, C., Snoeren, A.C., Vahdat, A.: PlanetLab Application Management Using Plush. ACM Operating Systems Review (OSR) 40(1), 33–40 (2006)
9. Peterson, L., Bavier, A., Fiuczynski, M., Muir, S.: Experiences Building PlanetLab. In: Proceedings of the ACM/USENIX Symposium on Operating System Design and Implementation (OSDI), pp. 351–366 (2006)
10. Irwin, D., Chase, J., Grit, L., Yumerefendi, A., Becker, D., Yocum, K.: Sharing Networked Resources with Brokered Leases. In: Proceedings of the USENIX Annual Technical Conference (USENIX), pp. 199–212 (2006)
11. ProtoGENI, http://www.protogeni.net
12. Orbit, http://www.orbit-lab.org/
13. GpENI, http://wiki.ittc.ku.edu/gpeni
14. MANFRED, http://geni.maxgigapop.net/
15. GeniWrapper, http://svn.planet-lab.org/wiki/GeniWrapper
16. Soltesz, S., Potzl, H., Fiuczynski, M., Bavier, A., Peterson, L.: Container-based Operating System Virtualization: A Scalable, High-performance Alternative to Hypervisors. In: Proceedings of the EuroSys Conference (EuroSys), pp. 275–288 (2007)
17. Barham, P., Dragovic, B., Fraser, K., Hand, S., Harris, T., Ho, A., Neugebauer, R., Pratt, I., Warfield, A: Xen and the Art of Virtualization. In: Proceedings of the ACM Symposium on Operating System Principles (SOSP), pp. 164–177 (2003)
18. White, B., Lepreau, J., Stoller, L., Ricci, R., Guruprasad, S., Newbold, M., Hibler, M., Barb, C., Joglekar, A.: An Integrated Experimental Environment for Distributed Systems and Networks. In: Proceedings of the ACM/USENIX Symposium on Operating System Design and Implementation (OSDI), pp. 255–270 (2002)

19. PlanetLab Application Manager, http://appmanager.berkeley.intel-research.net
20. Goldsack, P., Guijarro, J., Lain, A., Mecheneau, G., Murray, P., Toft, P.: SmartFrog: Configuration and Automatic Ignition of Distributed Applications. In: HP Openview University Association Conference (HP OVUA), pp. 1–9 (2003)
21. ViSE Project, http://vise.cs.umass.edu
22. Soroush, H., Banerjee, N., Balasubramanian, A., Corner, M.D., Levine, B.N., Lynn, B.: DOME: A Diverse Outdoor Mobile Testbed. In: Proceedings of the ACM International Workshop on Hot Topics of Planet-Scale Mobility Measurements (HotPlanet), pp. 1–6 (2009)

Interoperability of Lightpath Provisioning Systems in a Multi-domain Testbed

Alfred Wan, Paola Grosso, and Cees de Laat

University of Amsterdam – System and Networking Engineering research group
Science Park 107, 1098 XG Amsterdam, The Netherlands
{wan,p.grosso,delaat}@uva.nl

Abstract. On-demand services are a key feature of Future Internet architectures. Already today research networks around the world provide dedicated optical circuits (lightpaths) to scientists, to offer a higher quality of network service. Tools to request and instantiate these lightpaths within a single domain exist, but interoperability of the various provisioning systems in a multi-domain scenario is still in its infancy. We present here our work for end-to-end multi-domain circuits reservation and setup. We worked in a large-scale research testbed composed by different provisioning systems, spanning multiple network domains. We developed translation modules at the provisioning system boundaries that allow for seamless path reservation and setup. We conclude identifying the main elements for interoperability and provide some guidelines for future integration of research network provisioning systems.

Keywords: network provisioning systems, service and control planes, federated testbeds.

1 Introduction

Future Internet architectures will certainly provide on-demand network services, away from the one-size-fits-all feature of the current Internet, and toward a tailored approach to users and applications requirements.

Network researchers are focusing on this issue, backed up by several funding efforts. The European Commission promotes the Future Internet Research and Experimentation (FIRE) initiative [1]. The National Science Foundation in the USA does the same with the Global Environment for Network Innovation (GENI) project [2].

Research and Education Networks (NRENs) around the world have tackled the problem in the last years. Their answer has been to offer separated network circuits to users, called *lightpaths*. They have built hybrid network architectures, with coexisting IP and lightpath services in the same physical infrastructure. The SURFnet6 network in the Netherlands is an example: it has offered use of lightpaths since 2006 to the Dutch research community and academic centers.

T. Magedanz et al. (Eds.): TridentCom 2010, LNICST 46, pp. 412–427, 2011.

e-Science and Grid applications are the driving force behind this network model. These applications rely on guaranteed bandwidths and fixed latencies; concurrent traffic disturbs or, worse, completely disrupts their functioning. Traditional routed IP services are not sufficient, and scientists clearly profit from having a dedicated network path. For example, the high-energy physics experiments at the Large Hadron Collider at CERN [3] distribute their data to the computing centers via lightpaths; in very long interferometry radio astronomy experiments [4], telescopes also send the recorded signals to a correlation center via lightpaths.

We foresee that many of the solutions that satisfy scientists in the somehow closed world of research networks will be adopted in other communities, for large and small businesses and ultimately single users.

Requests for circuits within a single NREN are nowadays relatively easy to satisfy; each domain provides a reservation system to its end users, while ad-hoc software tools instantiate and configure the paths. The challenges arise whenever the circuits need to span multiple network domains, for the lack of interoperability between the various systems.

We present in this article the software tools we develop to facilitate interoperability along a multi-domain lightpath chain; we present the results obtained in several experiments; based on these we outline necessary future work to further ease the use of multi-domain lightpaths.

2 Provisioning Systems

Dedicated circuits in hybrid networks can be implemented at various network layers. Layer1 circuits are carried over a DWDM infrastructure, where different services run on different wavelengths [5]. Layer2 circuits are Ethernet paths extending into the WAN. A multi-domain circuit may comprise several segments at different layers, where (de)adaptation functions (de)encapsulate the data between layers [6][7].

Lightpaths can be static and persist for very long periods of time, to form an optical private network. They can also be dynamic and short-lived, being automatically unconfigured once the reservation time has expired.

A lightpath provisioning system always performs two major tasks, independently from the characteristics of a circuit:

- It provides the interfaces for path reservations to the end users. The user can make requests without being aware of the underlying technologies used e.g. SONET or Ethernet. This component constitutes the Network Service Plane (NSP);
- It configures the network equipment. These are the internal components that are vendor and equipment specific. We refer to them as the Control Plane (CP).

2.1 Network Service Planes

We identify four main components in a NSP, as shown in Fig. 1.

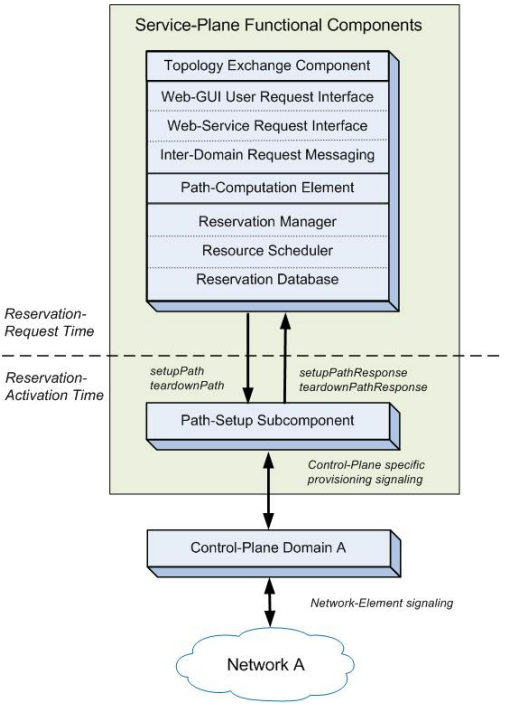


Fig. 1. The four components of the Network Service Plane. From top to bottom: the Topology Exchange Component, the Reservation Request Interface, the Path Computation Element and the Reservation Management System.

This model is based on observations of the design of existing provisioning systems. The User-Controlled LightPath (UCLP) provisioning system was one of the first tools that provided on-demand dynamic lightpaths. Later versions of UCLP [8] and other independent efforts rendered designs with common functionalities though with slightly different characteristics.

We distinguish between components that operate at reservation request time and components that operate at reservation activation time.

At reservation request time we have:

- A *Topology Exchange Component.* This exposes the local topology and facilitates topology exchange with other domains, a step necessary to setup multi-domain paths;
- A *Reservation Request Interface*. This allows users and neighboring domains to make reservation requests for paths in the local domain, and for the local domain

torequest reservations in other domains. In the most common scenario, a multi-domain path request originates from a user that utilizes a Web-GUI (*Web-GUI User Request Interface*). A user can also use a Web-Service client to do this (*Web-Service Request Interface*). If a domain receives a request that involves another domain, the NSP has to forward it to that domain and check its status using an *Inter-Domain Request Messaging* component;

- A *Path Computation Element.* This calculates a path between two endpoints specified in the request given the topologies of the domains involved;
- A *Reservation Management System*. This checks the availability of resources when it receives a request, schedule resources when a request can be accommodated and stores it in its local database.

The component that bridges the NSP and CP is the *Path-Setup Subcomponent* (PSS). It operates at reservation activation time. The PSS translates path-setup requests that carry the general request parameters, e.g., bandwidth, duration, VLANs, etc., to CP specific messages, e.g., path-refresh (RSVP-TE) messages that it then sends at regular intervals to the CP until the reservation ends.

3 The Combined Harmony-IDC Testbed

NRENs have developed independently their provisioning systems. These efforts resulted in software suites that cannot immediately interoperate. Although the purpose of each NREN NSP is the same, the functionalities it offers to the user differ; the communication between the NSP and the CP, for example the way to specify time and bandwidth parameters, or the way the communication is secured, vary in all implementations.

Several initiatives are underway to specify how different NREN NSPs should interact to successfully create inter-domain connections. One example is the Network Service Interface (NSI) workgroup in OGF [9]; one of its goals is to provide the interface between network domains for interoperability in a heterogeneous multi-domain environment. However, these initiatives attempt to formulate specifications from a top-down perspective, while the NSP designs of the various NRENs are often idiosyncratic and it is therefore challenging to unify their functions.

We have opted instead for a bottom-up approach and choose a specific usecase to determine what it takes to create multi-domain NSP interoperability and data-path setup.

We performed our research and development in a large-scale testbed, consisting of network resources from several Phosphorus [10] partners, UvA, SURFnet, i2CAT and VIOLA, and of a DICE partner [11], Internet2. Fig.2 depicts the testbed.

Our goal was to create a working lightpath starting in the Internet2 network and ending up in one of the Phosphorus domains, and vice versa.

In our testbed the i2CAT and VIOLA domains are controlled by the Harmony provisioning system. The UvA/SURFnet and the Internet2 domain are under control of the IDC provisioning system.

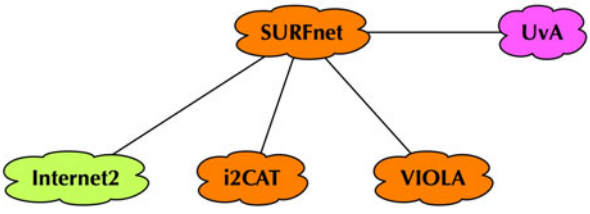


Fig. 2. The combined Phosphorus-DCN testbed where we conducted our work. We show a high level overview of the interconnected domains. Concerning the Network Service Plane, Internet2, SURFnet and UvA are under control of an IDC whereas i2CAT and VIOLA are under control of Harmony. SURFnet runs DRAC, i2CAT runs ARGIA and VIOLA runs ARGON.

3.1 Harmony

We participated in the development of Harmony, the NSP of the EU-funded project Phosphorus. The project has ended, but the network infrastructure is still available for experimentation.

The Phosphorus project aimed to make applications aware of the available computational and networking resources, and to provide dynamic, adaptive and optimized use of heterogeneous network infrastructures connecting these resources. In the project network domains connected to each other via inter-domain lightpaths, and each one ran its own NSP. Three NSPs are in use within the Phosphorus community:

- ARGIA, a commercial system used in the i2CAT network in Spain. It provides time slices or lightpaths to users or organizations, by virtualization of the network resources. An end user can create circuits on-demand based on the applications needs;
- ARGON, an MPLS/GMPLS enabled system used in the VIOLA testbed in Germany. ARGON stands for Allocation and Reservations in Grid-enabled Optical Networks; it provides interfaces for lightpath reservations;
- DRAC, a Nortel commercial product used in the SURFnet network in the Netherlands. DRAC is the Dynamic Resource Allocation Controller that also fulfills the purpose of lightpath reservation.

Harmony provides the APIs to control these three different NSPs.

3.2 IDC

We also contributed to the Inter-Domain Controller (IDC), the provisioning system of the DICE partners. The IDC is an architecture that facilitates both intra-domain path provisioning as well as the creation of circuits across domains. The IDC protocol defines formats and exchanges for:

- Network resource reservation requests messages;
- Intra-domain topology descriptions;

- Inter-domain topology exchange messages;
- Intra-domain network-element signaling and inter-domain request forwarding.

All functional components processing these requests and descriptions can be accessed through Web-Service (WS) interfaces. The WS interfaces can be categorized into:

- User to Network Interface (UNI) that provides users means to interact with the reservation system OSCARS;
- Internal Network to Network Interface (INNI) through which the IDC interacts with local resources, such as network elements;
- External Network to Network Interface (ENNI) that allows other IDCs to interact with the local IDC facilitating multi-domain path reservation and provisioning [12]. The possibility to interact with non-DCN Interdomain Brokers (IDBs) was not present when we started our work. We developed the ENNIs for this purpose as we will describe in the upcoming section.

  Furthermore, IDC interfaces to the DRAGON CP [13], that configures the network equipment to create paths.

4 Service-Plane to Service-Plane Translation Modules

In general, NSPs interpret advance reservation requests for lightpaths in such a way that they are stored in the reservation database unambiguously and consistently. Unambiguity regards the meaning of the parameters pertaining to the reservation, such as endpoint designations, reservation duration and bandwidth specification; consistency avoids conflicts such as overlapping reservations and requests exceeding the available bandwidth.

This also holds for reservation modification and reservation status inquiry requests, albeit in a somewhat lesser degree because while accommodating new reservation requests the NSPs already interpreted, disambiguated and consistently stored them in the reservation database.

When combining heterogeneous networks and NSPs to facilitate multi-domain lightpath reservation and provisioning, the syntax and semantics of the requests may have to be translated in order for the parameters in the requests to conform to the required format of the receiver, and for the requests to have the same operational semantics. For instance, endpoint-designations in the topologies of two domains may be different and have to be translated into the proper format of the receiving domains. In the case of a reservation modification, one domain may allow modifications of a reservation that has already been activated, while in another domain this may not be allowed: this causes differences in the effect of the same operation in different domains. For instance, in one domain a reservation may be cancelled while it is active, whereas in another domain this is not allowed.

We identified two general types of translations necessary at the boundaries between two heterogeneous NSPs, i.e., 1) Web-Service operation translations, and 2) Endpoint translations. For both types we produced code to perform the translations.

4.1 Web-Service Operation Translations

Operation translation requires that input used in a Web Services call in a certain NSP is mapped onto the correct input components in another.

Our first effort was to create translation modules to interface Harmony to IDC. Table 1 shows the operations we implemented.

Table 1. List of supported Web-Service operations. The first column indicates the name of the Web-Service operation. The second column shows if the operation is supported by IDC; the third column indicates support from Harmony. The last and fourth column shows whether we have implemented a translator service for the specific operation.

Operations	IDC	Harmony	Request translator
createReservation	✓	✓	✓
cancelReservation	✓	✓	✓
queryReservation (IDC) getStatus (NSP)	✓	✓	✓
isAvailable (NSP)		✓	
modifyReservation	✓	✓	
listReservation (IDC) getReservations (NSP)	✓	✓	✓
createPath (IDC) activate (NSP)	✓	✓	
refreshPath (IDC)	✓		
tearDownPath (IDC)	✓		

The translation may be:

- A parameter conversion whenever the two NSPs use different data formats to describe the same parameter. For example, start and end time of reservations are expresses in seconds from Epoch in one system and as time-day-month-year in the other;
- Be required to overcome that the results of a Web Service operation produce inconsistent or different results when invoked in the two systems.
- Necessary due to different security mechanisms deployed in the two systems. The IDC requires Web Services operation to be authenticated. IDC identifies requestors with X.509 certificates. Harmony on the other hand allows all users in its network domain to make requests. Our translation module will use a single X.509 certificate for all IDC requests coming from Harmony users.

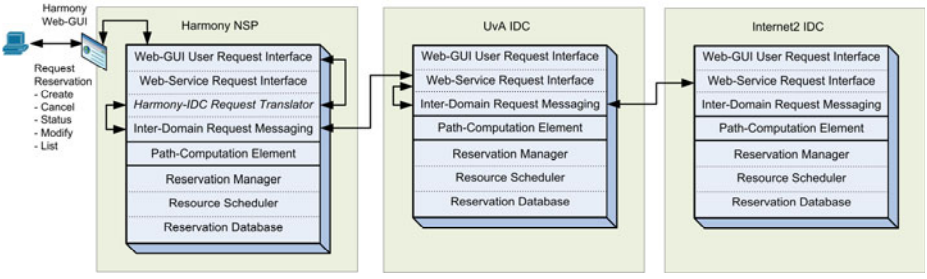


Fig. 3. The NSP components involved in multi-domain reservation requests. In the Harmony NSP (on the left) we insert our translator module; it translates a user request from Harmony format to the IDC format. Conversely, the responses sent to Harmony-NSP are translated from the IDC format to the format required by the Harmony NSP.

Fig.3 shows the insertion of our Harmony-IDC Request Translator among the components present in the Harmony NSP (on the left). The GUI User Request Interface couples to the Request Translator and the Request Translator interfaces internally to the Inter-Domain Request Messaging component.

4.2 Endpoint Translations

To provide interoperability between the NSPs in our testbed we also needed to identify the network endpoints in the proper format. Harmony and IDC express topologies in different manners.

Harmony identifies endpoints with Transport Network Addresses (TNAs). It uses for this a dotted decimal notation. IDC uses instead a URN identifiers for endpoint based on the topology schema developed by the OGF Network-Monitoring Working Group (NM-WG) [14], with specification of domain:node:port:link. Listing 1 shows an example of endpoint described according the IDC format.

We manually created topology descriptions for all the endpoints in the testbed in the proper formats. Listing 1 shows the XML structure used by IDC for the identification of endpoints. Using this schema an IDC endpoint would be described as:

link-id = urn:ogf:network:domain=uvalight.net:node=SURFnetOME4T_I2:port=s1p1:link10.9.2.10

Listing 1 – The XML schema representing a port in the IDC syntax.

```
<domain id="domain-id">
  <node id="node-id">
     <port id = "port-id">
        <link id="link-id">
            <remoteLinkId>remoteId</remoteLinkID>
        </link>
      </port>
   </node>
</domain>
```

Fig. 4 is a detailed representation of our topology. The goal is to connect a Phosphorus endpoint to an Internet2 endpoint. The Phosphorus endpoint is at the top left of the figure; the Internet2 endpoint is at the bottom left. In this use case both Web Services operation and endpoint translations take place.

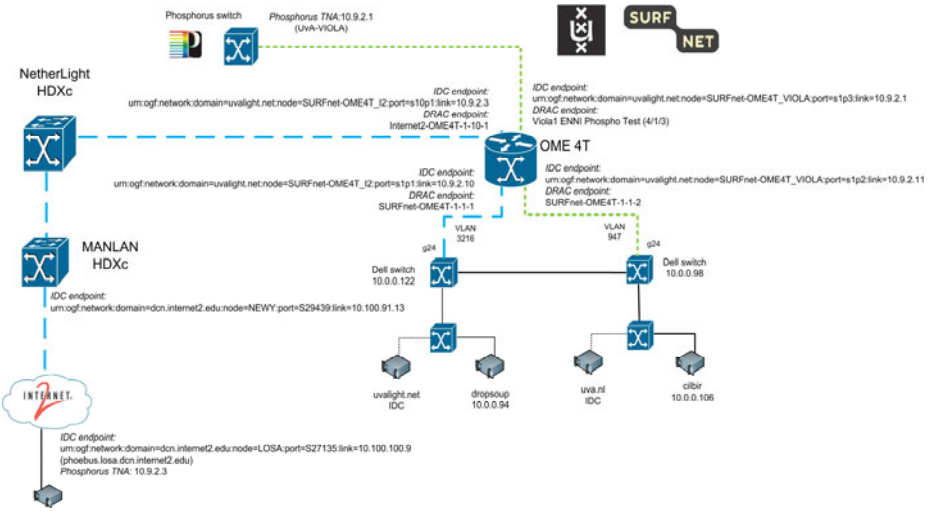


Fig. 4. Path setup scenario between a Phosphorus and an Internet2 host. Different line-patterns represent different VLAN tagging of the traffic running over the data-connections. The OME 4T is a Nortel OME 6500 and provisions Layer1 cross-connects between its ports. The Netherlight and MANLAN HDXc-s provide carrier services over TDM.

Harmony identifies the Phosphorus endpoint with TNA 10.9.2.1 and the Internet2 node with TNA 10.9.2.3. So the reservation request in Harmony contains TNA pair (10.9.2.1, 10.9.2.3) as the (*src_TNA, dst_TNA)* parameters for the *createReservation* Web Service operation. But the devices along the path between the source and the destination are under control of IDC and not Harmony. Harmony makes use of our Request Translator and performs:

- An endpoint translation, from the (*src_TNA, dst_TNA*) to an IDC (*src_URN, dst_URN*);
- A Web Service request translation, where Harmony uses our request translator to create a corresponding IDC Web Service *createReservation* request.
- A *createReservation* Web Service call to the IDC NSP.

Harmony in this manner delegates the actual setup to the IDC provisioning system, while offering to the user a single interface.

5 Service-Plane to Control-Plane Translation Module

The biggest challenge for Harmony/IDC interoperability stems from a fundamental incongruence in their respective designs.

The Harmony NSP is designed to interoperate with other single-domain NSPs that are built on top of CPs tailored to their individual domains, e.g., ARGON in the VIOLA domain. For this reason, CP functionalities such as path-setup and path-teardown components are absent in the Harmony NSP.

Conversely, the IDC design does not facilitate interoperability with any other NSP than another IDC. Moreover, CP functionality is an integral part of the IDC design and is tailored to interface with the DRAGON CP. The only possibility the IDC offers to interface with CPs other than DRAGON is to modify the Path-Setup Subcomponent (PSS) that signals path-setup, path-refresh and path-teardown events to the underlying CP.

The SURFnet/UvA network depicted on the right of Fig.4 is under control of IDC. Still IDC cannot configure paths because the DRAGON control plane cannot configure the OME testbed and Layer2 switches present in this portion of the network. The OME devices, a Layer1 switch from Nortel, can only be configured with DRAC; the Layer2 Dell switches can be controlled by using SNMP. In the following section we describe the high-level design of the PSS modifications we made to allow configuration.

5.1 Path-Setup Subcomponent

Fig. 5 depicts the components participating in the path setup through the two domains controlled by the IDC, the Internet2 domain and the UvA/SURFnet domains we saw in Fig.4.

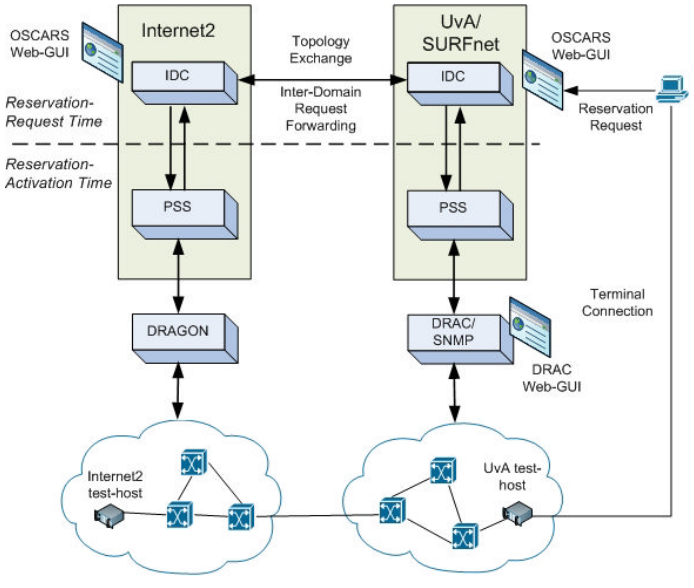


Fig. 5. The Internet2 domain and the UvA/SURFnet domain communicate in our testbed. At reservation activation time the Path Setup Subcomponent (PSS) communicate with the control plane to configure the network. In the Internet2 domain the PSS communicates with DRAGON; in the UvA/SURFnet domain the PSS triggers DRAC and SNMP calls to our equipment.

At reservation-activation time the IDC signals the CPs through the PSS. In the UvA/SURFnet domain we modified the PSS to signal the Layer1 network elements (OME 6500s) through interfacing with DRAC, and the Layer2 network elements (Dell Powerconnect 5324s) by interfacing with the SNMP agents running on these switches.

The parameters that are passed to the PSS are a Global Reservation ID (GRI), the requested bandwidth, a VLAN number, and a path consisting of a list of intra-domain hops calculated by the Path Calculation Element at reservation-request time.

The PSS has two main tasks: 1) to analyze the path and discover if there are ingress/egress hops (e.g., from/to the Internet2 domain) linked to intra-domain hops; these hop-pairs will be translated into DRAC request to set up a cross-connects on the OME 4T, and 2) to set up the Layer2 switches in order to retag traffic running between the Harmony and Internet2 domains, in case of VLAN mismatches.

The path contains an ingress/egress hop from the Internet2 domain to an internal UvA/SURFnet node, the PSS would perform the following translation:

- IDC hop urn:ogf:network:domain=uvalight.net:node=SURFnet-OME4T_I2:port=s10p1:link=10.9.2.3 translates into DRAC endpoint Internet2-OME4T-1-10-1;
- IDC hop urn:ogf:network:domain=uvalight.net:node=SURFnet-OME4T_I2:port=s1p1:link=10.9.2.10 translates into DRAC endpoint: SURFnet-OME4T-1-1-1

After mapping the ingress/egress hop to intra-domain hop pairs, the PSS calls DRAC with the corresponding DRAC-endpoint pairs to set up the Layer1 cross-connects on the OME 4T. In the case of the scenario depicted in Fig. 5 the only necessary cross-connect to be provisioned on the OME 4T is between *Internet2-OME4T-1-10-1* and *SURFnet-OME4T-1-1-1*. Subsequently, the port of the switch connected to the OME 4T will be configured by the PSS to accept traffic with the VLAN number included in the arguments of the PSS call, and to tag outgoing traffic with that VLAN number as well.

6 Workflow

Once the translation modules were ready we could prove the feasibility of an inter-domain path in our IDC-Harmony combined testbed. In this section we use an example end-to-end path to illustrate all the steps in the provisioning workflow. We use an example request for a circuit starting in the VIOLA domain and ending in the IDC domain.

6.1 Reservation Time

The user interacts with the NSP through the Reservation Request Interface. The UNI has two variants: a WS-UNI or a web-browser based UNI (WBUI). In both cases the user provides the reservation system with the parameters of the request: the source and destination IP addresses or transport-network addresses (TNAs), bandwidth, start time and duration. In the case of the IDC it's also possible to provide a VLAN number that will be used for the end-to-end Layer2 path.

Submitting requests is restricted to authorized users, however, the user authentication and authorization methods between Harmony and IDC differs considerably as we described before. In Harmony, requests from both the WBUI as well as from the WS-UNI are allowed from machines within the Harmony VPN. Both the IDC and DRAC WBUI require username/password authentication. The IDC WS-UNI requires messages to be signed by the user's private key, and DRAC authenticates WS requests through an included username/password token.

In Fig.6 we show the workflow associated to the setup of a path from the VIOLA domain to the Internet2 domain.

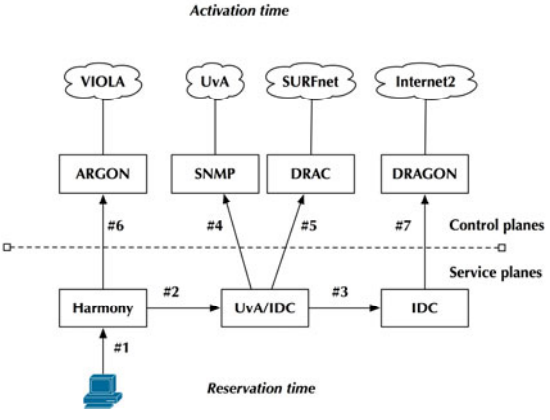


Fig. 6. Schematics of the workflow for a lightpath request starting in the VIOLA domain and ending in the Internet2 domain. The NSP operate at reservation time and the control planes operate at activation time.

At reservation time the user sends a requests for a path from VIOLA to Internet2 through the Harmony Web-GUI reservation form (*step #1*). The Harmony request is translated into an IDC request that is subsequently sent to the UvA IDC (*step#2*). This makes use of our Web Services and end-point translation modules.

The UvA IDC determines that the request comprises a multi-domain hop to the Internet2 domain, which causes the request to be forwarded to the Internet2 IDC (*step#3*). If this request can be accommodated a reservation is subsequently made in the Internet2 IDC and the UvA IDC.

6.2 Activation Time

At the start time of the reservation, both Harmony and the IDC signal the underlying control planes to set up the respective paths. The signaling consists of several distinct and heterogeneous processes:

- *DRAC-OME signaling* (*step #4*). The UvA IDC signals DRAC through its Path Setup Subsystem (PSS). DRAC processes this request as an immediate reservation and sets up the requested path through a Nortel OME testbed. This process sets up

the data-paths in the OME testbed and uses Nortel/SURFnet proprietary code and algorithms.

- *UvA-IDC-SNMP signaling* (*step #5*). The VLANs used in the Internet2 and Phosphorus domains differ. The switches in the UvA domain have to be configured to convert VLAN tags between the ones used by Phosphorus and Internet2. The signaling consists of sending SNMP commands through control connections from the IDC to the Dell switches. Here we use our PSS translation code.
- *Harmony NSP-ARGIA signaling* (*step #6*). This step is optional, because the reservation Harmony creates in ARGIA can both be of type automatic or signal-on-activation. Harmony-ARGIA (and other CPs in the Phosphorus testbed) signaling happens through modules dubbed 'adapters' that enable ARGIA and the Harmony NSP to exchange information about e.g., reservations and topology.
- *IDC-DRAGON signaling* (*step #7*). As in the case of IDC-DRAC signaling, the PSS signals the DRAGON CP to set up the reserved path at activation time. This happens through RSVP-TE messages.

7 Demonstrations and Results

We demonstrated path setup as described in the previous section between Internet2, i2CAT and VIOLA at Supercomputing 2008 (SC08; held in Austin Texas, USA) and the Internet2 Spring Member Meeting in April 2009 (Washington DC, USA). The latest demonstration has been at the Oct. 2009 GLIF workshop in Korea.

At SC08 and at the Internet2 Member meeting the focus was on the NSP-to-NSP translator (section 4). The demonstration in Korea focused on the service-plane to control-plane translation module (section 5.1). Fig.7 is the graphical representation of a path created with our software: one endpoint is in Amsterdam and the other is in Los Angeles.

Fig. 7. The graphical overview of an end to end path realized with our software. In the specific case we created a dedicated circuit between Amsterdam and Los Angeles. This makes use of our service-plane to control-plane software.

8 Challenges and Future Work

Currently in our testbed the request translation only flows from the Harmony NSP to the IDC and not the other way around (see Fig.3). To achieve full interoperability, we intend to develop a translator for the IDC that will be plugged in among the standard IDC components.

We also intend to enhance the IDC-DRAC integration. Currently, the IDC is designed to completely control the domain it resides in, and acts as both an SP as well as a CP. This means that the IDC is the only authoritative entity in one single domain and disregards the SP/CP that may already exists in the domain. In case of the IDC-DRAC interaction this means the IDC only sends path setup and path teardown requests to DRAC, while DRAC itself has a very sophisticated reservation system. Possible inconsistencies can arise because of this when the IDC tries to set up paths in the domain DRAC controls and there is alreday an existing reservation made by a DRAC user in the local domain.

A more general question concerning integration of provisioning system is whether interoperability will be achieved by following a centralized model or a federated model. Fig.8 exemplifies the two architectures.

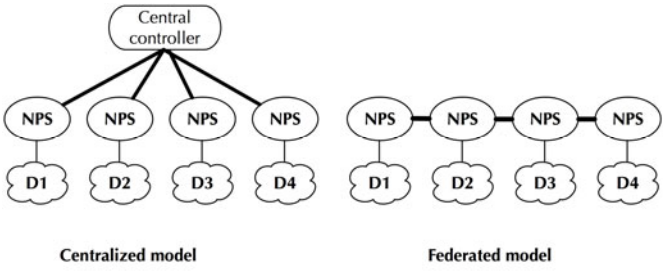


Fig. 8. Two possible architecture models for interoperability: a centralized model (left) and a federated model (right)

Centralized architectures see a vertical communication pattern (called north-south communication), where different domains relinquish control to a central management plane. This is the model implemented in Harmony. Federated architectures maintain a horizontal communication pattern (often called east-west communication). This chain model is the one realized by IDC.

We cannot predict which model will prevail. Our work provides tools to bridge one to the other by allowing a centralized model (Harmony) to co-exist and communicate with a federated model (IDC).

Furthermore, our work lead us to identify four components necessary for interoperability of provisioning systems that will need to be addressed in future work:

- *Terminology consistency*. Lightpaths is an all-encompassing term: at time lightpaths are configured purely at Layer1, at times they are Layer2 Ethernet paths. Consistent naming, clear technology information is a prerequisite for configuration across domains;
- *Topology descriptions consistency*. Network domains require some knowledge of the connected domain topology to set up on-demand paths beyond network borders.

For this reason domains must exchange network topologies. But what constitutes the essential and necessary information is matter of study. We expect standard topologies will come from efforts in the Open Grid Forum community, such as the one in the Network Markup Language working group [15];

- *Consistency of information on the status of resources*. Currently Harmony and IDC operate independently from each other at activation time: this means, a path that had been guaranteed during reservation might fail to be provisioned in one of the domains. Harmony or IDC will not be notified of this by the other NSP. As a result, the user will not have an end-to-end path available, while the NSP has guaranteed the existence of one. Notifications of resource availability and status in domains along a path can be a way to maintain consistency.
- *AAA features consistency*. This implies that higher or lower control on the resources must be harmonized across domains. As we mentioned in Sec. 4.1 Harmony and IDC have very different authentication models: with IDC authentication requiring X.509 certificates individually assigned to users. User authentication is a necessary condition to assign different permission levels and control of resources [16]. All NSPs should in the future implement this model.

9 Conclusions

Several challenges are present when trying to perform inter-domain circuit provisioning. Systems that work perfectly in one domain need to be interfaced to interoperate. We showed our solution to this problem in the case of a combined Harmony-IDC testbed. We developed translation modules at the service-plane-to-service-plane level, and at the service-plane-to-control-plane level. With these modules we could show the setup of intercontinental multi-domain lightpaths.

We identified many fundamental inconsistencies when providing tools for interoperability between Harmony and IDC. The creation of standardized interfaces between network domains, as pursued for instance in the NSI group at OGF, will provide the answer. Our work helps in that effort as it contributes to the identification of the necessary interoperability components.

Acknowledgements. We are grateful to SURFnet that sponsored this research through its Research on Network program. We would like to thanks our collegues Alexander Willner, Christian de Waal, Jan Gassen at the University of Bonn; Joan Antoni Garcia Espin, Jordi Ferrrer Riera, Carlos Baez Ruiz at i2CAT; and John Vollbrecht and Andrew Lake at Internet2. We also thank our collegues at SURFnet, Gerben van Malenstein and Hans Trompert.

References

1. Future Internet Research & Experimentation (FIRE) Initiative website, `http://cordis.europa.eu/fp7/ict/fire/`
2. Global Environment for Network Innovations initiative website, `http://www.geni.net`
3. The Large Hadron Collier website, `http://lhc.web.cern.ch/lhc/`

4. The European VLBI website, http://www.evlbi.org/
5. de Laat, C., Radius, E., Wallace, S.: "The Rationale of the Current Optical Networking Initiatives". iGrid 2002 special issue, Future Generation Computer Systems 19(6) (2003)
6. Dijkstra, F., van der Ham, J., Grosso, P., de Laat, C.: A Path Finding Implementation for Multi-layer Networks. Future Generation Computer Systems 25(2), 142–146 (2009)
7. Kuipers, F., Dijkstra, F.: *Path Selection in Multi-Layer Networks*. Elsevier Computer Communications 32, 78–85 (2009)
8. Grasa, E., Junyent, G., Figuerola, S., Lopez, A., Savoie, M.: UCLPv2: a network virtualization framework built on web services. IEEE Communications Magazine 46(3), 126–134 (2008)
9. NSI-WG website, http://forge.gridforum.org/sf/projects/nsi-wg
10. Phosphorus project website, http://www.ist-phosphorus.eu/
11. DICE initiative website, http://www.geant2.net/server/show/conWebDoc.1308
12. Lehman, T., et al.: Control Plane Architecture and Design Considerations for Multi-Service, Multi-Layer, Multi-Domain Hybrid Networks". In: INFOCOM 2007 High-Speed Networks Workshop, 26th Annual IEEE Conference on Computer Communications, Anchorage, AK, pp. 67–71 (2007)
13. Lehman, T., Sobieski, J., Jabbari, B.: DRAGON: a framework for service provisioning in heterogeneous grid networks Export. IEEE Communications Magazine 44(3), 84–90 (2006)
14. NM-WG website, http://nmwg.internet2.edu
15. NML-WG website, https://forge.gridforum.org/projects/nml-wg
16. Gommans, L., Xu, L., Wan, F., Demchenko, Y., Cristea, M., Meijer, R., de Laat, C.: Multi-Domain Lightpath Authorization using Tokens. Future Generation Computing Systems 25(2), 153–160 (2008)

The Great Plains Environment for Network Innovation (GpENI): A Programmable Testbed for Future Internet Architecture Research

James P.G. Sterbenz[1,5], Deep Medhi[2], Byrav Ramamurthy[3], Caterina Scoglio[4], David Hutchison[5], Bernhard Plattner[6], Tricha Anjali[7], Andrew Scott[5], Cort Buffington[8], Gregory E. Monaco[9,4], Don Gruenbacher[4], Rick McMullen[1], Justin P. Rohrer[1], John Sherrell[4], Pragatheeswaran Angu[3], Ramkumar Cherukuri[2], Haiyang Qian[2], and Nidhi Tare[4]

[1] The University of Kansas, Lawrence KS, USA
{jpgs,mcmullend,rohrej}@ittc.ku.edu
http://www.gpeni.net
[2] University of Missouri, Kansas City MO, USA
{dmedhi,rc5gd,hqian}@umkc.edu
[3] University of Nebraska, Lincoln NE, USA
{byrav,pangu}@cse.unl.edu
[4] Kansas State Univ., Manhattan KS, USA
{caterina,grue,ntare}@k-state.edu
[5] Lancaster University, UK
{dh,acs,jpgs}@comp.lancs.ac.uk
[6] ETH Zürich, Switzerland
{plattner,jpgs}@tik.ee.ethz.ch
[7] Illinois Institute of Technology, Chicago IL, USA
tricha@ece.iit.edu
[8] KanREN, Lawrence KS, USA
cort@kanren.net
[9] Great Plains Networks, Kansas City MO, USA
greg@greatplains.net

Abstract. The Great Plains Environment for Network Innovation – GpENI is an international programmable network testbed centered on a regional optical network in the Midwest US, providing flexible infrastructure across the entire protocol stack. The goal of GpENI is to build a collaborative research infrastructure enabling the community to conduct experiments in future Internet architecture. GpENI is funded in part by the US National Science Foundation GENI (Global Environments for Network Innovation) program and by the EU FIRE (Future Internet Research and Experimentation) Programme, and is affiliated with a project funded by the NSF FIND (Future Internet Design) Program.

Keywords: programmable Future Internet testbed, GENI, FIND, FIRE, PlanetLab, VINI, DCN, ResiliNets, PoMo, ResumeNet.

T. Magedanz et al. (Eds.): TridentCom 2010, LNICST 46, pp. 428–441, 2011.

1 Introduction and Motivation

The Great Plains Environment for Network Innovation – GpENI (pronounced with accent on the middle syllable and rhyming with GENI) is an international programmable network testbed centered on a regional optical network between The University of Kansas (KU) in Lawrence, Kansas State University (KSU) in Manhattan, University of Nebraska – Lincoln (UNL), and University of Missouri – Kansas City (UMKC) within the Great Plains Network, supported with optical switches from Ciena interconnected by Qwest fiber infrastructure, in collaboration with the Kansas Research and Education Network (KanREN) and Missouri Research and Education Network (MOREnet). GpENI is undergoing significant expansion to Europe and Asia. The goals of GpENI are to:

- Build a collaborative research infrastructure in the Great Plains region among GPN and other institutions
- Construct an international programmable network infrastructure enabling GpENI member institutions to conduct experiments in Future Internet architecture, supporting projects such as PoMo: PostModern Internetwork Architecture [1] and ResumeNet [2]
- Provide programmable optical infrastructure to the GpENI Midwest optical backbone, and expand optical connectivity to selected international sites
- Provide flexible infrastructure to support the GENI program as part of the PlanetLab control framework cluster B
- Deploy tools developed by GpENI and the GENI community such as Gush for experiment control and Raven for code deployment
- Provide an open environment for networking research community experiments

1.1 Previous Testbeds

It is important to remember that the idea of large-scale testbeds on which to conduct networking research is not new with GENI and FIRE. This section summarises a few of the most relevant previous efforts.

Gigabit testbeds. A set of testbeds was constructed in the early 1990s to further the state of high-speed networking research, funded by the US NSF and DARPA (Defense Advanced Research Projects Agency), managed by CNRI (Corporation for National Research Initiatives). Five separate testbeds were constructed: Aurora, Blanca, Casa, Nectar, and Vistanet [3], later supplemented by MAGIC [4].[1] The Gigabit Testbeds were a platform for research in high-speed networking, new bandwidth-enabled applications [5], and networked supercomputing (before the term *grid* was coined).

Active network testbeds. In the late 1990s, testbeds were constructed in the US and Europe to support active networks research. *Active networks* are programmable networks in which one of the programming modalities includes

[1] GpENI is currently using the research fiber originally installed for MAGIC at the University of Kansas.

capsules of mobile code that can dynamically program network nodes. In the US, the ABone [6] was constructed as part of the DARPA-funded Active Networks Program[2][7], to permit experimentation on programmable network languages, management and control [8], node operating systems [9], and security mechanisms [10]. The ABone had the goal of open access to the research community, with a separation into core nodes provided by program fundees, and edge nodes attached by the larger community. In Europe, the EU FP5 FAIN (Future Active IP Networks) [11] and related projects (e.g. LARA++ [12]) also investigated active and programmable networks, with testbeds constructed for experimentation. These active network architectures and testbeds permitted sharing of infrastructure by the simultaneous execution of active applications (AAs) in execution environments (EEs) on a network node operating system (NodeOS). While not generally recognised in this manner, the active network testbeds had many goals similar to that of GENI, and should be considered a conceptual precursor.[3]

Modern large-scale testbeds. More recently, two large scale testbed infrastructures have been constructed with the explicit goal of permitting open access for networking research. PlanetLab [13] is a large-scale worldwide infrastructure that permits users to run networked experiments at large scale. The infrastructure is shared using the *slice* paradigm. It is important to note that while PlanetLab permits experimentation in networked applications and end-to-end protocols, the network itself is not programmable, and experiments in lower-layer protocols can only be performed on overlays. VINI [14] is a version of PlanetLab that allows control over the network topology.

Emulab [15] is a network testbed consisting of a cluster of computing nodes interconnected by flexible network infrastructure, which permits researchers to experiment with network protocols and applications with complete root access to the systems. A number of Emulab facilities are located throughout the world, some of which provide access to external researchers in addition to the main facility at the University of Utah. Both PlanetLab and Emulab are the basis for GENI control frameworks; GpENI uses the PlanetLab GENI control framework.

1.2 Current Future Internet Initiatives

While a number of researchers were proposing alternatives to the Internet architecture as early as the 1980s (including research programs such as DARPA Next Generation Internet – NGI), there is now a general consensus in the research community that the current architecture is limiting in scale and support for emerging application paradigms such as mobile and nomadic computing and communications. Recent research initiatives include NSF FIND (Future Internet Design) [16] in the US, EU FP6 SAC (Situated and Autonomic Communications) [17], and the research component of FP7 FIRE (Future Internet Research

[2] Program managers were Gary Minden, Hilarie Orman, and Doug Maughan.

[3] Three of the GpENI PIs (Sterbenz, Plattner, Hutchison) performed research in these programs.

	GpENI Layer	Programmability
	experiment	Gush, Raven
7	application	PlanetLab
4	end-to-end	
3	router	Quagga, XORP, Click
	topology	VINI
2	VLAN	DCN
	lightpath	
1	photonics	site-specific

Fig. 1. GpENI Programmability Layers

and Experimentation) [18]. These research initiatives aim to investigate clean slate (greenfield) as well as incremental (brownfield) architectures to evolve the Future Global Internet architecture.

A key problem remains how to experiment with Future Internet architectures at reasonable scale. For this reason, the NSF GENI (Global Environments for Network Innovation) program [20], experimental component of the EU FP7 FIRE programme [18] , and Japanese JGN2plus [19] testbed plan to deploy large-scale programmable testbeds for experimentation of the Future Internet research. GpENI is a large-scale international programmable testbed currently under construction as part of the GENI, FIRE, and FIND programs.

1.3 GpENI Programmability and Flexibility

The defining characteristic of GpENI is programmability of *all* layers, as shown in Figure 1, implemented on a *node cluster* of general- and special-purpose processors, described in detail in Section 3. At the top layer Gush provides experiment control and Raven distributes code; both are software developed as part of the GENI program. Layer 7 and 4 programmability are provided by the GENIwrapper version of PlanetLab. At layer 3, programmable routers are implemented in Quagga, XORP, and Click, supplemented by any other technology for which GpENI institutions should choose to deploy.[4] Flexible network-layer topologies are provided by VINI. At layer 2, dynamic VLAN configurations are provided by DCN-enabled managed Gigabit-Ethernet switches at the center of each GpENI node cluster. GpENI institutions directly connected to the optical backbone use DCN-enabled Ciena switches to provide dynamic lightpath and wavelength configuration. At layer 1, the architecture even permits programmability at the photonic layer for switches that provide such support. Furthermore, each GpENI institution can connect site specific networking testbeds; plans include wireless, sensor, and cognitive radio testbeds (e.g. KUAR [22]). External users in the broader research community may request GpENI accounts with which to run network experiments.

[4] We are investigating including OpenFlow [21] as a core GpENI technology.

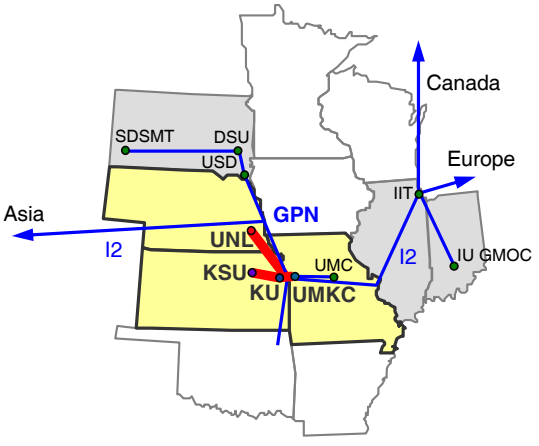


Fig. 2. GpENI Midwest Topology

2 Topology and Network Infrastructure

The core of GpENI is the regional optical backbone centered around Kansas City. This is extended by KanREN (Kansas Research and Education Network) to various GPN (Great Plains Network) institutions in the Midwest US. Connectivity in Kansas City to Internet2 provides tunneling access to the Canadian, European, and Asian GpENI infrastructure. Optical connectivity is currently in place in the UK between Lancaster and Cambridge, and will replace other GpENI L2TPv3 and IP tunnels as available. GpENI is growing, currently with about 200 nodes at 40 institutions in 20 nations. Institutions may connect to GpENI if they are interested in becoming part of the GpENI community, and have the resources to install, connect, and manage a node cluster.

2.1 Midwest US GpENI Core and Optical Backbone

GpENI is built around the core GpENI optical backbone centered in the Midwest, shown in Figure 2, among the principal institutions of KU, KSU, UMKC, and UNL, currently under extension to additional institutions, including the GMOC (GENI Meta-Operations Center). The optical backbone consist of a fiber optic run from KSU to KU to the Internet2 PoP in Kansas City, interconnected with dedicated wavelengths to UMKC and UNL, as shown in Figure 5.

Each of the four core institutions will have a node cluster that includes optical switching capabilities provided by a Ciena CoreDirector or CN4200, permitting flexible spectrum, wavelength, and lightpath configurations.[5]

2.2 GpENI European Topology

GpENI is extended to Europe across Internet2 to GÉANT2 and NORDUnet and then to regional or national networks, as shown in Figure 3. Currently,

[5] GpENI optical infrastructure is currently undergoing phased deployment; the UNL switch is installed and the KU switch is under procurement.

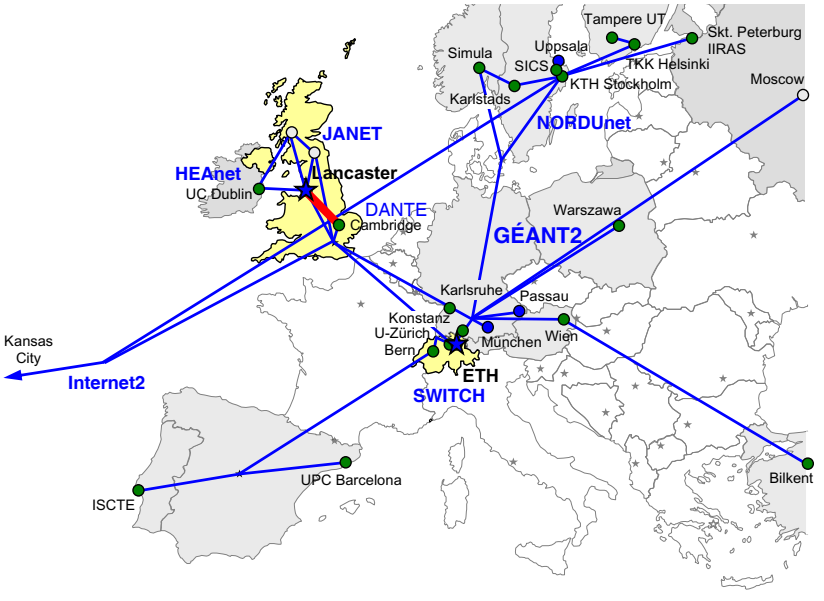


Fig. 3. GpENI European Topology

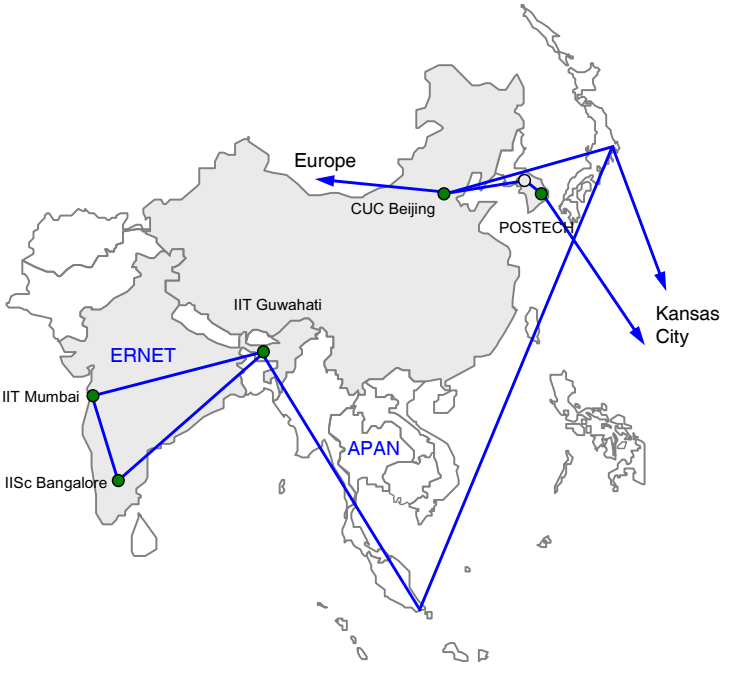


Fig. 4. GpENI Asian Topology

connectivity is achieved using L2TPv3 and IP tunnels. A direct fiber link over JANET is deployed between Lancaster and Cambridge Universities.[6] The principal European GpENI institutions are Lancaster University in the UK and ETH Zürich in Switzerland.

2.3 GpENI Asian Topology

Similarly, GpENI is extended to Asia across Internet2 to APAN, then to national research network infrastructure including ERNET. as shown in Figure 4.

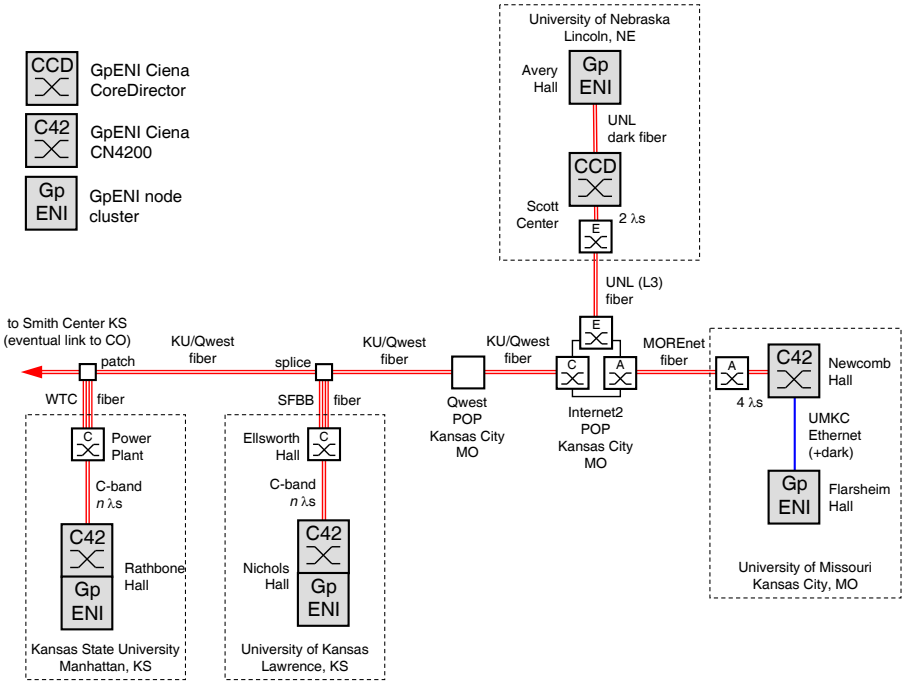


Fig. 5. GpENI Midwest Optical Backbone

3 Node Cluster Architecture

Each GpENI node cluster consists of several components, physically interconnected by a managed Netgear Gigabit-Ethernet switch to allow arbitrary and flexible experiments. GpENI uses a KanREN 198.248.240.0/21 IP address block within the `gpeni.net` domain; management access to the facility is via dual-homing of the Node Management and Experiment Control Processor. The node

[6] GpENI is currently working with research network providers to increase direct optical connectivity.

cluster is designed to be as flexible as possible at every layer of the protocol stack, and consists of the following components, as shown in Figure 6:

- GpENI management and control processor: general-purpose Linux machine
- PlanetLab control framework consisting of aggregate managers: MyPLC with GENIwrapper SFA (at KSU), myVINI (at UMKC), and DCN (at UNL)
- PlanetLab programmable nodes (enabling layer 4 and 7 experimentation)
- VINI-based programmable routers (providing flexible network topologies), with Quagga and other extensions such as XORP and Click (enabling layer 3 experimentation), as well as the ability for GpENI partners to install their own programmable routers
- Site-specific experimental nodes and testbeds, including software defined radios (e.g. KUAR), optical communication laboratories, and sensor testbeds
- Managed Gigabit Ethernet switch, providing L2 VLAN programmability and connectivity to the rest of GENI
- Ciena optical switch running DCN providing L1 interconnection among GpENI node clusters on the Midwest US optical backbone

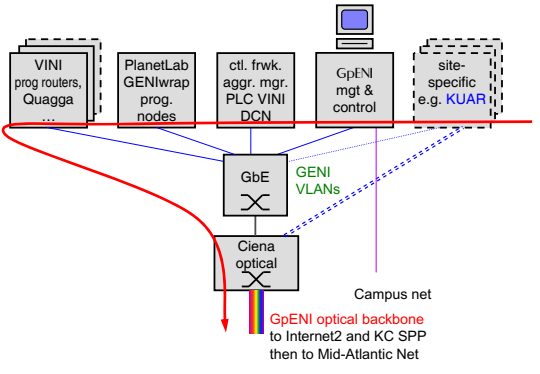


Fig. 6. GpENI Node Cluster

3.1 GpENI Management and Control

The GpENI management and control services are distributed across the Linux machines dedicated for the purpose at each of the node clusters. Open-source tools are used wherever possible to minimise the amount of GpENI-specific software development and maintenance required. Some of these services are installed at every node cluster, for example the Cacti monitoring tool [23] is used to monitor the per-port network usage on each of the Netgear Gigabit-Ethernet switches. Nagios [24] is used to monitor the status of individual nodes and services across all the clusters. Zenoss Core [25] is also being evaluated as an alternative to Nagios. On the other hand, some functions are specific to one site, such as the GpENI-Planetlab demo [27], running on Apache and hosted only on the management node at the KSU node cluster.

The control node for each cluster also provides firewall and NAT services using Firestarter [28] for that cluster's private subnet, thereby protecting insecure devices, such as the Netgear switch telnet and SNMP management interfaces, from direct exposure to the Internet.

3.2 Experiment Control

To ease experimenters role in resource discovery and preparation of experiments, the GENI User Shell (Gush) [29], provides a robust experiment control and management framework. Gush extends the PlanetLab control framework to implement an API that interacts with the GENI Clearinghouse. With Gush deployment on the GpENI aggregate, a user can reserve both PlanetLab and GpENI resources by downloading agents on the selected nodes and deploy their experiments subsequently with the help of the controller that communicates with the agent about node status. The Raven [30] provisioning service provides services such as the proper execution environment, software packages, configuration information and computational resources.

3.3 PlanetLab Control Framework Sub-aggregate

The PlanetLab Control Framework provides the control software to implement control plane, data plane, management plane, and operations plane functionalities in GENI Cluster B.[7] Cluster B contains a number of distinct *aggregates*: PlanetLab nodes, VINI nodes, Supercharged PlanetLab Platform (SPP) backbone nodes [31], OpenFlow switches [21], and GpENI node clusters. The GpENI aggregate consists of three sub-aggregates: the PlanetLab control framework (described in this section), the routing and topology sub-aggregate, and the DCN sub-aggregate (discussed in the following sections). A *slice* is a fraction of resources allocated to a particular experiment throughout an aggragate. GpENI has deployed the PlanetLab GENIwrapper slice facility architecture (SFA) [32] to export three interfaces: registry, slice manager (SM), and aggregate manager (AM). These GENI interfaces are accessible via the slice facility interface (SFI) implementing functions to get slice details, node details, and user accounts. Using the SFA, GpENI has *federated* with the public PlanetLab, which means that GpENI resources are available for authorised PlanetLab researchers using the public PlanetLab interface. In the long term, there are plans for federations among GENI, as well as to other Future Internet testbeds, such as OneLab [33].

3.4 Routing and Topology Sub-aggregate

For layer 3 programmability, GpENI provides programmable topologies using VINI and an arbitrary number of programmable routers in each node cluster.

Virtual Topology Control using VINI. VINI enables arbitrary virtual topology creation on top of physical networking infrastructure. VINI is

[7] GENI is divided into several clusters, each of which uses a distinct control framework.

essentially a flavor of PlanetLab with a set of enhancements to the PlanetLab kernel and tools called *Trellis*. Trellis allows users to create their own topology, either automatically using the IIAS (Internet in a slice) toolkit or manually designating links between *slivers*. MyVINI is the private VINI implementation within the GpENI nodes, permitting full control of virtual topology.

Programmable Routers. Trellis can host existing routing software such as Quagga [34], XORP [35], and Click [36]. Therefore, MyVINI with programmable routing software installed enables the implementations of layer 3 programmable routers for GpENI nodes. The routing softwares support a wide range of existing routing protocols. For example, Quagga supports RIPv1, RIPv2, OSPFv2, BGP-4, RIPng, OSPFv3. However, these programmable routers have very limited processing power and can only handle moderate size forwarding tables compared with realistic routers in backbone networks since they are running in commodity PCs. GpENI is initially running Quagga, with planned modifications to run XORP and Click.

The ultimate goal is to design a toolkit with a GUI permitting users to plug-in arbitrary research protocols. This will permit new routing protocols to be tested on the arbitrary layer-3 topologies provided by VINI, and on arbitrary layer-2 topologies provided by the DCN sub-aggregate described in the next section. Two areas of emphasis will be to study network resilience (in the context of the EU-FIRE ResumeNet project) and routing among heterogeneous realms (in the context of the NSF-FIND Postmodern Internet Architecture project).

3.5 DCN Sub-aggregate

GpENI uses DCN for control of VLAN interconnection among L2TPv3 tunneled node clusters as well as optical switches connected directly to the core backbone.

Internet2 ION Service and DCN Optical Control Plane Framework. In recent years, the Internet2 network has evolved from a pure IP-based packet-switching network into an advanced hybrid optical and packet network. Currently, the Internet2 network consists of three different, but related physical networks: Advanced IP network (provided by Juniper routers), Virtual Circuit network (provided by the multiservice switching capabilities of the Ciena CoreDirectors), and the core optical network (provided by the Infinera platform). Apart from the traditional IP service, the new Internet2 network offers a virtual circuit service to provision dedicated bandwidth across the network, called the Internet2 Interoperable On-demand Network (ION) [37]. The ION service can be used to set up on-demand, dedicated optical paths between endpoints in standard SONET bandwidth increments up to 10 Gbps. The ION service is dynamic and can thus be used to set up short term connections by a requestor or an application through a Web interface. The control plane software that automates the set up and tear down of the circuits was developed for the Internet2 Dynamic Circuit Network (DCN) research prototype and leverages technology developed by DRAGON (USC/ISI East, MAX, and George Mason University), GÉANT2,

and the DOE ESnet (OSCARS project). The DCN software suite is used to control the Internet2 ION service and is also available as open source to other institutions to use in creating their own DCN.

DCN in GpENI. GpENI has modified the DCN software suite to support the Netgear GSM7224 Ethernet Switch [38] used at the center of each GpENI node cluster. This will enable the creation of on-demand circuits at the required bandwidth for specified durations using the DCN software suite. There are two options to deploy DCN across the GpENI testbed [39]. The first option involves setting up static VLANs and Q-in-Q at the GPN switch in the Kansas City Internet2 PoP. The second option involves setting up a dedicated switch for DCN in the GpENI testbed. Deploying DCN across GpENI will also facilitate setting up VLAN circuits across the Ciena CoreDirectors located at various locations in Internet2. The CoreDirector Component Manager Interface [40] describes the use of the CoreDirector in the GpENI testbed. As additional GpENI optical switches are deployed, a common GpENI-wide DCN testbed will emerge over a multidomain network with CoreDirectors forming the optical domain and Netgear switches forming the Ethernet VLAN domain at each GpENI institution.

4 Early Demonstrations and Experiments

GpENI is currently operational at the core Midwest US nodes, principal European nodes, and several other institutions.[8] We have just begun experiments with GpENI, and are permitting limited accounts to researchers in the GENI, FIRE, and larger community.

The first public demonstration[9] creates a GpENI PlanetLab slice that generates an overlay loop among GpENI nodes; a screenshot example is shown in Figure 7. The next public demo will use the routing sub-aggregate to create a VINI topology slice giving control to the underlay in addition to the PlanetLab overlay.

Very preliminary performance experiments have begun using the current overlay capabilities, measuring throughput and delay for various topologies among the KU, KSU, UMKC, UNL, Bern, and Cambridge node clusters.

Table 1 shows the average delay between a set of 5 GpENI node clusters, measured on 23 Nov. 2009, beginning at 09:00. For each cluster pair, a `ping` command was issued 6 times at 2 hour intervals. Each command used 64KB packets, with a repeat count of 1000 and an interval of 1 sec. Thus the averages for each pair are computed for 6000 values over an 11 hour period. This table clearly indicates that the PlanetLab overlay topology is not representative of the layer-2 underlay, since traffic between Cambridge and Bern is backhauled to Kansas in the current tunnel confuguration. With the further deployment of L2TPv3 tunnels within Europe and DCN control of the L2 topology, we will be able to control both logical and physical topologies.

[8] As of this writing, KU, KSU, UMKC, UNL, Lancaster, Cambridge, ETH Zürich, and Bern nodes are operational, with many others under installation.

[9] Available at `http://control-1.ksu.gpeni.net/demo/#sites`.

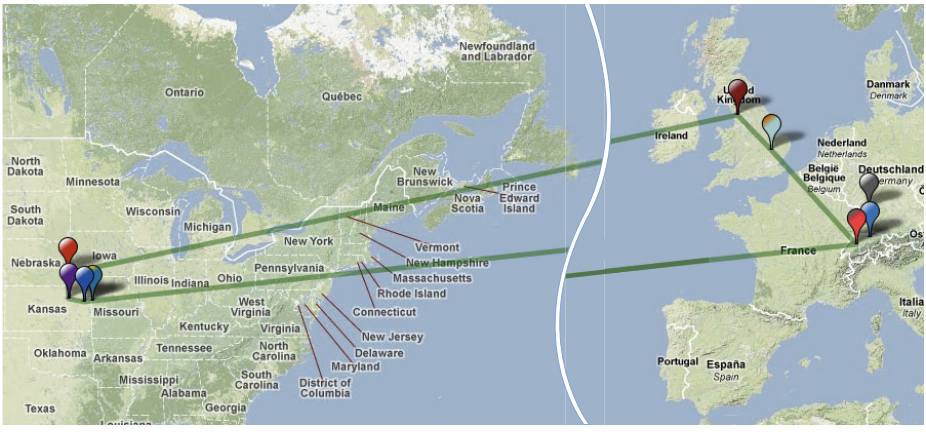


Fig. 7. GpENI-Planetlab Overlay Path Demonstration

Table 1. Average ping times in msec. between GpENI sites

Location	KSU	KU	UNL	Bern	Camb.
KSU	–	2	9	145	155
KU	2	–	6	143	153
UNL	9	6	–	152	158
Bern	145	143	152	–	295
Camb.	155	153	158	295	–

5 Future Outlook

GpENI is a testbed programable at every layer, intended to enable research in Future Internet architectures. The first GpENI nodes are operational, and we believe that it is the first such international testbed spanning the US GENI and EU FIRE programs, with plans to connect to JGN2plus in Japan, as well as Korean and Chinese testbeds. GpENI is currently under expansion to over 200 nodes at 40 institutions in 20 nations. We will use GpENI for experiments in the NSF-FIND Postmodern Internet Project at the University of Kansas, and the EU-FIRE ResumeNet project among KU, Lancaster University, ETH Zürich, TU München, Universität Passau, and Uppsala Universitet. GpENI has direct interconnection to ProtoGENI [41] linking GENI clusters B and C. Furthermore, efforts are underway to connect the GpENI to the MAX (Mid-Atlantic Crossroads) regional optical network at Layer 2 through the ProtoGENI network. The MANFRED project is in GENI cluster B with GpENI. In the longer term, we are under discussion with regional and national research networks to extend the optical substrate domestically and internationally.

Acknowledgements

This work is funded in part by the US National Science Foundation GENI program (GPO contract no. 9500009441), by the EU FP7 FIRE programme ResumeNet project (grant agreement no. 224619), and in large part by the participating institutions. GpENI is made possible by the involvement of many people in the participating institutions; in addition to the authors of this paper we particularly acknowledge: At KU: Co-I Joseph Evans; student Egemen Çetinkaya; net/sysadmins Michael Hulet, Wesley Mason. At UMKC: Co-I Baek-Young Choi; students Xuan Liu, Can Kanli, Sean Korzdorfer, Tim Sylvester; net/sysadmins James Schonemann II, George Koffler At UNL: student Mukesh Subedee; net/sysadmins Kent G. Christensen, Dale M. Finkelson (now at Internet2). At KSU: students Ali Sydney and Yunzhao Li (John Sherrell is now at Garmin); net/sysadmins Sam Hays and Richard Becker. At ETH Zürich: student Ajita Gupta. At the University of Cambridge: Co-I Andrew Moore. At Universität Bern: Co-I Torsten Braun. We further acknowledge many other Co-Is, students, and net/sysadmins in the institutions connected to GpENI, listed on the GpENI wiki [26].

References

1. Bhattacharjee, B., Calvert, K., Griffioen, J., Spring, N., Sterbenz, J.P.G.: Postmodern Internetwork Architecture. ITTC-FY2006-TR-45030-01 (February 2006), http://wiki.ittc.ku.edu/pomo
2. ResumeNet, http://www.resumenet.eu/project/index
3. CNRI. The Gigabit Testbed Initiative (1996), http://www.cnri.reston.va.us/gigafr/Gigabit_Final_Rpt.pdf
4. MAGIC Gigabit Testbed, http://www.magic.net/
5. Partridge, C., Davie, B., Campbell, R., Catlett, C., Clark, D., Feldmeier, D., McFarland, R., Messina, P., Richer, I., Smith, J., Sterbenz, J.P.G., Turner, J., Tennenhouse, D., Touch, J.: Report of the ARPA/NSF Workshop on Research in Gigabit Networking. National Science Foundation, Washington (July 1994)
6. Braden, B., Ricciulli, L.: A plan for a Scalable ABone – A Modest Proposal. TR, USC/ISI (January 1999)
7. Active Networks Program, http://www.darpa.mil/sto/strategic/an.html
8. Jackson, A.W., Sterbenz, J.P.G., Condell, M.N., Hain, R.R.: Active Network Monitoring and Control: The SENCOMM Architecture and Implementation. In: Proc. DARPA DANCE, pp. 379–393. IEEE, Los Alamitos (2002)
9. Shalaby, N., Peterson, L., Bavier, A., Gottlieb, Y., Karlin, S., Nakao, A., Qie, X., Spalink, T., Wawrzoniak, M.: Extensible Routers for Active Networks. In: Proc. DARPA DANCE, pp. 92–116. IEEE, Los Alamitos (2002)
10. Murphy, S., et al.: Security Architecture for Active Nets. In: DARPA Active Networks Security Working Group (July 1998)
11. Galis, A., Plattner, B., Smith, J.M., Denazis, S.G., Moeller, E., Guo, H., Klein, C., Serrat, J., Laarhuis, J., Karetsos, G.T., Todd, C.: A flexible IP active networks architecture. In: Yasuda, H. (ed.) IWAN 2000. LNCS, vol. 1942, pp. 1–15. Springer, Heidelberg (2000)

12. Chart, T., Schmid, S., Sifalakis, M., Scott, A.: Active Routers in Action: Evaluation of the LARA++, Active Router Architecture in a Real-Life Network. In: Wakamiya, N., Solarski, M., Sterbenz, J.P.G. (eds.) IWAN 2003. LNCS, vol. 2982, pp. 215–227. Springer, Heidelberg (2004)
13. PlanetLab, http://www.planet-lab.org/
14. VINI: A Virtual Network Infrastructure, http://www.vini-veritas.net/
15. Emulab: Network Emulation Testbed, http://www.emulab.net/
16. NSF NeTS FIND Initiative, http://www.nets-find.net
17. FP6 Situated Autonomic Communications, http://cordis.europa.eu/fp7/ict/fire/future-internet-projects_en.html
18. FIRE: Future Internet Research & Experimentation, http://cordis.europa.eu/fp7/ict/fire/
19. JGN2plus Testbed, http://www.jgn.nict.go.jp
20. GENI: Global Environments for Network Innovations, http://www.geni.net/
21. The OpenFlow Switch Consortium, http://www.openflowswitch.org/
22. Minden, G.J., et al.: KUAR: A flexible software-defined radio development platform. In: Proc. DySPAN, pp. 428–439 (April 2007)
23. Cacti, http://www.cacti.net/
24. Nagios, http://www.nagios.org/
25. Zenoss, http://www.zenoss.com/
26. GpENI wiki, http://wwww.gpeni.net
27. GpENI Demonstration, http://control-1.ksu.gpeni.net/demo/
28. Firestarter, http://www.fs-security.com/
29. Gush: Geni User Shell, http://gush.cs.williams.edu/trac/gush
30. Raven Provisioning Service, http://raven.cs.arizona.edu/
31. Supercharged PlanetLab Platform (SPP) Hardware Components, http://wiki.arl.wustl.edu/index.php/SPP_Hardware_Components
32. Peterson, L., Sevinc, S., Lepreau, J., Ricci, R., Wroclawski, J., Faber, T., Schwab, S.: Slice-based facility architecture. Technical report, Princeton (November 2007)
33. OneLab: Future internet test beds, http://www.onelab.eu/
34. Quagga routing suite, http://www.quagga.net/
35. XORP: Extensible Open-Source Routing Platform, http://www.xorp.org/
36. The Click Modular Router Project, http://read.cs.ucla.edu/click/
37. Internet2 DCN/ION Software Suite, https://wiki.internet2.edu/confluence/display/DCNSS/DRAGON+Supported+Switches
38. Lake, A., Tracy, C.: Dragon Supported Switches, https://wiki.internet2.edu/confluence/display/DCNSS/DRAGON+Supported+Switches
39. Angu, P., Ramamurthy, B.: DCN/ION in GpENI Investigation Report (October 2009), http://www.gpeni.net
40. Angu, P., Ramamurthy, B.: Ciena CoreDirector Component Manager Interface Document (October 2009), http://www.gpeni.net
41. ProtoGENI, http://www.protogeni.net
42. Mid-Atlantic Crossroads, http://wiki.maxgigapop.net

FIT: Future Internet Toolbox*

Thorsten Biermann, Christian Dannewitz, and Holger Karl

University of Paderborn, Research Group Computer Networks,
Pohlweg 47-49, 33098 Paderborn, Germany
{thorsten.biermann,christian.dannewitz,holger.karl}@upb.de

Abstract. Prototyping future Internet technologies is an important but complicated task, mainly caused by incompatibilities to existing systems and high implementation complexity. To reduce these problems, we have developed the *Future Internet Toolbox (FIT)*, consisting of four frameworks that cover data transport, information-centric networking, naming, and name resolution. These frameworks can be used separately or can be combined to a large testbed, covering many aspects at once. Experience has shown that FIT not only simplifies our own prototyping activities but is also useful for other projects due to its generality.

Keywords: Future Internet; network; testbed; framework; prototype; information-centric; data-centric; transport; naming; name resolution.

1 Introduction

Currently, a lot of research is done in the area of future Internet technologies. Hot topics include information-centric networking, data transport techniques and protocols, routing schemes, resource allocation, mobility, and cooperation/coding techniques. After developing new concepts and protocols in these areas, they have to be evaluated. Often, it is desirable to do this via prototyping and testing under real-world assumptions in real networks. This procedure, however, is very difficult today and is often avoided because (1) the new concepts are incompatible with existing systems or (2) it is too costly to implement a whole prototype from scratch to evaluate a small component of an overall architecture.

To overcome these difficulties, we developed *FIT* – the *Future Internet Toolbox*. FIT simplifies developing future Internet prototypes and testbeds by providing a set of frameworks, covering the following aspects of networking:

- Data transport, including related aspects like routing, resource management, mobility, and cooperation techniques
- Naming and name resolution
- Information-centric networking, including search, publish/subscribe, caching/storage, and information modeling

* The research leading to these results has been conducted in the 4WARD project and received funding from the European Community's Seventh Framework Programme (FP7/2007-2013) under grant agreement no. 216041.

T. Magedanz et al. (Eds.): TridentCom 2010, LNICST 46, pp. 442–453, 2011.

These frameworks solve both problems mentioned above as they (1) provide generic testbeds that integrate into today's network/system architectures and (2) provide a lot of ready-to-use building blocks that support and ease development. The downside of such high reusability is that frameworks always introduce a trade-off between reusability and flexibility. Having this in mind, the main goal was to mitigate this trade-off by designing the frameworks as generic as possible to allow implementing as many different concepts and protocols as possible.

The following sections introduce the frameworks that constitute FIT. Sec. 2 covers information-centric networking, Sec. 3 our naming framework. Data transport is addressed in Sec. 4 and Sec. 5 discusses generic name resolution.

An extended version of this paper, containing additional use cases and examples, is available in [1].

2 Information-Centric Networking Framework

This section describes the Information-Centric Network (ICN) framework. We will describe the framework itself and will illustrate how the framework can be used in specific information-centric architectures using the example of NetInf (and Content-Centric Networking (CCN) [2], see technical report [1]).

2.1 Overview

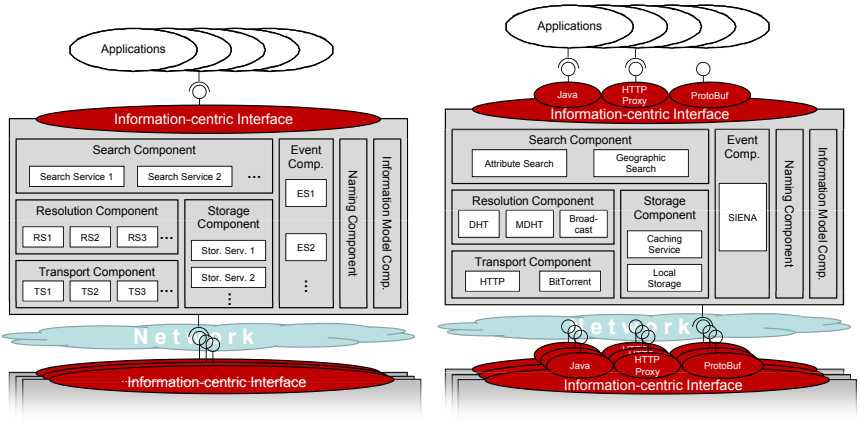
Several information/content/data-centric network architectures have been proposed lately [3,2,4]. To support the prototyping of ICN concepts, a framework providing the following aspects is desirable:

- A generic, adaptable *node structure* implementation
- A flexible, reusable implementation of various *architecture components*
- Support for defining *new services and protocols*, incl. especially communication

We are not aware of a framework that focuses on prototyping ICN architectures (see our technical report [1] for a discussion of related work). Therefore, we developed the ICN framework for rapidly and easily implementing and testing ICN designs as well as specific services and protocols.

The ICN framework solves the apparent conflict between flexibility and reusability by recursively addressing this conflict on four different levels: the *network architecture level* including communication between network nodes, the architecture of each *network node* that is composed of components, the architecture of each *component* containing multiple component services that implement architecture-specific services and protocols, and the design of those *component services*. On each level, we provide reusable structure and building blocks to accelerate the prototyping process while at the same time providing flexibility and extensibility based on a consistent plugin concept.

Fig. 1a gives an example of a node structure for a generic ICN node, consisting of components. Each component can contain a single (e.g., *Naming*, *Information Model* component) or multiple different implementations of its component



(a) Generic node structure **(b)** NetInf-specific node configuration

Fig. 1. Example testbed node configurations, consisting of adaptable components that each contain several service incarnations like *Resolution Services* (*RS*), *Transport Services* (*TS*), and *Event Services* (*ES*)

functionality (e.g., *Resolution* component). The API of the information-centric node as well as inter-node communication is combined in the same adaptable interface. To provide a broad and flexible mechanism to access the interface, we have implemented a layer of indirection on top of the interface (Fig. 1b) that provides access in different ways (e.g., via an HTTP proxy interface to connect legacy applications) and can easily be extended with new mechanisms.

Based on our NetInf prototype [5] and evaluation of related concepts like CCN [2], Content-Based Networking [6], and DONA [4], we identified the following main components of an ICN architecture: *Search*, *Naming*, *Name Resolution*, *Data Transport*, *Storage*, *Event Service*, and *Information Model*. The ICN framework provides ready-to-use implementations for those components.

Each component can be adapted with architecture-specific services and protocols based on a flexible plugin concept. Services and protocols are encapsulated in *component services*. To enhance flexibility, a component can contain multiple component services while each service fulfills the same interface but may implement and fulfill this service in a different way. A *component controller* is responsible for choosing between component services, and managing the order of execution. For example, the *Resolution Controller* manages several *Resolution Services* that implement specific name resolution mechanisms, e.g., via DHT or via DNS, that can be chosen depending on the type of namespace (flat/hierarchical) to resolve. Adding a new service to a component is done via the simple plugin concept and reconfiguring the controller to include the new service.

Besides the architecture of a network node, the communication between nodes is the second main aspect of an ICN architecture. The ICN framework offers two distinct components for implementing and testing various communication protocols. First, the *Transport* component is used to implement and test communication

protocols in general. Implementations for legacy protocols are already provided by the ICN framework and new protocols can be implemented using the data transport framework (Sec. 4). Second, as the publish/subscribe paradigm often plays an important role in ICN [6,3], we provide a dedicated *Event* component to simplify the integration of different publish/subscribe mechanisms.

In summary, the ICN framework provides a testbed for building custom nodes and for interconnecting those nodes to implement and test different ICN architectures. It has a flexible and adaptable structure, and offers ready-to-use implementations for the main components of many ICN architectures as well as component services to accelerate the prototyping process.

2.2 Implementation

The ICN framework is consistently based on the interface design pattern for flexibility. It is implemented in Java for platform independence and a flexible choice of devices. The code runs on FreeBSD, Linux, Windows, and Android.

2.3 NetInf Use Case

Fig. 1b shows the configuration of a NetInf node created with the ICN framework. Each component contains several specific service implementations that represent the main services and protocols of the NetInf architecture. For example, the NetInf architecture contains multiple name resolution services in parallel that are each represented as a *Resolution Service* (*RS*) in the *Resolution* component.

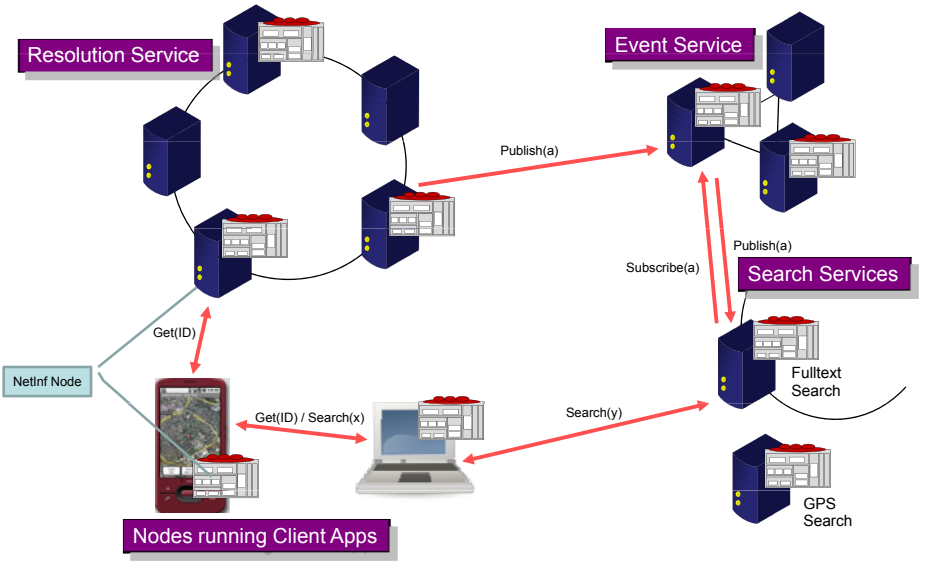


Fig. 2. Network of information-centric nodes

The NetInf prototype also illustrates the generality of the ICN framework by integrating the NetInf architecture with another content-based network architecture, SIENA [6]. SIENA is used to realize the publish/subscribe communication, integrated with the NetInf information model.

Our NetInf prototype extends the information-centric node interface with three different mechanisms to access the node interface (Fig. 1b): Java applications can simply use the provided Java interface. For applications that talk HTTP (e.g., a Web browser plugin), we provide an HTTP proxy interface. For inter-node communication, we provide access to the node interface via Google Protocol Buffers (Protobuf) [7]. Google Protobuf enables the simple and fast definition of custom protocol messages and provides efficient data transfer.

Fig. 2 shows an example of several NetInf nodes that form an ICN. Some are running client applications while others provide services like global name resolution and search services. Communication between those nodes is based on the information-centric node interface (implemented with NetInf-specific primitives like *Get(ID)*, *Publish(a)*) and Google Protobuf.

3 Naming Framework

A naming framework should provide mechanisms that support a wide variety of different naming schemes while at the same time minimizing the implementation overhead. As a result of those considerations, our naming framework [8] is based on a simple, yet flexible and powerful mechanism: names are composed of a sequence of labels, each of which is a 'labelName=labelValue' pair. There are two ways to handle the ordering of labels: *labelNames* are either themselves part of the name, e.g., 'label1=value1 & label2=value2 & label3=value3', thereby explicitly assigning *labelValues* to *labelNames*. Alternatively, *labelNames* are not part of the name, e.g., 'value1 & value2 & value3', which then requires to define the ordering of labels in advance.

Including the *labelNames* in each name allows for flexibility. This makes it possible to define naming schemes with a flexible set of labels as well as a flexible ordering of labels. On the other hand, namespaces with compact names can be achieved by excluding *labelNames* from the names and predefining an explicit label ordering. Via those two mechanisms, our naming framework supports common naming schemes like IP addresses and URIs as well as complex naming schemes like the NetInf naming scheme [3].

Some more complex naming schemes, especially in the area of information-centric networking [3,2], involve features like information-centric security that go far beyond common naming schemes. To support such features, we optionally combine names with a flexible set of metadata that can, e.g., contain security-related data like a hash value of the content. In addition, our framework integrates implementations for security-related algorithms like symmetric and asymmetric encryption, (self-)certification, and public/private-key-based authentication to simplify the implementation of complex naming schemes which include such security features.

4 Data Transport Framework

4.1 Overview

The main observation that triggered our work to develop a data transport framework was the difficulty to introduce new functionality/protocols into today's network stacks. One reason for this is that there is no coherent way to identify entities and to manipulate them as today's network architecture is based on a mainly statically layered stack and functionality is located in end systems.

To design a more flexible, powerful, and reusable data transport architecture, we need an approach to model, design, identify, and use data flows. This approach, however, needs to be generic enough to stretch over a wide range of technology levels and should encompass a wide range of data processing and forwarding functions in end systems as well as in intermediate nodes.

With such a set of requirements, it is impossible to come up with *the, single* solution for a *one-size-fits-all* flow type. Therefore, we decided to choose a design process that combines (1) a uniform appearance and interface for all different flow types and (2) flexibility in supporting a wide range of flow types, in as many different environments as possible. A network architecture that fulfills these requirements is the *Generic Path (GP) architecture* [9]. We use its concepts as basic building blocks for our data transport framework.

We chose an object-oriented approach to design network components while keeping them coherent in their interfaces and basic structures. This allows to incorporate new networking techniques more flexibly than in today's network architectures as networks can be arbitrarily composed of the components. Furthermore, networks can easily be adapted according to any cross-layer information during runtime, thanks to the unified interfaces. Examples for data transport aspects that have been modified/integrated into our framework are routing, mobility [10], cooperation and coding techniques [11], and resource allocation.

A discussion on related work in this area can be found in [1, Sec. 4.5].

4.2 Framework Components

This section introduces the components that constitute the data transport framework: Generic Paths (GPs), Entities, Compartments (CTs), Endpoints (EPs), and Hooks. An example of their interactions is given in Fig. 3. The scenario consists of six Entities (E1-E4, two Cores), four CTs (C1, C2, N1, N2), four EPs (EP1-EP4) forming two GPs, and two Hooks. The GP between EP1 and EP2 in C1 (e.g., IP) is realized by the GP between EP3 and EP4 in C2 (e.g., Ethernet).

Entity. An Entity is the generalization of an application that takes part in any kind of communication. Depending on the implementation, this can be a process, a set of processes, a thread. Communication between Entities is realized via GPs.

Furthermore, an Entity keeps state information that is shared among multiple GPs and runs processes or threads that manage this state. Examples for such state information are routing tables, resolution tables, and access control tables.

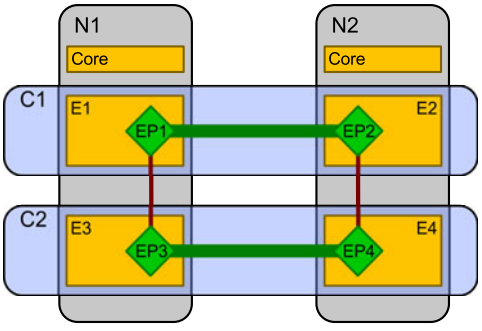


Fig. 3. Overview and interaction of GP architecture components. Entities are drawn as rectangles, CTs as rectangles with rounded corners, EPs as squares, GPs as horizontal lines, and Hooks as vertical lines.

Generic Path. A Generic Path (GP) is an abstraction of data transfer between communicating Entities located in the same or in remote nodes. The actual data transfer, including forwarding and manipulation of data, is executed by EPs.

Endpoint. An Endpoint (EP) keeps the local state information of a specific GP instance, i.e, it is a thread or process executing a data transfer protocol machine and doing any kind of traffic transformation. EPs are created by an Entity and may access shared information of that Entity.

Usually, GPs require other GPs to provide their service. E.g., a TCP/IP GP requires another GP that provides *unreliable* unicast, like an Ethernet GP, to provide a *reliable* unicast service at the end. Therefore, EPs are bound via Hooks to other EPs within the same node. Besides exchanging data, Hooks also hide names from other CTs to permit changing a GP's realization later on.

Compartment. A Compartment (CT) is a set of Entities that fulfill the following requirements:

- Each entity carries a name from a CT-specific name space (e.g., MAC addresses in the Ethernet CT). These names can be "empty" and do not need to be unique. Rules how names are assigned to entities are specific to each CT.
- All entities in a CT *can* communicate, i.e., they support a minimum set of communication primitives/protocols for information exchange. These protocols are implemented as different GP types. Hence, for joining a CT, an Entity must be able to instantiate the EP types required by the CT.
- All entities in a CT *may* communicate, i.e., there are no physical boundaries or control rules that prohibit their communication.

A special CT is the Node CT (N1 and N2 in Fig. 3). It corresponds to a processing system, i.e., typically an operating system that permits communication

between different processes (e.g., by using Unix domain sockets). By means of virtualization, multiple Node CTs can be created on one physical node.

An Entity is typically member of at least two CTs, the "vertical" Node CT and a "horizontal" CT. Furthermore, the Entity has a (possibly empty) set of names from each of the respective CT name spaces.

Note that GPs cannot cross CT boundaries due to the possibly different name spaces, protocols, etc. GPs always reside within a single CT.

Core. The Core is a special Entity. It exists once per Node CT, is only member of the Node CT, and is responsible for node-wide management. The Core's mainly supports other Entities in cross-CT (i.e., cross-layer) issues, like, name resolution, mobility, managing/controlling Hooks, and service discovery. E.g., in the example in Fig. 3, when setting up the GP between EP1 and EP2, it is the Core that tells E1 that E3 is able to provide the service required to realize its GP.

Note that the Core is also able to reconfigure the realization of existing, i.e., already established, GPs during runtime. This is done transparently to the involved Entities by moving a Hook from one EP to another, possibly in another CT. This feature is required, e.g., to realize mobility or to switch between different transmission modes, like cooperative and direct transmission.

4.3 Testbed Implementation

To be able to use Entity, EP, and Core implementations in various environments, like Linux, Windows, and embedded systems, we used C++ for efficiency and strictly separated the implementation of the logic parts from the environment-specific parts. In detail, this separation means that our testbed abstracts execution environment functions, e.g., by mapping Hooks to available IPC mechanisms. Entities, EPs, and Cores just use these abstractions, which enables to use their content (transport protocols, routing strategies, mobility schemes, data encoding, etc.) in different environments without changing code.

We realized this separation by inheritance, provided by the object-oriented programming paradigm. For this, we implemented all abstractions, like Hook, timeout, and callback handling, in a root class, called `AbstractEntity` (and analogously for EPs and Cores). From this root class, wrapper classes like `PosixEntity` and `OmnetEntity` inherit to map the abstractions to the appropriate execution environment APIs. Thereafter, an `Entity` class is derived from *one* of these wrappers (chosen during compile time). Own, environment-independent Entity classes inherit from this class. Environment-specific entities directly inherit from a wrapper class. Fig. 4 illustrates this.

Currently, we have ready-to-use wrappers for POSIX-compliant systems, like Linux, BSD, and Darwin (Mac OS X) and for OMNeT++ [12], an open-source discrete event simulator. The wrapper for OMNeT++ is especially useful during the development phase and for demonstrating scalability while at the same time using the implementation in real-world scenarios via the POSIX wrapper. Additionally, we are currently working on wrappers for Windows and OpenFlow [13].

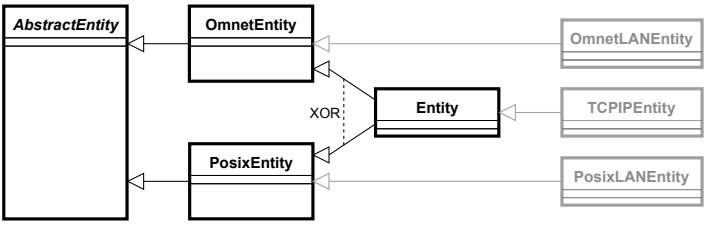


Fig. 4. Inheritance graph for Entity implementations. `Entity` is usually the base class from which user-specific Entities (gray) are derived.

5 Name Resolution Framework

Resolving "names" into "addresses" is used in everyday networking at various layers and abstraction levels. It is realized by a patchwork of individual techniques and concepts. In current networking practice, there is no clear consensus on what a "name" or an "address" really is and how they should be mapped to each other between different layers of a system. Closely linked to this confusion is a lack of a clear concept how the entities inside the individual layers in a system refer to each other and what is necessary to identify the mapping between such two entities; only patchwork solutions for typical combinations of layers (e.g., ARP, DNS) exist. These issues make it difficult to introduce a new protocol or a new layer, as this likely to break existing name resolution schemes.

We propose a flexible yet unified name resolution framework that has two advantages: (1) With a very small set of primitives, a vast range of resolution mechanisms, like ARP or DNS, can be captured. (2) Introducing new layers is much easier. We will discuss information-centric networking as an example.

In the following, we use the framework component definitions introduced in Sec. 4.2 to describe our name resolution framework.

5.1 Resolution Process

When resolving a name, an Entity knows the name of its desired peer Entity and the CT to which itself and this name belongs. The objective of name resolution is to find the following additional information for such a name:

- The name of a CT via which the peer Entity can be reached (e.g., "WLAN").
- An Entity inside this "lower CT", which can handle the communication on behalf of the originator Entity (typically, by means of sharing a Node CT).
- The name of a remote Entity in the lower CT that can pass on data to the actual peer Entity (typically, by means of sharing a Node CT).

The core point in designing a unified name resolution system is to avoid spreading knowledge of how to interpret a name outside of its CT. Neither does the upper CT understand names of the lower CT nor vice versa. Hence, the only thing an Entity can do to resolve a name (in absence of further knowledge) is to

contact all other Entities in its own CT and ask which Entity *has* this name (for optimizations, see [1]) – a WhoHas message is sent *inside* its own CT.

Horizontal CTs usually do not have direct communication means, i.e., spreading a WhoHas message requires assistance of suitable lower CTs (which CTs are suitable depends on the required communication service). A lower CT Entity, however, needs to be told where to send this message; it needs a remote address, which has to be provided by the higher CT Entity that initiates the resolution. Note that this lower CT remote address is, from the perspective of the higher CT Entity, a configuration parameter (opaque string) which it needs to provide but not to understand. Hence, the resolving Entity sends its WhoHas message to all local Entities in all suitable CTs, with the remote address (in the *lower* CT) being looked up, e.g., from a configuration file. This address could be a unicast, broadcast, or even anycast address inside the lower CT.

In the lower CT, the WhoHas message is distributed to its remote address, possibly to many Entities in this CT. The receiving lower-level Entity will receive the WhoHas message and will distribute this message to *all* entities in the original CT. It does *not* have to process names of the original CT.

Inside the original CT, Entities understand names contained in the WhoHas message. Each Entity checks whether it matches the desired name (it does not have to *be* the named Entity, cp. ARP). If no, it can silently discard the message. If yes, it answers with an IHave message, containing (1) the original CT name, (2) the name to be resolved in the original CT, (3) the lower CT name, and (4) the lower CT address over which the name in the original CT can be reached.

5.2 Testbed Implementation

We implemented the name resolution framework directly in conjunction with the data transport framework (Sec. 4.3). In consequence, the transport testbed is able to deal with any naming scheme and any name resolution implementation.

Our name resolution framework implementation spans the Core and Entities. When an Entity wants to resolve a name, it creates a WhoHas message, containing the following information: (1) its own name, (2) the name to be resolved, and (3) the CT name to which these two names belong. Thereafter, this message is passed to the Core, along with a descriptor that specifies which service will be

Table 1. Example `ResolvConf`. SrcCT is the CT from which a name is resolved, DstCT the CT to which the resulting address belongs, ResolverCT the CT in which resolution takes place, and Resolver the name of the resolver Entity.

SrcCT	DstCT	ResolverCT	Resolver
`TCP_IP`	`LAN_A`	`LAN_A`	`00:00:00:00:00:00`
`TCP_IP`	`LAN_B`	`LAN_B`	`00:00:00:00:00:00`
`NetInf`	`NetInfTransport`	`NetInfRes`	`12345`

Table 2. Example `ResolutionTable`. SrcEntity is the Entity that starts the resolution, DstEntity the name to be resolved (both within CT). LowerSrcEntity/LowerDstEntity are the Entity names in LowerCT that were found during resolution.

CT	SrcEntity	DstEntity	LowerCT	LowerSrcEntity	LowerDstEntity
TCP_IP	1.2.3.4	5.6.7.8	LAN_B	45:67:89	89:AB:CD
TCP_IP	1.2.3.4	2.3.4.5	LAN_A	45:67:AB	23:45:67

required for data transfer later on (after a successful name resolution). The Core uses this descriptor to determine for which of the available (horizontal) CTs a resolution is performed. For each of these suitable CTs, the Core checks if an entry exists in the `ResolvConf`. An example is illustrated in Tab. 1; it contains two entries for configuring name resolution within the CTs `TCP_IP` and `NetInf`. Resolving a `TCP_IP` name in the `LAN_B` CT also takes place in the `LAN_B` CT; resolving from `NetInf` to `NetInfTransport` is done in the `NetInfRes` CT.

The Core extends the WhoHas message with the information from the ResolvConf and sends it to all Entities in the own Node CT that are member of the CT defined by ResolverCT. These Entities send the WhoHas message to the resolver Entity (ResolverName) where it is processed and answered.

When the resulting IHave message arrives back at the originating Entity that sent the WhoHas in the CT defined by ResolverCT, it is passed to the Core. The Core adds the contained information to its `ResolutionTable`. An example is shown in Tab. 2. It holds two results; one for 5.6.7.8 in the `TCP_IP` CT that succeeded in the `LAN_B` CT and one resolution that was answered in `LAN_A`.

The Core now informs the Entity that requested the resolution (1.2.3.4 in the example) about the success and returns a reference to the `ResolutionTable` entry. The Entity cannot read the `ResolutionTable` content. It uses the pointer to reference the name resolution result when sending data to the resolved destination name. This strictly keeps names in their own CTs and makes common misuse impossible (e.g., to put "lower addresses" like IP into the payload).

To our knowledge, this is the first approach towards a unified name resolution framework that captures a wide range of existing and future resolution services and spans over all technological levels, i.e., network layers.

6 Conclusion

We presented four frameworks for prototyping future Internet techniques related to data transport, information-centric networking, naming, and name resolution. These frameworks can be used on their own to prototype individual concepts or can be combined to complex testbed implementations, like we did it for the integrated NetInf prototype.

Experience during our work confirmed that the frameworks simplify prototyping a lot. Due to their generality, they will also be useful for other projects

as they reduce redundant implementation of basic testbed functions and provide ready-to-use building blocks to rapidly complement new, small components with an overall network architecture. We will publish our code as open-source project.

Acknowledgments

We gratefully thank the project groups AugNet I and II for valuable discussions and their excellent implementation support. Furthermore, we would like to thank our colleagues in the 4WARD project for their input and fruitful discussions.

References

1. Biermann, T., Dannewitz, C., Karl, H.: FIT: Future Internet Toolbox — extended report. Technical Report TR-RI-10-311, University of Paderborn (February 2010)
2. Jacobson, V., Smetters, D.K., Thornton, J.D., Plass, M., Briggs, N., Braynard, R.L.: Networking named content. In: Proc. ACM CoNEXT (December 2009)
3. Ahlgren, B., D'Ambrosio, M., Dannewitz, C., Marchisio, M., Marsh, I., Ohlman, B., Pentikousis, K., Rembarz, R., Strandberg, O., Vercellone, V.: Design considerations for a network of information. In: Proc. ReArch 2008 (December 2008)
4. Koponen, T., Chawla, M., Chun, B.G., Ermolinskiy, A., Kim, K.H., Shenker, S., Stoica, I.: A data-oriented (and beyond) network architecture. In: Proc. ACM SIGCOMM 2007, pp. 181–192. ACM Press, New York (2007)
5. Dannewitz, C., Biermann, T.: Prototyping a network of information. Prototype demonstration at the IEEE LCN (October 2009)
6. Carzaniga, A., Wolf, A.L.: Forwarding in a content-based network. In: Proc. ACM SIGCOMM 2003, pp. 163–174. ACM Press, New York (2003)
7. Google: Google protocol buffers – Protobuf, Open source project (July 2008)
8. Dannewitz, C., Golic, J., Ohlman, B., Ahlgren, B.: Secure naming for a network of information. In: Proc. 13th IEEE Global Internet Symposium 2010 (March 2010)
9. Biermann, T., et al.: D-5.2.0: Description of Generic Path mechanism, Project deliverable (January 2009)
10. Bertin, P., Aguiar, R.L., Folke, M., Schefczik, P., Zhang, X.: Paths to mobility support in the future Internet. In: Proc. IST Mobile Comm. Summit. (June 2009)
11. Biermann, T., Polgar, Z.A., Karl, H.: Cooperation and coding framework. In: Proc. IEEE Future-Net (June 2009)
12. Varga, A., et al.: OMNeT++ discrete event simulation system Open source project
13. McKeown, N., Anderson, T., Balakrishnan, H., Parulkar, G., Peterson, L., Rexford, J., Shenker, S., Turner, J.: Openflow: Enabling innovation in campus networks. SIGCOMM Comput. Commun. Rev. 38(2), 69–74 (2008)

TridentCom 2010

Practices Papers Session 6: Experimentally Driven Research and User Experience Testbeds

Open Urban Computing Testbed

Timo Ojala, Hannu Kukka, Tommi Heikkinen, Tomas Lindén, Marko Jurmu, Fabio Kruger, Szymon Sasin, Simo Hosio, and Pauli Närhi

MediaTeam Oulu, University of Oulu
P.O. Box 4500, 90014 University of Oulu, Finland
firstname.lastname@ee.oulu.fi

Abstract. We present a unique urban computing testbed for studying the utilization of ubiquitous computing technology in the public urban space of a city center. The testbed comprises of a wide range of pervasive computing infrastructure and different middleware resources. We demonstrate the applicability and benefits of the testbed in evaluating technology pilots and prototyping new ubiquitous services in real-world urban setting. We conclude with a discussion on the challenges in deploying this kind of a large-scale testbed in a public urban space.

Keywords: urban computing, ubiquitous computing, public private partnership.

1 Introduction

We introduce a unique urban computing testbed in the City of Oulu in northern Finland, just 200 km south of the Arctic Circle. With 140000 citizens Oulu is the sixth largest city in Finland. The Oulu region has strong ICT competence in 14000 ICT jobs and the largest regional R&D expenditure per capita in Finland. In the late 1990's the Wired Magazine ranked Oulu as the number three 'silicon valley' in the world. The long-term goal of our testbed is to provide open horizontal resources for building incremental functional prototypes of a future ubiquitous city. There networked computing devices are seamlessly embedded into the urban space, turning it into a smart space providing different interaction modalities with the physical, virtual and social spaces. The utilization of ubiquitous computing technologies in urban space is studied by the multidisciplinary field of urban computing. It is driven by two important and related trends, urbanization and increasing deployment of pervasive computing infrastructure in the urban areas.

The mainstream research on ubiquitous computing suffers from a distinct lack of longitudinal, real-world case studies of system usage. The vast majority of research consists of studies that typically last a few days or weeks at best. Further, from the viewpoint of urban computing the studies are often executed in artificial settings such as labs and university campuses. While the immense research effort has produced numerous publications laying the theoretical foundation, few visible and lasting contributions to the urban digital fabric have emerged. This lack of coherent progress motivated the 2005 UbiApps workshop at Pervasive 2005, where 25

T. Magedanz et al. (Eds.): TridentCom 2010, LNICST 46, pp. 457–468, 2011.

researchers were invited based on their position papers. In their workshop summary Sharp and Rehman [10] identified several reasons underlying the crisis in the international ubiquitous computing research. One of them was the well-known fact that the research community values novelty over high-quality implementations and good engineering practices. This leads to 'reinventing the wheel' in tiny increments, which may be worth yet another publication, but very little else to the community, as the increments are not shareable due to their poor engineering. The consensus was that the scientific community should reward good engineering and encourage research that constructs open, reusable infrastructure for the wider community's benefit.

We argue that the lack of visible and lasting results (in terms of applications) in ubiquitous computing is partially due to the lack of open pervasive computing infrastructure in the public spaces. Successful public spaces are mixtures of activities and applications, which purposefully combine physical, virtual and social spaces. They link places and context, consciously avoiding the 'anything, anytime, anywhere' paradigm. Doing this in practice requires permanent local infrastructure, which for business reasons is often deployed as closed verticals. Further, dedicated pervasive computing infrastructure would facilitate long-term large-scale real-world studies. Such studies are important because real-world ubiquitous computing systems are culturally situated, which cannot be reliably assessed with lab studies detached from the real-world context. Infrastructure and time are needed to establish the required technical and cultural readiness and the critical mass of users, before a pervasive computing system can be evaluated '(un)successful' [12]. Interestingly, while some research communities have made long-term large-scale investments in shared infrastructure to support joint and transparent research, such as radio telescopes and networking testbeds, no such attempt has been made by the ubiquitous computing research community.

We have set out to deploy a novel city-wide ubiquitous computing testbed, which is provided as an open horizontal resource to the whole community. The infrastructure layer of the testbed comprises of three types of wireless networks (IEEE 802.11 WLAN, Bluetooth, IEEE 802.15.4 WSN), large public displays and various server machines connected via an aerially large layer 2 network. These heterogeneous computing resources constitute a large distributed system which is organized with a middleware layer. It provides various resources for supporting technology experiments, developing ubiquitous computing applications, and managing and monitoring the applications and the testbed. We demonstrate the applicability and benefits of our testbed with several case studies on deploying and evaluating technology pilots and prototype services in authentic urban setting.

To our best knowledge, a similar urban computing testbed with a matching range of computing resources deployed in a city center has not been reported in the literature before. In a historical context the PARCTAB ubiquitous computing experiment at the Xerox Palo Alto Research Center is probably the most similar deployment [15]. Related recent efforts such as the iDisplays at the University of Münster [13] and the e-Campus at the Lancaster University [14] are deployed at university campuses. This is understandable as it is much easier and often more practical to deploy research infrastructure on your own campus than at a city center. However, a campus environment cannot provide the authentic experimental setting pursued in urban computing.

2 Testbed Resources

Fig. 1 shows the simplified architecture and the building blocks of our urban computing testbed. The key infrastructure installed at downtown Oulu comprises of three different types of wireless networks and large public displays. The server machines are placed in several server farms located both at downtown and at the University of Oulu campus. In terms of network topology these nodes are aggregated into an aerially large VLAN, whose traffic goes via the main switch (corresponds to switch #1 in Fig. 2). The Recorder captures complete packet history with a set of high-speed probes attached to the main switch [2].

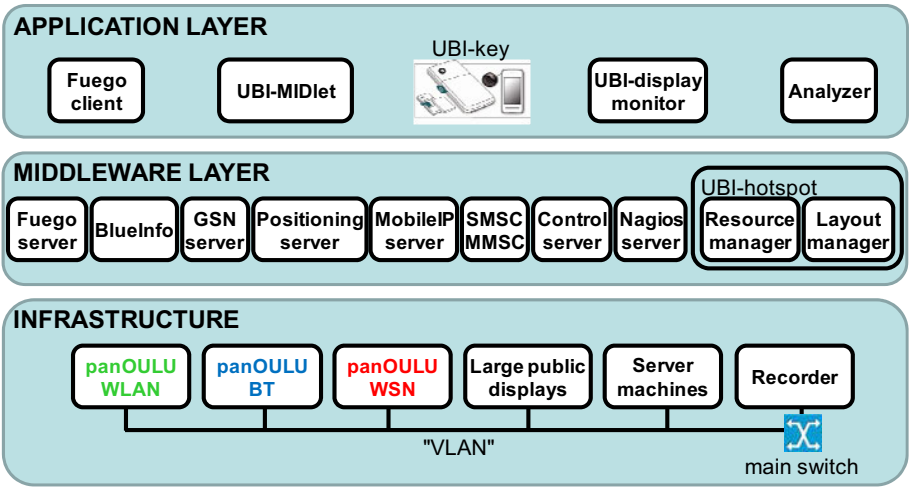


Fig. 1. The building blocks of our urban computing testbed

2.1 Wireless Networks

panOULU WLAN is a large multi-provider wireless network (IEEE 802.11) provided by a public-private partnership [7]. The network comprises of two parts, 'CITY' and 'REGION', as illustrated in Fig. 2. The 'CITY' comprises of two types of WLAN zones, the campus networks of five public organizations (City of Oulu, University of Oulu, Oulu University of Applied Sciences, VTT Technical Research Centre of Finland, and Pulmonary Association Heli) and the panOULU subscriptions sold by four ISPs (DNA, Elisa, LAN&WAN, Netplaza). panOULU subscription is an ISP product, which allows any organization to acquire panOULU WLAN hotspot into its premises to enhance image and customer service. The topology also includes a simple approach for integrating mobile APs (e.g. in buses) into the same IP subnet.

The WLAN zones are aggregated at the central layer 2 switch #1 into a single IP subnet, which effectively simplifies the multi-provider network into a single-provider network. The design comes with a rudimentary built-in session mobility support for the WLAN clients using the network, without any additional client software. The need for session mobility is motivated by the fact that the WLAN zones of

different providers overlap in terms of radio coverage, particularly in the city center. If the zones would reside in different IP subnets, a station's roaming from one zone to another would result in the change of IP address which would interrupt ongoing socket connections and application sessions. The session mobility is based on the self- learning property of the layer 2 switches used to connect the APs (access points) into the backbone (not necessarily switches #1 and #2 shown in Fig. 2). When a mobile node roams between APs in two different BSSs (Basic Service Sets), the layer 2 switch connecting the two BSSs will eventually receive a frame from the mobile node, thus automatically learning the new location of the mobile node and updating its forwarding table accordingly.

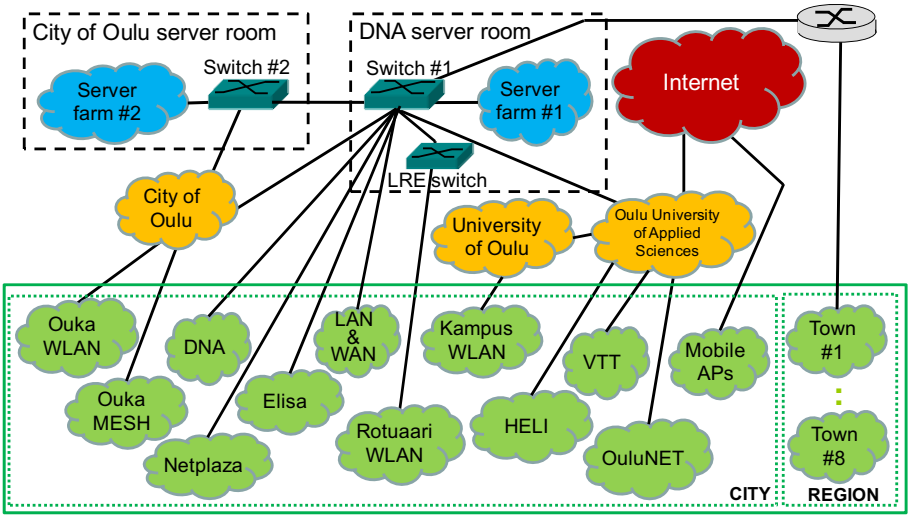


Fig. 2. Simplified topology of the panOULU WLAN network

The 'REGION' subnet comprises of the WLAN zones covering key public locations in eight nearby townships, which are connected to the core via a layer 3 router. The WLAN zones total currently ~1200 access points, of which ~500 reside within a 1 km radius of the city center of Oulu. From the user's point of view the APs appear as one large uniform network with SSID 'panoulu'. The APs provide both indoor and outdoor coverage in places deemed relevant for public access. The city center and its immediate surroundings are blanketed with a WLAN mesh network, but otherwise the coverage is provided in a hotspot manner.

In its coverage area the panOULU network provides open (no authentication or registration) and free (no payment) wireless Internet access to the general public equipped with a WLAN device. Excluding the blocking of outgoing port 25 (SMTP), which is required by the Finnish legislation, there are no limitations or restrictions on the use of the network. Currently, about 20000 WLAN devices use the network every month so that 25-40% of them are visitors and about 30% are WLAN phones.

The large coverage combined with the open and free access make the panOULU WLAN network a valuable R&D resource, as well. The network has been employed

by numerous R&D pilots and research projects. Further, the panOULU WLAN network provides backhaul connectivity to some panOULU BT and panOULU WSN AP's, as well. The panOULU WLAN is also used by the municipality for streaming feeds from video surveillance cameras and controlling digital parking guidance signs.

In a historical note, the panOULU WLAN fulfills Weiser's vision on ubiquitous infrastructure from the late 1980's: public access points provide short-range wireless connectivity on license-free spectrum, which allows the user's wireless devices to communicate with the surrounding smart space. A promo video of the panOULU WLAN is available at http://www.strixsystems.com/video/Strix_PanOULU.html.

panOULU BT network is a cluster of Bluetooth APs around downtown Oulu. A BT AP provides open and free connectivity to Bluetooth devices. It effectively establishes a WPAN (Wireless Personal Area Network) hotspot for providing services within the wireless coverage of few tens of meters in range. Since we know the location of the AP, we can also reliably estimate the location of the user to provide context-aware services. We currently have a network of 12 BT AP's co-located with the UBI- displays (see Section 2.2). The APs are equipped with three Bluetooth radios, which can establish their own piconets of seven clients, thus 21 clients can communicate within a single AP simultaneously. We are in the process of expanding the network with additional AP's, which will be using the panOULU WLAN for their backhaul connectivity.

panOULU WSN network will be a cluster of WSN AP's around downtown Oulu. The AP's conform to the IEEE 802.15.4 specification and have dual radios (868 MHz, 2.4 GHz). An AP provides open and free multi-hop half-duplex connectivity with the 6LoWPAN protocol stack, the light-weight version of the IPv6 protocol stack intended for low-power devices. The multi-hop connectivity means that sensors form multi-hop paths, where a sensor can forward the packets of other sensors towards an AP. The half-duplex connectivity means that we can also send packets to individual sensors if needed. The upcoming installation on the 868 MHz band in spring 2010 was preceded with a small trial on the 2.4 GHz band in 2009. However, given the large amount of interference on the 2.4 GHz band and the much better range and penetration of the sub GHz band the APs will only use the 868 Mhz radios.

2.2 Large Public Displays

We are deploying two different types of large public displays, UBI-displays and UBI-projectors. They provide large visual capacity for representing information and realizing visual interfaces to the ubiquitous city. Thus, they play a very important role in creating visible artifacts of the new pervasive infrastructure – 'seeing is believing'. The UBI-displays are large interactive public displays (Fig. 3) installed on street level. The first phase installation deployed in summer 2009 comprises of six indoor displays in public buildings and six outdoor displays in the city center. The indoor displays are movable and have one 57" Full HD LCD panel in landscape orientation. The outdoor displays are installed permanently on streets and they have two adjacent LCD panels. The displays are equipped with various accessories such as Internet connection, quad core control PC, 500 MB RAID1 disk, two overhead video cameras,

NFC/RFID reader, and loudspeakers. They also contain panOULU WLAN, BT and WSN access points. The UBI-projectors are implemented with data projectors on large surfaces. First two UBI-projectors will be deployed at the City Theatre in spring 2010.

Fig. 3. (a) Outdoor UBI-display at downtown Oulu; (b) Components of an indoor UBI-display

2.3 Middleware Layer

The middleware layer comprises of a number of components that provide various services to the application layer. We have aimed at a cost efficient implementation, utilizing ready-made open source and commercial components when available. We provide just a brief description of each component, while the details can be found in related original publications when applicable.

Fuego server. Open source Fuego architecture provides distributed event-based communication overlay based on the publish-subscribe paradigm. A process publishes an event, which is routed based on its content to those processes that have subscribed to that type of events. This corresponds to so-called degenerative communication, where communicating processes are temporally and referentially uncoupled. This is very practical in a large distributed system, where processes can join and leave dynamically, particularly those executing in mobile clients. A process who wishes to publish and/or subscribe to Fuego events has to execute the Fuego client process. [11]

BlueInfo is an in-house architecture for deploying web services in the panOULU BT WPAN hotspots for cost-free context-aware mobile access. A BlueInfo hotspot either pushes subscribed services at desired intervals to registered devices (BlueInfo Push) or alternatively the user invokes a particular service by sending a simple keyword query to the hotspot (BlueInfo Pull). The BlueInfo hotspot requests the service from the origin server in the Internet and relays the response to the mobile device, possibly after adaptation for mobile viewing. [3]

GSN server. The deployment of WSN applications is supported by the open source GSN (Global Sensor Network) architecture. It comprises of several parts: a data acquisition module, a database module, a web-based query module and an external web services module. [9]

Positioning server keeps track of the current location of mobile nodes. The location is updated when a node establishes a connection with a panOULU WLAN AP or a turned-on Bluetooth radio bypasses a panOULU BT AP.

MobileIP server. A commercial MobileIPv4 solution is provided with a limited number of client licenses. This allows utilizing MobileIP in the management of vertical handovers between different access networks.

SMSC/MMSC. Message (SMS, MMS) delivery to mobile clients is supported by access to commercial SMSC/MMSC hosted by an ISP.

Control server is an in-house component that is responsible for runtime service discovery, user authentication and hosting of application metadata.

Nagios server. Nagios is a popular open source computer and network monitoring software. It allows remote monitoring of multiple computers simultaneously, reporting important metrics such as CPU load, memory usage, and network services. It can be extended to monitor custom metrics from various components. It supports also automatic notifications of service or host problems.

Resource manager is an in-house component that controls temporal access to the resources placed under its administrative domain (the UBI-hotspot) according to predefined policies. Temporal access is enforced with different types of leases, ranging from open multi-user leases to private single-user leases. Users are also able to place leases in queue, thus allowing future reservation of a resource that is currently unavailable or busy. [1]

Layout manager is another in-house component that controls the spatial access to the screen real estate of an UBI-display included in an UBI-hotspot. The Layout manager provides a SOAP interface for triggering state changes and assigning virtual screens with URL's of arbitrary web applications. [16]

2.4 Application Layer

We provide some general purpose resources for application development and for monitoring the testbed and the applications executed atop it.

UBI-MIDlet is a lightweight J2ME software layer (aka stub) that provides native service support for mobile applications by inheriting them from the standard J2ME MIDlet application framework. The UBI-MIDlet implements session control, authentication and transparent integration with the Resource manager.

UBI-key is effectively an RFID tag that serves as the electronic identity of an 'ubiquitous ouluensis'. By showing his/her UBI-key to the RFID reader of an UBI-hotspot, a user can obtain control of the hotspot for further interaction.

UBI-display monitor is provided for monitoring the visual state of the UBI-displays for maintenance purposes. The tool periodically fetches the screenshots from each UBI-display and renders them as a collage on a web page.

Analyzer summarizes the packet data collected by the Recorder with various visual presentations representing different entities, for example individual events, identities, flows between identities, or causal relationships of flows. The available of complete packet data and the visualizations facilitate visual drilling down from high-level visual abstractions to the level of individual packets and back. This in turn allows high-level visual analysis of complicated events without tedious and time consuming detailed study of large amounts of packet data. [2]

3 Usage Examples

We demonstrate the practical applicability of our testbed with six case studies representing two different types of R&D, technology pilots and service prototypes. Details of the case studies can be found in the related publications, when available.

3.1 Technology Pilots

Three technology pilots illustrate the utilization of the testbed and especially the panOULU WLAN in the deployment and empirical assessment of a particular technology in real-world setting

UMA pilot. Nokia's first public UMA (Unlicensed Mobile Access) pilot was conducted atop our testbed in June-September 2006 in collaboration with DNA (local ISP) and the City of Oulu [6]. An UMA-enabled dual-mode handset is configured to access GSM core services over unlicensed wireless network (panOULU WLAN in our case) if it is available, otherwise licensed cellular network is used. The purpose of the pilot was to evaluate the functionality of the UMA technology in authentic setting. About 60 UMA-phones were distributed to the City of Oulu personnel, who used the phones for three months, totaling 1.03 million online seconds in the panOULU WLAN.

Mobile IP technology pilot. Mobile IP is a mobility management protocol standardized by the IETF. A Mobile IP pilot was conducted atop our testbed by the City of Oulu, Fujitsu Services and Secgo Software (later acquired by Birdstep). The City's mobile workers were equipped with laptops that provided transparent and seamless connectivity in a multi-access network. The laptops were furnished with various network interfaces and Mobile IPv4 client for mobility management. The successful pilot started in September 2006 and eventually led to the purchase of a production system a year later.

HIP technology pilot. HIP (Host Identity Protocol) is a security and mobility protocol standardized by the IETF. HIP-based distributed user authentication architecture was empirically evaluated in the panOULU WLAN network by the WISEciti project in 2008. A HIP proxy was installed in the network for establishing connections with HIP mobile clients. The proxy authenticated clients, provided terminal mobility and encryption of user data over unprotected wireless links. [5]

3.2 Service Prototypes

The following three cases demonstrate the exploitation of the testbed in deploying a novel service prototype to the general public. The usage of the service is then monitored, providing feedback on the usefulness and user experience of the service.

UBI-hotspot provides rich interaction between the physical, virtual and social spaces. The first version of the UBI-hotspot is effectively an UBI-display embedded with other co-located computing resources such as panOULU WLAN, WSN and BT APs. UBI-hotspot offers a wide range of services via different interaction modalities including mobile. In the current interaction model the UBI-hotspot alternates between a passive broadcast mode and an interactive mode. The transition to the interactive mode is triggered when a user touches the touch screen or presents an UBI-key to the RFID reader, or a face is detected from the video feed of the two overhead cameras. In the broadcast mode the whole screen is allocated to a digital signage service called UBI-channel. In the transition to the interactive mode the UBI-channel is smoothly squeezed into the upper left hand part of the screen and two additional virtual screens are created, one assigned for a touch screen portal called UBI-portal and another assigned for mobile services. The UBI-portal is effectively a web portal of various information and leisure services (web pages), which can reside on any web server in the public Internet. The default view of the UBI-portal can be configured on per hotspot basis. A video illustrating the services of the UBI-hotspot version 1.0 is available at http://www.ubioulu.fi/node/133. We have deployed a network of 12 UBI- hotspots in pivotal indoor and outdoor locations around downtown Oulu. The UBI- hotspots have been in everyday use by the general public since June 2009, attracting on average about 50 interactive sessions per panel each day. [8]

UBI-AMI prototype for advanced metering demonstrates the capabilities of the panOULU WSN network. The UBI-AMI socket sensors measure the power consumption of the devices attached to it together with temperature and illumination in that location. The UBI-AMI mains sensor measures the power consumption from the main electricity meter. The sensors packetize the measurement data and transmit the packets to a panOULU WSN AP. It forwards the packets to the UBI-AMI server (based on the GSN server), which filters and stores the data. The user can explore the data via a web interface, for example study how expensive is the electricity consumption of particular devices. Further, the user can configure to receive a SMS or email alert if a particular device (e.g. a freezer) is accidentally turned off. Thanks to the half-duplex connectivity provided by the 6LoWPAN protocol stack, the user can also remotely turn off the devices (e.g. a coffee machine) connected to a particular sensor. An UBI-AMI pilot with ten households as test users commenced in January 2010.

panOULU Luotsi was a location-based information mash-up for the users of the panOULU WLAN. XML content in various forms, such as RSS/ATOM feeds from several content providers were automatically merged into the Luotsi database using a flexible XML aggregator. It allowed mapping different heterogeneous information feeds into the Luotsi database without any changes to the application source code. The location of the user was provided by the Positioning server, estimated by identifying the WLAN AP to which the user's wireless device was connected. The information relevant to the user's current location is imposed on a map for a location-based browsing. [4]

4 Discussion

This type of large-scale testbed deployment comes with many challenges, first and foremost financial and technological viability and sustainability. The public sector has made a considerable capital investment in our testbed, thus we wish to offer it as open horizontal resource to the whole community. Many (academic) infrastructure deployments have fallen apart, because they did not have any long-term financial basis for covering operational and renewal expenses. The panOULU WLAN has survived seven years with its PPP model, where the owners of the WLAN zones are responsible for all the expenses incurred by their zones, and the City of Oulu sponsors the core services outsourced to subcontractors. In 2010 the City of Oulu uses about 75000 € to the panOULU WLAN, which corresponds to 0.04 € per citizen in a month – a very smart investment in an open and free public service used by 20000 people every month. The panOULU WLAN is due a major renewal of APs in a few years, if a city-wide WLAN network is still deemed a worthwhile resource amidst improving mobile data networks. To cover the operational expenses of the UBI-hotspots we are selling a portion of their capacity for commercial use. However, as of now we do not have such business model for the panOULU BT and panOULU WSN networks.

Shared urban spaces can be very dynamic, diverse and dense with complex ownership and decision making mechanisms. While you may have full control and command of your own research lab, the public city space falls under the authority of the city administration. We are very fortunate to have the unwavering support of the City of Oulu's administration, which is mandatory for the implementation and sustainability of this kind of testbed deployment. At the same time you have to comply with the municipal decision making procedure, which can introduce delays at times. For example, the installation of the panOULU WLAN APs and UBI-hotspots was subject to the formal urban planning process of the city. Further, you may need favorable cooperation with private parties such as property management companies and housing cooperatives, for example to acquire cost-free sites for APs.

Another major challenge is the operational execution of maintenance, where research organizations are typically not good at. Outsourcing maintenance or allocating designated maintenance personnel is highly recommended, if you can afford it. If researchers have to take care of maintenance, then you need to specify clear roles to avoid confrontations between research and maintenance tasks. Further, maintenance expenses and the availability of the testbed can be greatly optimized by automating maintenance.

Research on a real-world testbed in a city center differs from lab research in few important aspects. First is the demand for high quality engineering. For example, the engineering of the UBI-hotspots the user community expects to be available 24/7 is a totally different ball game than presenting a one-shot demo to your sponsors in your own lab. Second, you have to comply with the many ramifications of the real-world setting. A research prototype may depend on particular infrastructure which will not be available in the real-world in the foreseeable future. No matter how fascinating a novel mobile application developed for a particular exotic mobile device may be, it is rather pointless to bring the application to the real-world setting, if the market penetration of the mobile device is negligible. Further, while most lab studies happily

ignore value networks, business models and economic sustainability, we are trying to build a sustainable ecosystem around our infrastructure, for example by selling a portion of the capacity of the UBI-hotspots for commercial use to cover their operational expenses. This commercial dimension limits the research use of the UBI- hotspots, which may be difficult for researchers to accept. Another important difference is the public scrutiny in the real-world setting. While nobody cares about the mistakes you do in your own lab, a deployment in a city center is subject to daily scrutiny by the general public and media, which in our case has been rather ill-tempered at times. Finally, while usability evaluation has established itself as the de facto yardstick in lab studies, there is no such universally accepted methodology for evaluating real-world deployments. This is probably one of the reasons why the mainstream research community values lab studies over real-world studies.

We have a number of ongoing activities to make our testbed available to the whole community and to stimulate open innovation. For example, students build new services to the UBI-hotspots as their course works. We currently have a public tender for businesses to purchase rights to offer commercial services in the UBI-hotspots. We are executing a national 'UBI-challenge' where both individuals and businesses are challenged to innovate and implement novel services to the UBI-hotspots so that best proposals are supported with grants. An international 'UBI-challenge' prepared with a number of leading international researchers will be launched in 2010, inviting the international research community to show what they are able to do atop our urban computing testbed.

Acknowledgments. The authors are grateful for the financial support of the Finnish Funding Agency for Technology and Innovation, the European Regional Development Fund, the City of Oulu and the Urban Interactions consortium.

References

1. Jurmu, M., Boring, S., Riekki, J.: ScreenSpot: Multidimensional resource discovery for distributed applications in smart spaces. In: Fifth Annual International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services, Dublin, Ireland (2008)
2. Kenttälä, J., Viide, J., Ojala, T., Pietikäinen, P., Hiltunen, M., Huhta, J., Kenttälä, M., Salmi, O., Hakanen, T.: Clarified Recorder and Analyzer for visual drill down network analysis. In: Tenth Passive and Active Measurement Conference, Seoul, South Korea, pp. 122–125 (2009)
3. Kukka, H., Kruger, F., Ojala, T.: BlueInfo: Open architecture for deploying web services in WPAN hotspots. In: 7th IEEE International Conference on Web Services, Los Angeles, CA, USA, pp. 984–991 (2009)
4. Kukka, H., Ojala, T., Tiensyrjä, J., Mikkonen, T.: panOULU Luotsi: A location based information mash-up with XML aggregator and WiFi positioning. In: 7th International Conference on Mobile and Ubiquitous Multimedia, Umeå, Sweden, pp. 80–83 (2008)
5. Kuptsov, D., Khurri, A., Gurtov, A.: Distributed user authentication in Wireless LANs. In: 10th IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks, Kos, Greece (2009)

6. Nokia press release: Nokia's first public UMA pilot over WLAN is underway in Oulu, Finland (July 27, 2006),
http://www.nokia.com/press/press-releases/showpressrelease?newsid=1066083
7. Ojala, T., Hakanen, T., Salmi, O., Kenttälä, M., Tiensyrjä, J.: Supporting session and AP mobility in a large multi-provider multi-vendor municipal WiFi network. In: Third International Conference on Access Networks, Las Vegas, NV, USA, pp. 89–101 (2008)
8. Ojala, T., Kukka, H., Lindén, T., Heikkinen, T., Jurmu, M., Hosio, S., Kruger, F.: UBI-hotspot 1.0: Large-scale long-term deployment of interactive public displays in a city center. In: Fifth International Conference on Internet and Web Applications and Services, Barcelona, Spain (2010)
9. Salehi, A., Aberer, K.: GSN, quick and simple sensor network deployment. In: 4th European conference on Wireless Sensor Networks, Delft, The Netherlands (2007)
10. Sharp, R., Rehman, K.: The 2005 UbiApp Workshop: What Makes Good Application-Led Research? IEEE Pervasive Computing 4(3), 80–82 (2005)
11. Tarkoma, S., Kangasharju, J., Lindholm, T., Raatikainen, K.: Fuego: Experiences with mobile data communication and synchronization. In: 17th Annual International Symposium on Personal, Indoor and Mobile Radio Communications, Helsinki, Finland, pp. 1–5 (2006)
12. Greenberg, S., Buxton, B.: Usability evaluation considered harmful (some of the time). In: twenty-sixth annual SIGCHI Conference on Human Factors in Computing Systems, Florence, Italy, pp. 111–120 (2008)
13. Müller, J., Jentsch, M., Kray, C., Krüger, A.: Exploring factors that influence the combined use of mobile devices and public displays for pedestrian navigation. In: 5th Nordic Conference on Human-Computer Interaction, Lund, Sweden, pp. 308–317 (2008)
14. Storz, O., Friday, A., Davies, N., Finney, J., Sas, C., Sheridan, J.: Public ubiquitous computing systems: lessons from the e-Campus display deployments. IEEE Pervasive Computing 5(3), 40–47 (2006)
15. Want, R., Schilit, B., Adams, N., Gold, R., Petersen, K., Goldberg, D., Ellis, J., Weiser, M.: An overview of the PARCTAB ubiquitous computing experiment. IEEE Personal Communications 2(6), 28–43 (1995)
16. Lindén, T., Heikkinen, T., Ojala, T., Kukka, H., Jurmu, M.: Web-based framework for spatiotemporal screen real estate management of interactive public displays. In: 19th International World Wide Web Conference, Raleigh, NC, USA (2010)

Experimentally-Driven Research in Publish/Subscribe Information-Centric Inter-Networking

András Zahemszky[1], Borislava Gajic[2], Christian Esteve Rothenberg[3], Christopher Reason[4], Dirk Trossen[4], Dmitrij Lagutin[5], Janne Tuononen[6], and Konstantinos Katsaros[7]

[1] Ericsson Research, Finland
andras.zahemszky@ericsson.com
[2] RWTH Aachen, Germany
gbo@mobnets.rwth-aachen.de
[3] University of Campinas, Brazil
chesteve@dca.fee.unicamp.br
[4] BT Research, UK
{dirk.trossen,christopher.reason}@bt.com
[5] Helsinki University of Technology, Finland
dmitrij.Lagutin@hiit.fi
[6] Nokia Siemens Networks, Finland
janne.tuononen@nsn.com
[7] Athens University of Economics and Business, Greece
ntinos@aueb.gr

Abstract. Testing and evaluating new architectural propositions is a challenge. Given the usual variety of technologies and scales involved in the necessary evaluation, a one-size-fits-all approach does hardly suffice. Instead, a collection of evaluation and experimentation methods must be chosen for a comprehensive testing of the proposed solutions. This paper outlines some of the approaches chosen for an architectural proposition that establishes a publish/subscribe-based internetworking layer for the Future Internet. For that, we outline challenges we identified when turning to experimentation as a means of evaluation. We then present the variety of emulation as well as experimental test bed efforts that attempt to address these challenges. While this is not to be seen as a conclusive summary of experimental research in this space, it is an attempt to summarize our efforts as a work-of-progress for others working the architectural field.

Keywords: publish-subscribe, experimental research, NetFPGA, testbed.

1 Introduction

Future Internet research requires at least three key ingredients to have chances of success. First, a clear *vision* that outlines the direction and sets the goals and requirements of the envisioned global communication infrastructure. Second, *experimentally-driven research* to validate the architectural proposals at scale and under realistic

T. Magedanz et al. (Eds.): TridentCom 2010, LNICST 46, pp. 469–485, 2011.

scenarios. Third, understanding the *business* incentives for adoption, which requires socio-economic market evaluations and industry engagement early in the feedback and re-design loops.

The PSIRP (Publish-Subscribe Internetworking Routing Paradigm) project [1] is an EU FP7 funded project that started in January 2008 and aims at covering all three research fronts of a clean-slate design approach that departs from the current host-centric IP inter-networking to an information-centric future Internet. The PSIRP vision is inspired by the observation that information/content – what a user wants – should have a more central role in future network architectures than it does in today's Internet host-to-host conversation model [2,3,4,5,6]. To this end, architectures based on data-oriented primitives like publish/subscribe [7] and the similar (e.g., get/response, find/register) [8] are well-suited for the unwieldy amounts of named linked data retrieved from the Web and exchanged via overlay networks like P2P and content delivery networks.

The project has already outlined the direction to realize this vision by defining design principles [9] and proposing several design choices towards a novel pub/sub-based Internet-scale architecture [10, 11]. Time has come to accomplish the second key component of clean slate future Internet research: experimental validation and evaluation at scale. From the design phase of the project, prototyping work is one major component in the development of the architecture to provide fast feedback from the practical experiences enabling a fruitful top-down and bottom-up dialogue.

Most researchers have at some point faced questions such as “what is the performance of my new protocol”, “how does my new technology perform in a highly distributed environment” or “how does my pre-commercial code perform under more realistic networking conditions”. When developing clean slate technologies this is no different but arguably more challenging. In the most part, these questions are solved either experimentally or using models, and it is the former of the two which this paper focuses on; “How do I experimentally test and evaluate in a realistic setting?” It is important to note the inclusion of the “in a realistic setting” clause, as testing any multi-domain protocol designed to be run over a potential future Internet architecture over the current Internet will rarely provide irrefutable evidence for its performance.

In order to enable large-scale experimental research with the required levels of flexibility of future Internet architectural proposals, big efforts are undergoing on both sides of the Atlantic in projects such as GENI, FIRE, FEDERICA and OneLab2. Their common denominator is their goal of providing a playground for researchers to validate their visions under "realistic" scenarios. Typical experimental evaluation methods such as emulation, simulation and (experimental and usually local) testbeds, have particular strengths and weaknesses, so an evaluation architecture which combines all three should provide the greatest flexibility while retaining the best features of each. Such a rich evaluation playground is the ultimate goal of our validation efforts.

In this paper, we describe the experimental approach taken by the PSIRP project and the components of the underlying research infrastructure. We account for the experiences gained when working with the selected evaluation tools and implemented prototypes. First, we introduce the background fundamentals of the conceptual architecture and the evaluation challenges (Section 2). Then, we dissect the experimentally-driven visionary research divided by implementation work (Section 3) and the experimental verification efforts (Section 4). Finally, we describe the lessons learned and the ongoing work towards a unified evaluation approach (Section 5).

2 Background

The current Internet architecture focuses on communicating entities, largely leaving aside the information to be exchanged among them. However, trends in communication scenarios show that *what* is being exchanged is becoming more important than the *who* is exchanging information. Van Jacobson describes this as moving from *interconnecting machines* to *interconnecting information* [6]. The ambition of the PSIRP project is to investigate major changes to the current IP layer, up to the point of replacing this layer with a new form of inter-networking. To this end, PSIRP undergoes all phases of a clean-slate design project, from state-of-the-art survey over outlining basic design principles and understanding design choices through the definition of conceptual and actual architectures and their implementation. Architectural and technological choices are evaluated from the angles of security, socio-economic and quantitative design constraints. In the following, we briefly outline the underlying conceptual architecture of PSIRP, appreciating that we cannot present the full breadth of the architectural concepts in the given space. Hence, the interested reader is referred to [1] for more information.

2.1 Conceptual Architecture

Based on the design principles outlined in [9], the following information-based architecture relies on basic labeling (cf. *Everything is information*) and grouping of information (cf. *Information is scoped*), while providing a publish-subscribe service model (cf. *Equal control).* The main objective of the architecture is to provide the required mapping of these concepts onto concrete forwarding relations between endpoints, producing and consuming information. This keeps the network architecture simple, while enabling more complex application-level naming structures, as suggested in [6] and similar work [2,3,4,8].

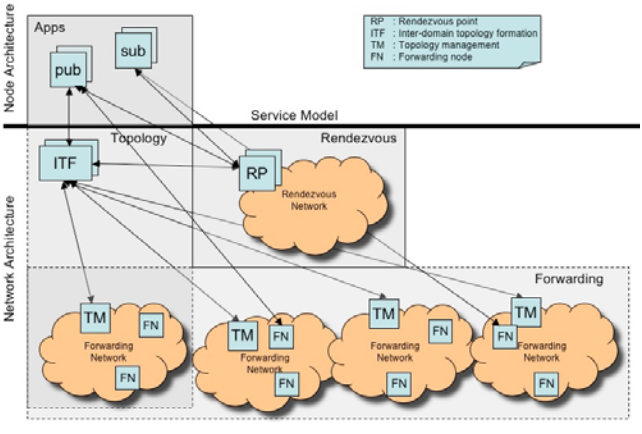


Fig. 1. Conceptual Architecture

Figure 1 presents the main architectural components implementing these design concepts. The *pub* and *sub* components at the application level implement applications based on basic publish/subscribe network services, enabling *publications* and *subscriptions* towards information items labeled by RId (Rendezvous ID) within particular scopes, identified by SId (Scope ID) – see more below.

Transactional services, operating in request-reply mode, can easily be supported through a publish/subscribe model [7], with the server subscribing to receive requests. From this basic mode of communication, we can bootstrap internal network operations as well as offer a new information-centric service API, similar to [8]. Such a new communication API replaces in many ways the role of traditional middleware layers since it conflates low-level information discovery as well as location determination of publishers and subscribers into a single network service, therefore largely eliminating the need for such mapping functions to exist on application level. However, there is still a need for mapping application-level information concepts onto the basic concepts provided by our architecture – something being left outside the scope of the network architecture considered here.

The architecture itself consists of three main functions: *rendezvous*, *topology* and *forwarding*. Generally, the *rendezvous* function implements the matching between publishers and subscribers of information items, each identified via a RId. Information items logically reside within at least one scope. Each scope is identified via a SId, which is in turn provided by dedicated rendezvous points (RP). Hence, rendezvous points match the semantic-free information items within the scope they are serving. There is at least one rendezvous point per scope, each of which subscribes to the SId through a global rendezvous system. Upon subscription to an information item in the scope, the request can be routed either to all or to the 'best' rendezvous point, using anycast-like functionality. Furthermore, rendezvous points implement policies associated with the matching, such as access control.

Once the rendezvous point has matched a publication and one or more subscriptions, the forwarding topology is created in negotiation with the inter-domain topology formation (ITF) function. This is based on the publisher and subscriber “locations” on the level of autonomous systems (ASes), the applicable policies and the ITF information that includes peering and transit relationships among ASes. This is similar to BGP or (G)MPLS PCE, but the underlying networks forward information, not (opaque data) packets, i.e., there exists a rich set of policies attached to potentially every information item.

In addition to building inter-domain paths between the forwarding networks to which the publisher and subscribers are attached to, appropriate intra-domain paths need to be constructed. This is done in collaboration with the *topology management* function that resides in every AS. This function is responsible for instructing its local *forwarding nodes* (FNs) to establish paths to local publishers and/or subscribers or to serve as transfer links between ASes. As in the current Internet architecture, this approach does not prescribe any particular intra-domain forwarding mechanism, with the one constraint that the local mechanisms should support ITF compliant policies.

2.2 Challenges

The architecture presented in Figure 1 aims at providing an internetworking architecture made for the foreseen scale of a future (information-centric) Internet. Evaluating the concepts for such architecture therefore face particular challenges that comes with that ambition:

- *Scale*: inter-domain functions, such as for rendezvous and forwarding, are built for large scale. This requires experimental methods that can scale to appropriate sizes. Experimentation alone is unlikely to suffice for scaling experiments.
- *Technology Variety*: inter-domain functions such as forwarding are designed to work over a variety of technologies, similar to today's Internet. This, however, requires the availability of such wide variety of technologies when evaluating crucial parameters, such as delay or efficiency.
- *Usage Variety*: it is hard to predict potential usages for any network architecture – the current Internet is the best example for this. Hence, potential user involvement is crucial but also a variety of different usage models for isolated experiments.
- *Economic Variety*: inter-domain functions, such as rendezvous and forwarding, heavily depend on the underlying business relations of ASes in their overall performance. Hence, a proper understanding of various business relations, possibly vastly different from today's peering relations in the Internet, is required to provide insight in the effectiveness of novel inter-domain functions.
- *Platform Variety*: it is obvious that a single platform for testing is hardly achievable given different operation systems, virtualization approaches and simulation/emulation platforms available. The experimental approach must cater to this variety.

In the following, we outline the project's approach to cope with these challenges. It is the ambition of this paper to outline a coherent testing and experimentation approach although its creation is driven by bottom-up testing and evaluation activities and a post-rationalization of these activities in a coherent framework that might aid similar activities in the future that need to address the challenges outlined above. Before doing so, however, we present a brief summary of the implementation work done in order to better understand the chosen evaluation methods.

3 Implementation Work

The PSIRP project works towards a publish/subscribe solution, where even IP forwarding is re-considered. This creates a need for a prototype with a different structure than existing systems. The prototype development [12] is divided into two separate areas, namely the Lower Layer implementation (LoLI) and the Upper Layer implementation (UpLI). The LoLI is motivated by the need for a new kind of data handling at the end-host (i.e., internal publication management) as well as for forwarding data packets due to the pub/sub architectural approach. The requirements on locating publications and managing the network topology are considered to be "upper layer" tasks. This section goes first through the LoLI, including the FreeBSD-based blackboard

implementation (Blackhawk), the security and PLA-related implementation, and the NetFPGA implementation of zFilter forwarding. Finally, we describe the Rendezvous and Topology functions in the UpLI developments.

3.1 Blackhawk: FreeBSD Node Implementation

The current node architecture [12] implements parts of the PSIRP service model and API [9]. It consists of the following pieces:

- A blackboard that implements a simple memory object model inside a node. Conceptually, the blackboard is the place where data items are stored as publications that can be subscribed to.
- An API for publishing data items to the blackboard, subscribing to them, and getting notifications when new versions of them are published.
- Applications that can communicate with each other via the blackboard. This includes helper applications that implement different network-level functions.

In the *Blackhawk* prototype for FreeBSD, the blackboard is implemented as a kernel module, as shown in Figure 2. It is integrated with the operating system's virtual memory system, i.e., publications in the blackboard correspond to virtual memory objects and pages. The API is implemented as a library that communicates with the kernel module using system calls. It provides functions for creating, publishing, and subscribing to publications. Notifications about publish operations on the blackboard can be acquired via the *kevent* system of FreeBSD. In addition, a file system view to the blackboard is provided as a hierarchy of scopes, publications, publication versions, and memory pages (which all have their own RIds).

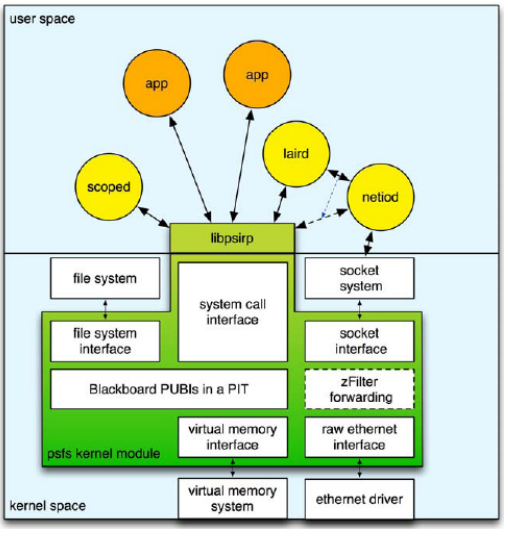


Fig. 2. Node Architecture

As mentioned above, user space applications can use pub/sub-based inter-process communication by publishing data to the blackboard, from where other processes can retrieve the publications by subscribing to them. The component wheel functions (e.g., rendezvous and topology management) are also implemented as applications. However, some parts of the component wheel (e.g., forwarding) may also be implemented in the kernel space to achieve better performance. As an example of helpers, the current version of the Blackhawk prototype features three helpers:

- A scope helper (*scoped*) takes care of instantiating, updating, and re-publishing scope publications, i.e. data items that contain collections of RIds.
- A local-area rendezvous helper (*laird*) extends the blackboard model into the network. It provides local monitoring for publish/subscribe operations, and it advertises the locally published publications to the local-area rendezvous node.
- A network I/O helper (*netiod*) implements packet fragmentation/assembly in addition to forwarding, using sockets for sending and receiving packets over links in the network.

The rendezvous helper communicates with the network I/O module when publication metadata or data needs to be sent into the network. In the reverse direction, the network I/O helper dispatches received metadata to the rendezvous helper.

3.2 Security and PLA Implementation

Packet Level Authentication (PLA) [13] is a novel method for providing availability at the network layer by using per packet cryptographic signatures. PLA was originally implemented for IP networks; however it does not depend on IP and therefore can also be used with other network layer solutions such as PSIRP. PLA's main aim is to allow nodes on the path to independently verify packets without having separate security associations with the sender, or previous nodes that have handled packets. Any node can verify whether packets has been modified, duplicated or delayed, therefore invalid packets can be dropped immediately, before they reach the destination.

PLA works by adding a security header including the sender's cryptographic identity, certificate from a trusted third party, timestamp, sequence number and the cryptographic signature. The timestamp and sequence number offer protection against replay attacks, while the signature protects the packet's integrity and offers accountability. PLA uses elliptic curve cryptography (ECC) since it offers a good security with compact key sizes. While public key signatures are computationally intensive, they can scale to high speed networks and low power devices as long as dedicated hardware is used for accelerating signature calculations [14]. Preliminary simulation results have shown that a 90nm dedicated ASIC would be able to perform almost one million signature verifications per second, such performance would be enough verify 5Gbps of average traffic.

In PSIRP, PLA is used mostly used to secure control messages (publish/subscribe), and can be optionally be used for securing all traffic. In our system, the most important security properties of control messages are integrity protection and authentication. For example, has the packet been modified? Does the publisher have a permission from the scope to publish in certain SId:RId? PLA functionality has been implemented as a separate library [15], which is used by the PSIRP networking daemon to add and verify PLA headers.

3.3 NetFPGA Forwarding Implementation

Forwarding nodes implement the multicast source-routing mechanism described in [15] based on an in-packet Bloom filter referred to as *zFilter*. The mechanism allows for compactly representing a delivery tree within the limited header space of a packet. Basically, a routable Bloom filter is formed by ORing the Bloom masks of the network links of a delivery tree. Forwarding decisions are based on simple logical AND operations between the zFilter and the forwarding node Link ID table. As with all Bloom filter based approaches, false positives of such AND operations can occur, leading to false deliveries along the AS links. Hence, determining a rate for such false positives is a typical performance evaluation objective. Mechanisms to minimize such false positives have been proposed in [16], such as the introduction of virtual link identifiers, which combine certain paths/trees into a virtual (single) link, 'thinning' out the Bloom filter space and therefore reducing the potential for false positives.

The zFilter forwarding algorithm has been implemented on NetFPGA [17], a flexible and open hardware platform for research and classroom experimentation in terms of networking and traffic processing. The implementation [18] was based on the Stanford reference switch implementation, which was modified to create a simple zFilter switch. According to early measurements, the efficiency of the NetFGPA-based zFilter forwarding is very good and requires about 3-5μs per hop, which represents a lower latency than the reference IP router implementation.

Our experiences with NetFPGA as the prototyping platform are overall positive. It is a solid development platform that is available in a complete package and it is suitable for clean-slate developments requiring line-speed operations.

3.4 Rendezvous

The PSIRP rendezvous architecture, defined in [9, 27], is a composition of modular rendezvous networks that are interconnected to form a globally reachable inter-domain rendezvous system. The rendezvous networks are formed by rendezvous nodes (RNs) that are organized as a policy controlled inter-domain hierarchy. Each RN may host multiple rendezvous points (RPs) that are logical meeting places in the pub/sub system for a certain <SId, RId> pair i.e. for each pair there exists at least one RP in the rendezvous system. In many cases the same RP is shared by the publications in the same scope. In the event of rendezvous, RP initiates creation of the forwarding path creation in the topology function that, when finished, enables transmission of data between the publisher and the subscriber.

The rendezvous function is implemented in two separate instances; the local rendezvous helper, described in the Blackhawk prototype, that handles rendezvous in local node and in small-scale local area networks, and the rendezvous node, which implements the rendezvous network. Large and geographically dispersed Autonomous systems (ASes) may be covered by multiple rendezvous networks, but in the typical case a rendezvous network would be a collection of rendezvous nodes from cooperating ASes. Therefore, in the same way as peering between two rendezvous nodes is supported, the current rendezvous network implementation can be seen also as a simple version inter-domain rendezvous system.

Figure 3 illustrates the architecture for the existing two versions of the rendezvous node implementation: the UDP forwarding based standalone version and the new Blackhawk integrated version. The architecture includes a stub version of the topology function that cooperates with the rendezvous function in the pub/sub system. Both versions implement the same functions to establish and operate pub/sub rendezvous networks.

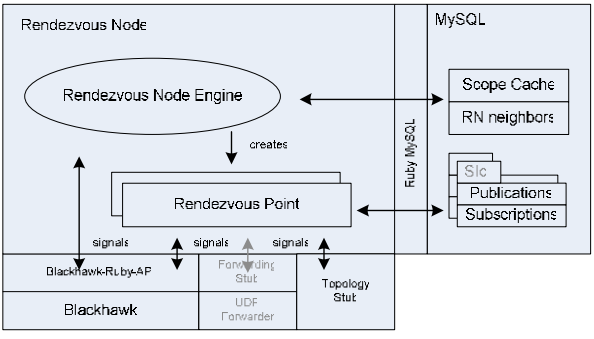


Fig. 3. Rendezvous node implementation architecture

3.5 Topology

Following the administrative division of the current Internet, we distinguish between intra-domain and inter-domain topology management mechanisms as two functionally separated units retaining a strong interconnection between their structural pieces. The main role of intra-domain topology management is the discovery of topology information, using it as an input for computing necessary network states, and sending updated forwarding information to the relevant nodes. The inter-domain topology formation functions on the domain level, in addition to discovery, is responsible for configuring and maintaining inter-domain topology states for creating forwarding paths based on various policy compliance requirements.

Our current Python based implementation of topology management is divided into two modules: client and server, which can simultaneously coexist on each node. The client module runs on each forwarding node and is mainly responsible for discovering local connectivity information, whereas the server module collects these local information pieces and structures them together to form a picture of the overall network topology within the domain of operation. The server module is also responsible for computing the optimal forwarding paths and publishing that information towards forwarding nodes. Additionally, each forwarding node runs a link state helper module which maintains the table of "known" links along with link related available information, e.g., throughput, and delay. Information about relevant link properties can be provided by low level helper functions, which collect physical data about the link.

In order to facilitate the exchange of required data different scopes are defined for exchanging particular information, e.g., a scope for exchanging the existence of information coming from each particular node, a scope for distributing and collecting information representing the set of neighboring nodes, a scope for dissemination and

collection of link data messages. Using the publish/subscribe paradigm, forwarding nodes and topology servers receive and update the required information ("Hello", "LSA" messages). The topology manager implementation also performs simple forwarding tree and zFilter creation using shortest path tree calculation.

4 Addressing the Evaluation Challenges

With the understanding of how we implemented the architectural foundations that we outlined in Section 2 in a component architecture that we presented in Section 3, we can now move on with addressing the evaluation challenges presented in Section 2.2. For this, we give specific examples from evaluation tasks within PSIRP targeting the identified challenges.

4.1 Simulation and Emulation: Addressing Scale and Variety of Technologies

For rapid evaluation of different networking solutions in terms of packet-level performance, simulations are the standard approach. In particular, inter-domain solutions, such as the outlined rendezvous and inter-domain topology formation (see Section 2.1), are targeted in simulative evaluations. But also our developed forwarding solutions are candidates for simulations, in particular when coupled with emulation methods. In the following, we outline the used technologies for these tasks.

4.1.1 NS-3

Simulation is a common means to achieve scale of evaluation, not requiring direct equipment to be handy for evaluation. Given our scale challenge, it is natural to resort to simulations as a central method to address this challenge. Examples of architectural solutions that are simulatively evaluated are the rendezvous and forwarding solutions. Our natural choice for performance analysis of parts of the PSIRP architecture is ns-3 [18], as it offers a clean simulator architecture, easy extendibility, and features for network emulation. Evaluation work with ns-3 included network coding solutions, as well as the performance of the multicast forwarding mechanisms based on zFilters. On-going work includes exploitation of network emulation functions and integration with the prototype implementation (Blackhawk).

Ns-3's easy extendibility is very attractive for projects designing clean-slate architectures, as new protocols can be installed into any desired level of the networking stack. Following this design philosophy, we extended the simulator to support the zFilter-based forwarding. Primarily written in C++, in ns-3 new mechanisms can be added intuitively via the techniques of class inheritance and new class creation. The implemented forwarding layer supports the major design elements presented in [15], including the optimization of using multiple parallel forwarding tables, various loop prevention techniques, fast reroute mechanisms and virtual links. With the usage of the simulator, we showed that zFilter forwarding is feasible, and supports unicast comfortably, as well as sparse mode multicast communication up to topologies of the size of metropolitan area networks.

The integration of ns-3 simulations with real world traffic is possible via two modes. *Virtualization* allows real hosts to communicate via simulated networks, while

network emulation allows simulated nodes to exchange information through real links, i.e. a networking testbed. So far, an early inter-operation test has been carried out between a simplified forwarding simulator and the first iteration of the BSD prototype, while future work includes further integration into a PSIRP testbed (cf. Section 5).

4.1.2 Network Emulation

Emulation allows for inserting a "sense of reality" into a simulative framework by emulating part of the real-world in combination with developed solutions, e.g., running our real-world forwarding implementation (see Section 3.3) in an emulated Ethernet network of a larger size than we achieve with a testbed setup.

For tests and evaluation activities involving a larger number of nodes in a controlled environment we use the network emulation testbed at RWTH. The testbed consists of powerful servers equipped with multi-core CPUs and four Gigabit Ethernet network interfaces each. All the servers are connected to a high-performance switch allowing for different network topologies to be set up for the experiments. Control traffic is sent over a dedicated network interface, over a different switch, so as not to interfere with measurements. Each of the servers is capable of hosting a large number of virtual machines functioning either as communication endpoints, or nodes normally associated with the network infrastructure such as forwarding nodes, rendezvous servers or topology management nodes.

Experiments with even larger number of nodes can be realized by combining the use of VMs with network emulation and tap techniques provided by ns-3 network simulator. In such a setup individual VM instances can emulate complete forwarding infrastructures within individual domains, while other VMs connected to those emulated forwarding domains can act as traffic sources or as nodes offering rendezvous or topology management services. On the other hand, each VM can be connected with simulated networks via ns-3 tap device, acting as a regular PSIRP node. Therefore we can implement the case of PSIRP traffic originating on VM instances running the developed prototype implementations and traversing over large scale simulated networks.

4.1.3 OMNet++ and OverSim

A medium term alternative to a native implementation of PSIRP, operating directly on top of the network hardware, is an overlay implementation on top of IP. The overlay work has so far focused on network support for scalable multicast, the main enabler for providing the entire PSIRP functionality as an overlay solution. In particular, a solution has been designed to operate on top of the Pastry DHT based content routing scheme [19] and the Scribe overlay multicast scheme [20]. Special attention is paid to the incremental deployment process of the overlay architecture as well as to the potential benefits of in-network caching. In addition, the concept of hierarchical DHTs has been explored with the purpose of making routing conform to the policy-compliant interconnection of networks on the Internet. For the work on DHT based overlays, the OverSim [21] platform has been used, which is an overlay network simulation framework for the OMNeT++ simulation environment. The OverSim framework provides implementations of several overlay schemes and applications, including Chord and Pastry, as well as overlay multicast schemes, such as Scribe.

Due to the popularity of P2P content distribution applications, BitTorrent was chosen as the main application model for benchmarking. In order to be able to perform a comparison study between our overlay multicast based BitTorrent alternative [22] and the regular BitTorrent of today's Internet, the BitTorrent suite of protocols for OMNeT++ and a churn generator module for OverSim based on an analysis of real BitTorrent traces was created [23].

4.2 Application Innovation Process: Addressing Usage Variety

Running component and architectural evaluation according to identified performance parameters is crucial. But the real test for any solution is that of being applicable to a certain (often large) set of real-life applications. It was recognized early in the planning phase that our efforts would not be able to address the potential usage variety for a Future Internet. Hence, an application innovation process was established that would attract developers to the new platform for trying out novel usages of the platform. This process is facilitated by the open source release of major node and network components, allowing for developing applications on an open platform with existing network technology like Ethernet.

One straightforward example of such usage is the development of a plug-in for Firefox, which provides mechanisms for users to subscribe to publications using the PSIRP protocol through their web-browser. The plug-in intercepts all PSIRP protocol calls in the address bar or in a link embedded in a webpage (e.g. psirp://), passing the PSIRP parameters (SId:RId) to the XPCOM component. This component interacts with the PSIRP library by subscribing to the publication identified by the SId and RId pair and, after retrieving it, the component saves it as a local file. Finally, the plug-in opens the fetched publication and displays it in the web-browser. Currently, the retrieved publication is saved as a local file, which is later opened by the web browser. As a future improvement, publications will be displayed directly in the web browser without requiring copies to be saved locally.

Other applications are currently explored in collaboration with external partners. But it is obvious that this challenge is a difficult and time-consuming one to be addressed.

4.3 Testbed Infrastructures: Addressing Scale and Variety of Technologies

Experimental testing of the technology solutions developed in PSIRP is crucial for evaluating the viability of the overall proposition of the project, namely to develop a viable alternative to the current IP paradigm. For this to happen, the implementation work is integrated into a single coherent prototype (see Section 3) [24]. As a result, not only is the architecture work converging, but also the various implementation efforts on these architectural components are starting to converge into a coherent and running system. In particular, core components like the node architecture, forwarding and rendezvous node (see Section 2.1 and 3.2) are coming together, enabling the progression towards a first networked setup of a PSIRP network. Although crucial components, such as the ITF function (see Section 2.1) are still missing from this coherent prototype, the foundation has been set to perform experimental testing in testbed infrastructures. A crucial step in this testing is the extension of a limited laboratory

prototype towards a fully networked test network that operates based on the central components of the architecture.

A first step in this direction is the establishment of localized test network at the BT and University of Essex facilities in the UK. These facilities are based on a heterogeneous network infrastructure that was built under the recently finished UK TSB (Technology Strategy Board) funded project HIPnet. It provides a variety of access technologies in the wireless and wireline domain, e.g., WiMax, WiFi, and all-optical fixed infrastructure. The infrastructure spans the local campus at Essex University, located at the edge of Colchester (UK). The university's facilities hold about 2500 students in their dorms, with access to their infrastructure. The wireless coverage has recently, in June 2009, been extended to full campus coverage with a single SSID. The WiMax coverage spans most of the campus, using a rotating antenna. The physical connectivity on the optical level extends to Cambridge University as well as to the BT facilities at Adastral Park (UK). This variety of access technologies is currently utilized for a fully networked PSIRP testbed. Given the largely Ethernet basis of the infrastructure, this is relatively easy for the wired part. Specifically, there are currently three types of machines being installed for a simple PSIRP network setup. The two end nodes are publisher and subscriber, respectively, currently running the latest release of the PSIRP node architecture (Blackhawk). These will be available at BT premises, at Essex as well as Cambridge University, enabling a variety of test cases through dedicated test applications running on these nodes. The third type is a forwarding node, utilizing the current NetFPGA implementation [16] as well as the FreeBSD-based forwarding engine.

Such setup will not only enable demonstrations but also provide a testing ground for the implementation itself. For instance, real network load performance experiments can be conducted for evaluating (a) end node architecture performance and (b) forwarding node performance. In addition, the setup will be utilized for extensions at the technological level. One such extension is the development of a topology management module, which will demonstrate the applicability of the PSIRP information concepts for optimizing resource utilization on the optical level. For this, we will utilize the existing optical infrastructure in the testbed in partnership with Essex University. Furthermore, the integrative demonstrator will be used as the basis for a UK-funded project between Essex University and Cambridge University in the area of lifestyle management [25]. This project targets novel services in the user-centric health area through self-monitoring and information processing. This is an area where we expect novel input from an underlying information-centric architecture like PSIRP.

While the localized experimental facilities allow for testing components and parts of the architecture under a variety of access and network technologies as well as in possibly diverse application settings, the issue of required scale still remains for crucial, in particular inter-domain, functions. For this reason, PSIRP is collaborating with the European Onelab2 efforts to establish a direct experimental platform support for architectural propositions as pursued by PSIRP. The immediate result of this collaboration is the connection of the localized testbed to the Planetlab Europe facilities, enabling the ability of experimentally testing technologies over the current Internet. In addition, direct connections to the experimental FEDERICA platform are currently explored to enable large-scale testing of PSIRP technologies that directly operate on Ethernet level, such as the forwarding solution [15].

5 Lessons Learned: The Attempt of a Coherent Approach

Section 4 highlighted the approaches being taken [26] for evaluating the variety of technologies and solutions being developed in PSIRP, according to the presented concept architecture. It is not surprising that the mix of simulations, emulations and testbeds of various kinds have been explored, given the variety of challenges to be addressed by the evaluation. But when looking closer at the variety of evaluation techniques, as presented in Section 4, one can recognize certain patterns that help formulating principles for a coherent approach that enables evaluating large-scale architectures like the one envisioned in Section 2.1. These principles for a coherent approach, directly addressing our challenges laid out in Section 2.2, are as follows:

- *Enable scale*: The scale challenge is most prominent in an evaluation of this kind and any approach must achieve this. While this is seemingly obvious, given our challenges, the approach must achieve this scale while preserving aspects like locality and component integration. We implement this principle by combining simulative and emulative elements.
- *Enable component-level testing*: Any solution development goes hand in hand with component development of the envisioned architecture. A comprehensive evaluation approach must enable component-level testing while being integrated into a scalable testing environment. This is achieved in our approach through the usage of emulation methods.
- *Enable locality*: Large-scale architectures do not live off inter-domain components and technologies only. Intra-domain solutions, such as for high-speed forwarding are part of this larger picture and need to be evaluated. Such testing requires often localized availability of components for the ability to quickly reconfigure and manipulate the test environment, while still being integrated into a large-scale framework of evaluation. Furthermore, locality enables a certain set of user experiments that are often not possible in global environments, such as sensing or local content scenarios. Hence, local test networks are crucial in a coherent approach.
- *Enable inter-domain operation*: The workings of protocols and technologies across multiple technological and administrative domains, lead to an often very different set of problems than a mere intra-domain operation. Hence, enabling such inter-domain operation, directly addressing the variety challenges of Section 2.2, is crucial in a coherent approach.

Taking into account these principles and our lessons learned from the evaluation, as presented in Section 4, we can formulate our attempt for a unified approach in Figure 4. This approach combines all evaluation environments: simulation (ns-3), emulation (in isolated environments) as well as testbeds like global solutions (PlanetLab) and a small general purpose testbed (at Essex University) to provide the best of all worlds while enabling a larger evaluation environment to be built than using just one environment alone. As a consequence, the localized testbed would enable the creation of large, high bandwidth tier-one equivalent ASes, with a direct connection to the raw Ethernet network provided by FEDERICA through the UK NREN.

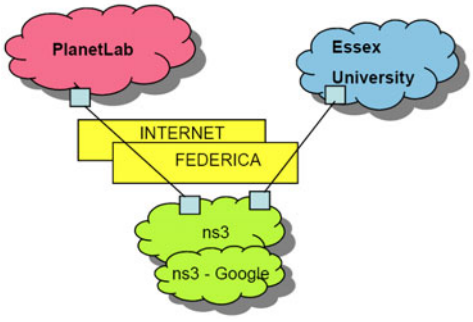


Fig. 4. Proposed environment for testing the PSIRP prototype

The ns-3 simulator(s), connected either directly to FEDERICA or to the Internet would allow the creation of a large number of smaller ASes, which could then be used to generate traffic to be injected into the PlanetLab and Essex University testbed. The PlanetLab environment, connected over the Internet, could be used to evaluate the performance of technologies over the current internet and its associated protocols, and would also be a prime candidate for creating a number of smaller ASes (depending on the available node bandwidth) with real world traffic properties (e.g., delays, link failures. etc). Combinations of tests, such as real-world experiments conducted at the local network, e.g., through applications that students would use, and background Internet-type traffic from PlanetLab, can be conducted with this approach.

The integration of an all-Ethernet testbed through FEDERICA and the localized testbed would provide the ability for evaluating native inter-domain technologies which would not run over IP. It also gives the experimenter the flexibility to choose the network environment with the most appropriate underlying physical topology to partially (or fully) match the overlay topology. The integration of the Planetlab environment, however, allows for amending the experiments with overlaid solutions, e.g., global rendezvous alternatives, in which an overlay execution suffices.

The above solution would also provide a large amount of flexibility for creating network topologies, as all three environments provide tools for this purpose. Ns-3 by design allows any network topology. PlanetLab has now been federated with VINI to enable layer 2 overlay creation and work is underway for a pure PlanetLab topology manager [27]. FEDERICA provides researchers the ability to specify a topology map containing V-nodes, virtual IP routers and virtual links. Early consultations with FEDERICA have determined that, for instance, forwarding techniques of PSIRP could easily be experimented with within FEDERICA without (logical) topology constraint. This eventually targets the last remaining challenge of Section 2.2, namely the economic variety. Testing inter-domain technologies, like global rendezvous or topology formation (see Section 2.1) over a platform like Planetlab forces the current business relations of ASes, represented through their underlying peering relations, on the tests. Approaches like VINI federation or FEDERICA integration would allow for creating inter-domain topologies that are significantly different from today's peering relations. The structure of such peering relations, however, is left for socio-economic considerations rather than experimental verification. But it is the proposed combined approach in Figure 4 that would facilitate the latter.

6 Conclusion

Experimental verification of new architectural approaches for the Future Internet is a difficult and challenging undertaking. Scale and variety on levels of technology, usage and economics places a burden on any evaluation task within an architectural effort. In this paper, we outlined the challenges that were faced by an exemplary architecture effort in the area of information-centric networking. While our approaches to implementation and experimentation are based on the particular efforts, many of the lessons learned easily apply to other, similarly ambitious, efforts currently being undertaken. Our combined approach of simulation, emulation and localized as well as global testbeds allows for addressing the various challenges. However, it is well recognized that more work is required for a coherent testing and experimentation approach for Future Internet solutions.

Acknowledgments

This paper outlines the combined efforts of the PSIRP project in various areas of design, implementation and evaluation. We therefore acknowledge the contributions of a large number of individuals in these areas. This paper would have not been possible ithout their work.

References

1. EU FP7 Publish-Subscribe Internetworking Routing Paradigm (PSIRP), http://psirp.org
2. Gritter, M., Cheriton, D.R.: An architecture for content routing support in the internet. In: Usenix Symposium on Internet Technologies and Systems, USITS (2001)
3. Balakrishnan, H., Lakshminarayanan, K., Ratnasamy, S., Shenker, S., Stoica, I., Walfish, M.: A Layered Naming Architecture for the Internet. In: SIGCOMM (2004)
4. Koponen, T., Chawla, M., Chun, B., Ermolinskiy, A., Kim, K., Shenker, S., Stoica, I.: A Data-Oriented (and Beyond) Network Architecture. ACM SIGCOMM Computer Communication Review 37(4), 181–192 (2007)
5. Esteve, C., Verdi, F.L., Magalhães, M.F.: Towards a new generation of information-oriented internetworking architectures. In: ReArch 2008 workshop at CoNEXT (2008)
6. Jacobson, V., Smetters, D.K., Briggs, N., Plass, M., Stewart, P., Thornton, J.D., Braynard, R.: Networking Named Content. In: ACM CoNEXT 2009 (2009)
7. Eugster, P.T., Felber, P.A., Guerraoui, R., Kermarrec, A.-M.: The many faces of publish/subscribe. ACM Comput. Surv. 35(2), 114–131 (2003)
8. Demmer, M., Fall, K., Koponen, T., Shenker, S.: Towards a modern communications API. In: 6th ACM SIGCOMM Workshop on Hot Topics in Networks, HotNetsVI (2007)
9. Ain (ed.): Architecture Definition, Component Descriptions and Requirements. PSIRP deliverable D2.3 (February 2009)
10. Särelä, M., Rinta-aho, T., Tarkoma, S.: RTFM: Publish/Subscribe Internetworking Architecture. In: ICT-MobileSummit 2008 Conference Proceedings (2008)
11. Tarkoma, S., Trossen, D., Särelä, M.: Black Boxes: Making Ends Meet in Data Driven Networking. In: MobiArch 2008, Seattle, Washington, USA, August 22, pp. 67–72 (2008)

12. Jokela, P. (ed.): Progress Report and Evaluation of Implemented Upper and Lower Layer Function, PSIRP deliverable D3.3 (June 2009)
13. Lagutin, D.: Redesigning Internet - The Packet Level Authentication architecture. Licentiate's thesis, Helsinki University of Technology (June 2008)
14. Forsten, J., Järvinen, K., Skyttä, J.: Packet Level Authentication: Hardware subtask final report. Technical report, Helsinki University of Technology (2008), http://www.tcs.hut.fi/Software/PLA/new/doc/PLA_HW_final_report.pdf
15. Jokela, P., Zahemszky, A., Esteve, C., Arianfar, S., Nikander, P.: LIPSIN: Line speed Publish/Subscribe Inter-Networking. In: Proc. of ACM SIGCOMM (2009)
16. Lockwood, J.W., et al.: NetFPGA – an open platform for gigabit-rate network switching and routing. In: MSE 2007 (2007)
17. Keinänen, J., Jokela, P., Slavov, K.: Implementing zFilter based forwarding node on a NetFPGA. In: Proc. of NetFPGA Developers Workshop (August 2009)
18. Henderson, T.M., Lacage, M., Riley, G.F.: Network Simulations with the ns-3 Simulator. In: SIGCOMM demonstration (2008)
19. Rowstron, A., Druschel, P.: Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems. In: Guerraoui, R. (ed.) Middleware 2001. LNCS, vol. 2218, p. 329. Springer, Heidelberg (2001)
20. Castro, M., Druschel, P., Kermarrec, A.-M., Rowstron, A.: SCRIBE: A large-scale and decentralized application-level multicast infrastructure. IEEE Journal on Selected Areas in Communications 20(8), 100–110 (2002)
21. Baumgart, I., Heep, B., Krause, S.: OverSim: A Flexible Overlay Network Simulation Framework. In: Proceedings of 10th IEEE Global Internet Symposium (May 2007)
22. Xylomenos, G., Katsaros, K., Kemerlis, V.P.: Peer assisted content distribution over router assisted overlay multicast. In: 1st Euro-NF Workshop on Future Internet Architecture, Paris, France (November 2008)
23. Katsaros, K., Kemerlis, V., Stais, C., Xylomenos, G.: A BitTorrent module for the OMNeT++ simulator. In: MASCOTS (2009)
24. Trossen, D. (ed.): Integration and Demonstration Plan, PSIRP deliverable D3.4 (June 2009)
25. PAL Personal Social Communication Services for Health and Lifestyle Monitoring (August 2009), http://pal.dalore.net
26. Riihijarvi, J. (ed.): Description of validation and simulation tools in PSIRP context, PSIRP deliverable D4.4 (July 2009)
27. Lischka, J. (ed.): Routing-in-a-slice platform extension requirement, OneLab2 Deliverable D7.3 (October 2008)
28. Rajahalme, J., Särelä, M., Nikander, P., Tarkoma, S.: Incentive-Compatible Caching and Peering in Data-oriented Networks. In: Re-Arch 2008 (December 2008)

QoE Testbed Infrastructure and Services: Enriching the End User's Experience

Frances Cleary Grant[1], Eileen Dillon[1], Gemma Power[1], Thomas Kaschwig[2], Ahmet Cihat Toker[2], and Christian Hämmerle[3]

[1] Waterford Institute of Technology, TSSG, Cork Rd, Ireland
{fcleary,edillon,gpower}@tssg.org
[2] Technische Universität Berlin, DAI-Labor, Berlin, Germany
{thomas.kaschwig,ahmet-cihat.toker}@dai-labor.de
[3] Fachhochschule Vorarlberg, Hochschulstr. 1, 6850 Dornbirn, Austria
christian.haemmerle@fhv.at

Abstract. Quality of Experience (QoE) is the subjective judgment of the satisfaction an end user perceives from an application running over a given network topology and configuration. The information provided by end users regarding their QoE preferences, experience and feedback is invaluable in providing a service that meets with their mobile activity needs within various access networks. The PERIMETER project progresses the QoE thematic research area by taking end user-related QoE factors for end user-centric mobility experimentation, thus empowering them to always have a service in which their QoE is high. This paper will detail the components of the PERIMETER framework and the user centric scenario based process adopted to implement and develop such a framework. This paper provides an insight into the federated testbed infrastructure, testing methodology and tools, operating system and applications used in the project, thus demonstrating PERIMETER's innovative advances within the QoE end user domain.

Keywords: Quality of Experience, testbeds, federation, PERIMETER, Future Internet, Always Best Connected.

1 Introduction

Quality of Experience (QoE) is a measure of the end-to-end performance at the service level from the end user perspective and an indication of how well the system meets the end user's needs [1]**.** Consideration of QoE parameters and preferences allows a more *user-centric*, rather than network-centric, approach to be adhered to in areas of seamless mobility. Enabling the end user to control the way their identity, preferences and credentials are used empowers them to be Always Best Connected (ABC) [2] in multiple access and multiple operator networks of the Future Internet.

The PERIMETER project [3] progresses the QoE thematic research area by taking user-related and non-technical QoE factors into account in order to provide a solid baseline for future user-centric mobility experimentation. This is achieved by studying the parameters that define the "user-centric seamless mobility", resulting in new

T. Magedanz et al. (Eds.): TridentCom 2010, LNICST 46, pp. 486–495, 2011.

network selection algorithms for achieving QoE based ABC paradigms and the development of a QoE framework that supports generic QoE definition, QoE signalling and QoE based content adaptation. To demonstrate such a QoE specific PERIMETER framework, a scenario based approach was adopted and a suitable federated testbed infrastructure was created to provide a valuable environment to verify the innovative QoE aspects of the PERIMETER project.

This paper gives an account of the PERIMETER middleware, before elaborating on the scenario based process used to derive the functional requirements of the system. Next, the underlying testbed infrastructure used to demonstrate and validate the middleware and its required applications is provided, in conjunction with details of the operating system used and the testing methodology employed. The results from the first round of testing are then presented. The paper concludes with a summary of the work of the PERIMETER consortium to date, before detailing its future trials in the QoE testbed infrastructure and services area.

2 PERIMETER Middleware

The PERIMETER middleware is composed of a QoE management system, QoE delivery system and PPA^{3}R (Privacy Preserving Authentication, Authorisation, Accounting and Reputation) system. These systems are supported by a Storage Layer for storing and retrieving information using a distributed peer-to-peer approach [4] and an Application Layer (which contains an Application Manager and a Graphical User Interface (GUI) that provides the end user with control over their QoE parameters, preferences and settings). The PERIMETER middleware, depicted in Figure 1, is hosted on PERIMETER aware mobile device terminals and on support nodes in the testbeds involved. Both terminals and support nodes can be directly connected to the Internet or to Virtual Private Networks (VPN) behind a Network Address Translation (NAT)/Firewall.

The central goal of PERIMETER is to devise a framework where end users are always in an ABC state. To achieve this goal, PERIMETER must gather relevant information, and make decisions on whether to generate a network switch based on the analysis of this information. The ABC state is measured using QoE metrics. PERIMETER makes its decisions using information from the end user's preferences, the end user's context (application under use, location and conclusions inferred from this), network performance parameters, other PERIMETER end users' QoE information and PERIMETER end users' feedback, which are collected in a QoE Descriptor (QoED) [5]. The Data Network Processor is responsible for computing these QoEDs. Dedicated Trust and PPA^{3}R components are employed to handle trust and security issues related to the sharing of QoEDs between the end users. The Decision Maker (DM) uses local and a selected subset of remote QoEDs to decide on the most suitable available network. The QoE delivery system performs the actual vertical handover based on the DM's analysis [6]. The QoE delivery system also conducts network measurements to aid the DM in making this decision.

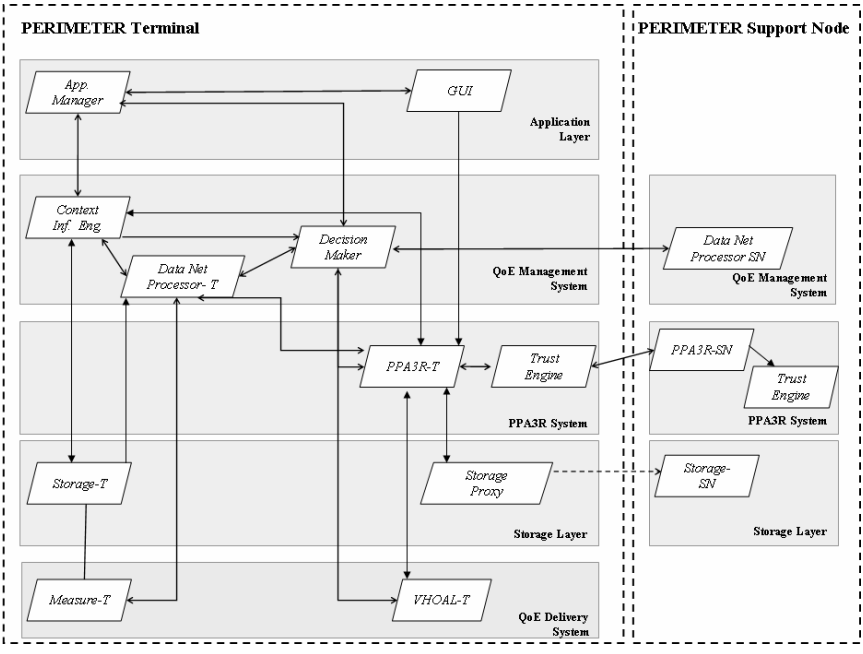


Fig. 1. PERIMETER Support Node and Terminal

3 User Centric Scenario Process

Adopting an end user scenario based process allowed the definition of a suitable federated testbed infrastructure that will be capable of supporting an end user QoE PERIMETER demonstrator. Through a scenario sub-step breakdown and component mapping activities, the PERIMETER consortium detailed the relevant component functionalities, interfaces and network technologies, therefore kick starting the process of identifying the testbed initial requirements. PERIMETER Scenario 1, entitled 'user-centric agnostic ubiquitous communication', follows the daily activity of an end user as they seamlessly roam between different technologies, connecting to services using various access technologies and devices.

A summary of the PERIMETER Scenario 1 is as follows:

Yvette is waiting for a taxi in the same room as her colleague Bob. Both have the PERIMETER system installed on their devices and are connected to the building's Ethernet network. A ***phone conference*** *starts. Bob participates with his laptop and Yvette with her handheld device. The taxi arrives and Yvette gets in. As the taxi leaves, Yvette's handheld begins to face network problems so the PERIMETER system begins searching for a suitable network to handover to Yvette's handheld has already detected a pair of Universal Mobile Telecommunications System (UMTS) networks; the PERIMETER system chooses the cheaper network. It has, in fact, analyzed a few statistics, previously cached from the PERIMETER overlay infrastructure, reporting that the majority of the end users had satisfactory experience with the*

cheaper UMTS network in that location. While in the taxi, Yvette receives an important ***video call*** *from her boss. Given its nature, the call requires high quality video parameters and a stronger confidentiality. PERIMETER scans for a network that will meet these requirements and analyses the QoE descriptors of the end users on various networks to find a suitable network. It chooses a different UMTS network and seamlessly connects to this to guarantee exceptional video call performance.*

The following work flow shows the step by step flow of actions taking into account the testbed infrastructure and the PERIMETER middleware:

- Scan for connection options and discover the nearby WLAN device. Join the phone conference.
- Monitor the active network interfaces for acceptable QoE.
- Detect changes in the location and degrading channel quality. Scan for other connection options.
- Collect QoS (Quality Of Service) data and interact with the PERIMETER Support
- Node to update QoS information of thecurrent geographic location. Collect QoE data over the PERIMETER overlay.
- Process collected data using the Decision Maker component of the system. Switch from WLAN to UMTS based on Decision Maker's decision.
- Receive the incoming call and detect QoE requirements (high quality and confidentiality).
- Probe the QoE descriptors of other users on this network.
- Select the more expensive and reliable UMTS network and switch between them.

Figure 2 conveys a high level overview of the mapping of this scenario in the PERIMETER federated testbed.

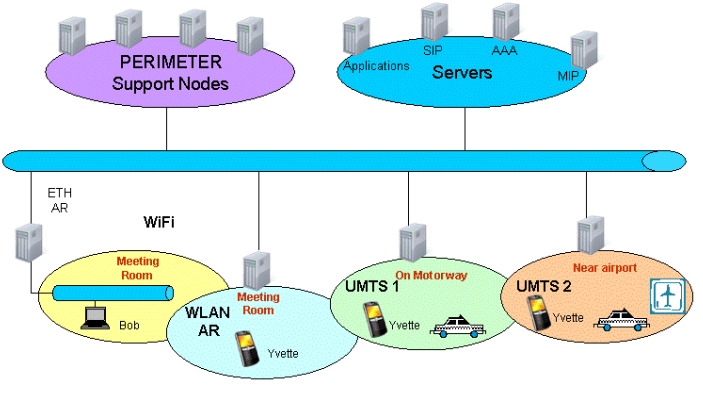


Fig. 2. Mapping Scenario to Testbed Requirements

4 PERIMETER Testbed Infrastructure and Testing

To effectively validate the results of the PERIMETER project, two main testbeds were setup to co-exist within the PERIMETER project, the first is housed at Waterford Institute of Technology (WIT) Ireland and the second main testbed is housed at Technische Universität Berlin (TUB) in Germany. The mapping of the scenario helped identify the main testbed requirements such as terminal devices (mobile and fixed), network support technology hardware and equipment and software requirements. WIT's testbed focused on the application services, while TUB's testbed focused on the network access infrastructure required to demonstrate the scenario, as illustrated in Figure 3 and Figure 4. The PERIMETER middleware requires both terminal and support nodes to be deployed in the QoE testbed infrastructure. There are four category actors in the PERIMETER system that will be part of the testbed design, User terminals, Support nodes, Network service providers and Application service providers.

WIT testbed hardware elements	TUB testbed hardware elements
• Hudson build server. • Asterisk SIP server • G1&G2 Mobile devices. • Gateway machine. • VMware server (perimeter support nodes) • Advent netbook x 2 • Wireless router, accesspoints	• Xen virtualization servers • (perimeter support nodes). • End terminal devices (laptops, netbooks, mobile devices G1/G2). • Semantic IPTV server and webcam. • gateway machine, routers, accesspoints • UMTS Femto Cells.

Fig. 3. PERIMETER Testbed Hardware

Between the two official test sites layer 3 Internet Protocol security (IPsec) interconnectivity was adopted to allow interconnection in a secure manner. The PERIMETER testbed used IPsec tools [7] such as M0n0wall [8], a complete embedded firewall package, Racoon [7] for Internet key exchange and Setkey [7] to manipulate security associations and policies within the implemented IPsec tunnel on the hosts from both testbeds.

The testbed interconnection has been designed with the following functionalities in mind:

- Service environment component integration.
- Testbed adaptation.
- Exposure, composition and redeployment of services and components.
- Horizontal Interconnection: to achieve greater scale.
- Vertical Interconnection: to support system-level testing of new Internet networking and services paradigms across layers.

The purpose of the PERIMETER testbed federation is to deliver the scenario testing, architecture validation testing, end user evaluation testing and final project demonstration for PERIMETER by interconnecting the diverse wireless network access systems and terminals and application environments which can provide a multi-faceted mobile communications environment.

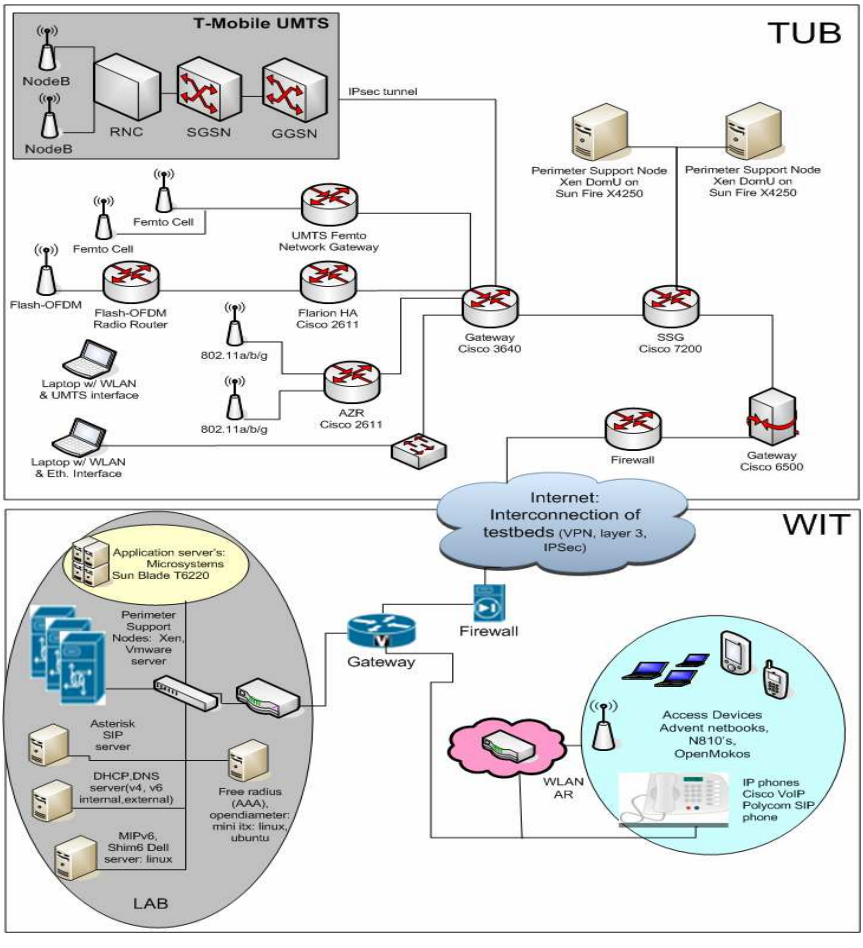


Fig. 4. PERIMETER Testbed Infrastructure

4.1 Operating System and Applications Used in the QoE Testbed

Adopting a user-centric scenario based approach incurs the need to address and have available certain applications in order to support the final end user demonstrator. In the PERIMETER scenario the two main applications required included a phone conference and a video conference application. It was agreed upon in the project to incorporate the use of the Google Android [9] operating system. This is a mobile operating system that runs on top of a Linux kernel [10].

The use of the Google Android operating system has allowed the PERIMETER consortium to invest in Android compatible mobile devices in addition to allowing the project avail of applications distributed by the Android Market. The following Google applications were initially assessed on the G1 mobile device containing firmware version 1.5, to determine their compatibility with the PERIMETER project:

1. For the phone conferencing application PERIMETER are examining **Sipdroid** [11]**.** Sipdroid is an open source SIP client implemented in Java which is capable of running on the Google Android platform.
2. For the video call application PERIMETER are investigating **Semantic IPTV** [12] provided by TUB. Semantic IPTV permits video streaming on the Android phone.

All applications used within the project must be made PERIMETER aware. This involved interfacing the PERIMETER specific Application Manager (AM). The AM's main functionality is to provide the end user with the ability to control the running applications, edit their preferences and set the network selection manually if needed.

4.2 GUI Testing Results

To validate the usability of the AM's GUI, usability tests were performed with a group of actual users from the PERIMETER consortium [13], taking into account the Living lab [13][24] methodology concepts. This step yielded important clues about the end users' acceptance of the system and their ability to grasp and utilize the functionality represented by the GUI.

As a first step for the GUI testing, a usability pre-test was initiated to evaluate statistical data of end users with different demographical (Ireland, Germany, Austria and Turkey) and technical background, with a set of interview questions. After the pre-test, the end users were provided with the Android G1 mobile phones with the PERIMETER GUI installed, shown here in Figure 5 and Figure 6. If the end user was not familiar with using an Android device, a tutorial was provided, which explained the basic tasks the end user needed to know for the tests. The end users were allowed to familiarize themselves with the phone and the operating system.

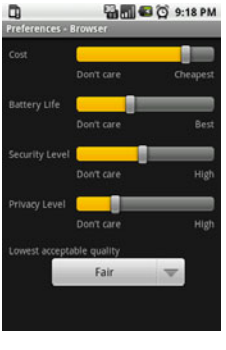


Fig. 5. Browser Preferences

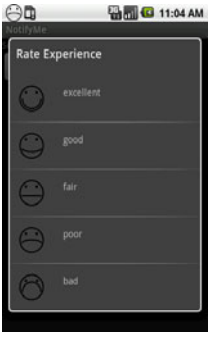


Fig. 6. QoE Rating

The main feedback points were in the following areas:

- The application launcher needed to be redesigned. Functionality such as the play/pause/stop metaphor for the applications was not deemed intuitive.
- Recommendation that the privacy and security preferences, shown in Figure 5, should be merged
- The dynamic feedback, involving the 'smileys' (Figure 6) was well received.
- The meaning of the cost preference (Figure 5) was not transparent.

4.3 PERIMETER Testing and Test Tools

In order to ensure that the PERIMETER system was brought to a level where it achieved its functional and end user objectives, a development and testing methodology was applied. The Agile software methodology [14][15][16] was deemed suitable for this project as it generally promotes a project management process that encourages frequent inspection and adaptation, a set of engineering best practices, such as Test Driven Development (TDD) [17] that allow for rapid delivery of high-quality software, and a business approach that aligns development with customer and end user needs. In conjunction with these Agile and TDD processes, the testing process also took into account test management 'best practises' in testing cycles [18], to include a testing cycle of five phases, as shown in Figure 7, These phases consisted of Definition (of the functionality to be tested), Commissioning (setting up the test environment), Execution (of the tests), Reporting (of the test results) and Evaluating (of the results achieved). These steps were mapped to the PERIMETER testing procedures for unit, functional, integration and scenario testing of the code base to ensure the testing cycle was robust, scalable, interoperable and secure.

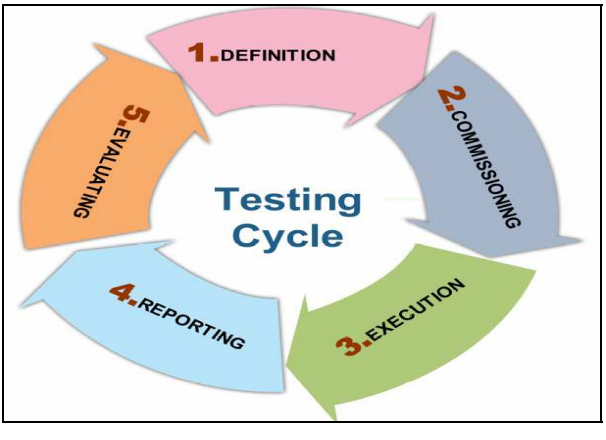


Fig. 7. PERIMETER Testing Cycle

Testing Tools are essential for team collaboration, continuous testing and structured software development. In PERIMETER, a number of supporting tools, such as Hudson [19], Trac [20] and Subversion [21] (SVN), were used to facilitate and structure the integration, validation and verification processes.

Subversion [21] was used for the code and document repository version control. ***Trac*** [20] was used for project management tasks and especially for bug tracking. ***Hudson*** [19] a continuous integration engine was used because of its flexibility and seamless integration with Subversion. In conjunction with ***Apache Ant*** [22], Hudson was configured to constantly execute builds as new or modified source code was checked in to the PERIMETER SVN.

5 Conclusions and Future Work

This paper introduced PERIMETER's approach for user-centric seamless mobility in Future Internet. The PERIMETER middleware, which manages the QoE and PPA^{3}R system and pictured the PERIMETER architecture, consisting of terminals and support nodes was described. A typical scenario for ubiquitous networking is shown that utilizes PERIMETER's overlay infrastructure with QoE based handover decisions to achieve an ABC networking environment. Within the PERIMETER project, a federated testbed was built with main sites at TUB and WIT, interconnected over the GÉANT2 research network. The testbed offers large-scale experimental facilities, sharing specialized networking infrastructure and services and enables integration and validation of the PERIMETER system in a heterogeneous and realistic environment.

On the terminal side, PERIMETER is based on the Android operating system. An Application Manager and GUI is being implemented, which allows the user to start PERIMETER aware applications, set preferences, returns QoE feedback to the user and enables the user to evaluate his QoE. The usability of the projected terminal software was evaluated in a field test, with primary focus on the GUI and the overall acceptance of the PERIMETER approach.

The PERIMETER consortium will mature the testbed activities, by progressing towards collaboration with and applying for a 'slice' of the FEDERICA [23] network infrastructure. This would enable the WIT and TUB federated testbeds to have core connectivity (layer 2) and also provide additional virtual hardware resources that can be used by the PERIMETER consortium for PERIMETER specific experimental research. Currently the PERIMETER system is being further developed in order to provide a complete system which can be used by an end user. This is being done in a user co- creation process, for which further usability and Living Labs [24] testing needs to be performed.

Acknowledgement

This research activity is funded under the EU ICT FP7 project, PERIMETER (Project No.: 224024).

References

[1] DSL Forum Technical Report TR-126, Triple-play Services Quality of Experience (QoE) Requirements, Produced by Architecture & Transport Working Group (December 2006)

[2] Isaksson, L.: Seamless Communications: Seamless Handover Between Wireless and Cellular Networks with Focus on Always Best Connected. Ph.D. Thesis 2007:06. Blekinge Institute of Technology (March 2007) ISBN: 978-91-7295-079-9

[3] PERIMETER project website, http://www.ict-PERIMETER.eu/
[4] Rieder, M., Schumacher, J., Hämmerle, C.: Improving QoE on mobile Internet by applying a P2P network for QoE aggregation, SMART EVENT (September 2009)
[5] Dillon, E., Power, G., Ramos, M.O., Callejo Rodríguez, M.A., Argente, J.R., Fiedler, M., Tonesi, D.S.: PERIMETER: A Quality of Experience Framework. In: Zseby, T., Savola, R., Pistore, M. (eds.) FIS 2009. LNCS, vol. 6152. Springer, Heidelberg (2010)
[6] PERIMETER - User-Centric paradigm for Seamless Mobility in Future Internet - Architecture Specifications Technical Report, http://www.ict-perimeter.eu/
[7] IPSec Tools, http://ipsec-tools.sourceforge.net/
[8] M0n0Wall, http://m0n0.ch/wall/
[9] Android Operating System, http://www.android.com/
[10] Open Handset Alliance, http://www.openhandsetalliance.com/
[11] SIPdroid, Native SIP/VoIP client for Android, http://sipdroid.org/
[12] Semantic IPTV project DAI-Labor, TU-Berlin, http://semantic-iptv.de/
[13] Dillon, E., Power, G., Hämmerle, C.: PERIMETER Phase 1 Usability Testing Technical Report, http://www.ict-perimeter.eu/
[14] The Agile Manifesto for Agile Software Development, http://agilemanifesto.org/
[15] Subramaniam, V., Hunt, A.: Practices of an Agile Developer (April 2006) ISBN: 9780974514086
[16] Racheva, Z., Daneva, M., Buglione, L.: Complementing Measurements and Real Options Concepts to Support Inter-iteration Decision-Making in Agile Projects. IEEE Computer Society Press, Los Alamitos (2008)
[17] Beck, K.: Test Driven Development: By Example. Addison-Wesley Longman, Amsterdam (2002)
[18] Ponce de Leon, M., Cleary, F., GarcÍa Moreno, M., Sobrino Jular, A., Romero Vicente, A., Roddy, M., Ryan, P., Jedrzejek, C.: Large scale interoperability, Integrating the Daidalos project, eChallenges 2006 (October 2006)
[19] Hudson Extensible continuous integration server, http://hudson-ci.org/
[20] [Trac Open Source Project, http://trac.edgewall.org/
[21] Subversion software project, http://subversion.tigris.org/
[22] Apache Ant Project, http://ant.apache.org/
[23] FEDERICA - Federated E-infrastructure Dedicated to European Researchers Innovating in Computing network Architectures, http://www.fp7-federica.eu/
[24] Eriksson, M., Niitamo, V., Kulki, S.: State-of-the-art in utilizing Living Labs approach to user-centric ICT innovation – a European approach. Centre for Distance-spanning Technology at Luleå University of Technology (2005)

From Learning to Researching
Ease the Shift through Testbeds

Guillaume Jourjon, Thierry Rakotoarivelo, and Max Ott

NICTA*
Australian Technology Park
Eveleigh, NSW, Australia
{guillaume.jourjon,thierry.rakotoarivelo,max.ott}@nicta.com.au

Abstract. This papers presents an e-learning platform that improves the current state of the art by successfully integrating four features.

Firstly, it provides a web interface incorporating lecture notes, labs instruction and results. This remote interface also allows the teacher to easily implement new experiments using a high level description language. Secondly, the proposed architecture will provide a low deployment cost without limiting the experimental scope. Thirdly, the new platform can take advantage of many existing and emerging testbeds. Finally, we introduce a new framework for teaching and learning network concepts. Thus a student using this new tool during an introductory course will embrace a less difficult path to perform more advanced studies on currently widely deployed testbed.

1 Introduction

Knowledge of networking concepts and technologies has become essential to most computer, software, and information technology engineers. Indeed, an ever increasing number of devices are now network-enabled, e.g. fridges emailing grocery lists, cars pulling information from highways, TV pre-fetching favourite shows. Thus many educational institutions have included networking subjects in the syllabus of most of their computer or electrical engineering courses.

When teaching introductory or advanced networking subjects, lecturers face the difficulty of illustrating both the concepts and technologies, and assessing students with various knowledge backgrounds. This illustration phase is usually implemented in the form of laboratory classes, where students use some software tools to experiment on various networking scenarios. While different solutions may be used for different course levels, it would be desirable that such tools allow the experimentations of both basic and advanced networking scenarios.

The use of simulation software (e.g. ns-3, OPNET [1,4]) is a possible approach to address both these illustration and assessment challenges. This solution offers the benefit of an easy installation, maintenance, and access to many different

* NICTA is funded by the Australian Government as represented by the Department of Broadband, Communications and the Digital Economy and the Australian Research Council through the ICT Centre of Excellence program.

T. Magedanz et al. (Eds.): TridentCom 2010, LNICST 46, pp. 496–505, 2011.

type of simulated resources, but may involve a steep learning curve, e.g. understanding the scripting interface to the simulator and unrealistic and simplified models.

Another possible approach is to use deployed experimental platforms (e.g. testbed) to both illustrate and assess the learning of networking knowledge. The main advantage of this approach is that it allows students to interact with real protocol implementations in realistic environments, thus potentially enhancing their learning experience. On the other hand, deploying and maintaining a testbed often involves significant financial costs and engineering overheads, which in turn limit the number and type of available experimental resources with which to experiment. A solution to these limits is to use existing open research testbeds such as ORBIT, PlanetLab, or Emulab [11,9,13]. However, similar to the simulation approach, the use of testbeds may also involve steep learning curves for students.

This paper presents a new alternate testbed-based approach, which combines the features of a previously introduced e-learning platform (Internet Remote Emulation Experiment Laboratory, IREEL [6]) with the experiment control and resource management services of a widely used testbed framework (cOntrol and Management Framework, OMF [10]). From a student perspective, this integrated tool addresses the learning curve issue associated with using testbeds. Indeed, it allows both beginner and advanced students to experiment with networking concepts through an intuitive web-based interface, or with their own protocols through detailed experiment descriptions and automated executions.

The remainder of this paper is organised as follows. Section 2 presents our e-learning platform vision that motivates the presented work. Section 3 reviews some of the related works.Section 4 presents the design of the proposed IREEL-OMF integration, and describes some of the key implementation decisions and their benefits. Section 5 discusses the contributions of the proposed IREEL-OMF integration in terms of new and enhanced learning methods and tools for networking students and future young researchers. Finally, Section 6 concludes this paper and presents some potential future works.

2 Motivation

We envision the future of network-course e-learning platform as depicted in Fig. 1. We believe this architecture provides four improvements compare to the state of the art.

Firstly, we think that lecture note and labs instruction must be accessible directly on the web interface that pilots the possible configurable experiments. This feature is necessary to allow student to work efficiently. Furthermore, the configuration of these experiments must be available in a high-level interface to allows student with no programming language skills to participate in these lectures and the teacher to add easily new experiments.

Secondly, the architecture should have a low-deployment cost and a large range of network configurations. This requirement leads us to consider emulation solution for the basic platform.

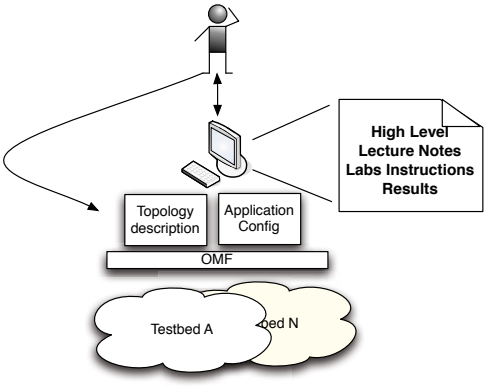


Fig. 1. General Architecture of the Future e-learning Environment

Thirdly, with the recent advancement in the worldwide deployment of network testbed, the future e-learning platform must facilitate the reproducibility of simple experiments over multiple testbed. In Fig. 1, this is illustrated by OMF's capability and current deployment over multiple testbed.

Finally, this new e-learning architecture should facilitate the shift from a beginner to advanced user of testbed as it is illustrated by the left arrow in Fig. 1.

3 Related Work

Few initiatives currently provide software services and tools to control experiment executions, access, and manage resources on networking platforms. Some examples of such software suites are the Emulab [13], the PlanetLab [9], and the OMF frameworks. These frameworks were primarily designed to address the experimental needs of the networking research community. Although, some of them could be used in a educational context, their usage requires a learning curve which is not suitable nor necessarily relevant to a semester-based academic course focusing on networking concepts and technologies.

Emulab [13] is a large network emulator. It provides experimenters with a set of computers, which can be configured into various topologies through emulated network links. The Emulab control framework supports three different experimental environments: simulated, emulated, and wide area networks. It unifies all three environments under a common user interface. Emulab provides tools to describe a required experiment topology and map it to actual resources. Some control tools are also provided, but with minimal features. Emulab and OMF share many design principles with the differences primarily shaped by a focus on different hardware and a different user community.

PlanetLab [9] is a global research platform based on more than 1000 distributed computers, which are hosted by independent organisations. It is the primary large-scale testbed used for experimental overlay and service oriented

systems (e.g. distributed storage, peer-to-peer contend distribution). PlanetLab provides a suite of software, which uses virtualisation tools to efficiently share the global resources among simultaneous short or long-lived experiments. Similar to Emulab, these tools are essentially focused on resource allocation and access, and only provide minimal supports in describing, controlling and measuring experiments. PlanetLab is also limited by its default offered layer-3 abstraction, i.e. it could not be used to illustrate layer-2 schemes (e.g. wireless MAC mechanisms) to networking students.

The Open Network Laboratory (ONL) [7] is a testbed-based educational resource, which has been used in the teaching of several academic networking courses. It consists of several computers interconnected by multiple extensible routers, which can be linked in various network topologies through a central virtual network switch. ONL allows the users to extend the routing functionalities through software plugins insertion. A user (e.g. a student) remotely access the platform through a stand-alone graphical interface, which easily allows the construction of various topologies, their configurations (e.g. route, bandwidth), and their monitoring. Similar to PlanetLab, the current ONL platform is limited to the illustration of layer-3 and above networking concepts.

Academic and industry research communities have developed many networking simulators, which provide inexpensive alternatives to testbed platforms as educational tools. Some of these simulators (e.g. OPNET [4]) have been successfully used in academic courses as the base of complementary laboratory activities [5]. Others (e.g. the ns-3 network simulator [1]) requires similar learning curves as the frameworks mentioned earlier, which may impede the learning of networking concepts. In any case, the inherent model assumptions within simulators limit the illustration of some networking aspects, e.g. there is no accurate simulation model for losses on wireless channels.

4 IREEL and OMF Integration: Design and Implementations

Facing the integration of an e-learning platform and a testbed management framework we have identified several challenges including the need to identify: a new communication scheme between the platform controller and the two computers that constitute the platform, a new measurement scheme based the a generic measurement library, and a new script generation for the experiment and the student.

4.1 New Architecture Overview

In the first version of the IREEL platform, communication between the platform controller and the two end-systems was done using remote calls with CORBA. This way of communication allows us to deploy the platform regardless of the operating system. On the counterpart, this scheme forced the teacher to follow strict rules for the deployment of new features.

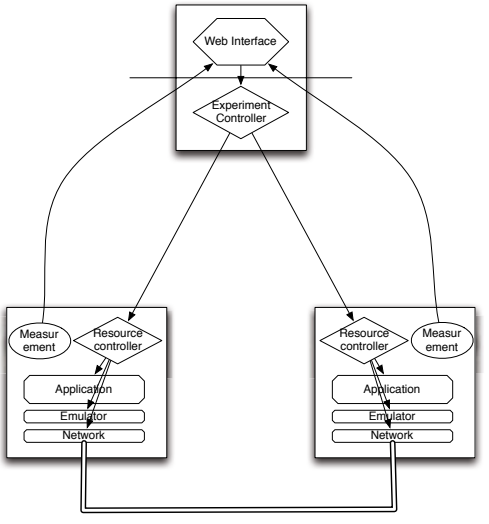


Fig. 2. New Architecture of the e-learning platform

In this new architecture, there is no longer a need for an intermediate computer thanks to the integration of a new emulation module inside OMF, as it will be detailed in the following of this section. Furthermore, as it is depicted in Fig. 2 we have added a new measurement channel using OML [12]. In the following of this section, we present the new communication channel using OMF, the measurement capability and finally the new emulation module in OMF.

4.2 New Communication and Measurement Scheme

In order to build the new architecture, we had to completely change the end-systems. This has been made easy thanks to the OMF application description. Then we had to decide how to start an application on these new end-systems. For that we changed all the application and the controller that was waiting for an XML file to decide which application to start. Therefore, we built ruby wrappers around all the legacy applications and remove the end-system controller to put instead the management framework.

The second challenge we had to resolve was the collection of measurement on the new end-system. Indeed, the management framework allows us to use a modular measurement library [12] that dynamically stores the results into a database on the platform controller. Previously, three kind of measurements were available on IREEL: network, application specific and application output such as a video over a network with impairments. For the moment we decided to use the measurement library to collect both network and application specific result and by-pass this library for the output of the video streaming. The answer to this challenge remains open for the moment.

4.3 Integration of an Emulation Module in OMF

Finally, in order to make this integration possible we had to develop a new module for OMF. As explained in Section 2, one of the main goal of this e-learning platform is to be easily deployable and therefore we need to maintain a low deployment cost. Thus, in order to provide a large scope of network configurations we decided to use an emulation solution. However, in the current state of OMF it was not yet possible to manage emulation tools but skeletons to implement such controls were available. Therefore we developed a new emulation module that allows us to firstly maintain a low deployment cost of the e-learning and secondly provide networking community an emulation enhancement to all testbeds that use OMF. As a side effect, this makes OMF even more suitable for wired networks and allows testing on more complex topology using existing testbed.

This emulation module has been make possible by the integration of a management procedure of Netem [8] on the end-systems. This integration has been possible through the enhancement of the topology description inside OMF and the interfacing of netem specific API. In OMF Experiment Description Language OEDL [3], it was possible to easily integrate this feature.

The addition of this emulation module inside OMF brings us two main benefits, an extension the topological abstraction and capacities of OMF and the student is given a theoretical graph representation of the configured network. Indeed, in addition to the configuration of the network, OMF's topology builder gives the experimenter a graph that represents the network. In this model, the network is represented as a directed graph $G(V, E)$, where V represents the set of vertices and E the set of edges.

This feature allows the student to verify if (s)he configured the network as (s)he was asked to do to finish his assignment. On the teacher side, this feature provides the opportunity to introduce the graph theory as a representation of the network. Furthermore, as we demonstrate in the following section, it also allows the teacher to point out the need of real experiments compare to pure theoretical approach to networking.

4.4 Integration Procedure: A More Flexible Approach

As explained in the previous section, we changed the way to initialise the application on the end-systems. This modification also facilitates the integration of new applications on both the end-system and the platform controller. In the first version of the platform developers have to follow a Java skeleton.

In the new architecture, the developer must simply install the new application on the end-systems and then write a ruby script that will be called by the management framework. This script is presented through an example in Listing 1. Starting the new application is therefore transparent for the developer. Furthermore, we plan to extend this application description to automatically generate the front end for the student. Thus, the teacher writes a script describing both the application to use and the different network impairments required for this experiment. This new feature can be illustrated in the second part of Listing 1.

```
defProperty('delay1', '10ms', 'between 1 and 2')
defProperty('bandwidth1', '1000kbit', 'between 1 and 2')
# Create an application representation from scratch
defApplication('ireel:app:newApp', 'newApp') {|app|
  app.version(1, 1, 2)
  app.shortDescription = "Illustration of the CS course"
  app.path = "/usr/bin/newApp"
  app.defProperty('numPackets', 'number of packets')
  app.defProperty('timeout', 'ping timeout in second')
  app.defProperty('size', 'user data size')
  app.defProperty('output', 'Output type binary or text')
}
#Specify the link characteristics from node 1 to 2
defLink('ireel:link:newLink', 'newLink12'){|link|
  link.shortDescription = "Configuration of the link characteristics "
  link.setProperty('bandwidth', prop.bandwidth1)
  link.setProperty('PLR', 0.0)
  link.setProperty('delay', prop.delay1)
}
```

Listing 1. Example of a new application

The teacher then submitted this experiment description using a wiki-like interface. It is afterward processed by the system in order to generate an IREEL experiment interface with the desired network impairments and application configuration.

5 From Learning to Researching: Benefits of a Unified Solution

5.1 Introducing a New Learning Plane

As illustrated on Figure 3, the integration of IREEL with OMF provides a new *learning plane*, which serves as an interface between students/lecturers and the bare resources of a testbed platform. This plane provides a seamless access from a user-friendly graphical front-end to the full visualisation/edition of an OMF Experiment Description. This caters both for undergraduate basic courses and

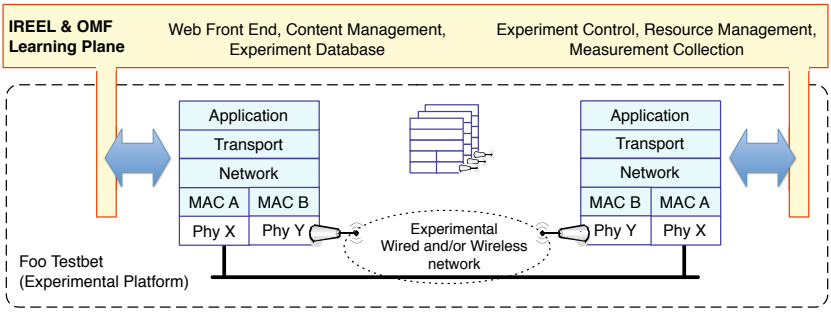


Fig. 3. New Learning Plane

postgraduate advanced ones. Indeed, students of the former courses use graphical menus to develop, run, and analyse experiments, thus focusing on the networking concepts without the difficulty of a steep usage learning curve, nor the need for any programming skills.

Through the use of OMF's systematic descriptions and automatic executions of experiments, this learning plane further introduces new students to scientific concepts, such as reproducibility, statistical significance, or peer-verification. For example, students can easily cross verify their results, or run multiple trials of the same experiment with/without parameter modifications, and observe the effects on their result statistics. This feature also benefits advanced and research students, as they have now an uncomplicated tool to produce more accurate scientific results.

Another major benefit of this learning plane is that through the use of OMF as the interface with the underlying testbed, it potentially gives students and lecturers access to large number of heterogeneous resources within many different testbed platforms. Indeed, OMF is currently used to control experiments and managed various resources at many institutions. For example, the particularities of an existing protocol (e.g. TCP) can be illustrated on real PC-based resources connected via links with Internet-like characteristics, and then seamlessly evaluated on real mobile wireless resources connected via ad-hoc 802.11 links. Comparison between results from both environments would give students valuable insights into the studied protocol. An example of such a case is developed in the next section.

5.2 Demonstration of the Portability

In this section, we demonstrate through a simple example the benefit of the combined use of the new platform and a wireless testbed. In this scenario the student is first asked to configure the simple IREEL's network and then starts a traffic generator. Once this experiment is finished the student can now take the experiment script provided by the platform and execute it with the same topology on the wireless testbed. The aim of this simple scenario is to put in relief the physical characteristics between wireless and wired network even if the theoretical topology remains the same.

Execution on IREEL. In order to execute this experiment on IREEL, the student must first log on the IREEL portal. Once the student on his or her personal page, they may configure a simple scenario using iperf in its UDP mode and a topology with a bottleneck of 1 $Mbit/s$ and an RTT of 200 ms.

The student must then waits for the experiment to finish and analyzes the results provided by the platform. These results comprise: the output of iperf; a graph of the throughput perceived by both the sender and receiver; the file generated by TCPdump [2]; the newly introduced measurement database; and can be summarised by the Table 1.

Execution on a Wireless Testbed. Once the results have been analysed, the student will be in possession of an OMF Experiment Description. Advanced

Table 1. UDP Results on a Wired Network

Average Throughput	1010 $kbit/s$
Average Jitter	2.21ms
Packet Loss Rate	59.56%

student can now use this script to better understand the importance of the access medium.

We performed this simple experiment on the NICTA wireless testbed and able to observe results displayed in Table 2. In these results, slight differences are evident even if the exact same theoretical network has been configured. Based on these differences, the teacher can open the discussion on the importance of the access network and start a deeper explanation on this technology and its limitations.

Table 2. UDP Results on a Wireless Network

Average Throughput	987 $kbit/s$
Jitter (min/avg/max)	5.43ms
Packet Loss Rate	60.51%

6 Conclusion and Future Work

Based on feedbacks from students and teachers, we have presented a new e-learning platform that improves the state of the art on several aspects. Firstly, the proposed architecture provides both students and teachers with a rich web interface incorporating lecture notes, labs instructions and results of the experiments. Furthermore, this interface allows an easier integration of new experiments for the teacher through the use of a high level description language.

Our solution also provides a low deployment and maintenance cost in its basic architecture and facilitates the use of already deployed testbed. Finally, we introduces a new teaching plane for network course that ease the shift from beginners to advanced students.

On a technical aspect, we plan to extend this platform to support directly wireless experiments. Thus the new platform will be one of the first e-learning platform integrating a wireless component.

On the teaching counterpart aspect of this platform, we plan to provide it to Universities and follow its use. We also plan to evaluate the final product based on students and teachers evaluation.

Acknowledgements

This work was achieved in the context of the NaDa and Onelab2 projects funded by the E.U. 7^{th} Framework Program, and the GENI (Global Environment for Network Innovations) initiative funded by the U.S. National Science Foundation.

References

1. NS3 Network Simulator, http://www.nsnam.org/
2. Tcpdump/libpcap public repository, http://tcpdump.org
3. The OMF Testbed Control, Measurement and Management Framework, at: http://omf.mytestbed.net
4. The OPNET Network Simulator, http://www.opnet.com/
5. Aboelela, E.: Network Simulation Experiments Manual, 2nd edn. Morgan Kaufmann, San Francisco (2008)
6. Dairaine, L., Jourjon, G., Lochin, E., Ardon, S.: IREEL: Remote Experimentation with Real Protocols and Applications over Emulated Network. Inroads, the SIGCSE Bulletin 39(2), 92–96 (2007)
7. DeHart, J., Kuhns, F., Parwatikar, J., Turner, J., Wiseman, C., Wong, K.: The Open Network Laboratory. In: Proceedings of SIGCSE 2006 (March 2006)
8. Hemminger, S.: Network emulation with netem. In: Linux Conf. Au. (April 2005)
9. PlanetLab Consortium. Planetlab: An open platform for developing, deploying, and accessing planetary-scale services, http://www.planet-lab.org/
10. Rakotoarivelo, T., Ott, M., Jourjon, G., Seskar, I.: OMF: A Control and Management Framework for Networking Testbeds. In: Proc. of ROADS 2009: Fourth Workshop on Real Overlays and Distributed Systems (October 2009)
11. Raychaudhuri, D., et al.: Overview of the ORBIT Radio Grid Testbed for Evaluation of Next-Generation Wireless Network Protocols. In: Proc. IEEE Wireless Communications and Networking Conference, WCNC (2005)
12. Singh, M., Ott, M., Seskar, I., Kamat, P.: ORBIT Measurements Framework and Library (OML): Motivations, Design, Implementation, and Features. In: Proceedings of IEEE Tridentcom 2005 (February 2005)
13. White, B., Lepreau, J., Stoller, L., Ricci, R., Guruprasad, S., Newbold, M., Hibler, M., Barb, C., Joglekar, A.: An integrated experimental environment for distributed systems and networks. In: Proc. of the Fifth Symposium on Operating Systems Design and Implementation, Boston, MA, pp. 255–270. USENIX Association (December 2002)

TridentCom 2010

Full Papers Session 5: Monitoring in Large Scale Testbeds

Distributed Ontology-Based Monitoring on the IBBT WiLab.t Infrastructure

Stijn Verstichel, Eli De Poorter, Tim De Pauw, Pieter Becue, Bruno Volckaert, Filip De Turck, Ingrid Moerman, and Piet Demeester

Dept. of Information Technology - IBBT, Ghent University, Ghent, Belgium
G. Crommenlaan 8/201, 9050 Ghent, Belgium
Tel.: +32 9 331 4981; Fax: +32 9 331 4899
<name>.<surname>@intec.UGent.be
http://www.ibcn.intec.UGent.be/

Abstract. Testbeds as a means to evaluate protocol and software development are gaining importance, not least because of the oftentimes unpredictable influence of environmental behaviour. IBBT, the Interdisciplinary Institute for Broadband Technology, recognizes the importance of such testbeds and has therefore invested in WiLab.t, a wireless sensor and mesh testbed. It contains over 200 wireless and programmable nodes. The monitoring and management of such a testbed is very important so as to guarantee a proper functioning and stable environment to be used by researchers. This is however not a trivial task, even more so when in the future, the testbed is expanded with new devices and as such becomes a heterogeneous environment. Therefore, we have developed an ontology-based monitoring approach, which allows hiding the heterogeneity from the monitoring application and enables to process the data in a formal manner. Additionally, it allows adaptation according to characteristics of the local deployment, without the need to re-engineer the entire monitoring application every time alterations are made to the testbed.

Keywords: wsn, wmn, ontology, semantics, monitoring, reasoning.

1 Introduction

Whereas network simulators can be used to evaluate the performance of applications in general and for wireless sensor networks in particular, these do not have the capabilities to simulate all effects of real-life deployments, leading to a considerable discrepancy between experiments and simulations. Therefore, both wireless and other testbeds are a valuable tool for evaluating the performance of such applications.

IBBT, the Interdisciplinary Institute for Broadband Technology has recognized the importance of such testbeds. IBBT is an independent research institute founded by the Flemish government to stimulate ICT innovation. The IBBT team offers companies and organizations active support in research and development. It brings together companies, authorities, and non-profit organizations to

T. Magedanz et al. (Eds.): TridentCom 2010, LNICST 46, pp. 509–525, 2011.

join forces on research projects. Both technical and non-technical issues are addressed within each of these projects. Because of its belief in testbeds to enhance research and development, it has therefore invested heavily in the installation of a number of testbeds, to support a variety of research topics.

To enable intelligent monitoring and management of this testbed, an ontology-based monitoring framework has been developed. This framework has been deployed on the WiLab.t wireless sensor and mesh network testbed, but because of the adoption of a generic ontology approach, the framework could be used on other testbeds as well. Apart from offering monitoring information, the ontology-based approach also allows classification and inference to be performed on the monitored information. As such, only pre-processed and important information is exposed to the administrative team, avoiding the need to constantly manually analyze the data being produced by the monitoring application.

Although the monitoring use-case presented in this paper is very important on itself, it has supported and demonstrated the research and development of an ontology-based collaboration and data-aggregation platform, which can also be used in a wider context. Originally, the development of such a distributed ontology-based collaboration platform in a constrained environment was the main research task, but the monitoring use-case has proven very important and worthwhile to further exploit.

The remainder of this paper is structured as follows. The next section introduces related work on similar research topics. Section 3 describes in more detail the nodes and topology of the WiLab.t testbed. In Section 4 the complete architecture is introduced, starting from the software components on the sensor devices, through the mesh and back-end modules, concluding with the monitoring application. The platform has been thoroughly evaluated and the results are presented in Section 7. Finally, in Section 8 we present our main conclusions and introduce aspects for future research.

2 Related Work

The first generation of experimental set-ups' main purpose was the evaluation of nature monitoring applications. These set-ups did not have any advanced benchmarking facilities or have any flexibility regarding the reconfiguration of test set-ups [1]. To increase the reuse of existing testbed set-ups, newly developed testbeds offer more advanced management functions, such as automatic code deployment and scheduling mechanisms. These testbeds are deployed in a wide range of scenarios, from city monitoring [2] to office monitoring [3,4]. The number of active nodes in a single testbed ranges from a few nodes to more than 150 nodes. As of recently, efforts are being made to merge the different testing facilities into a single, world-wide testing environment [5].

However, many innovations are still missing in regards to the flexibility of wireless sensor testbeds. *(i)* Even though energy efficiency is very important for wireless sensor networks, very few testbeds have fine-grained and detailed power management and measurement capabilities. Similarly, the ability to emulate different battery types is still missing in most current testbeds. *(ii)* The topology

of current deployments is fixed, resulting in inaccurate results when testing different sensor deployment scenarios. *(iii)* Accurate timestamp logging and time synchronization is frequently missing from the management platform. *(iv)* Sensor testbeds consist of at most a few hundred nodes, which is far less than the thousands of nodes which are foreseen in the vision of "the future internet of things" for office buildings. *(v)* Finally, the types of nodes deployed in a single testbed are very similar, whereas future sensor networks are foreseen to contain heterogeneous nodes with very diverse capabilities.

In this paper, we propose an ontology-based approach towards the monitoring of the WiLab.t testbed. A brief, but all-embracing definition of an ontology, can be found in [6]: "An ontology is a specification of a conceptualisation in the context of knowledge sharing." Accordingly, an ontology describes in a formal manner the concepts and relationships, existing in a particular system and using a machine-processable common vocabulary within a computerised system.

OWL, a modelling language for ontologies, consists of three sublanguages, each of them varying in their trade-off between expressiveness and inferential complexity. They are, in order of increasing expressiveness: *(i)* OWL Lite: supports classification hierarchies and simple constraint features, *(ii)* OWL DL: OWL Description Logics, a subset providing great expressiveness without losing computational completeness and decidability and *(iii)* OWL Full: supports maximum expressiveness and syntactic freedom, however without computational guarantees.

Using one of the three sublanguage flavours of OWL, one can easily adapt to the required expressiveness. Arguably the most interesting sublanguage for many application domains is OWL DL, balancing great expressiveness with inferential efficiency. The efficiency is guaranteed by the underlying Description Logics. Due to its foundation in Description Logics, OWL DL is also very flexible and computationally complete.

A number of initiatives were investigated previously to incorporate web semantic technology in wireless sensor environments. [7] presents a proposal that combines the benefits of autonomic and semantic sensor networks to build a semantic middleware for autonomic wireless sensor networks. Ontology-based data provisioning mechanisms for wireless sensor networks, in order to deal with varying applications, are presented in [8], while [9] defines a set of ontologies and accompanying architecture for knowledge sharing.

To conclude, even as there are currently a wide variety of testbeds available, many of these could be improved by providing more flexibility in regards to the configuration, power management, scale and topology of the testbeds. Also the adoption of ontologies and more specifically distributed reasoning mechanisms within the specific nature of wireless sensor and mesh networks to support reasoning in constrained environments is an additional feature presented in this research.

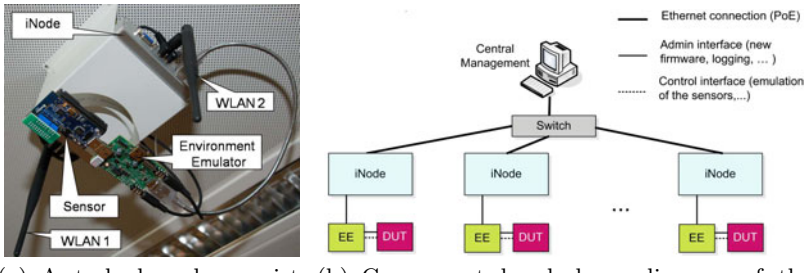
3 WiLab.t Infrastructure

The WiLab.t test infrastructure is located at the IBBT office building of Ghent University, Belgium. This testbed consists of 200 TMoteSky [10] sensor nodes,

spread over 3 floors. Wireless communication between the different floors is possible through 2 air shafts, in which several sensor nodes are installed.

Each sensor node is connected with an 'Environment Emulator' (EE) (see Figure 1(b)). This device can be used to control the physical properties of connected sensor nodes, *i.e.* sensor values can be emulated, the battery voltage can be regulated, general purpose pins can be connected with the sensor node and energy harvesting or battery models can be programmed. Finally, the EE also enables very accurate power measurements, in the order of microsecond intervals.

Finally, for ease of programming and debugging as well as to experiment with software in constrained environments, the wireless sensor nodes are connected with intermediate nodes, called iNodes, which are Alix 3C3 devices [11] running Linux Voyage [12]. Management of the testbed is performed using a modified version of the motelab [4] management software. This software is expanded with additional features such as a visualizer tool, a tool for graphical analysis of measured data and a customizable SQL database. A picture of a node in the WiLab.t testbed can be seen in Figure 1(a).



(a) A tesbed node consisting out of an Alix device, a TMoteSky sensor board and an Environment Emulator

(b) Component break-down diagram of the testbed nodes

Fig. 1. The nodes in the WiLab.t testbed infrastructure

4 Monitoring Architecture

This section introduces the architecture of the software components developed to monitor the behaviour of the WiLab.t infrastructure. The TinyOS NesC components to be run on the TMoteSky devices are described. Additionally, the more heavy-weight reasoning components, deployed on the iNodes, are presented. These components reason on the raw data being produced by the sensors. Finally, the overall workflow to trigger a monitoring task from the client, through the back-end and iNodes, finishing at the TMoteSky sensors is detailed.

Our general ontology-based monitoring approach has already been thoroughly evaluated in a back-end heavy-duty environment. This was published in [13]. Here, the ontology processing modules and mechanisms for query partitioning

and execution have been detailed. However, because of the constrained environments taken into account in this scenario, we have had to define a number of additional modules and enhance certain mechanisms, in order to facilitate the deployment of the platform in this constrained environment. The extended platform architecture is given in Figure 2.

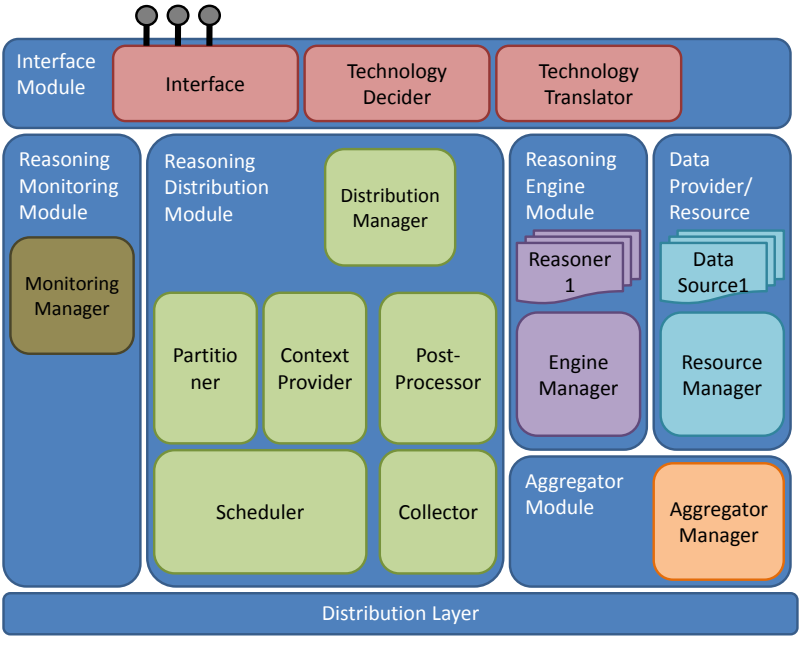


Fig. 2. Extended ontology-based agent collaboration platform

The main building blocks of the architecture are the *Reasoning Distribution Module* and *Reasoning Engine Module*. Additionally, to improve the transparency for the outside world, an extra indirection layer has been included at the interface level, namely the *Interface Module*. This layer is introduced to facilitate multiple reasoning technologies without the need for the clients to be aware of this. As such, only generic interface operations should be defined, avoiding the usage of reasoning technology dependent query and invocation mechanisms, *e.g.* SPARQL [14]. Additionally, to decouple the reasoning from the data storage, which was not the case in the original architecture presented in [13], two extra modules are introduced, namely the *Data Provider/Resource Module* and the *Aggregator Module*. The *Data Provider/Resource Module* will collect the data from the resources on which it has been deployed or for which it is responsible and feed it to the *Aggregator Module*. Upon request of the *Reasoning Engine Module*, the *Aggregator Module* will feed this collected data to the reasoning process. This way of working allows including sensor devices in the workflow and thus facilitates the monitoring of the sensor network, by means of sensor

information generated from the *Data Provider/Resource Modules* deployed on the sensor devices.

4.1 The Data Components

Each node that contains an *Aggregator Module* needs to collect the appropriate data. To this end, each sensor node regularly gathers *system statistics*, such as internal voltage, node ID, total queue occupation, *network statistics*, such as number of bytes and packets sent, number of bytes and packets received, number of packet drops, number of failed transmissions and number of faulty packets received and *measurement data*, *e.g.* external temperature, internal temperature and humidity.

The information is sent at-runtime and wirelessly over an IEEE 802.15.4 wireless interface to the sensor node connected to the Alix board responsible for the reasoning on the data from that sensor node. This particular node is called the aggregator. It forwards the incoming data immediately to the back-end database. Since we want to include information about the networking layer in the reasoning, we need to set up a wireless sensor cloud with sufficient network traffic. Therefore we do not use every local iNode with its WiFi connection to transmit its information, but use the senor nodes for this matter. However, due to the large scale of typical sensor networks, measured information cannot be sent directly to the aggregator node. Instead, sensor nodes use a multi-hop approach whereby measured information is sent over intermediate nodes to reach the *Aggregator Module*. Each *Aggregator Module* contains a software component responsible for notifying nearby sensor nodes. Therefore, a software component is installed that regularly broadcasts "sink" notification messages [15]. Each sensor node that receives a sink message checks if the hop count is lower than any previously received sink message. If it is, the notification is further forwarded by the node, and the address of the neighbour from which the sink message was received is used as the default next hop address when forwarding measured data. This way, each sensor node sends its information from neighbour to neighbour until the information reaches the nearest aggregator node. Thus, to collect the appropriate data, the following software components are installed on each sensor node:

SensorMeasurement. This component regularly gathers system, network and measures sensor data from the appropriate data components. All information is encapsulated in a packet and is sent to the *DataDistribution* component.

DataDistribution. This component is responsible for selecting the next hop neighbour to which packets are forwarded. As part of this component, the sink with lowest distance is selected, and unreliable routes are regularly purged.

4.2 The Reasoning Components

As indicated earlier in Section 3, every node in the WiLab.t infrastructure, consists out of two devices. One of the devices is a TMoteSky [10] sensor, the other is

an Alix [11] board. This allows for a hybrid approach concerning the deployment of software components. More specifically for components having different functionalities, requiring different specifications. The software components on the sensors producing the data in a non-intrusive manner, were detailed in Subsection 4.1. However, as detailed in Section 1, an intelligent monitoring framework, based on distributed formal first-order logic reasoning using ontologies, is pursued. This goal resulted in a number of reasoning components being developed and deployed in the WiLab.t testbed. The reasoning modules impose more stringent requirements on the hardware running these components. Using the iNodes, reasoning components can be deployed to process the data in a more intelligent way. These reasoning modules, using standard reasoning software, such as Pellet [17], analyze the data using the ontology model to draw conclusions about the status of the nodes in the testbed.

4.3 Ontology Used for WiLab.t Monitoring

Starting from the Sensor Node Ontology [16], which describes various states of a sensor node depending upon states of its constituent modules, additions and enhancements were modelled to take the specific situation of the WiLab.t into account. An important addition to the ontology is the location information. In this way, we can model the physical location of the nodes in the ontology. Additionally, a further component breakdown of the sensor nodes was modelled. To facilitate this, the concepts *SensorBoard* and *SensorPart* have been introduced. The general goal of the ontology is to classify the sensor nodes based on the values of the monitored metrics. Therefore, we used a typical observation pattern.

The two most important concepts in the ontology in terms of reasoning are *Fault* and *Solution*. A *Fault* subconcept is defined based on a logical statement

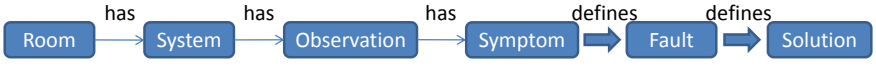


Fig. 3. Main property chain

Table 1. Additional properties and their characteristics in the WiLab.t Monitoring Ontology

Object Property Name	Characteristics	Domain	Range	Inverse Property
hasObservation		System	Observation	isObservationOf
hasSolution		Fault	Solution	isSolutionForFault
hasSensorPart		System	System	isSensorPartOf
hasSymptom		Observation	Symptom	isSymptomOf
hasNextObservation	Functional	Observation	Observation	hasPrevObservation
requiresAction		Solution	Action	isActionFor
hasFault		Symptom	Fault	isFaultOf
hasSystem	Inv. Functional	Room	System	isLocatedIn
hasRoom	Inv. Functional	Floor	Room	isOn

mainly combining *Symptoms*. In turn a *Solution* is defined mainly using a combination of *Faults*. Example definitions of two *Fault* concepts can be found below.

Detected TMoteSkyFault definition

```
[hasSensorPart some
  (hasObservation some
    (hasSymptom some HumidityZeroSymptom)) and
 hasSensorPart some
  (hasObservation some
    (hasSymptom some LightIntensityZeroSymptom)) and
 hasSensorPart some
  (hasObservation some
    (hasSymptom some TemperatureZeroSymptom))] or
[hasObservation some
  (hasSymptom some MissingReportsSymptom)]
```

This definition specifies that a TMoteSky sensor node which has a sensor part that outputs at the same time zero as value for temperature, humidity and light intensity or which does not produce anything is to be classified as faulty.

Incipient HVACFault definition

```
hasSystem some
  (hasSensorPart some
    (hasObservation some
          ((hasSymptom some
            (TemperatureBelow15Symptom or
TemperatureAbove25Symptom))
and
(hasSymptom some
  TemperatureNotZeroSymptom))))
```

In this definition, a room which has a system - *i.e.* a TMoteSky - which in its turn has a sensor part that outputs a non-zero temperature value below 15 °C or above 25 °C, probably has a faulty HVAC system.

4.4 The Coordinating Back-End Component

In the context of the WiLab.t infrastructure monitoring application, two typical queries are used. The first type queries for the inferred *Fault* individuals, while the second type triggers the reasoner to infer the possible *Solution* individuals for a given *nodeID*. The information contained in an ontology is modelled in a triple type format. This means that a subject is linked to an object by means of a predicate. This mechanism is used by the SPARQL query language to specify queries. By inserting unbound variables in the triple patterns, the reasoner will search the model and its data to find the individuals satisfying the triple. For more information we refer to [14].

***Incipient fault* query**

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX wsn: <http://users.atlantis.ugent.be/svrstich/deus/wsn#>
SELECT ?x0 WHERE {
    ?x0 rdf:type wsn:IncipientFault .
    ?x0 rdf:type wsn:System
}
```

This first query searches for all individuals which belong at the same time to the *IncipientFault* and *System* concept. As described in the previous section, the *IncipientFault* concept is modelled by means of a description logic statement. As such the reasoner will check at-runtime which of the *System* individuals, i.e. the sensors, satisfies this logic statement and will only return those sensors that do match this description.

***HVAC* query**

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX wsn: <http://users.atlantis.ugent.be/svrstich/deus/wsn#>
SELECT ?x0 WHERE {
    ?x0 rdf:type wsn:PossibleHeatingFault .
}
```

Dually to the previous query, this query triggers the reasoner to search through the entire set of data to find those objects which satisfy the logic description of a *PossibleHeatingFault*. Its definition is presented in Section 4.3.

***Solution* query**

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX wsn: <http://users.atlantis.ugent.be/svrstich/deus/wsn#>
SELECT ?x0 ?x1 WHERE {
(1) ?x0 rdf:type wsn:DetectedFault .
(2) ?x0 wsn:hasID \"24\"^^xsd:integer .
(3) ?x0 rdf:type ?x1 .
(4) ?x1 rdfs:subClassOf wsn:Solution
}
```

The line indicated with (1) in this last query can be replaced with an appropriate *Fault* concept from the ontology. The *nodeID* mentioned as object in pattern (2) might obviously be replaced with the id of the node concerned in the query.

Additionally, for that sensor node, the reasoner is asked to infer which other types can be linked to the individual representing the faulty sensor node. This is defined by line (3). However, by also including line (4) in the query, we limit the search for other types, to those types that are modelled in the ontology as a subclass of *Solution*. After all, the goal of this query is to find the solution for a given node with a certain fault.

5 Deployment Overview

This section demonstrates that the architecture defined in the previous sections can be deployed on the different nodes of heterogeneous networks, such as the WiLab.t infrastructure, facilitating a distributed monitoring platform which can be tuned to the needs of each individual deployment. Although only a single type of sensors is currently deployed on the testbed, future extensions with different types of hardware are planned. After the integration of this new hardware, the addition of new ontology models and data providers will suffice to include them in the monitoring workflow. After all, the specific definitions of the concepts against which the observations are checked to realise the correct *Fault* and *Solution* classification, can be changed independently from the end monitoring application.

An example of this claim is the detection of faulty HVAC (Heating, Ventilation and AirConditioning). In a normal office environment, an upper threshold of 25 °C can already indicate an HVAC problem, while in a lab environment this threshold could easily be 35 °C or 40 °C. To support this kind of adaptation, the ontology T-Box which is deployed on the reasoning agent can be altered according to the needs of this particular situation. The other parts of the platform do not need to know about this, because the communication will only involve the request for rooms in which the HVAC system might be corrupted. The reasoner will use the locally deployed definition to check the local data. Figure 4 presents this deployment in a graphical manner.

The monitoring of the sensor network within the WiLab.t infrastructure has been defined in an office environment. In this setting, every office has a number of deployed sensors, working together in a wireless sensor network. Each of the offices is networked together by means of a light-weight dedicated access point, establishing a mesh network for communication between the offices. This mesh network in its turn is supported by a back-end network to facilitate more services, *e.g.* an uplink to the internet. Starting from the data generated at the source of a sensor node, we deployed the *Data Provider/Resources Modules* on the TMoteSky sensor nodes in the sensor network. As described, these modules provide the data to be included in the reasoner for the monitoring process. It makes this information available to the *Aggregator Module*, which stores it in a MySQL database. We envisage deploying such a database either in the backend network, or ideally on a mesh node. In this situation, the mesh node can handle locally the data coming from the sensor nodes attached to it. Therefore we included the *Aggregator Module* on the mesh node as well. Additionally, the

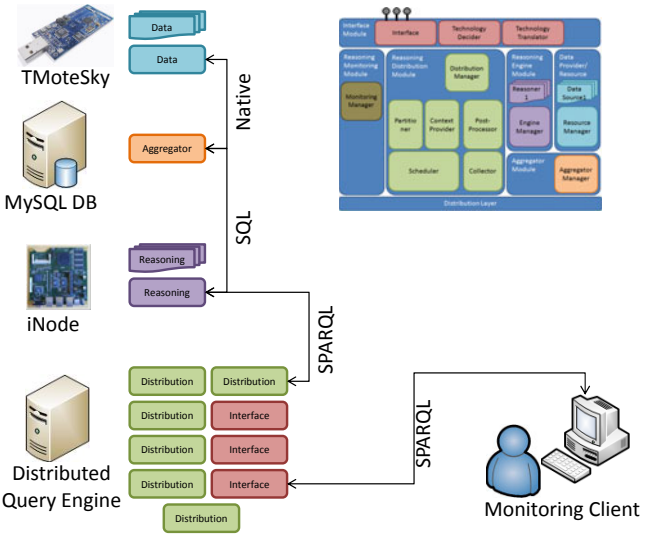


Fig. 4. Deployment view of the platform modules on the devices of a DEUS network

Reasoning Engine Module was deployed on the same mesh node. After all, since there is a mesh node for every office in which a number of sensors have been deployed, this mesh node is the ideal place to locally process and reason on the information coming from the locally deployed sensor nodes. As indicated, the description of the local situation in the ontology T-Box, the model, included in the reasoning process on this mesh node, can be tuned to the particular needs. Using D2R [18], an automatic conversion between raw data in the MySQL database and the ontology A-Box, the data, is supported. An example mapping for a certain *Symptom* concept, namely the observation that no information is being generated by the nodes, can be found below. By using a left-join SQL statement, even the absence of observations can be represented in specific A-Box individuals. For a detailed description of the constructs used in this mapping language, we refer to [18].

```
map:TMoteNoObservationSymptoms a d2rq:ClassMap;
    d2rq:uriPattern "NoObservationSymptoms/@@coordinates.id@@";
    d2rq:class vocab:MissingReportsSymptom;
    d2rq:classDefinitionLabel "MissingReportsSymptom";
    d2rq:condition "coordinates.id NOT IN
        (SELECT DISTINCT sensorinfo.moteid FROM sensorinfo)";
```

All other remaining modules are deployed on a back-end server, which can be used by monitoring clients to trigger the monitoring process.

The adoption of ontologies has not only been driven by the logical formalisms defining the constructs in such models, but also by their ability to be used

for communication. After all, an ontology can easily by serialized in a number of specific and well-defined languages, such as RDF/XML, n-triple, *etc.* Their communication nature has always been considered as one of the main strongholds of ontologies. Not surprisingly so, given their origin in the Semantic Web. This means that not only the data is serialized and/or transmitted, but also the meaning of this data. This turns it into machine-processable information. The architecture described earlier takes full advantage of this.

All high-level communication is performed using the SPARQL [14] query language. As such, only certain parts and certain views of the ontology are transferred. Additionally, only one interface operation is necessary, no matter what the content of the ontology T-Boxes is. The argument of this operation is a SPARQL query, and according to the T-Box of the ontology deployed, other SPARQL queries can be used. However, by making use of logically defined concepts, even these queries themselves won't have to change all that drastically. Of course, if a complete new ontology is used in a given deployment, serving a completely different use case, this argument does not hold. But for similar use cases, a specific alteration of the definition of the logically defined concept should be sufficient to handle the different scenarios. Additionally, by introducing the *Interface Module*, even the usage of SPARQL as implementation within the reasoning platform is transparent to the monitoring client. The interfaces and protocols used in the low-level part of the platform, namely the sensors, are specifically implemented to be used on those devices and for the information to be exchanged.

6 The Front-End Monitoring Application

In addition to the monitoring framework as detailed in the previous sections, a front-end application to visualise the reasoned and inferred information was developed as well. The goal of this application is to demonstrate the transparency of the approach, as well as its ability to be used as monitoring application as such. The *Data Provider/Resource modules* are only deployed and actively collecting information about the environment when no other jobs are scheduled on the WiLab.t testbed. This ensures that the monitoring framework does not interfere with the experiments scheduled by other users. A screenshot of this monitoring application can be seen in Figure 5.

Three queries were predefined in this application, one for *Incipient Faults*, *Detected Faults* and *HVAC Faults*. The exact implementation of the queries has been detailed in Section 4.4.

The complete workflow results in a list of nodes with their IDs in the case of node classification, and a list of room numbers in the case of Heating, Ventilation and AirConditioning monitoring. These logical IDs are captured in a tree-based view, which the administrative staff can expand to check the results of the reasoned monitoring process. This process is initiated iteratively by the monitoring application. This results in continuous monitoring and reasoning on the information available in the MySQL database. However, by doing this on

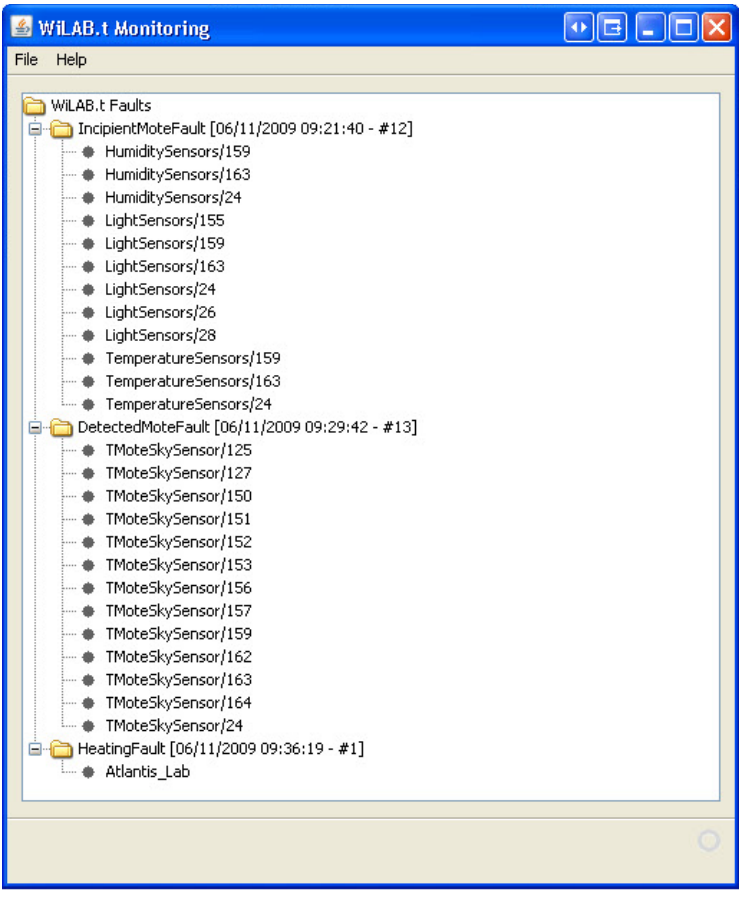


Fig. 5. WiLab.t ontology-based monitoring application front-end

historical data, superseded information is sometimes taken into account even after the fault has been rectified. This is a result of the design of the database and the *Data Provider/Resource Modules*' implementation. We plan to enhance this mechanism into an online deployment in future versions of the monitoring framework, where the information is to be fetched at-runtime during the reasoning process, thus also eliminating the need for maintaining a MySQL database with all raw data.

7 Performance Evaluation

Having presented the developed architecture and components supporting an ontology-based monitoring framework using distributed reasoning mechanisms for the IBBT WiLab.t infrastructure in the previous sections, this section details the evaluation of the platform. Apart from checking whether the conclusions

drawn automatically by the distributed reasoning process are really correct, we evaluated the overall round trip time starting from the triggering of the reasoning process until the reception of the conclusions.

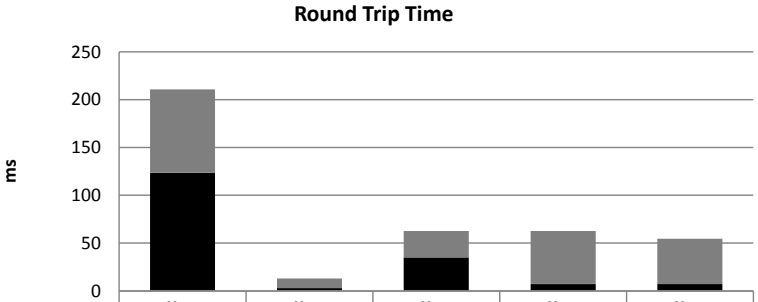
We constructed a WiLab.t test setup, using 5 offices. Each of the offices has on average 6 nodes, producing in total between 500 and 700 measurement reports per office. The scenario measured is as follows:

1. The request is initiated in the back-end, by the monitoring application,
2. The query is analysed by the back-end engine [splitting],
3. The correct iNodes are located by the back-end engine [locating],
4. The reasoning tasks are scheduled on the iNodes,
5. The 5 iNodes collect the local data and convert it into ontology A-Box data [populating],
6. The 5 iNodes execute the reasoning on the local data and return the results to the back-end [reasoning],
7. The back-end merges the results and returns it to the monitoring application.

Of the 7 tasks enumerated here, only task 2, 3, 5 and 6 significantly contribute to the overall round-trip time. The others can be neglected. After all, Task 1 only triggers the action. Secondly, since there is only a single concurrent reasoning task being executed, the scheduling has no effect. Task 7 marks the end of the process, by returning the list of merged results.

The results can be seen in the graph in Figure 6. The reasoning process was triggered 30 times. The average was recorded in the graph for each of the offices. A number of important conclusions can be drawn from this graph. First of all, it is clear that two main contributing phases in the workflow are the *"populating"* and *"reasoning"* tasks. The time the reasoning takes for even a limited number of sensor nodes per iNode clearly underlines the need for a distributed approach in constrained environments. Secondly, the influence of the amount of reports to be included in a single iNode is also of great influence. Office 1 contains the most sensor nodes, namely 13, while office 2 contains the least nodes, namely 3. Thirdly, the populating phase of the reasoning process has a significant contribution to the overall processing times. However, by implementing the intended transition towards an online approach, the need for this population approach can be avoided. We expect this to result in a lower overall round-trip time. Finally, the distribution mechanism implemented in this platform penalises the quicker offices, by not returning the results to the monitoring client until all contributing reasoning tasks have completed. Therefore, the iNode in the office taking the most time to complete will define the overall round-trip time. However, because a post-processing merger phase is included to eliminate potential duplicates, this would otherwise have to be handled by the invoking client.

On the iNodes, Java 1.6.0 update 13 was used to run an Apache Tomcat 6.0 Webserver. The reasoning and populating modules deployed in this container were implemented using Pellet 2.0.0rc4 [17] as reasoner and d2rq6 [18] as populator to convert the raw data from the MySQL database into an ontology A-Box.



	Office 1	Office 2	Office 3	Office 4	Office 5
reasoning	87181,2	10041,07	27419,53	55282,4	47328,27
populating	123099,97	2628,53	34768,4	6897,93	6670,3
locating	443,9	443,9	443,9	443,9	443,9
splitting	6,23	6,23	6,23	6,23	6,23

Fig. 6. Total round-trip time including a component break-down for the 4 major contributing phases

8 Conclusions and Future Work

In this paper, we have presented how an ontology-based monitoring framework was developed for IBBT's WiLab.t infrastructure. The adoption of the ontology technology supported by distributed reasoning mechanisms facilitates the transparent monitoring in a heterogeneous environment, where the rules according to which nodes and the environment variables need to classified can be changed according to the locally deployed hardware, specifications and environment. We have detailed the implementation of the contributing components on sensors, iNodes, back-end and front-end. The platform has been evaluated on the testbed through analysis of the processing times of the different contributing components. Moreover, the presented platform is currently in daily use for the monitoring of the WiLab.t testbed. It demonstrates that by introducing intelligent and advanced technologies in a cross-domain manner, great improvements can be achieved both in extendibility and maintainability. As can be concluded from the presentation of the monitoring platform in this paper, changes over time of classification rules are easily implemented through adaptation of the ontology deployed on the iNodes. New hardware can also be included with minimal effort if new ontology models are created modelling the new hardware. The inclusion of this new information, or the adaptation of local rules based on the local environment is achieved transparently from the monitoring application.

We plan to further exploit the developed platform to introduce a permanent monitoring application, by migrating from an offline database backed deployment towards an online fully distributed and autonomous platform. This will not only result in faster response times, but will also avoid the inclusion of potentially superseded information which might still be present in the database, even after

a given detected fault has been rectified. Finally, we plan to develop mechanisms to take changing network topologies and deployments into account in an online manner and optimise the deployment based on certain networking metrics.

Acknowledgement

Stijn Verstichel and Eli De Poorter would like to thank the IWT (Institute for the Promotion of Innovation through Science and Technology in Flanders) for financial support through his Ph.D. grant. Tim De Pauw would like to thank the University College Ghent Research Fund for financial support through his Ph.D. grant. This research is partly funded through the IBBT-DEUS project.

References

1. Xu, N.: A survey of sensor network applications. IEEE Communications Magazine 40(8), 102 (2002)
2. Murty, R.N., Mainland, G., Rose, I., Chowdhury, A.R., Gosain, A., Bers, J., Welsh, M.: CitySense - An Urban-Scale Wireless Sensor Network and Testbed. In: IEEE Conference on Technologies for Homeland Security, Waltham, MA, pp. 583–588 (2008)
3. Ertin, E., Arora, A., Ramnath, R., Nesterenko, M., Naik, V., Bapat, S., Kulathumani, V., Sridharan, M., Zhang, H., Cao, H.: Kansei: a testbed for sensing at scale. In: The Fifth International Conference on Information Processing in Sensor Networks, IPSN 2006, Nashville, TN, pp. 399–406 (2006)
4. Werner-Allen, G., Swieskowski, P., Welsh, M.: MoteLab - a wireless sensor network testbed. In: Fourth International Symposium on Information Processing in Sensor Networks, IPSN 2005, pp. 483–488 (2005)
5. WISEBED - Wireless Sensor Network Testbeds. Part of the Seventh Framework Information Communication Technologies program from the European Commission under project number 224460, `http://www.wisebed.eu/`
6. Gruber, T.R.: A translation approach to portable ontology specifications. Knowledge Acquisition (1993)
7. da Rocha, A.R., Delicato, F.C., de Souza, J.N., Gomes, D.G., Pirmez, L.: A semantic middleware for autonomic wireless sensor networks. In: Proceedings of the 2009 Workshop on Middleware For Ubiquitous and Pervasive Systems, WMUPS 2009, Dublin, Ireland, June 16, vol. 389, pp. 19–25. ACM, New York (2009), doi:http://doi.acm.org/10.1145/1551693.1551697
8. Hu, Y., Wu, Z., Guo, M.: Ontology driven adaptive data processing in wireless sensor networks. In: Proceedings of the 2nd International Conference on Scalable Information Systems, June 6-8. ACM International Conference Proceeding Series, vol. 304 (2007)
9. Delicato, F.C., Pirmez, L., Pires, P.L., Ferreira de Rezende, J.: Exploiting Web Technologies to Build Autonomic Wireless Sensor Networks. In: Mobile and Wireless Communication Networks, vol. 211, pp. 99–114. Springer, Boston (2006)
10. TMoteSky, `http://www.moteiv.com/`
11. Alix 3C3, `http://www.pcengines.ch/alix3c3.htm`
12. Voyage Linux, `http://linux.voyage.hk/`

13. Verstichel, S., Ongenae, F., Volckaert, B., De Turck, F., Dhoedt, B., Dhaene, T., Demeester, P.: An autonomous service-platform to support distributed ontology-based context-aware agents, In: Special Issue of Expert Systems: The Journal of Knowledge Engineering on Engineering Semantic Agent Systems (2009) (accepted for publication)
14. Sparql: Query Language for RDF, W3C Candidate Recommendation (2007), http://www.w3.org/TR/rdf-sparql-query/
15. Gnawali, O., Fonseca, R., Jamieson, K., Moss, D., Levis, P.: Collection Tree Protocol. In: SenSys 2009, Berkeley, California, USA, November 4-6 (2009)
16. Avancha, S.: Sensor Node Ontology, http://ebiquity.umbc.edu/resource/html/id/131/Sensor-Node-Ontology
17. Parsia, B., Sirin, E.: Pellet: An OWL DL Reasoner. In: Proceedings of the International Workshop on Description Logics (2004), http://pellet.owldl.com/
18. Bizer, C., Cyganiak, R.: D2R Server, Publishing Relational Databases on the Semantic Web. In: Proceedings of the 5th International Semantic Web Conference. Athens, GA, USA (2006)

FLAME: Flexible Lightweight Active Measurement Environment

Artur Ziviani, Antônio Tadeu A. Gomes,
Marcos L. Kirszenblatt, and Thiago B. Cardozo

National Laboratory for Scientific Computing (LNCC)
Av. Getúlio Vargas 333, 25651-075, Petrópolis-RJ, Brazil
{ziviani,atagomes,marcoslk,thiagoc}@lncc.br

Abstract. We propose a platform for the rapid prototyping of active measurement tools to collect network characteristics. The proposed platform provides its users with basic active measurement primitives upon which sophisticated active measurement tools can be prototyped quickly, practically, and efficiently through scripts in the Lua scripting language. We validate the platform as well as show its flexibility and accuracy through experiments on a local testbed and also on Planet-lab.

Keywords: Network measurements, Rapid prototyping.

1 Introduction

Network measurements [1,2] aim at characterizing the performance, behavior, dynamics, and structure of different kinds of networks. From an implementation viewpoint, the currently available measurement tools (see [3] for a survey) are typically coded in low-level programming languages (usually C) to avoid the impact of high-level programming language features—e.g. garbage collection and exception handling mechanisms—on the accuracy of measurement results. As a consequence, most of such tools present a potentially high development time. Besides, such tools are based on very low-level network APIs (usually BSD socket-like APIs), which hinders higher levels of code reuse across tool projects—this is easily observable in the open-source codes of several publicly available measurement tools. Further, the absence of standards in these tools with respect to the collection and storage of measurement data brings inconveniences to their integrated use in the existing measurement platforms.

In this paper, we propose a platform for the rapid prototyping of *active* measurement tools, i.e. tools based on the sending of *probes* (packets with the single purpose of performing measurements) between network nodes, thus allowing the measurement of network properties along the path linking such nodes. Our proposal, named FLAME (Flexible Lightweight Active Measurement Environment),[1,2] allows the rapid prototyping of active measurement tools even if the

[1] The source code is available at `http://martin.lncc.br/main-software-flame/`

[2] Disambiguation: We recently became aware of a set of tools also called FLAME [4] to facilitate the manipulation of flow traces, a totally different purpose than ours.

T. Magedanz et al. (Eds.): TridentCom 2010, LNICST 46, pp. 526–541, 2011.

targeted network metric depends on a *cooperative* destination node, i.e. a destination node that hosts (part of) the measurement tool.

The FLAME platform is based on the distribution of measurement agents among some network nodes. Such agents send and receive probe packets in response to commands from a central manager. These agents publish the collected measurement data in a standardized way on a central repository, simplifying the management and further analysis of such data.

The FLAME platform offers its users active measurement primitives to be executed in the agents. Users can prototype active measurement tools upon such primitives in a rapid, practical, and efficient manner. The central manager is responsible for deploying such tools and starting their execution in the agents.

Tool prototyping in the FLAME platform is based on the Lua scripting language [5]. Lua is adopted in FLAME as an *extension* language: its interpreter is embedded as a library into the measurement agents. On the one hand, the Lua interpreter gives to the scripts running in the agents access to active measurement primitives through a high-level, minimalist API [6]. On the other hand, the measurement agents and the measurement API are implemented in C, preventing significant overheads in the measurement results due to the execution of Lua scripts. We validate the platform as well as show its flexibility and accuracy through experiments on a local testbed and also on the Planet-lab platform.

The remainder of the paper is organized as follows. Section 2 briefly reviews related work. The FLAME architecture is presented in Section 3. Section 4 describes the The minimalist, high-level measurement API offered by the FLAME platform. In Section 5 we give examples of tool prototypes in the FLAME platform and present experimental results conducted using such tools to validate the platform. Finally, in Section 6 we conclude the paper, also indicating some possible future work.

2 Related Work

Related work on network measurements generally targets two different kinds of results: (i) the development or improvement of measurement tools specialized in measuring a specific subset of network environment properties—Michaut and Lepage [3] provide a survey of several existing network measurement tools; (ii) the deployment of large scale measurement platforms that employ measurement tools (usually developed independently by other projects) to analyze network metrics in a more comprehensive way. Such tools and platforms are described in the following subsections.

2.1 Tools for Active Network Measurement

Each active measurement tool typically aims at evaluating network performance according to a limited set of metrics. The most used metrics and some of the tools proposed to measure them are pointed out in Table 1, adapted from [3]. This table also indicates which tools depend on the presence of cooperative destination nodes to work correctly.

Table 1. Examples of typical active measurement metrics and tools

Tools	COP	OWD	OWV	RTT	PLR	PRO	ROT	PHC	ECP	ABW
`Iperf` [7]	✓		✓		✓					
`owping` [8]	✓	✓			✓	✓				
`pchar` [9]			✓					✓		
`pathload` [10]	✓									✓
`pathrate` [11]	✓								✓	
`ping`				✓	✓					
`QoSMet` [12]	✓	✓	✓		✓	✓				
`sprobe` [13]									✓	
`sting` [14]					✓	✓				
`traceroute`							✓			
`traceroute-paris`[15]							✓			

Legend:
COP: tools that depend on cooperative destination nodes.

OWD: one-way delay; **OWV**: OWD variation; **RTT**: round-trip time;
PLR: packet loss rate; **PRO**: packet reordering; **ROT**: route tracing;
PHC: per-hop capactity; **ECP**: end-to-end link capacity;
ABW: available bandwidth.

An important point to highlight in the active measurement tools such as those in Table 1 is the low (if none) code reuse, even for those tools developed in the same research group (e.g. `pathrate` and `pathload`) or with very similar functionalities and purposes (e.g. `traceroute` and `traceroute-paris`). Another key point to remark is that generated results are provided in an *ad hoc* format (e.g. consider the extreme case of popular tools as `ping` and `traceroute`, which provide different outputs in different operating systems), making it difficult to manage this data and eventually analyze measurement results in an integrated way. As an example, although there exists some proposals to standardize log formats [16,17], few active measurement tools (e.g. `pathrate` and `pathload`) do use these standards to present the results of their measurements.

2.2 Platforms for Active Network Measurement

NIMI [18] and ETOMIC [19] are large-scale active measurement platforms that apply restrictive, domain-based trust models among measurement managers and agents. Both platforms use third-party active measurement tools as plug-ins in the measurement agents, which hinders code reuse across tools, making it harder the implementation of innovative measurement techniques. Besides, neither NIMI nor ETOMIC provides a central result repository, making it more difficult to analyze measurement results in an integrated way.

DIMES [20] goes in a different direction by employing an approach similar in concept to projects on voluntary computing (such as SETI@home [21]), rendering a potentially high coverage of the platform. The main goal of DIMES is to make available a detailed connectivity graph of the Internet. DIMES offers a XML-based language, called PENny, for the specification of the experiments. This language is rather limited in its expressiveness, however, only offering primitives for measurement of bidirectional delay and route tracing, which constrains the flexibility of the DIMES platform.

ANEMOS [22] and Flexmon [23] focus on the provisioning of central result repositories. ANEMOS allows the definition of rule-based alarms according to the measured network performance, whereas Flexmon is able to constrain probes according to the resources available at the measurement agents. Like NIMI and ETOMIC, ANEMOS and Flexmon employ third-party active measurement tools.

ATMEN [24] and NetQuest [25] aim at reducing the overhead of active network measurements. ATMEN avoids wasted measurements by judiciously reusing measurement results. NetQuest employ inference algorithms that select data collected in a measurement experiment to maximize the amount of information gathered about the properties of certain network paths, given the set of constraints about the experiments (e.g. the maximum allowed amount of issued probes). Like most of the aforementioned platforms, ATMEN uses third-party active measurement tools. The way such tools are implemented/used in NetQuest is not presented in [25].

Scriptroute [26] is the approach we regard as closest to ours. In this platform active measurement tools are specified in Ruby, a fully object-oriented scripting language, whereas measurement agents, which interpret the Ruby scripts, are implemented in C. Like Flexmon, Scriptroute is able to constrain probes according to the resources available at the measurement agents. Nevertheless, the flexibility of Scriptroute is limited by the impossibility of implementing tools that depend on cooperative destination nodes. Furthermore, Scriptroute does not have a centralized repository in which the measurement results are stored. Thus, the gathering and output of results must be explicitly coded in the Ruby scripts that implement the tools. This approach renders scripts that tangle probing and data output functionality, hindering higher levels of reuse.

3 FLAME Architecture

The FLAME architecture is presented in Figure 1. The communication between the components of the architecture—namely the *user console*, the *measurement manager*, and the *measurement agents*—is performed through the XMPP protocol [27]. An XMPP service (comprising one or more XMPP servers, either dedicated or public ones) acts as a message bus among FLAME components. We describe the role of each FLAME component in the following.

Users issue measurement sessions to the measurement manager using a text-based console application. The measurement manager is responsible for initiating the issued measurement sessions—called *experiments* in FLAME—in the measurement agents and for storing the results obtained in these sessions in a central repository (a relational database in the current version of platform). The measurement agents are responsible for executing the experiments, caching the collected measurement results and sending them to the manager at the end of these experiments—such delay is to avoid remote repository updates to interfere with the experiments.

An experiment is specified as a script that describes the prototyped tools to be adopted in the experiment and the commands that invoke such tools. Such a

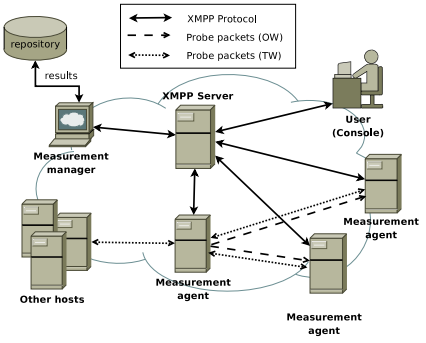


Fig. 1. FLAME architecture

script has access to the basic primitives of the measurement API implemented by the agent (see Section 4 for further details about such primitives). Both the script and the corresponding collected measurement results from an experiment are published in the central repository. This allows for better data provenance, since the prototyped tools and the commands that invoke them can be later analyzed (for instance, concerning their correctness and accuracy) based on the collected results.

It is worth noting that measurements may be performed between sets of measurement agents, or between measurement agents and ordinary hosts (i.e. hosts that do not have a measurement agent but may reply to probes, for instance a host answering ICMP requests), as depicted in Figure 1. In the case of measurements between sets of agents, besides two way (TW) measurements such as those performed between agents and ordinary hosts, it is also possible to have one-way (OW) measurements.

The use of XMPP presence controls permits the measurement manager to ask agents to perform an specific experiment in *exclusive* mode, meaning that the XMPP service will make such agents temporarily unavailable for other experiments. This functionality allows carrying out experiments that would otherwise be disturbed by concurrent experiments in the same set of agents.

4 The Minimalist, High-Level Measurement API

In the FLAME platform, experiments are specified in the Lua scripting language [5]. We chose Lua to prototype active measurement tools due to the following convenient characteristics it presents:

– Lua has a simple (procedural) syntax and a high abstraction level, thus having a great potential to reduce software development time. In the FLAME platform, the combination of Lua and a central repository allows the developer of measurement tools to focus more on the probing techniques than on the particularities of lower-level languages (e.g. memory management) and data output functionality;

– Lua presents extensible semantics (e.g. allowing the use of threads in different collaboration levels [28]) and a reduced memory footprint,[3] thus making measurement agents deployable on nodes with diverse levels of resource availability.

Each measurement agent in the platform hosts an adapted Lua interpreter that provides a sandboxing environment (like the Scriptroute platform) for the controlled execution of Lua scripts. The agents make the measurement primitives available to the Lua scripts through an API in C that is exported to the Lua interpreter with the name `lamp` (*Lua Active Measurements Primitives*).

The names of the probing operations offered by the `lamp` API follow the `send[protocol][type](...)` structure, where `protocol` indicates the protocol used for sending probes (in the current implementation, UDP, TCP, and ICMP are available), and `type` indicates if the probes are bidirectional (TW – *Two Way*), unidirectional (OW – *One Way*), or sent in a burst (PT – *Packet Train*). The TW operations do not depend on cooperative destination nodes and the results obtained with these operations are collected in the source of the experiment. In this case, the source node is responsible for sending the results to the measurement manager. The OW and PT operations depend on cooperative destination nodes. In this case, a destination node is responsible for collecting the results and sending them to the measurement manager. Such operations, however, also instruct the destination nodes to send the collected results back to the source node after the operation execution. This is important when the developer needs to implement active measurement tools that rely on successive iterations based on the feedback from the cooperative destination nodes to properly measure certain network characteristics (as in the case of `pathload`, for instance).

All probing operations of the `lamp` API return a Lua table[4] containing the collected results, in case of success. Such probing operations are extensibly parametrized, but without impacting significantly on the usability gained with a minimalist API, given that several parameters are optional and, when omitted, receive default values.

Besides probing operations, the `lamp` API also offers other operations, such as `sleep(...)` (to suspend the execution of a script for a certain amount of time), and a set of operations for querying the central repository, allowing the reuse of previously implemented tools—and collected results, like the ATMEN platform—in the context of other experiments (due to space restrictions we omitted in this paper a detailed explanation of such operations).

[3] A comparative benchmark study, available at `http://shootout.alioth.debian.org`, points out an average memory consumption 530%, 260%, and 190% larger than Lua for Ruby, Python, and Perl, respectively.

[4] Lua offers a single type of data structure, called *table*, that implements associative arrangements of the kind $\{k_1 = v_1, \ldots, k_n = v_n\}$, i.e. each value v_i stored in the structure is associated to a key k_i. Keys and values can assume any type, even though numbers and strings are the most common types of keys. Values are indexed by numeric keys in the form `table[key]` and by alphanumeric keys in the form `table[''key'']`, or simply `table.key`.

5 Experimental Validation

To validate the FLAME platform as well as to illustrate its flexibility and accuracy, we conducted a set of measurements on two experimental scenarios: the Planet-lab platform and a local testbed. We stress that these experiments are intended to validate the FLAME platform and not to provide a performance evaluation of FLAME.

The following subsections describe these experimental scenarios and compare a set of measurements collected in such scenarios with publicly available tools and equivalent tools prototyped on the FLAME platform. In the case of the publicly available tools, the measurement results were obtained by post-processing the textual output of such tools. In the case of our prototyped tools, the measurement results were obtained by querying the central repository of FLAME directly and post-processing the returned data. It is important to emphasize here that this approach to data collection encourages the untangling of probing and data output functionality, which can be observed from the compactness and neatness of the prototype sources presented in the following subsections.

5.1 Experimental Scenarios

The Planet-lab platform was used for RTT and route tracing experiments. For such experiments, we employed six nodes spread throughout the globe, one node (S1) working as the source and the other five nodes (T1, ..., T5) as targets. Such nodes are presented in Table 2.

Table 2. Source and target nodes of the Planet-lab experiments

	IP address	Domain name
Source		
S1	200.19.159.34	`planetlab1.pop-mg.rnp.br`
Targets		
T1	130.83.166.198	`host1.planetlab.informatik.tu-darmstadt.de`
T2	128.112.139.80	`alice.cs.princeton.edu`
T3	132.65.240.100	`planet1.cs.huji.ac.il`
T4	130.216.1.22	`planetlab-1.cs.auckland.ac.nz`
T5	131.112.243.102	`node2.planet-lab.titech.ac.jp`

Our local testbed was used for one-way delay and link capacity experiments. It comprises four Linux nodes (Ubuntu distribution) connected in a row topology; the two nodes at the edges of the row working as the source and destination nodes and the other two nodes as intermediate routers. For this scenario we adopted two link configurations. In the first configuration, the three links worked at 10 Mbps (referred to in this section as the "10-10-10 topology"). In the second configuration, the link in the middle of the row topology worked at 100 Mbps (the "10-100-10 topology").

5.2 RTT Measurements

For our RTT experiments in Planet-lab, we used the `ping` tool available in node S1 and prototyped an equivalent tool (`flamePing`) using the `lamp` API, deploying it in the FLAME measurement agent running on node S1. The code in Listing 1 illustrates our `flamePing` prototype.

```
1  function flamePing(params)
2    -- Ping params (and corresponding defaults)
3    local _target = params.target or "127.0.0.1"
4    local _size = params.size or 56
5    local _interval = params.interval or 250000 -- microsecond resolution
6    local _npackets = params.npackets or 10
7    local _protocol = params.protocol or "icmp"
8    local _timeout = params.timeout or 5000000 -- microsecond resolution
9    local _ttl = params.ttl or 30
10
11   for i = 1,_npackets do
12     local _response
13
14     -- Choose probing protocol (for UDP and TCP primitives,
15     --  since port is not indicated it is randomly chosen above 1024)
16     if _protocol == "icmp" then
17       _response=lamp.sendICMPTW{ip=_target, size=_size,
18                                 timeout=_timeout, ttl=_ttl}
19     elseif _protocol == "udp" then
20       _response=lamp.sendUDPTW{ip=_target, size=_size,
21                                timeout=_timeout, ttl=_ttl}
22     elseif _protocol == "tcp" then
23       _response=lamp.sendTCPTW{ip=_target, size=_size,
24                                timeout=_timeout, ttl=_ttl}
25     else
26       print("INVALID PROTOCOL:", _protocol)
27       return
28     end
29
30     -- Check host/net unreachability
31     if _response and _response.loss == ICMP_HOST_UNREACH then
32       print("DESTINATION UNREACHABLE: ", _target)
33       return
34     end
35
36     -- Wait time between probes
37     if type(_interval) == "number" then
38       lamp.sleep(_interval)
39     elseif type(_interval) == "table" then
40       lamp.sleep(_interval.func(_interval.params))
41     end
42   end --for i
43 end
```

Listing 1. FLAME ping prototype

It is worth noting in Listing 1 that our `flamePing` prototype is pretty simple, yet provides functionality for sending UDP and TCP probes besides ICMP probes (lines 14-28), and for spacing probes according to arbitrary functions (lines 36-41). In Table 5.2 we illustrate two different sets of commands that issue the execution of `flamePing` on node S1 in Planet-lab, the first one with probes uniformly spaced (as the original `ping` tool does) and the second one with probes exponentially spaced.

Table 3. Commands to execute `flamePing` over Planet-lab nodes

Probes uniformly spaced	Probes exponentially spaced

`local targets = {`
`  "130.83.166.198",`
`  "128.112.139.80",`
`  "132.65.240.100",`
`  "130.216.1.22",`
`  "131.112.243.102"`
`}`
 	
`for k,v in ipairs(targets) do`
`  flamePing{target=v}`
`end`
	`local function expdist(lambda)`
	`  local r = 0`
	`  repeat r = math.random() until(r ~= 0)`
	`  return math.floor(-math.log(r)/lambda)`
	`end`
	
	`local targets = ... -- same as left`
	
	`for k,v in ipairs(targets) do`
	`  flamePing{`
	`  target=v,`
	`  interval={ func=expdist, params=1/10000000 }`
	`  }`
	`end`

Figure 2 illustrates some RTT measurement statistics collected in Planet-lab with `ping` and `flamePing` (both uniform and exponential distributions). Such statistics comprise, for each target in Table 2, 36 interleaved executions of each tool spaced by 10 minutes, to minimize the effect of traffic synchronization. For each tool, one execution sent out 10 probes of 56 bytes for a total of 360 samples per tool and a total of 18 hours of measurement collections. The error bars in the figure show for each tool the 05- and 95-percentiles among the 360 samples, and the data points show the corresponding averages. It is important to note how similar the `ping` and `flamePing` tools characterized the network behavior with regard to RTTs among the used Planet-lab nodes.

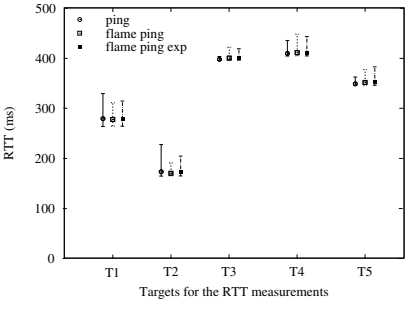


Fig. 2. RTT measurement statistics in Planet-lab

5.3 Route Tracing

For our route tracing experiments in Planet-lab, we used the `traceroute` tool available in node S1 and prototyped an equivalent tool (`flameTrace`) using the `lamp` API, deploying it in the FLAME measurement agent running on node S1. The code in Listing 2 illustrates our `flameTrace` prototype.

It is worth noting the similarity between Listing 1 and Listing 2, which becomes more apparent as the FLAME design encourages the separation between packet probing and data output/processing functionality. For triggering route

```
1  function flameTrace(params)
2    local _target = params.target or "127.0.0.1"
3    local _size = params.size or 60
4    local _interval = params.interval or 250000
5    local _npackets = params.npackets or 3
6    local _protocol = params.protocol or "udp"
7    local _timeout = params.timeout or 5
8    local _maxhops = params.maxhops or 30
9
10   for h = 1, _maxhops do
11     local _response
12
13     for i = 1,_npackets do
14       -- Choose probing protocol (for UDP and TCP primitives,
15       --  since port is not indicated it is randomly chosen above 1024)
16       if _protocol == "icmp" then
17         _response=lamp.sendICMPTW{ip=_target, size=_size,
18                                   timeout=_timeout, ttl=h}
19       elseif _protocol == "udp" then
20         _response=lamp.sendUDPTW{ip=_target, size=_size,
21                                  timeout=_timeout, ttl=h}
22       elseif _protocol == "tcp" then
23         _response=lamp.sendTCPTW{ip=_target, size=_size,
24                                  timeout=_timeout, ttl=h}
25       else
26         print("INVALID PROTOCOL:", _protocol)
27         return
28       end
29
30       ... -- same as lines 30-41 in flamePing
31     end --for i
32
33     -- Is current node the target?
34     if (_response.remIP == _target) then break end
35   end --for h
36 end
```

Listing 2. FLAME route tracing prototype

tracing experiments in Planet-lab using FLAME, we issued the commands below on node S1:

```
local targets = {"130.83.166.198", "128.112.139.80",
                 "132.65.240.100", "130.216.1.22", "131.112.243.102"}

for k,v in ipairs(targets) do flameTrace{target=v} end
```

Like the RTT experiments, the route tracing experiments were conducted using the `traceroute` and `flameTrace` tools in an interleaved way for a total of 18 hours. Figure 3 shows a graph representation of the paths linking the Planet-lab nodes used in the experiments, as identified by both the `traceroute` and `flameTrace` tools. The results of both tools were again quite similar, showing no changes on such routes during the experiments. The only perceived difference was on the process of route tracing between S1 and T1 nodes. In Figure 3, the dashed loop arrow on T1 indicates that in most of our experiments the `traceroute` tool couldn't realize it had arrived at the destination node, continuing to increment the TTL field of probing packets until its maximum value (30 hops). This situation didn't happen with the `flameTrace` tool in any of the experiments. We attribute this phenomenon to an unexpected situation that `traceroute` couldn't

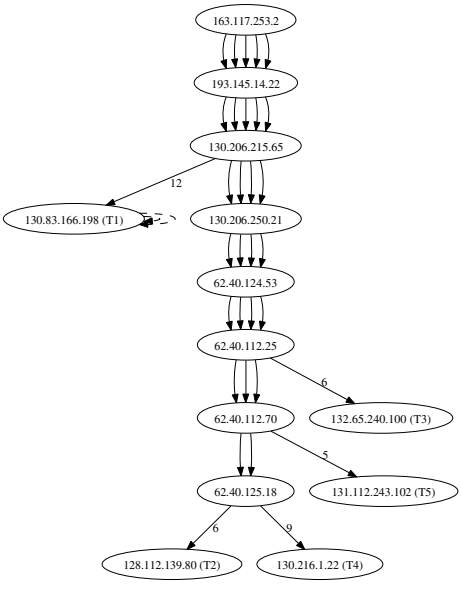


Fig. 3. Graph representation of `traceroute` and `flameTrace` results. The numbered edges indicate the number of omitted hops between the corresponding nodes.

sort out. With the rapid prototyping facilities offered by FLAME we expect such kind of situation to be easily identified and corrected by the tool developer.

5.4 One-Way Delay Measurements

Unlike the tools used for the RTT and route tracing experiments, the tools we used for one-way delay measurements depend on cooperative destination nodes. To minimize clock synchronization errors, these experiments were conducted in our local testbed with the source and destination nodes synchronizing on a local NTP server. For these experiments, we installed the `owping` tool in the source and destination nodes and prototyped an equivalent tool (`flameOWD`) using the `lamp` API, deploying it in the FLAME measurement agent running on the source node (in these experiments we also ran a management agent on the cooperative destination node). The code in Listing 3 illustrates our `flameOWD` prototype.

The single command `flameOWD{target=‘‘10.0.2.2’’, npackets=200}` was used for issuing the execution of `flameOWD` on the source node in our testbed.

Table 4 illustrates some one-way delay measurement statistics collected in our local testbed using the 10-10-10 and 10-100-10 topologies with `owping` and `flameOWD`. Such statistics comprise, for each topology, 200 probes of 56 bytes. The table shows for each tool and topology the median, 05- and 95-percentiles among the 200 samples. It is important to note that in both topologies the `flameOWD` prototype provided lower one-way delays, but the NTP estimated synchronization error (the same for both tools) rendered statistically equivalent results between the two tools.

```
1  function flameOWD(params)
2    local _target = params.target or "127.0.0.1"
3    local _size = params.size or 56
4    local _interval = params.interval or 750000
5    local _npackets = params.npackets or 5
6    local _protocol = params.protocol or "udp"
7    local _timeout = params.timeout or 5000000
8    local _port = params.port or nil
9
10   for i = 1,_npackets do
11     local _response
12
13     -- Choose probing protocol (if port is not indicated the destination node
14     -- chooses a port above 1024. In any case, for OW operations,
15     -- the source and destination nodes agree on the used port
16     -- through XMPP messages)
17     if _protocol == "udp" then
18       _response=lamp.sendUDPOW{ip=_target, size=_size,
19                                timeout=_timeout, port=_port}
20     elseif _protocol == "tcp" then
21         _response=lamp.sendTCPOW{ip=_target, size=_size,
22                                timeout=_timeout, port=_port}
23     else
24       print("INVALID PROTOCOL:", _protocol)
25       return
26     end
27
28     -- Check port unavailability and host/net unreachability
29     if _response then
30       if _response.loss == ICMP_HOST_UNREACH then
31         print("DESTINATION UNREACHABLE: ", _target)
32         return
33       elseif _response.loss == PORT_ALREADY_IN_USE then
34         print("DESTINATION PORT ALREADY IN USE: ", _target, _port)
35         return
36       end
37     end
38
39     ... -- same as lines 36-41 in flamePing
40   end --for i
41 end
```

Listing 3. FLAME one-way delay measurement prototype.

Table 4. Measured one-way delay in ms. NTP estimated synch error is ± 0.41 ms

	10-10-10 Topology			**10-100-10 Topology**		
	P05	Median	P95	P05	Median	P95
`owping`	1.28	1.39	1.50	1.10	1.21	1.32
`flameOWD`	0.87	0.91	0.98	0.46	0.51	0.58

5.5 Link Capacity Estimation

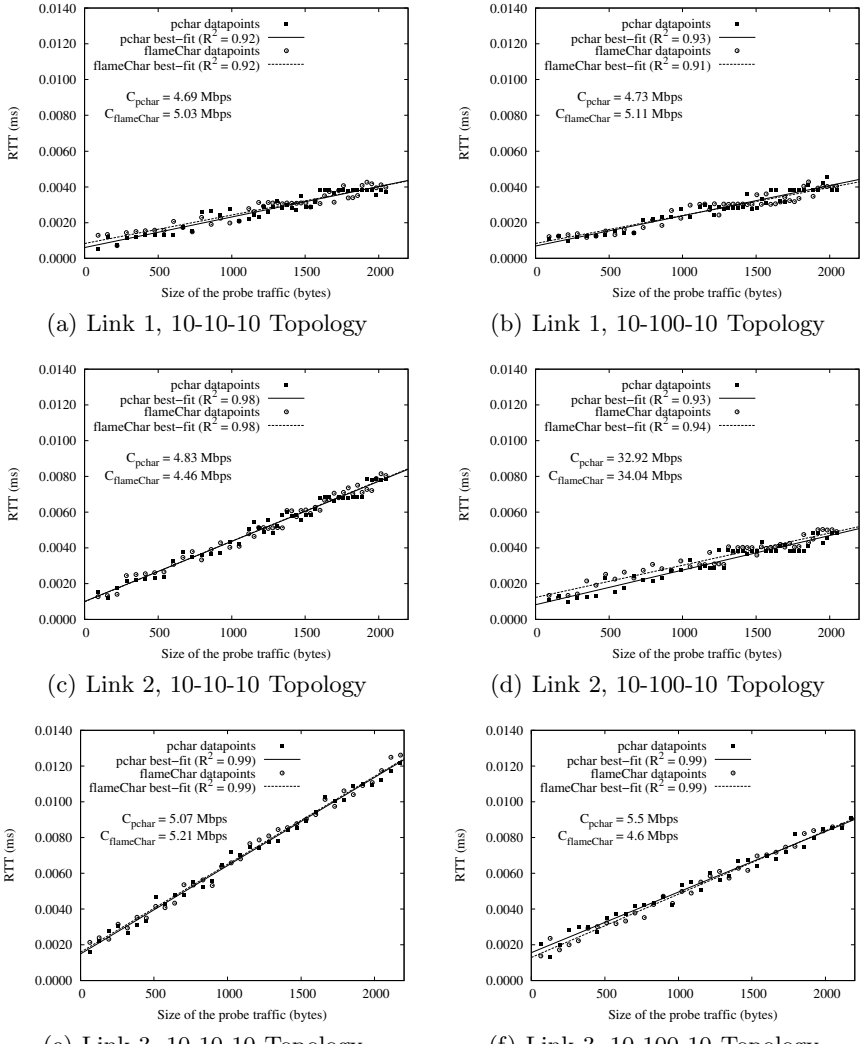
For the link capacity experiments, we employed our local testbed so that we could have more precise information about the actual link capacities we were interested in estimating. For these experiments, we installed the `pchar` tool in the source node and prototyped an equivalent tool (`flameChar`) using the `lamp` API, deploying it in the FLAME measurement agent running on the source node. The code in Listing 4 illustrates our `flameChar` prototype.

```
function flameChar(params)
  local _mtu = params.mtu or 1500 -- probe size limit
  local _increment = params.increment or 32 -- difference between probe sizes
  local _npackets = math.floor(_mtu/_increment)
  local _maxhops = params.maxhops or 30 -- maximum number of allowed hops
  local _repetitions = params.repetitions or 32 -- number of tests per hop
  local _target = params.target or "127.0.0.1"
  local _timeout = params.timeout or 3
  local _interval = param_table.interval or 500000 -- time between probes

  local _sizelist = generateProbeSizeList(_npackets, _increment)

  for h = 1, _maxhops do
    local _response
    for t = 1,_repetitions do
      for i = 1,_npackets do
        local _response

        -- Choose probing protocol (for UDP and TCP primitives,
        --  since port is not indicated it is randomly chosen above 1024)
        if _protocol == "icmp" then
          _response = lamp.sendICMPTW{ip=_target, size=_sizelist[i],
                                      ttl=h, timeout=_timeout}
        elseif _protocol == "udp" then
          _response = lamp.sendUDPTW{ip=_target, size=_sizelist[i],
                                      ttl=h, timeout=_timeout}
        elseif _protocol == "udp" then
          _response = lamp.sendTCPTW{ip=_target, size=_sizelist[i],
                                      ttl=h, timeout=_timeout}
        else
          print("INVALID PROTOCOL:", _protocol)
          return
        end

        ... -- same as lines 30-41 in flamePing
      end --for i
    end --for t

    -- Is current node the target?
    if (_response.remIP == _target) then break end
  end --for h
end

-- Function to create a shuffled package size list
function generateProbeSizeList(npackets, increment)
  ... -- implementation omitted due to space limitations
end
```

Listing 4. FLAME link capacity estimation prototype.

The single command `flameChar{target=''10.0.2.2''}` was used for issuing the execution of `flameChar` on the source node in our local testbed.

Figure 4 illustrates some link capacity measurements and related statistics collected in our local testbed using the 10-10-10 and 10-100-10 topologies with `pchar` and `flameChar`. Such measurements and statistics comprise, for each topology, a single execution of each tool. For each tool, the execution sent out 32 probes of varying sizes (from 32 to 1472 bytes in increments of 32 bytes) in a particular topology. The figure shows for each tool and topology the minimum RTTs per probe size and the "best-fit" lines obtained through the linear least squares fitting technique. As shown in [9], the slope of such lines is used for estimating the capacity of each link in both topologies. Figure 4 also shows the



(a) Link 1, 10-10-10 Topology

(b) Link 1, 10-100-10 Topology

(c) Link 2, 10-10-10 Topology

(d) Link 2, 10-100-10 Topology

(e) Link 3, 10-10-10 Topology

(f) Link 3, 10-100-10 Topology

Fig. 4. Link capacity measurements

capacity estimation per link obtained with both tools (C_{pchar} and $C_{flameChar}$). As can be seen from Figure 4, both tools returned quite similar results.

6 Conclusions

We proposed the FLAME platform for the rapid prototyping of active measurement tools and the execution of experiments using them. Overall, compared with the main characteristics of previous work, FLAME provides the following contributions: (i) an environment for the rapid prototyping of active measurement

tools based on a small but comprehensive set of probing primitives, allowing the implementation of tools that depend or not on cooperative destination nodes; (ii) a centralized repository for implicitly gathering measurement results in a common data format, encouraging the untangling of probing and data output functionality and easing the process of further analysis and comparison among different result datasets. It is also interesting to highlight the declared future goal of CAIDA's newest Ark active measurement infrastructure [29] to provide a high-level API that eases the challenges of writing measurement tools. Inline with this trend, we believe our FLAME platform achieves this goal.

As future work, we plan to implement other well-known active measurement tools to further validate the comprehensiveness of our measurement API. Moreover, we intend to evaluate the performance of the FLAME platform and its prototypes in face of other platforms such as scriptroute and low-level developed tools for active measurements. We also intend to port our FLAME platform to resource-constrained devices such as PDAs and sensors, taking profit from the low footprint of the Lua interpreter, so that we can collect measurement data on networks such as cellular, ad-hoc, disruption-tolerant, and sensor networks.

Acknowledgement

This work was supported by the Brazilian Funding Agencies FAPERJ, CNPq, CAPES, and by the Brazilian Ministry of Science and Technology (MCT).

References

1. Crovella, M., Krishnamurthy, B.: Internet Measurement: Infrastructure, Traffic and Applications. John Wiley & Sons Inc., New York (2006)
2. Ziviani, A.: Internet measurements. In: Freire, M., Pereira, M. (eds.) Encyclopedia of Internet Technologies and Applications, pp. 235–241. IGI Gobal (2007)
3. Michaut, F., Lepage, F.: Application-oriented Network Metrology: Metrics and Active Measurement Tools. IEEE Comm. Surv. & Tut. 7(2), 24 (2005)
4. Brauckhoff, D., Wagner, A., May, M.: Flame: a flow-level anomaly modeling engine. In: CSET 2008: Proceedings of the conference on Cyber security experimentation and test, pp. 1–6. USENIX Association, Berkeley (2008)
5. Ierusalimschy, R., Henrique, L., Waldemar, F., Filho, C.: Lua – an extensibleextension language. Software: Practice and Experience 26, 635–652 (1996)
6. Henning, M.: API design matters. Queue 5(4), 24–36 (2007)
7. Hsu, C., Kremer, U.: IPERF: A framework for automatic construction of performance prediction models. In: Proceedings of the Workshop on Profile and Feedback- Directed Compilation (1998)
8. Boote, J.: One–way ping, OWAMP (2009), `http://e2epi.internet2.edu/owamp`
9. Downey, A.B.: Using pathchar to estimate Internet link characteristics. In: Proceedings of the ACM SIGCOMM 1999, pp. 241–250 (1999)
10. Jain, M., Dovrolis, C.: End-to-end available bandwidth: measurement methodology, dynamics, and relation with TCP throughput. IEEE/ACM Trans. Netw. 11(4), 537–549 (2003)

11. Dovrolis, C., Ramanathan, P., Moore, D.: Packet-dispersion techniques and acapacity-estimation methodology. IEEE/ACM Trans. Netw. 12(6), 963–977 (2004)
12. Prokkola, J., Hanski, M., Jurvansuu, M., Immonen, M.: Measuring WCDMA and HSDPA Delay Characteristics with QoSMeT. In: Proceedings of the IEEE International Conference on Communications, pp. 492–498 (June 2007)
13. Saroiu, S., Gummadi, K.P., Gribble, S.D.: SProbe: A fast tool for measuring bottleneck bandwidth in uncooperative environments (2002), http://sprobe.cs.washington.edu
14. Savage, S.: Sting: A TCP-based network measurement tool. In: Proceedings of the USENIX Symposium on Internet Technologies and Systems, pp. 71–79 (1999)
15. Augustin, B., Cuvellier, X., Orgogozo, B., Viger, F., Friedman, T., Latapy, M., Magnien, C., Teixeira, R.: Avoiding traceroute anomalies with paris traceroute. In: Proceedings of the ACM Internet Measurement Conference, pp. 153–158 (2006)
16. Abela, J., Debeaupuis, T.: Universal format for logger messages (1999), http://tools.ietf.org/id/draft-abela-ulm-05.txt
17. Open Log Format, The OLF standard (2009), http://www.openlogformat.org/
18. Paxson, V., Mahdavi, J., Adams, A., Mathis, M.: An architecture for large-scaleInternet measurement. IEEE Comm. Magazine 36(8), 48–54 (1998)
19. Magana, E., Morato, D., Izal, M., Aracil, J., Naranjo, F., Astiz, F., Alonso, U., Csabai, I., Haga, P., Simon, G., Steger, J., Vattay, G.: The European traffic observatory measurement infrastructure (ETOMIC). In: Proceedings IEEE Workshop on IP Operations and Management (October 2004)
20. DIMES, The DIMES project, (2009), http://www.netdimes.org/
21. Anderson, D.P., Cobb, J., Korpela, E., Lebofsky, M., Werthimer, D.: Seti@home:an experiment in public-resource computing. Comm. ACM 45(11), 56–61 (2002)
22. Danalis, A., Drovolis, C.: Anemos: An autonomous network monitoring sys- tem. In: Proc. of 4th Passive and Active Measurements Workshop, San Diego, CA (2003)
23. Johnson, D., Gebhardt, D., Lepreau, J.: Towards a high quality path-oriented network measurement and storage system. In: Claypool, M., Uhlig, S. (eds.) PAM 2008. LNCS, vol. 4979, pp. 102–111. Springer, Heidelberg (2008)
24. Krishnamurthy, B., Madhyastha, H.V., Spatscheck, O.: ATMEN: a triggered network measurement infrastructure. In: Proceedings of the 14th ACM International Conference on World Wide Web, New York, USA, p. 499 (2005)
25. Song, H.H., Qiu, L., Zhang, Y.: NetQuest: a flexible framework for large-scalenetwork measurement. IEEE/ACM Trans. Netw. 17(1), 106–119 (2009)
26. Spring, N., Wetherall, D., Anderson, T.: Scriptroute: a public internet measurement facility. In: Proceedings of the USENIX Symposium on Internet Technologies and Systems, pp. 17 (2003)
27. Saint-Andre, P.: Extensible Messaging and Presence Protocol (XMPP): Core. RFC 3920 (2004)
28. Moura, A.L.D., Ierusalimschy, R.: Revisiting coroutines. ACM Trans. Pro- gram. Lang. Syst. 31(2), 1–31 (2009)
29. CAIDA, Archipelago measurement infrastructure (2009), http://www.caida.org/projects/ark/

TopHat: Supporting Experiments through Measurement Infrastructure Federation

Thomas Bourgeau, Jordan Augé, and Timur Friedman

LIP6 laboratory, UPMC Paris Universitas and CNRS

Abstract. Researchers use the PlanetLab testbed for its ability to host experimental applications in realistic conditions over the public best-effort internet. Such applications form overlays whose performance is affected by the underlying topology and its evolution. While several topology information services have been proposed for PlanetLab, the TopHat system that we describe here fills a special niche. It is designed to support the entire lifecycle of an experiment: from setup, through run time, to retrospective analysis. TopHat does so in a new way, by drawing upon excellent, proven third party services, notably the DIMES and ETOMIC measurement infrastructures, for specialized measurements. TopHat has been developed as the active measurement component of PlanetLab Europe, the flagship testbed of the OneLab experimental facility. It is part of OneLab's larger effort to pioneer the federation of previously independent testbeds and measurement systems in order to provide a diverse global scale environment for Future Internet research.

1 Introduction

PlanetLab nodes are fully open to the internet, and this allows experimenters to deploy applications such as novel overlays, peer-to-peer systems, content distribution networks, and the like. The collection of topology information is of particular interest to experimenters because the testbed consists only of the nodes at the edges of the network, not the underlying network. The ability to expose these applications to real-world network conditions is one of the prime motivating factors that leads experimenters to use PlanetLab, rather than a simulation or emulation environment. However, it also means that experimenters require information about the network topology in order to guide their experiments and make sense of their results.

There are many tools to measure the interesting properties of network paths. These properties range from the IP topology, which can be obtained thanks to the popular traceroute tool, to the available bandwidth between two nodes, for which there exists several possible tools. A review by the MOME project provides more details on available tools [1].

TopHat proposes an alternative to the deployment and use of such tools independently by each user by providing a topology monitoring service for applications running on the PlanetLab testbed. TopHat's originality in this regard lies in its support of the entire lifecycle of an experiment. During the setup phase,

T. Magedanz et al. (Eds.): TridentCom 2010, LNICST 46, pp. 542–557, 2011.

it assists PlanetLab users in choosing the nodes on which the experiment will be deployed, allowing them to base decisions on measured characteristics of the network as seen from each node. At run time, it provides live information to support adaptive applications and experiment control, providing measurements via a simple query interface and through the use of callbacks. The measurement data collected by the system are archived, and are thus available for retrospective analysis of an experiment, as well as being available for the community at large. Sec. 3 of this paper gives further insight into how TopHat supports users.

Following this description of the service that TopHat provides, Sec. 4 gives an overview of the system's architecture. In particular, it presents the user interfaces, focusing on the web services API.

Another specificity of TopHat is that it draws upon third party services – notably the DIMES and ETOMIC measurement infrastructures – that have a proven track record of excellence in providing specialized measurements to the research community. TopHat tunnels information from these systems to its users transparently. This interconnection is an instance of the larger effort, pioneered by the OneLab experimental facility [2], to federate previously independent testbeds and measurement systems in order to provide a diverse global scale environment for Future Internet research. Sec. 5 presents the details.

TopHat gets its inspiration from a number of proposed and existing systems, which are described in Sec. 2, the related work section of this paper.

2 Related Work

Network measurement systems draw on two principal sources of information to learn about the network topology: BGP feeds, which describe how the IP address space is divided into routable prefixes and which provide coarse-grained routes, at the autonomous system (AS) level, to reach those prefixes; and active measurements, starting with the well-known traceroute tool, which learns about routes at the IP interface level.

Infrastructures making BGP information publicly available include Route-Views [3], Team Cymru [4], and pWhoIs [5]. The Route Views project, headquartered at the University of Oregon, provides much of the data that researchers use to study BGP routing tables and their dynamics. Route Views servers get their information by peering directly with BGP routers, typically at large ISPs. Team Cymru is a not-for-profit network security firm that provides an IP to AS number mapping service based on information collected from a large number of BGP feeds (including Route Views). The corporately-run pWhoIs service is similar to that offered by Team Cymru, with the specificity that it offers geographical information about ASes and IPs. TopHat currently sources its IP to AS translations from Team Cymru.

Two notable active measurement infrastructures offered as a public service are Scriptroute and perfSONAR. Scriptroute [6], from the Universities of Maryland and Washington, consists of a set of ready-to-use tools deployed on PlanetLab nodes that a user can access through a simple scripting language. It supports

queries for traceroute information and measurements of delay and available bandwidth. As Scriptroute is accessible to any internet user, it places an emphasis on measurement safety. TopHat support a similar set of measurement types, but sources them from third party services if those services can offer superior measurements. Measurement safety in TopHat is achieved with the PlanetLab model [7] of restricting users to those whose institutions have committed to an acceptable use policy, and by ensuring traceability of each measurement back to its originator; in exchange, there are no hard-coded limits on what can be measured.

perfSonar [8], a product of the national research and education network (NRen) community, is deployed in the NRen networks and provides uniform access to measurements to users in multiple administrative entities. It serves as an example to TopHat of a system that interconnects measurement systems. Whereas perfSonar offers a uniform set of tools from a set of peer entities, TopHat federates heterogeneous systems. The focus of perfSonar is on troubleshooting across network boundaries; TopHat's is on experiment support. In addition to performing standard active measurements, perfSonar has direct access to router information such as queue lengths and packet drop rates. Because of this unique source of measurements, we consider it a prime candidate for future TopHat interconnection.

TopHat draws upon the Dimes [9] infrastructure for its large number of measurement vantage points. Dimes consists of thousands of software agents hosted by volunteers under the coordination of researchers at Tel-Aviv University. There are also agents on PlanetLab. The Dimes web service interface provides access to IP level traces and AS information. The Ono project [10] marshals a much larger number of agents: there have been more than 650,000 downloads of its plugin for the popular Vuze BitTorrent client. These agents conduct traceroutes between clients in order to support better peer selection. However, the sensitive nature of measurements conducted by individuals' peer-to-peer clients means that the Ono data would not be freely available to TopHat users.

Whenever possible, TopHat conducts delay and available bandwidth measurements from Etomic boxes. Etomic [11] is an infrastructure consisting of GPS-synchronized servers equipped with measurement cards that are capable of measuring delays to a precision of tens of nanoseconds. There are a few dozen Etomic boxes, many of them collocated with PlanetLab nodes. Since these boxes must be reserved, another platform has been deployed alongside Etomic to allow on-demand measurements. This platform is called Sonoma [12], and it offers medium-precision resolution (tens of microseconds) measurements through a webservice interface. TopHat interconnects with Sonoma as well. The Ripe TTM infrastructure [13] also provides GPS-synchronized measurement boxes. Interconnection with TTM would be of interest to TopHat because there are a few hundred of these, located primarily with network operators (including commercial operators), and providing regular measurements in a full mesh between boxes. As opposed to Etomic, though, TTM does not provide on-demand measurements or the same degree of precision.

In designing TopHat, we have been inspired by the Nakao et al. paper [14] that argues for the deployment of a topology information service within a testbed, aimed at providing users with a variety of measurements through a common API. The basic service can then be used to offer higher-level functionalities. One interest of such a service is that aggregation of requests allows for measurement reuse, thus reducing the strain on the network. Also, the user can benefit from best-of-breed tools and measurements, leaving him to focus on developing his overlay application. iPlane [15] is an infrastructure that implement this sort of service. Run by researchers at the University of Washington, iPlane provides overlays with predictions of network path characteristics such as delays, loss rates, and available bandwidth. The focus is on making predictions concerning paths between endpoints for which direct measurements are not possible or would be costly to obtain. It follows in the line of other predictive services such as IDMaps and Vivaldi (see the iPlane paper for further references). iPlane makes use of its own agents on PlanetLab nodes and within BitTorrent clients, as well as connecting to public traceroute servers. TopHat's federation with external measurement infrastructures can be seen as an extension of iPlane's drawing upon external traceroute servers. As a service for PlanetLab users, TopHat does not face the same necessity for measurement prediction as does a universal measurement service such as iPlane; in most cases, there are TopHat dedicated agents available to directly perform measurements from the PlanetLab nodes. TopHat could benefit, though, by adding predictions in circumstances where on-demand measurements are not possible.

TopHat also shares characteristics with Atmen [16]. Atmen reduces measurement overhead through reuse of measurements taken from similar vantage points or at points in time close to those that have been requested. The system supports a mechanism to trigger alarms (which in turn can start up active measurements) when it detects topological changes. TopHat's callback mechanism operates on a similar principle. Atmen is not available to the research community at large.

TopHat provides historical measurement data through the Network Measurement Virtual Observatory [17]. The best-used source of historical data comes from the CAIDA center. CAIDA's Archipelago, or Ark, measurement infrastructure [18] (the successor of the well-known Skitter), consists of a few dozen monitors worldwide. Ark aims for regular, comprehensive measurements to all /24 network prefixes, whereas TopHat provides measurements focused on testbed users' demands.

3 An Infrastructure in Support of Testbed Applications

This section presents the topology information services that TopHat offers to support PlanetLab applications, from setup through completion. It also discusses how usage monitoring in TopHat might help us gain a better understanding of users' needs and how this could provide insight into how the platform should evolve.

3.1 Supporting Experiments from Setup through Completion

TopHat offers its users four broad services that follow the experiment lifecycle:

Setup. A large part of the interest of deploying an application on PlanetLab is to expose it to a diversity of network locations and conditions. Examples of characteristics that a researcher might seek include: locations in Europe, Asia, and North America; nodes that are far from each other in terms of traceroute hops, AS path, or delay; nodes that are collocated with high-precision measurement boxes; particularly stable routes between nodes; paths that have load balancing routers; or a range of available bandwidths. The core PlanetLab services do not aid researchers in choosing their nodes on such bases, leaving it to a service such as TopHat.

Live. The underlying topology between PlanetLab nodes, and characteristics of that topology, will typically evolve during an experiment, due to network anomalies, such as path failures, the emergence of bottlenecks, and other sources of network dynamism. These changes are of interest for experiment control: for instance, a researcher might want to restart an experiment if certain paths have changed. They are also of interest for the applications themselves. A peer-to-peer application might adapt its overlay as a function of changing delays and available bandwidth in the underlay. TopHat offers measurements on demand. Also, to avoid the need for polling, TopHat offers a callback service. Sec. 3.2 provides more details.

Rewind. The service we brand "rewind" offers a researcher access to measurements related to an experiment once it is finished. He can use these data to understand application performance. A user will typically repeat the same experiment several times while varying some control parameters. The retrospective data can help him tease apart the effects that are due to changes in the parameters from those that are due to evolutions of the network topology. The data could also serve as inputs to a simulation, allowing the changes to be replayed while further parametric variations are explored.

Viz. This service consists of a collection of visualization tools that a researcher can use to obtain graphical representations of his experimental data.

3.2 User and Application Interfaces

To promote ease of use, TopHat presents the user with a simple measurement query interface. The user need not concern himself with cumbersome details if he does not wish to do so. For instance: he can simply ask for an available bandwidth measurement without specifying which tool TopHat should use; he can request a one-way delay measurement without himself synchronizing the measurement agents on two hosts; he does not need to parse the output of diverse tools, or handle the various error conditions that might arise. TopHat

provides an opportunity for the user to benefit from best-of-breed tools while only focusing on the core of his experiment.

The user can specify a class of measurement, such as traceroute, latency, available bandwidth, or topological distance at the IP or AS level, and TopHat itself will select the specific tool for that class. For example, when asked for a traceroute, TopHat would normally select our own team's Paris Traceroute tool [19] because of its ability to avoid many of the measurement artifacts that the standard traceroute tool encounters in the presence of load-balancing routers. The user does not need to know this, but he can learn it if he so wishes: by requesting the information, the user can obtain the name and the version of the tool that TopHat has selected. Furthermore, if the user does wish to request a particular tool, among the tools that are available, he is free to do so.

In addition to providing direct responses to user requests, TopHat allows requests to be registered for later reply. These can be either requests that require some time to fulfill, and therefore are more adapted to an asynchronous reply; requests for periodic updates; or requests for callbacks to be triggered by measurements and other events. These latter two forms of reply allow the user or his experiment to take actions in response to change. Such actions might include starting or stopping an experiment, readjusting an overlay topology, or triggering a set of measurements.

Examples of events that can trigger a callback are: a routing change as measured by traceroute, a delay increase of more than 20% along a given path, or the availability of a new available bandwidth measurement from the background measurement service. The callback information consists of the change that has occurred, a reference to the callback conditions that the change triggered, and a timestamp. Channels to inform the user of changes include: the XML-RPC interface, updates to in the user's space on the TopHat website, e-mail alerts, and RSS feeds. A user can browse the event history on the TopHat website.

3.3 Leveraging Historical Data

TopHat regularly conducts its own background measurements, compiling a general use archive. It can provide data to those interested in the long term evolution of network topology, or to those who want to look back to a specific point in time, for example to see what happened during a network failure or an attack. These measurements are also available to serve users' requests.

A user may specify a time frame when requesting a measurement, indicating the period over which he considers the measurement to be valid. He might be interested in a very recent measurement (no older than one minute, for instance) or be more flexible (if, say, a measurement from any time during the past day would do). If TopHat has an archived measurement that corresponds to the requested time frame, it can serve the user with that measurement, avoiding the need to launch additional probe traffic. In cases where measurements require significant time to carry out, serving the result from the archive provides the user with a faster response.

In addition to being able to specify a time frame for measurement validity, the user can specify a time interval for measurement aggregation. For instance, a user might be interested in collecting a set of traceroutes between a source and a destination. If the motivation is to understand the current state of the network, he might want the past few hours' traceroutes. On the other hand, if the motivation is to uncover rarely seen alternate links between routers, he might want information from several weeks of measurements.

Similarly, a user can request summary information from aggregated data. Requests might include: an average value, its variance, or the most frequently seen values. Such information can be used for node selection in the setup phase of an experiment: is some path characteristic between two nodes, such as delay or available bandwidth, stable or not? If the experiment is to be a short one, perhaps only the current state of the network is of interest. If, on the other hand, a user plans to deploy a long term service then he might wish to select nodes based upon measurements that indicate historic stability.

3.4 Understanding User Requirements

Although TopHat's main objective is to provide a service for users and the applications that they run on PlanetLab, we currently have no direct information about how people use the testbed. Our understanding comes to us through experiment descriptions in the literature and from what hints we can extract from traffic that originates from testbed nodes (the proportion of traceroute traffic, etc.) We therefore instrument TopHat to give us a better picture of users' measurement needs.

Logging of TopHat usage helps us to determine which features users exploit regularly and which ones either do not interest them or are too complicated or inaccessible to be much used. We can shape our future design of the system accordingly. The logs are also important for us to report to our sponsors, to indicate the extent to which the system that they paid for is in fact being used, and to what ends. And the usage logs are of interest to those who study testbed usage in general. Finally, usage logs help us to monitor the service and debug any problems, as well as engineer the system over the long term.

Beyond automated usage monitoring, we plan to work closely with application developers to understand their needs and to integrate the features that they find most useful.[1]

4 Description of the TopHat Measurement Infrastructure

4.1 Architecture Overview

Fig. 1 presents a global overview of the TopHat architecture, divided into functional blocks. The black boxes represent tasks run by the system, and the arrows indicate the flow of data through the system. Users and applications are at the

[1] We welcome inquiries.

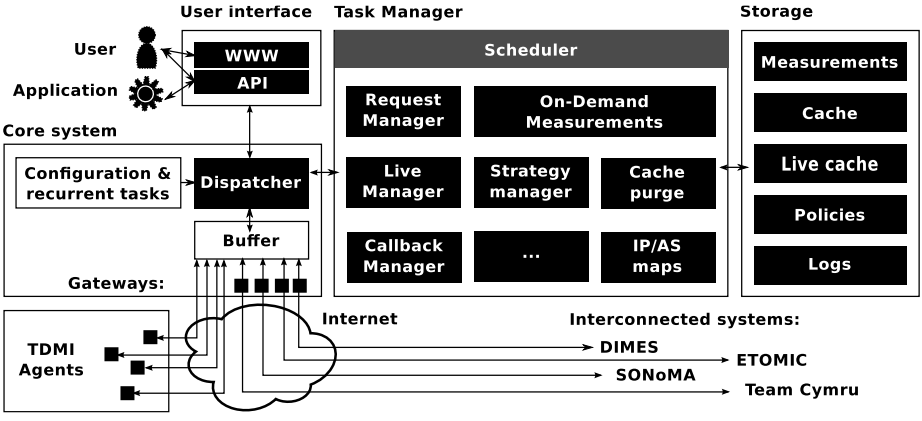


Fig. 1. TopHat architecture

top left. Users access TopHat either through the web interface or, via the XML-RPC API, at the command line. Applications use the XML-RPC API. The measurement infrastructures that conduct the measurements are at the bottom. On the bottom left are TopHat's own measurement agents, which are deployed within a slice on PlanetLab nodes. This is the TopHat Dedicated Measurement Infrastructure (TDMI), supplying measurements when no other system can do so. Other measurement infrastructures are on the bottom right. These are interconnected to TopHat via gateways. Mediating between the user and application requests and the measurement infrastructures is the core system, at center left. The core system dispatches requests and measurements to the task manager, top center. Data are stored in the storage subsystem, top right.

Origin of Measurements. The measurements originate either from TDMI or, via gateways, from the interconnected measurement systems. TopHat's dedicated agents are modular daemons that consist of wrappers around common measurement tools and basic services, like file upload and packet forging. The tools are invoked by dynamically loadable modules that perform tasks ranging from periodically starting a set of measurements to providing an XML-RPC interface to the agents. TDMI agents incorporate some improved measurement mechanisms that authors of this paper have helped develop, including Paris Traceroute [19], which more accurately measures internet paths that contain load balancing routers, and Doubletree [20], an algorithm to enhance the efficiency of a distributed probing infrastructure.

Gateways are specialized versions of the agents just described. They authenticate themselves to the external measurement infrastructures and ensure the exchange of data with TopHat. Gateways translate the requests originating from TopHat into platform-specific requests and wrap the results in a format that TopHat can understand, appending metadata to the measurements.

The Flow of Data through the System. Data files are uploaded to a buffer in the core system which, upon reception, associates each one with a task for further processing. The tasks are modular entities that can dynamically be added to the task manager to add new functionalities. Two classes of task are scheduled according to two levels of priority, depending on whether they are part of the live flow of information or whether they can be delayed to some extent. High priority tasks are those that are mandatory to ensure interactive communication with the user: measurement requests, updates from an agent that might trigger a callback, etc. Low priority tasks are generally those regularly created by the server as part of background activities such as ongoing IP and AS-level mapping, updates to the dataset that informs the probing strategy, cache purges, etc.

Both the core system and the agents maintain caches adequate to avoid the transmission and processing of redundant information, and to support high rates of measurement data transfer. Thus, the task manager only asks the agents for a new on-demand measurement when it is unable to fulfill the request from its own cache. Similarly, a traceroute measurement that has not changed since the previous measurement won't trigger a new database entry, but simply an update (first in the cache, then when the information is synchronized to the database). This extends the notion of measurement reuse by accounting for user needs. A user can obtain a measurement more quickly if his request specifies a time interval tolerance sufficient to serve the response out of the cache. The mechanism could be adapted in the future to serve predictions, such as those proposed by iPlane [15] and Atmen [16]. While the system does not limit measurements to avoid abuse, as Scriptroute [6] does (in TopHat, this is handled by the acceptable use policy and the ability to trace measurements back to users), policies are implemented to protect the system itself from being overwhelmed by measurement demands.

Data Storage The TopHat database records measurements as well as metadata such as agent system logs. Certain types of measurements, such as traceroutes, typically don't change much, if at all, from one measurement to the next. TopHat reduces the load on its database in such circumstances by avoiding rewrite of the entire measurement and only writing the differences and new timestamps. TopHat also employs caches at the agents, for robustness, and along the paths from agent to database. Caching is important in our architecture as it allows quick responses to the most frequent requests, the storage of snapshots representing the state of measurements over a given time interval, and, when a set of data is going to be subject to calculation, ready access to that set. TopHat also stores information such as its own system logs, the dataset of IPs it is currently probing, and policy-related information such as blacklists, rate limits, etc.

Scalability Considerations TopHat uses a centralized server architecture much like other monitoring systems, such as CoMon [21], that are currently deployed on PlanetLab. It does so for the same reason: the architecture is simpler and therefore more robust. System logs allow us to uncover bottlenecks, and

none have been insurmountable so far. We will continue to study the system's scalability as it grows. TDMI agents conduct measurements in a full mesh, which increases server load as the square of the number of agents. As the number of agents increases, we will look to improved algorithms for avoiding measurement redundancy, along the lines of Doubletree [20], to improve the system's scaling characteristics.

4.2 User Interface and API

TopHat provides two main interfaces: an XML-RPC API that allows an application, or a sophisticated experimenter (using a command-line interface) to interrogate the system, and a web interface that provides greater ergonomy for many of an experimenter's tasks. Typically, the web interface is more convenient during experiment setup for such tasks as node selection, while the API will be used by the application to perform measurements when it is running, or to react to changes in the underlying network.

The core API is the set of functions made available to the user, and that allow him to benefit from the functionalities presented in Sec. 3. The most important functions are:

Get allows the user to request information and measurements about nodes and paths in the monitored topology. A `Get` can be a standard request, an asynchronous request, or a request for periodic updates, as described in Sec. 3.2. In the latter two cases, the system responds via a callback.

Filter is a convenience function that filters a set of nodes or paths according to specified criteria. For instance, suppose the experiment requires twenty nodes that are each at least ten IP hops away from all of the others. The user can request a long list of nodes via the `Get` function and then pass that list to the `Filter` function along with the conditions on path length and number of nodes.

SetCallback is used to configure conditions on which the system will react by triggering a callback function, as described in Sec. 3.2. The API also features a set of related functions to help the user manage his list of callbacks (list, deletion, etc.)

An full description of the up-to-date API is available on TopHat's website.[2] This paper restricts itself to illustrating the `Get` function.

4.3 Requesting a Measurement

The prototype of the `Get` function is as follows:

```
RET = Get(Auth, Method, Timestamp, Input, Output, Callback)
```

[2] `http://www.top-hat.info/`

The Parameters of the Request. The first parameter, `Auth`, is an authentication token similar to the one used to authenticate with PlanetLab [22]. Authentication can be password based or key based. We are currently working on a common authentication mechanism for all OneLab platforms, and the use of this parameter will be updated accordingly.

`Method` describes the type of information or measurement we are requesting. A simple request is generic, using a keyword from a high level taxonomy (e.g., *traceroute* or *nodeinfo*). A more sophisticated request asks for a specific measurement tool.

The `Timestamp` parameter specifies the time that the request refers to. This can be a simple textual description (e.g., *now*, *latest*, or *today*), a UNIX timestamp (to get the closest measurement), or an interval. In the case of an interval, the user can ask for a variety of information, such as the first or last measurement in the interval, a list of measurements, or an average value.

`Input` specifies the object or objects to be measured, such as a path, or a set of paths, a node, or a set of nodes. The allowed values depend upon `Method`. The standard way to specify a node is to give its hostname or its IP address.

Each method returns a set of fields that are particular to that method. For example, the *traceroute* method returns the source and destination IP addresses (*src_ip* and *dst_ip*); a list of entries for each hop, consisting of the hop number (*hops.ttl*), the IP address (*hops.ip*), and the DNS name (*hops.hostname*) of the node; as well as additional information such as the presence of load balancing on the path, a timestamp, the platform the measurement originates from, etc. The *nodeinfo* method returns the IP address and hostname of a node (*ip* and *hostname*); the autonomous system that it is part of (*asn* and *as_name*); the city in which it is located (*city*); a *precision* field indicating the type, if any, of high-precision measurement equipment at that location (thanks to collocation with an ETOMIC node, for example); etc. The user specifies which fields he wants to receive by providing a set of their names to the `Output` parameter.

Finally, the `Callback` parameter is used for asynchronous requests, which typically take some time to answer, or requests for periodic updates. The `Timestamp` specifies the desired frequency update. For the simplest requests, this parameter will go unused, as in the sample query below.

Sample Query. Fig. 2 illustrates a Python query. The request calls for traceroutes from two nodes. One of the nodes belongs to the TDMI platform, the other to SONOMA.

This sample query returns a list of associative arrays that each describe a traceroute with the requested fields: source and destination IP, then, for each hop, the TTL, the IP address, and the corresponding hostname, and finally the platform that performed the measurement. Additional fields such as *tool*, *version* and *timestamp* can be added to obtain further information about the measurements; for the first traceroute this would have given for instance: *tool*='Paris Traceroute' and *version*='0.92b'.

Note how supplementary information can communicate the provenance of the measurements, which is an important feature for an interconnected measurement

Query:

```
path_list = [('planet2.elte.hu', 'planetlab-europe-02.ipv6.lip6.fr'),
             ('ape.onelab.elte.hu', 'planetlab-europe-02.ipv6.lip6.fr')]
print TopHat.Get(auth, 'traceroute', 'now', path_list,
    ['src_ip', 'dst_ip', 'hops.ttl', 'hops.ip', 'hops.hostname', 'platform_name'])
```

Result:

```
[{'src_ip': '157.181.175.248',  'dst_ip': '132.227.62.19',
    'hops': [ {'ttl': '1', 'ip': '157.181.175.254',  'hostname': None},
              {'ttl': '2', 'ip': '157.181.126.45',  'hostname': 'taurus.taurus-leo.elte.hu'}, ...],
    'platform_name': 'TDMI'},
 {'src_ip': '157.181.175.247', 'dst_ip': '132.227.62.19',
    'hops': [ ... ],
    'platform_name': 'SONoMA'}
]
```

Fig. 2. Sample traceroute request dispatched to two platforms

system: a point elaborated upon in Sec. 5. This issue of provenance also arises when supplying inferred data to the user. For instance, when a set of IP aliases to a router has been inferred, the user might want a pointer to the technique and/or data source that was used. (This inferred information is also distinguished from raw measurements in the database.)

5 Interconnection

The example in the previous section shows how TopHat makes use of its connections with other platforms to satisfy measurement requests. This section elaborates on the systems with which TopHat is currently connected, DIMES, ETOMIC, SONOMA, and Team Cymru, which were briefly described in Sec. 2, explaining the motivations for this interconnection. It also sets forth the case for a future connection with perfSONAR. The section wraps up by describing ways in which we can generalize our approach to interconnection.

5.1 Infrastructures Connected to TopHat

DIMES [9] is notable for the large number of vantage points that it offers (1700 measurement agents were active on a recent day), and the fact that many of these vantage points are on people's home computers, providing diversity compared to the NREN environment in which most PlanetLab nodes are located. DIMES is able to freely share its data with other measurement infrastructures, which is not the case for other systems of this type, such as Ono [10]. The interest of having access to measurement agents located outside of the testbed stems from PlanetLab's openness to the internet as a whole. Experimental applications on PlanetLab make use of this openness. For instance, a content distribution network (CDN) deployed on PlanetLab might be designed to serve content to

a user via its nearest PlanetLab node, with distance calculated at the AS or IP level. Outside measurements can help determine these distances. Similarly, a network coordinate system can benefit from measurements taken from a large number of vantage points.

The ETOMIC [11] and SONOMA [12] systems are notable for the higher than ordinary precision that they bring to measurements. They are GPS-synchronized dedicated measurement boxes. ETOMIC boxes, which must be reserved, provide delay measurements with a precision of tens of nanoseconds. SONOMA boxes can run measurements concurrently with a precision of tens of microseconds. A couple of dozen PlanetLab sites currently house both ETOMIC and SONOMA boxes, which clearly makes them of interest to TopHat. The precision of these boxes is useful for calculating one-way delays and available bandwidth, and for geolocalization.

The Team Cymru IP to AS mapping service [4] is notable for offering information drawn from a large number of BGP feeds and making it easily accessible to be queried over the internet. By connecting with the Team Cymru service, TopHat avoids the need to receive and process these BGP feeds itself.

We are currently exploring the possibility of interconnecting TopHat with perfSONAR [8], which is notable for offering extensive measurements from privileged vantage points within the network. perfSONAR obtains measurements directly from routers and from agents located at network points of presence (POPs). Some of the information that it provides, such as router queue lengths and packet drop rates, is not normally available to outside researchers and can at best be inferred. The perfSONAR information is of particular interest to researchers on PlanetLab because communication between most PlanetLab nodes crosses NREN infrastructure that is instrumented by perfSONAR.

5.2 Generalizing Our Approach to Interconnection

Our work on TopHat takes place in the context of the larger effort to federate computer networking testbeds. Initiatives such as FIRE [23] in Europe and GENI [24] in the United States are pursuing the vision of a worldwide federation of testbeds that will allow experimentation with new networking technologies at a global scale. OneLab [2] pioneered this vision, starting its work on federation in September 2006. An effort to develop measurement infrastructures for these testbeds has been a part of OneLab since the beginning, and TopHat is one of the results.

Our approach with TopHat has been to build on the interconnection mechanisms that emerge from the work on testbeds, as TopHat exists to be at the service of these testbeds. We see a first example of this in the TopHat authentication mechanism. TopHat operates on PlanetLab, which as a result of OneLab is now a federation of testbeds, PlanetLab Europe having joined the original PlanetLab in the United States. The same mechanisms that allow users to authenticate themselves to the global PlanetLab system serve to authenticate them with TopHat.

We plan to extend this authentication mechanism to encompass the infrastructures with which TopHat interconnects. At present, each interconnection has its own particular authentication mechanism, encapsulated in our gateway architecture. However, some of these platforms, such as ETOMIC, are testbeds in their own right, with users who can log in and run experiments. The potential exists, with a common authentication mechanism, to allow users of these systems access to TopHat and the OneLab facility, just as TopHat now has access to these systems. Rakotoarivelo et al. [25] have underlined how researchers are ever more inclined to deploy their experiments across different testbeds on different administrative entities, or to repeat the same experiment on different testbeds.

Other aspects of interconnection can also be standardized. For instance, the language that a system uses to describe the resources that one system requires from another. Here too, we can borrow from work currently being done on testbed interconnection. Various proposals for generic resource specifications (RSpecs) are currently emerging from PlanetLab [26] and OMF [25], among others, and efforts are taking place, notably within GENI [27], to harmonize them.

Another area that can benefit from standardization is the way in which measurements are described. Work in the IP Performance Metrics Working Group at the IETF has led to an XML specification for traceroute information [28], for instance.

Finally, we believe it is important to standardize a system for usage accounting across systems. Since active network measurements are potentially disruptive, these systems can only function if there is accountability, meaning the ability to trace a measurement back to the user who requested it. Accounting is also valuable to system operators, to enable them to better understand who is using their systems and for what purposes, and plan system development accordingly.

6 Conclusion

This paper has presented TopHat, a topology information service for the PlanetLab testbed. TopHat is oriented specifically towards the support of experiments running on the testbed, from setup through completion. The service provides data about the underlying network that help a PlanetLab user to choose the nodes that will be part of his experimental overlay. In so doing, it enables users to exploit the geographical and topological and diversity that PlanetLab uniquely offers. The service also assists a running experiment by offering live measurements through a simple query language, and by allowing the user to define callbacks that will keep him informed of significant changes, such as a change in the topology, or increased delays along a path. Other aspects of the service include measurement archiving and data visualization.

TopHat aggregates measurements coming from several infrastructures, including DIMES, which has measurement agents at a large number of vantage points, and ETOMIC and SONOMA, which offer high precision measurements. The user benefits transparently from this range of capabilities, accessing them through a simple common interface with PlanetLab based authentication.

This interconnection of measurement systems is an instance of the more general issue of testbed federation. TopHat provides one model for handling common authentication of users, description of resources, and the exchange of control messages.

As new features emerge for TopHat, they are being deployed in PlanetLab Europe, the European arm of the global PlanetLab testbed, in preparation for roll-out worldwide. This is part of a larger development effort: PlanetLab Europe is the flagship testbed of the OneLab experimental facility, and TopHat is a component in MySlice, a more general experiment management facility for OneLab.

Acknowledgments

We thank our OneLab colleagues, and in particular those of the ETOMIC, DIMES, and MySlice teams, for their thoughtful comments on TopHat as it has evolved.

The research leading to these results has received funding from the European Community's Seventh Framework Programme (FP7/2007-2013) under grant agreement n°224263-OneLab2.

References

1. Schmoll, C., Quittek, J., Bulanza, A., Zander, S., Kundt, M., Boschi, E., Sliwinski, J.: State of interoperability. Deliverable D11, FP6 IST MOME project (2004)
2. The OneLab experimental facility, `http://www.onelab.eu/`
3. University of Oregon Route Views project, `http://www.routeviews.org/`
4. Team Cymru IP to ASN mapping service, `http://www.team-cymru.org/ervices/ip-to-asn.html`
5. The Prefix WhoIs Project, `http://pwhois.org/`
6. Spring, N., Wetherall, D., Anderson, T.: Scriptroute: A public internet measurement facility. In: Proc. USITS (2003)
7. Spring, N., Peterson, L., Bavier, A., Pai, V.: Using PlanetLab for network research: myths, realities, and best practices. SIGOPS Oper. Syst. Rev. 40(1), 17–24 (2006)
8. Hanemann, A., Boote, J.W., Boyd, E.L., Durand, J., Kudarimoti, L., Lapacz, R., Swany, D.M., Trocha, S., Zurawski, J.: PerfSONAR: A service oriented architecture for multi-domain network monitoring. In: Benatallah, B., Casati, F., Traverso, P. (eds.) ICSOC 2005. LNCS, vol. 3826, pp. 241–254. Springer, Heidelberg (2005)
9. Shavitt, Y., Shir, E.: DIMES: let the internet measure itself. ACM SIGCOMM Comput. Commun. Rev. 35(5), 71–74 (2005)
10. Chen, K., Choffnes, D., Potharaju, R., Chen, Y., Bustamante, F., Pei, D., Zhao, Y.: Where the Sidewalk Ends: Extending the internet AS graph using traceroutes from P2P users. In: Proc. ACM CoNEXT (2009)
11. Morato, D., Magana, E., Izal, M., Aracil, J., Naranjo, F., Astiz, F., Alonso, U., Csabai, I., Haga, P., Simon, G., Steger, J., Vattay, G.: The European Traffic Observatory Measurement Infraestructure (ETOMIC): A testbed for universal active and passive measurements. In: Proc. Tridentcom (2005)
12. SONoMA measurement infrastructure, `http://www.complex.elte.hu/sonoma/`

13. Georgatos, F., Gruber, F., Karrenberg, D., Santcroos, M., Susanj, A., Uijterwaal, H., Wilhelm, R.: Providing active measurement as a regular service for ISP's. In: Proc. Passive and Active Measurements Workshop, PAM (2001)
14. Nakao, A., Peterson, L., Bavier, A.: A routing underlay for overlay networks. In: Proc. ACM SIGCOMM (2003)
15. Madhyastha, V.H., Isdal, T., Piatek, M., Dixon, C., Anderson, T., Krishnamurthy, A., Venkataramani, A.: iPlane: An information plane for distributed services. In: Proc. OSDI (2006)
16. Krishnamurthy, B., Madhyastha, H.V., Spatscheck, O.: ATMEN: a triggered network measurement infrastructure. In: Proc. WWW (2005)
17. Matray, P., Csabai, I., Haga, P., Steger, J., Dobos, L., Vattay, G.: Building a prototype for network measurement virtual observatory. In: Proc. MineNet (2007)
18. Archipelago measurement infrastructure, `http://www.caida.org/projects/ark/`
19. Augustin, B., Cuvellier, X., Orgogozo, B., Viger, F., Friedman, T., Latapy, M., Magnien, C., Teixeira, R.: Avoiding traceroute anomalies with Paris traceroute. In: Proc. ACM IMC (2006)
20. Donnet, B., Raoult, P., Friedman, T., Crovella, M.: E_cient algorithms for large-scale topology discovery. In: Proc. ACM SIGMETRICS (2005)
21. Park, K., Pai, V.S.: CoMon: a mostly-scalable monitoring system for PlanetLab. SIGOPS Oper. Syst. Rev. 40(1), 65–74 (2006)
22. PlanetLab Central API Documentation: Authentication, `https://www.planet-lab.eu:443/db/doc/PLCAPI.php#Authentication`
23. FP7 Future Internet Research and Experimentation (FIRE) initiative, `http://cordis.europa.eu/fp7/ict/fire/`
24. NSF Global Environment for Network Innovations (GENI) initiative, `http://www.geni.net/`
25. Rakotoarivelo, T., Ott, M., Seskar, I., Jourjon, G.: OMF: a control and management framework for networking testbeds. In: Proc. SOSP Workshop on Real Overlays and Distributed Systems, ROADS (2009)
26. Peterson, L., Sevinc, S., Baker, S., Mack, T., Moran, R., Ahmed, F.: PlanetLab implementation of the Slice-Based Facility Architecture. Draft technical report (2009), `http://www.cs.princeton.edu/~llp/geniwrapper.pdf`
27. Faber, T., Ricci, R.: Resource description in GENI: Rspec model. In: Presentation given at the Second GENI Engineering Conference (March 2008)
28. Niccolini, S., Tartarelli, S., Quittek, J., Dietz, T., Swany, M.: Information model and XML data model for traceroute measurements. RFC 5338, IETF (December 2008)

TridentCom 2010

Posters

Multi-hop Wireless Network Emulation on StarBED

Lan Tien Nguyen[1], Razvan Beuran[1,2] and Yoichi Shinoda[1,2]

[1] Japan Advanced Institute of Science and Technology
1-1 Asahidai, Nomi, Ishikawa, 923-1292 Japan
lannt@jaist.ac.jp, shinoda@jaist.ac.jp
[2] National Institute of Information and Communications Technology
2-12 Asahidai, Nomi, Ishikawa, 923-1211 Japan
razvan@nict.go.jp

Abstract. In this paper, we present an architecture to emulate multi-hop wireless networks on StarBED, a wired-network testbed at Hokuriku Research Center of NICT, Japan. The architecture uses a distributed approach, and it can effectively emulate in real time the properties of WLAN contention-based media access mechanism.

Keywords: emulation, real-time, testbed, routing, wireless.

1 Introduction

Network emulation is an experimental technique that intends to bridge the gap between simulation and real-world experiments, and thus, it delivers a significant impact on the wireless research community. There are two approaches for emulation, one is the centralized approach, and the other is the distributed approach.

In the centralized approach, all the nodes connect to a central server and direct their traffic to the server. Then, the sever forwards the traffic to the destination according to the parameters which characterize the current state of the emulated network, i.e. reachability, link quality, collision etc.

Opposed to the centralized approach, in the distributed approach, all the nodes are mutually connected via wired or wireless media. The nodes themselves are responsible for directing and forwarding traffic. Since all the nodes are mutually connected, network topology is created by using logical connectivities which are computed from geographical information, radio parameters, and medium information in a distributed fashion.

Due to the bottleneck that can occur at the central server in the centralized approach, only distributed emulators are able to support real-time evaluation of topology-related protocols. In the following sections, we will focus on the design and implementation of a distributed architecture, named AEROMAN (Architecture to Evaluate Routing PrOtocols for Multi-hop Ad-hoc Networks) which allows us to emulate multi-hop wireless networks on a wired network testbed, StarBED [2].

T. Magedanz et al. (Eds.): TridentCom 2010, LNICST 46, pp. 561–563, 2011.

2 AEROMAN

2.1 How to Emulate a Wireless Link

In order to emulate wireless links, we use Dummynet [3], a link emulation tool designed for FreeBSD. It works by intercepting packets on their way in the protocol stack, and passing them through its pipes, which simulate the effects of bandwidth limitation, propagation delay, and packet loss. These pipes can be either at the sending side or the receiving side.

In AEROMAN, pipes for unicast traffic are located at the sending side, while pipes for broadcast traffic are located at the receiving side. The reason for this is that dummynet classifies packets based on their IP addresses, and thus, in order to emulate multi-hop wireless networks, the incoming/outgoing link of a packet has to be determined from the addresses in its IP header. For unicast traffic, a sending node can easily find the link it will send a packet on by looking at routing table for the next-hop node. For broadcast traffic, it is impossible to locate pipes at sending side since links cannot be identified by using broadcast IP address. However, at the receiving side, a node can find out the source node of a broadcast packet by looking at the source IP address, and hence it knows the link through which the packet has passed. Being aware of the links used to forward packets, a node can direct a packet to the appropriate pipe configured with parameters (bandwidth, delay, packet loss rate) equivalent to parameters of the link through which the packet has traveled (broadcast) or will travel (unicast).

2.2 Design of AEROMAN

AEROMAN uses a two-stage approach to emulate multi-hop wireless networks. In the first stage, parameters of all wireless links are computed in contention-free conditions by using deltaQ library of QOMET [1]. This information is distributed to all the experimental nodes before the experiment. In the second stage, AEROMAN use the Adaptive Traffic (AT) model to adjust these parameters in a contention-aware fashion. Figure 1 shows AEROMAN node internals which includes six modules:

- *Pipes Controller*: Applying appropriate links parameters, which are generated by Adaptive Real-time Parameter Generator, to dummynet pipes.
- *Multicasting Module*: Exchanging traffic information between experimental nodes.
- *Local Node Real-time Traffic Collector*: Collecting traffic information of the current node.
- *Remote Nodes Real-time Traffic Collector*: Collecting traffic information of other experimental nodes.
- *Routing Support Module*: Identifying the link which will be used to forward a given IP packet that goes through the current node. Parameters of this link will be used for configuring the pipe which handles the packet.
- *Adaptive Real-time Parameters Generator*: Using Adaptive Traffic Model to adjust contention-free links parameters based on real-time traffic information of the wireless channel.

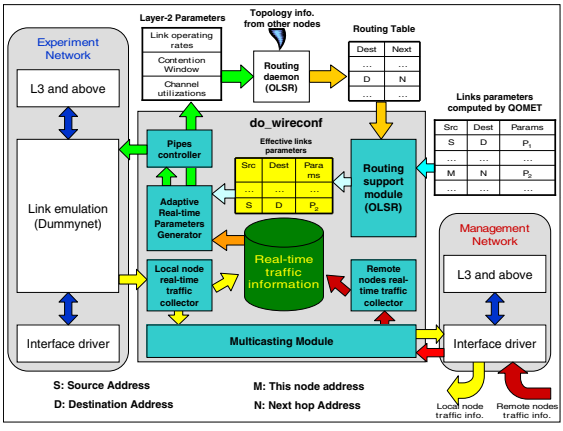


Fig. 1. AEROMAN Node Internals

2.3 Adaptive Traffic Model

This model is used by the Adaptive Real-time Traffic Generator module to compute link parameters on the fly from both Effective Link Parameters and Real-time Traffic Information. Firstly, Frame Error Rate (FER) is recomputed based on current channel utilization. Secondly, both packet delay and bandwidth limitations are adjusted based on the value of FER computed in the previous step.

3 Conclusions

In this paper, we present the design and implementation of AEROMAN as an architecture for evaluating routing protocols for multi-hop wireless networks. AEROMAN follows a distributed approach, while still effectively emulating properties of the wireless environment, such as bandwidth limitation and shared media. Due to the lack of space, no experiment results are shown in this paper; however, such results will be displayed in the poster.

References

1. Beuran, R., Nguyen, L.T., Latt, K.T., Nakata, J., Shinoda, Y.: QOMET: A Versatile WLAN Emulator. In: Proceedings of the 21st International Conference on Advanced Information Networking and Applications, Niagara Falls, Ontario, Canada, pp. 348–353 (2007)
2. Miyachi, T., Chinen, K., Shinoda, Y.: StarBED and SpringOS: Large-scale General Purpose Network Testbed and Supporting Software. In: Proceedings of the 1st International Conference on Performance Evaluation Methodologies and Tools, page 30, ACM Press, New York (2006)
3. Rizzo, L.: Dummynet FreeBSD Network Emulator

The LambdaCat Open Testbed Facility: An Innovation Facility to Test Multi-layer Devices on a Real Environment

Carlos Bock, Sergi Figuerola, and Víctor Jiménez

i2cat Foundation,
C/ Gran Capità 2-4, Nexus I building, 2nd floor, office 203
08034 Barcelona, Spain
{carlos.bock,sergi.figuerola,victor.jimenez}@i2cat.net

Abstract. This paper describes the LambdaCat experimental infrastructure, which provides an open facility to test and validate research, experimental, and pre-production equiment and sevices aligned with Future Internet technologies. The LambdaCat validation platform is composed by three transparent and colourless ROADM-based nodes, which are deployed in the Barcelona metropolitan area. The experimental infrastructure is virtualized in order to offer logical isolated substrates to enable simultaneous disruptive research experiments in productive environments without interfering to parallel research users. An IaaS (Infrastructure as a Service) model is adopted to be aligned with the user infrastructure needs. The implemented services allow end users to test new telecommunications equipment and services. The experimental services are accessible at different network layers (L1, L2 & L3) to test new technologies and protocols, from core to access networks.

Keywords: Future Internet Testbed, Optical Open Infraestructure.

1 Introduction

Internet traffic is constantly growing and network applications require more restrictive connection parameters to transmit properly their data. Thus, to allow an optimized way to test new devices and applications aligned with the Future Internet, open collaborative environments should be deployed, driving R&D&i to a Public and Private Partnerships (PPP) model.

Moreover, many organizations can not afford the deployment of their own testbeds. Therefore, most devices, applications and services are tested in environments far of a real production context. These impacts negatively on the robustness of novel network devices and sevices. Thus, offering an open testbed to the R+D+i sector in Catalonia increases the developments quality, which also impacts positively on the emergence of future technologies.

T. Magedanz et al. (Eds.): TridentCom 2010, LNICST 46, pp. 564–569, 2011.

In this paper, we present the architecture and topology of LambdaCat experimental infrastructure, describing the devices that compose the experimental facility.

Secondly, the different services offered within LambdaCat optical open testbed are described. This section contextualizes the experimental services at the different network layers.

Thirdly, three research and innovation projects experiences that have been developed using LambdaCat experimental services are listed.

Finally, the paper describes the main goals and innovation keys of the LambdaCat open testbed facility. The section tries to contextualize R+D+i impact into public and private organizations, focusing on the benefits of collaborative environments.

2 Architecture and Topology

The LambdaCat open testbed is composed by three optical reconfigurable 2.5/10G ROADM-based nodes, with add and drop capabilities. The three nodes are deployed in the Barcelona metropolitan area establishing a multi-wavelength fibre ring. The points of presence (PoP) of the optical testbed are strategically placed on industrial and academic locations. The main research centres, technical universities and industrial clusters in Catalonia are connected to LambdaCat facility to use, test and offer their experimental services. Additionally, to demonstrate services and applications, the LamdaCat testbed offers a pool of distributed virtual machines. Thus, the platform can be configured to perform ad hoc multivendor tests at different network layers.

To complement the experimental connectivity services, LambdaCat facility offers monitoring services to analyze the performance of the devices and services under test. Moreover, HD media content 10G services are offered, as well as an extension to access to a 4k demonstration laboratory to test advanced media services. Finally, LambdaCat open testbed has two more interesting research extensions: a FTTH industrial extension and a GRID/cloud computing extension.

2.1 Network Structure

The network structure is composed by three open optical nodes, which are accessible at different OSI layers. It means that the network devices, services and protocols can be tested at different layers depending on the user needs and devices under test.

The three ROADMs have 2.5/10G connections on bidirectional fibres to implement redundancy and resiliency. The following illustration shows the LambadaCat testbed structure.

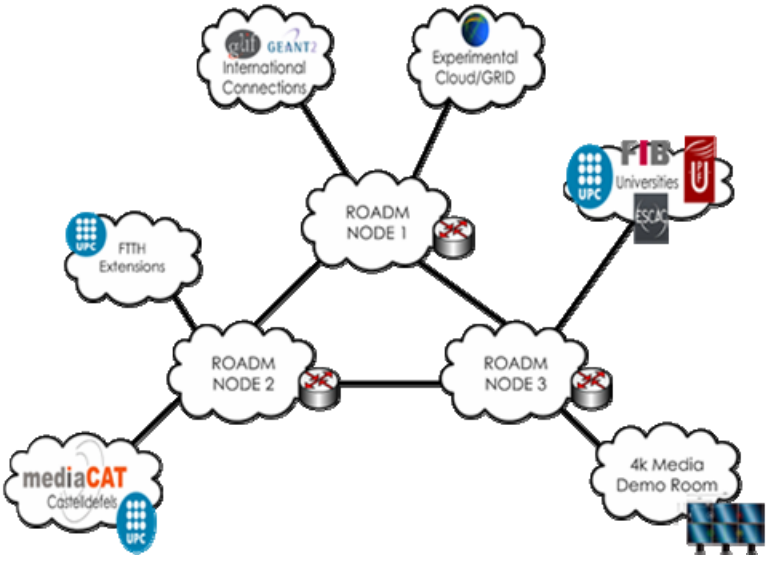


Fig. 1. LambdaCat extensions

Also, in each Point of Presence, a L2 switch, a L3 switch and a virtual machine server is located. These devices allow to the users to test devices, protocols and services.

The LambdaCat open testbed provides extensions to test technologies on more heterogeneous environments. These experimental extensions are described below.

2.2 Network Nodes

Each LambdaCat Point of Presence is composed by heterogeneous network equipment. Each node hosts multiple network equipment, from the optical to the service and application layer.

First, a ROADM is used to offer 2.5 and 10G connectivity at optical layer between LambdaCat PoPs. The ROADMs allow to the users to add and drop lambdas at each node. Second, each ROADM is connected to a layer 2 with the aim of offering different ports to access and test layer 1 and layer 2 services. Third, layer 2 switches are connected to a layer 3 switch to offer IP connectivity to test devices, protocols and services.

Finally, each layer-3 switch is connected to virtual machines servers to offer parallel instances to testing applications and services at higher OSI layers.

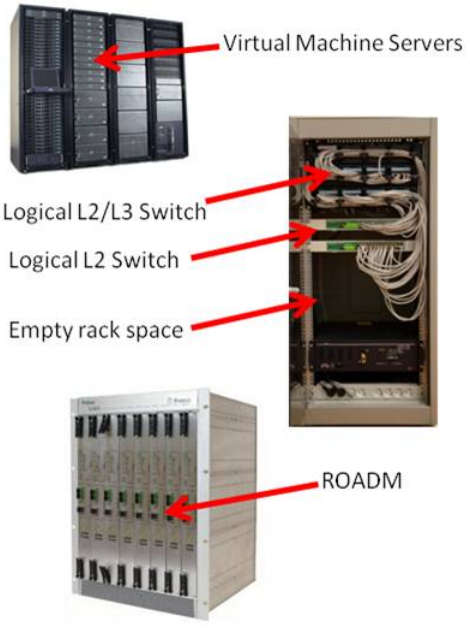


Fig. 2. PoP Topology

2.3 Network Extensions

Currently, LambdaCat open testbed has two FTTH extensions. The facility has connectivity to a FTHH laboratory placed in the Technical University of Catalonia in Castelldefels. This laboratory is aimed to provision experimental FTTH services to the university research community to research and develop new technologies and protocols.

Also, the LambdaCat open testbed provides another FTTH extension to test industrial and residential solutions in production environments. This FTTH extension offers two different platforms to test FTTH active and passive technologies, having connected a GPON OLT and an EPON one.

One of the LambdaCat PoPs is connected to a very high definition demo laboratory to offer 4k experimental media transmission services.

The LambdaCat facility has two international 10G connections, to offer international experimental services. These international connections will allow to connect the LanmbdaCat facility at other research locations and testbeds.

3 Experimental Services

In this section are described the different services offered for the different OSI layers. Each PoP has enough empty rack space to host simultaneous testing equipment. All the equipment that composes each PoP is virtualized to offer logical instances to the users and support parallel disruptive testings and experiments between them.

Applying IaaS techniques, the users are allowed to configure themselves their dedicated network resources without the intervention of a network manager.

Also, the LambadaCat experimental facility is connected to MediaCAT platform, where multiple media experimental services are offered. Currently, the MediaCAT media services are working in production environments, transmitting 2.5G and 10G streams with high definition media content between Barcelona and Castelldefels.

Over the LambdaCat testbed there are also monitoring and performance services to allow the users to analyze the behaviour of the technology that they are testing.

3.1 Layer 1 (L1) Services

LambdaCat experimental layer 1 services offer connections of 2.5 and 10G with add and drop capabilities on each optical reconfigurable ROADM node. The three ROADMs nodes are the outcomes from the Spanish R+D+i project DREAMS (PROFIT/CIDEM 2007) and an e-Infrastructure Catalan research project PAIS (InfoRegió 2009).

L1 experimental services are capable of establishing lightpaths between two or more network edges. This is done by using a network management tool, ARGIA [1]. ARGIA applies virtualization techniques to implement intelligency, flexibility and dynamic connections to the physical network. To test L1 equipment, the users will locate physically their devices on one of the PoP.

3.2 Layer 2 (L2) Services

To test L2 devices, the LambdaCat facility offers experimental connectivity services and empty port space in each PoP of the metropolitan ring. Thus, the experimental services allow to the users to manage their own connection.

3.3 Layer 3 (L3) Services

L3 services are characterized by offering IP connectivity streams between two or more network destinations. the LambdaCat experimental facility is capable of deliverying an IP network to the final users. This service allows configuring their own IP networks to the end users so they can configure their own IP network according to their test requirements.

3.4 Grid and Cloud Computing Services

The LambdaCat experimental grid computing services are mainly the outcome of PHOSPHORUS FP6-Project [2].

To offer cloud computing experimental services, the hosted servers on each node have eyeOS (Cloud computing operative service) installed and have access to the experimental cloud computing services.

End users can access these services through a remote connection to the LambadaCat experimental facility or by connecting or installing directly their developments on the experimental infrastructure.

3.5 Media Services

The LambdaCat open testbed is connected to the MediaCAT media platform, where users can access experimental media services through a connection to the facility. The platform is connected to a 4k visualization room to experiment with very high definition 4k media content services.

4 First Experimental Activities

LambdaCat open testbed facility has been used to test several R&D&i projects. In this section, four interesting research experiences are explained.

First, LambdaCat L1 connectivity services have been used to develop, deploy and debug the ARGIA management network tool. At present, ARGIA is used to manage the LambdaCat testbed, virtualizing L1 devices to offer dedicated experimental networks to the users.

The DREAMS project has used LambdaCat to develop and deploy the three ROADMs that nowadays compose the LambdaCat testbed.

Finally, the LamdaCat facility was used to deploy a GPON and an EPON OLT on UPC Castelldefels to test FTTH high-quality voice and novel business services.

5 Conclusions

LambdaCat experimental facility is aimed to offer an open experimental platform to improve the quality of network developments in Catalonia. This open experimental environment should promote and enhance public-private partnership of organizaions inside and outside Catalonia.

The LambadaCat experimental facility presents an ideal environment to researh on Future Internet technologies, architectures, protocols and services. One of the main innovation keys of LambdaCat is the capability of the end users to reconfigure their assigned resources according to their testing and research needs.

References

1. Argia, `http://www.inocybe.ca/frontpage/home`
2. Ferrao, L., et al.: Advance and Immediate Reservations of Virtualized Network Resources. In: Proceedings on Terena meeting (2008)

Polymorphic Ubiquitous Network Testbed RUBIQ

Junya Nakata, Razvan Beuran, Takashi Okada,
Ken-ichi Chinen, Yasuo Tan, and Yoichi Shinoda

Hokuriku Research Center,
National Institute of Information and Communications Technology
2-12 Asahidai, Nomi, Ishikawa Japan
{jnakata,razvan}@nict.go.jp
Internet Research Center,
Japan Advanced Institute of Science and Technology
1-1 Asahidai, Nomi, Ishikawa Japan
{tk-okada,k-chinen,ytan,shinoda}@jaist.ac.jp

Abstract. In this paper we present RUBIQ, a polymorphic ubiquitous network testbed. RUBIQ works on StarBED, which is a network testbed consisting of hundreds of PCs connected to each other. RUBIQ consists of a set of subcomponents such as RUNE and QOMET that make it possible to simulate ubiquitous network systems in huge scale that can hardly ever be experimented in the real world. We illustrate the structure of the testbed, how it functions, and the results of some of ubiquitous network system simulations. We show some results demonstrating that the testbed achieved an accurate simulation of a pedestrian tracking system by using appropriate modules, such as a wireless communication emulator and processor emulator.

Keywords: ubiquitous networks; wireless network; distributed testbed; supporting software; simulation; emulation.

1 RUBIQ

In this paper we present RUBIQ, a polymorphic ubiquitous network testbed implemented on StarBED, and a couple of its subcomponents such as RUNE and QOMET. RUBIQ enables to perform various kinds of simulation environments with the available simulation modules.

StarBED [1] is a network testbed which consists of over 1,000 PCs connected to each other. StarBED provides a simulation supporting software, SpringOS, to implement an easy-to-use simulation environment with which the users can write simulation scenarios that can be executed automatically.

In order to be able to use StarBED for simulation we developed a set of subcomponents, called RUBIQ. The major components of RUBIQ are the simulation support software RUNE (Real-time Ubiquitous Network Emulation environment) [2] and the wireless network emulator QOMET (Quality Observation and Mobility Experiment Tools) [3] described in the following part.

T. Magedanz et al. (Eds.): TridentCom 2010, LNICST 46, pp. 570–572, 2011.

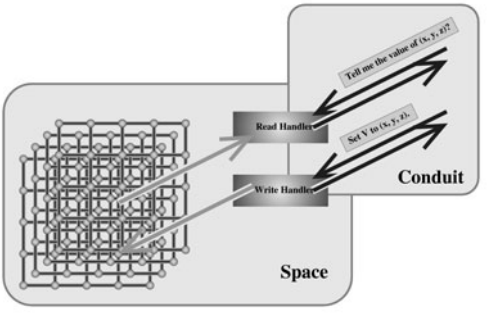


Fig. 1. Space and Conduit

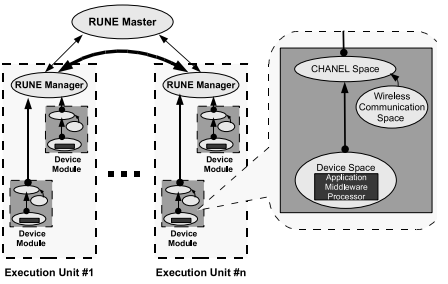


Fig. 2. RUBIQ based simulation

The simulation-support software RUNE is used to effectively run and manage the simulation. The basic elements of the logical structure of a RUNE-driven simulation are the *space* and *conduit* shown in Fig. 1. A *space* is an entity that behaves as one of the simulated elements. *Spaces* can simulate: (i) nodes, i.e., physical devices, (ii) environments, such as the thermal field; (iii) networks. *Spaces* are connected with each other by elements called *conduits*. Their role is to create an abstract error-free communication channel between two *spaces*.

One of the most important elements when using simulation for studying ubiquitous network systems is to be able to recreate with sufficient realism the communication. QOMET emulates several wireless communication technologies such as WLAN, ZigBee, Active tag by using a scenario-driven architecture. QOMET calculates a network quality degradation (ΔQ) description from a scenario representation. The ΔQ description represents the varying effects of the network.

RUBIQ also provides other supporting software modules belonging to different layers of emulation, such as processor emulation and middleware emulation. The major processor and middleware emulation modules provided by RUBIQ can emulate PIC 16F series processor, OpenRISC OR1200 processor, and ZigBee protocol stack. The users can choose those modules accordingly and combine them to implement the simulation they want to carry out as shown in Fig. 2.

2 Simulations Carried Out on RUBIQ

So far we took advantage of RUBIQ to evaluate a number of ubiquitous network systems such as an in-home sensing system, a motion planning robot system and an active tag based pedestrian tracking system [4] by leveraging hundreds of StarBED nodes.

As a representative example, we describe the simulation of the active tag based pedestrian tracking system shown in Fig 3. In the simulation, we started by reproducing a real-world 16 pedestrian experiment carried out with the prototype and eventually simulated the system with over one hundred pedestrians. In order to simulate the system, we utilized QOMET, a PIC 16F processor emulator and the firmware for the real system. The results showed a good agreement between the real-world experiment and the simulation. Moreover, we obtained

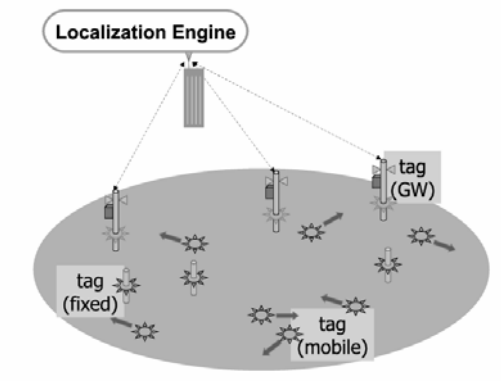


Fig. 3. Pedestrian tracking system

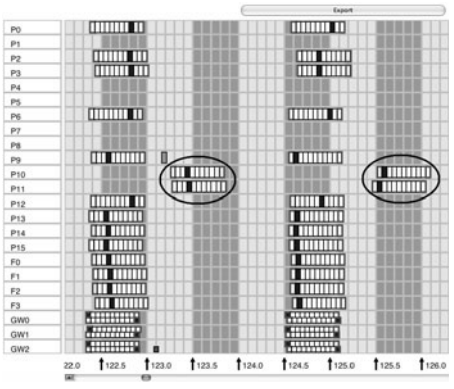


Fig. 4. Firmware issues

a significant achievement by finding some implementation issues. These issues, the quality of the random number generator and accuracy of the time synchronization protocol shown in Fig. 4, were fixed in the next version.

3 Conclusion

In this paper we presented RUBIQ, a polymorphic ubiquitous network testbed, and showed some results obtained by executing simulations. RUBIQ allows the accurate simulations of ubiquitous network systems by using its subcomponents.

So far we simulated a number of ubiquitous network systems with up to a few hundred entities, and the scale of simulations could be extended up to thousands by fully using the abilities of StarBED.

References

1. The StarBED Project, `http://www.starbed.org/`
2. Nakata, J., Miyachi, T., Beuran, R., Chinen, K., Uda, S., Masui, K., Tan, Y., Shinoda, Y.: Starbed2: Large-scale, realistic and real-time testbed for ubiquitous networks. In: The 3rd International Conference on Testbeds and Research Infrastructures for the Development of Networks and Communities (TridentCom 2007), Orlando, Florida, U.S.A (2007)
3. Beuran, R., Nguyen, L.T., Latt, K.T., Nakata, J., Shinoda, Y.: Qomet: A versatile wlan emulator. In: AINA 2007: Proceedings of the 21st International Conference on Advanced Networking and Applications, pp. 348–353. IEEE Computer Society Press, Washington (2007)
4. Nakata, J., Beuran, R., Kawakami, T., Chinen, K., Tan, Y., Shinoda, Y.: Distributed emulator for a pedestrian tracking system using active tags. In: UBICOMM 2008: Proceedings of the The Second International Conference on Mobile Ubiquitous Computing, Systems, Services and Technologies, Valencia, Spain, pp. 219–224. IEEE Computer Society, Los Alamitos (2008)

IMS IPTV: An Experimental Approach

Nguyen Tai Hung, Nguyen Huu Thanh, Nguyen Giang Nam,
Tran Ngoc Lan, and Dinh Thai Hoang

Department of Communication Engineering- Hanoi University of Technology
hungtai-fet@mail.hut.edu.vn

Abstract. IMS has been widely recognized as the control and signaling framework for delivering of the rich communication & multimedia services to broadband users. Amongst others, it's deploying as the service (middleware) platform for interactive and personalized IPTV services. The goal of this paper is to provide a short description and analysis of the (IPTV) use cases that have been selected for design and implementation at Hanoi University of Technology (HUT) in scope of its initiatives for NGN researching program. Major use cases, or we called intelligent features, are the advanced electronic service guide, video on demand (VoD), (IPTV) session continuity, and parental control. Development results for each of the use case are depicted.

Keywords: IMS; IMS IPTV; enhanced EPG; Parental Control; Blending Service, Intelligent Features.

1 Introduction to the IMS Based IPTV Testbed

In the scope of a joint-research program between Hanoi University of Technology and Fraunhofer Institute of FOKUS, we was setting up a next generation Test-bed in our lab for purpose of prototyping of new multimedia and rich-feature communication services using IMS framework. The test-bed consists of all three layers: media layer for transportation of media traffic in the modes of unicast, multicast and broadcast. The core layer of signaling and session/service control, that uses the FOKUS' open source IMS Core [3][4][5], consisting of CSCF servers and a light user profile database (HSS). Our project main focus is on the application layer in which we specified and developed prototypes for value added services to IP Telephony and IP Television using the open source platform (Sailfin). A Media Server was also developed at our lab using VLC (VideoLAN) media stack. Besides that we had developed a comprehensive framework and prototype of IMS IPTV Client that based on the open source IMS Communicator. Finally, several IMS interfaces, namely, Sh, Mw, etc are implemented on our own effort. Figure 2 depicts high level view of our Test-bed setup.

T. Magedanz et al. (Eds.): TridentCom 2010, LNICST 46, pp. 573–576, 2011.

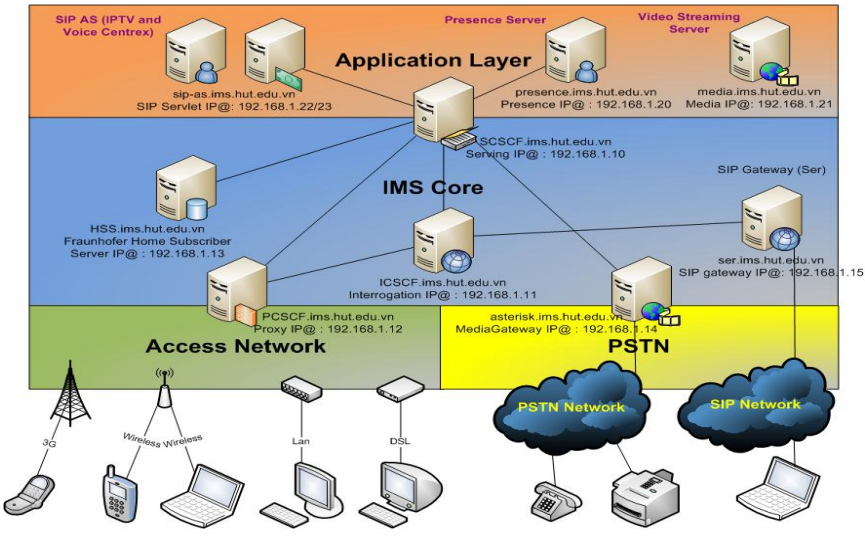


Fig. 1. The HUT's Next Generation Test-bed

2 IMS IPTV Use Cases

In our research, we focus on design and implementation of special use cases for IPTV services that utilize the strength of IMS framework, like User Interaction, Service Authorization and Authentication and Session Blending. In particular, we considered and implemented following use cases (we call Intelligent Features): *Standard Video on Demand, Parental Control, Enhanced EPG and Session Continuity.*

Standard Video on Demand
We considered a scenario in which an IMS user initiates a call to a specific content (a movie, a song or other resources) at the content provider through an IMS domain. The request would be routing through different SIP servers (CSCFs) and at S-CSCF a suitable Trigger(s) would be invoked to forward the request to IPTV AS, the AS will then proxy the request to MRF (Media Server). Media Server, after accepting the request, will send back the successful responses (via AS) as well as the RTP streams directly to Emulated STB.

Parental Control
A special feature, called Parental Control, had been designed and implemented which allows the parent to control their child from requesting and viewing classified contents. The control could be provided based on the registered (IMS) identity or the time slot of requesting.

Enhanced EPG
Personalization is a key feature in the IMS IPTV solution. In this sense, we have complemented the user profile with a new XML-formatted [9] service profile for each

IMS identity to contain the personalized information. That leads to another intelligent feature for IMS-based IPTV, we called Enhanced EPG. With this feature viewer will be classified in to different groups (via subscription) with different service levels and will receive the different channel list from the portal. The user also will receive the different channel list when requests at different time slots.

Session Continuity

The issue of session continuity is also studied in our test-bed, in which we investigated a new approach [10] that allows handing over of an on-going IPTV session between different access heterogeneous environments. We propose a new component in the IMS domain, namely an proxy based on mSCTP (mobile Stream Control Transmission Protocol) that acts as an anchor point for soft vertical handover of mobile nodes, which have multiple physical interfaces (e.g., WLAN/UMTS). The mSCTP-based proxy also supports QoS provisioning and adaptation for the mobile nodes when moving in a heterogeneous wireless environment. Our simulation results show that the signaling cost for handover in our approach can be up to 23 times smaller than that in the conventional approach.

Example Result

Figure 2 illustrates a personalized user portal that provides a different content metadata (channel list) to registered user from different groups.

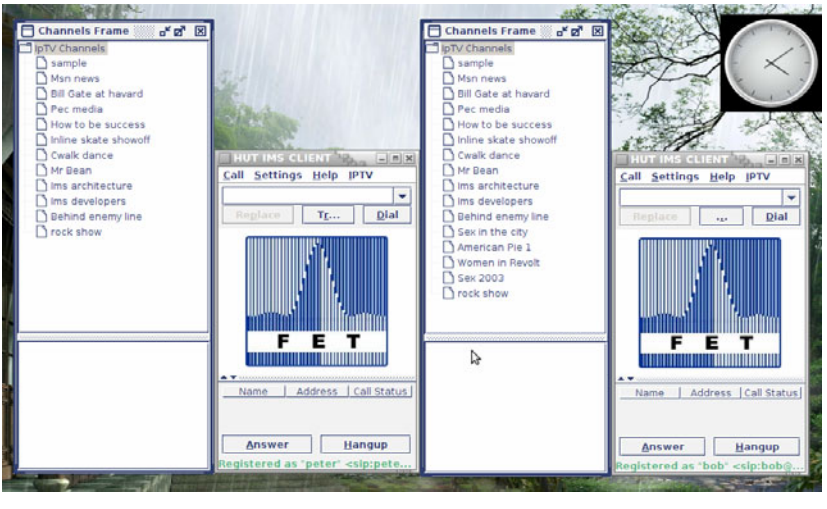


Fig. 2. Channel List for different User Categories of Enhanced EPG feature

3 Conclusions and Further Discussions

This paper presents our investigation, in experimental perspective, of using IMS framework to provide intelligent features for IPTV services. In particular, it focuses on the

video on demand, remote parental control, blending services, session handover without interruption and other interactive features which some of the use cases were discussed and presented here above. It shows how SIP [2][7] signaling and IMS can be used to provide the Interactive and Blending features for the entertainment video services.

The initial results promise the great potential of those IMS-based TV interactive and differentiated features, which offer attractive and rich multimedia experiences to the end user. We are currently investigating and developing several other intelligent features of IMS-based TV, namely, the context-based session continuity that allows to seamlessly handover the IPTV sessions across different screens/terminals on different access networks.

References

[1] TS 23.328, IP Multimedia Subsystem, 3GPP, Release 6
[2] Rosenberg, J., et al.: SIP: Session Initiation Protocol. RFC 3261, IETF (June 2002)
[3] F. inc., XDMS: The FOKUS XML Document Management Server, http://www.fokus.fraunhofer.de/bereichsseiten/
[4] Open IMS Core Playfround, http://www.openimscore.org/
[5] SIP Express Router, http://www.iptel.org/ser
[6] UCT IMS Client,, http://uctimsclient.berlios.de
[7] Handley, M., Jacobson, V.: SDP: Session Description Protocol. RFC 2327, IETF, (April 1998)
[8] Service Capability Interaction Manager, http://en.wikipedia.org/wiki/Service_Capability_Interaction_Manager
[9] Rosenberg, J.: The Extensible Markup Language (XML) Configuration Access Protocol, http://tools.ietf.org/html/draft-ietf-simple-xcap-12
[10] Thanh, N.H., Hang, L.T., Yem, V.V., Thu, N.Q., Dung, N.X.: Multimedia Session Continuity with Context-Aware Capability in IMS-based Networks. In: Proceedings of IEEE Sixth International Symposium on Wireless Communication Systems 2009 (ISWCS 2009), Siena-Tuscany, Italy, September 7 – 10 (2009)

Real-World G-Lab: Integrating Wireless Sensor Networks with the Future Internet

Daniel Bimschas[1], Sándor Fekete[2], Stefan Fischer[1], Horst Hellbrück[3], Alexander Kröller[2], Richard Mietz[4], Max Pagel[2], Dennis Pfisterer[1], Kay Römer[4], and Torsten Teubler[3]

[1] Institute of Telematics, University of Lübeck
{bimschas,pfisterer,fischer}@itm.uni-luebeck.de
[2] Braunschweig University of Technology
{pagel,kroeller,s.fekete}@ibr.cs.tu-bs.de
[3] University of Applied Sciences Lübeck
{teubler,hellbrueck}@fh-luebeck.de
[4] Institute of Computer Engineering, University of Lübeck
{mietz,roemer}@iti.uni-luebeck.de

1 Introduction

Based on technologies and algorithms that were developed about 30 years ago, today's Internet is approaching the limits of its legacy architecture. This has spawned a wide range of intensive studies on the future internet, including the German-Lab (G-Lab) initiative.

One of the promising emerging technologies of more recent years are wireless sensor networks (WSNs). The idea is to use sensor-equipped devices such as cellphones and other embedded systems that sense and interact with their environment for obtaining valuable information about the real world. Only a few mature techniques exist to integrate heterogeneous WSNs with the Internet; it is clear that upcoming massive amounts of data widely exceed the capabilities of classical approaches. The goal of *Real-World G-Lab* is to overcome these obstacles by working on the different levels of protocols, services and applications. We will enable developers to write applications that rely on sensor data input, without knowledge of the underlying hardware platform and the network communication algorithms. This implies that sensors are able to participate in the future internet as peer hosts. This enables new fields of applications but likewise opens a set of new challenges in the context of efficient request processing by WSNs. We will verify our concepts and applications inside the controllable environment of the G-LAB research network, by adding several outdoor WSN deployments to the experimental facility of the G-LAB project.

In summary, Real-World G-LAB will contribute to the integration of resource-constrained (wireless) sensor devices into the future internet by investigating several key challenges, ranging from low-level energy efficiency to improved high-level application development.

T. Magedanz et al. (Eds.): TridentCom 2010, LNICST 46, pp. 577–579, 2011.

2 Integration of Testbeds

Real-World G-Lab will deploy three outdoor and one indoor WSN testbed, as well as one indoor wireless mesh network testbed. To the best of our knowledge, this will be the first federation of sensor and backbone networks that will be permanently available to run experiments on all network layers. By integration of outdoor networks it offers evaluation of realistic applications, e.g. in the fields of environmental and area monitoring including a small mobile network of wireless sensor nodes that is available on demand. A major contribution of the Real-World G-Lab is the extension of the G-Lab experimental facility by a federation of sensor network testbeds as shown in Figure 1.

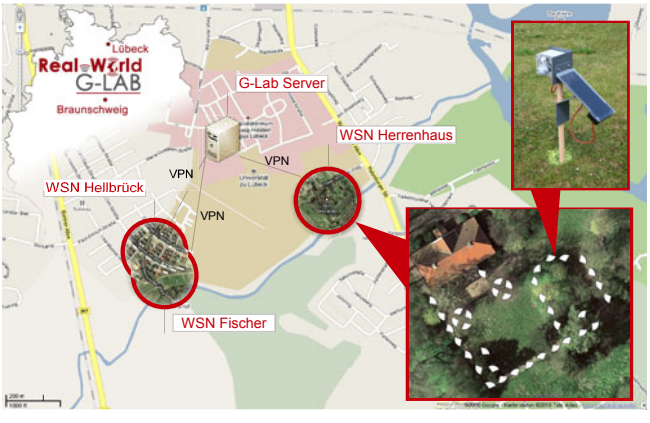


Fig. 1. Integrating WSNs into the G-Lab-Infrastructure

The realization benefits from previous experience and re-usage of components already developed in previous research projects like the WISEBED project that allows for reservation-based utilization of the infrastructure. By that, research results can be evaluated in a large scale on a real world deployment in a fast way. The several testbeds are the base for the experimental real-world evaluation of the research goals described in the next sections.

3 Protocols, Services, and Applications

Protocols: Integrating resource-constrained sensor networks with the Future Internet raises a number of fundamental questions that need to be addressed. One of these deals with the basic communication paradigms, which are currently not sufficiently interoperable. First steps to let sensor nodes participate in Internet communication have already been taken by the research community. Most prominently these are the 6LoWPAN [2] standard and its implementations. However it does not sufficiently address all issues, and there is still a substantial amount of groundwork necessary. We are investigating mechanisms to connect a

sensor network to the Future Internet without compromising some crucial mechanisms of the heavily limited nodes, such as keeping energy consumption at the minimum possible level. This is done by allowing nodes to do duty cycling and energy-aware communication. Furthermore we extend the schemes to support dynamic and mobile WSNs. Another approach is the reduction and avoidance of communication wherever possible. This includes work not limited to the low-level communication level (e.g., compression techniques), but includes very high-level algorithmic mechanisms.

Services: Building upon the protocol layer, we will study two key services for a future Real-World Internet: monitoring and management, as well as service discovery. Although these services have already been investigated in other contexts, the specific properties of the Real-World Internet need careful reconsideration.

With respect to monitoring and management we envision a service that allows users to tradeoff the degree of visibility and control of the system state with resource consumption. In particular, the user will be able to specify a resources budget in terms of network bandwidth, memory, CPU cycles, and energy such that our service will offer best possible visibility and control while not exceeding the given budget.

Integrating embedded sensors into the Internet will allow online and real-time access to the state of the real world. We envision a discovery service that allows finding people, places, and objects that exhibit a certain state at a given point in time, utilizing the information gathered by embedded sensors. We will address this problem by using prediction models, by exploiting correlations between sensors, and by applying peer-to-peer strategies [1]. These services will be the base for the application development.

Applications: Real-World G-Lab will contribute to the application development by the simplified evaluation on large scaled, permanently available sensor network. The consortium encourages all other G-Lab partners to contribute their applications or develop further applications. The sensor networks integrated into the testbed are a piece of a larger federation of extensions which will be contributed by the new G-Lab projects. The applications as illustrated in Figure 1 include environmental monitoring, animal observation. The setup will be extended by audio and video signal processing as well as further event detection.

References

1. Elahi, B.M., Römer, K., Ostermaier, B., Fahrmair, M., Kellerer, W.: Sensor ranking: A primitive for efficient content-based sensor search. In: IPSN 2009: Proceedings of the 2009 International Conference on Information Processing in Sensor Networks, Washington, DC, USA, pp. 217–228. IEEE Computer Society, Los Alamitos (2009)
2. Montenegro, G., Kushalnagar, N., Hui, J., Culler, D.: Transmission of IPv6 Packets over IEEE 802.15.4 Networks (RFC 4944) (September 2007)

QoE Prediction for Radio Resource Management*

M. Hirth, B. Staehle, F. Wamser, R. Pries, and D. Staehle

University of Würzburg, Chair for Communication Networks, Würzburg, Germany
{hirth,bstaehle,wamser,pries,dstaehle}@informatik.uni-wuerzburg.de

Abstract. In this paper, we introduce a monitoring tool which measures application-specific parameters. These parameters are used to predict the QoE and to perform resource management based on QoE thresholds. We demonstrate the tool for YouTube traffic in an IEEE 802.11 mesh network. Thereby, the QoE is based on the player video buffer size and the resource management can include rerouting, throttling of best effort traffic, or a gateway handover.

Keywords: QoE, radio resource management, wireless.

1 Introduction

Today's Internet traffic is transmitted on a best effort basis without supporting quality of service (QoS). Normally, there are no service guarantees for the predominant consumer Internet traffic which is composed of applications like P2P or client-server file sharing, web browsing, or video streaming which make up for more than 80% of today's traffic [1]. Technical solutions enforcing quality guarantees exist, but in general the network does neither know which Internet applications it is carrying nor which quality requirements have to be met.

The prerequisite for QoS support for Internet applications is hence to detect the flows/packets belonging to the application in the packet stream which is currently done using deep packet inspection (DPI). However, DPI is rather challenging as it is not very reliable and does not work if the payload is encrypted. In addition, DIP is very resource intensive and hence not suitable for real-time traffic classification. For guaranteeing application-specific QoS parameters, it is moreover necessary that the network knows about appropriate quality parameters. Deriving and monitoring the appropriate QoS parameters for an application on the network layer is also a very complex task. Both the flow classification problem and the difficulties finding the appropriate QoS parameters can be overcome by information exchange between the application and the network. An application can announce its presence to the network and provide a feedback about its current QoS level.

* This work was funded by the Federal Ministry of Education and Research of the Federal Republic of Germany (Förderkennzeichen 01BK0800, GLab). The authors alone are responsible for the content of the paper.

T. Magedanz et al. (Eds.): TridentCom 2010, LNICST 46, pp. 580–583, 2011.

Even if application demands are known, static resource assignments in IEEE 802.11 wireless mesh networks (WMNs) can cause problems, as the link capacity between two nodes typically changes over time. This can be overcome by using a dynamic resource management. Resource management in WMNs covers routing including gateway selection, channel and interface allocation in multi-radio multi-channel mesh networks, prioritization of medium access through contention parameters, and finally traffic shaping. The user's quality of experience (QoE) can be used to toggle these different measures [2].

QoE is a measure for the subjective quality that a user experiences. Today, a large number of QoE models exist, e.g. for VoIP traffic [3,4] or video streaming [5,6] but these models only allow to quantify the user satisfaction after the application has been carried out. Our goal is however to use QoE as an input for a network management tool. Therefore, we need to know the user satisfaction during the execution of the application. To avoid a QoE degradation, a network management tool has moreover to be notified if a QoE degradation has not yet happened, but is only about to occur. We therefore propose to install a generic tool at the client that monitors and predicts the QoE and communicates this information to the network.

2 Prediction of YouTube QoE

In [7] we introduce a YouTube Monitoring tool (YoMo) which is able to predict the QoE of a YouTube video and uses this QoE information for radio resource management. YouTube videos are distributed via TCP streaming which, unlike UDP streaming approaches, always assures a constant video quality. However, the QoE of a YouTube user is affected by video stallings. For our proof of concept we use a very simple QoE metric assuming the QoE of a user is good as long as the video does not stall and degrades as soon as the video stalls. The length and the frequency of the stallings are not taken into consideration.

The main issue of this approach is to exactly predict the stalling time of the video. The YouTube player offers a programming API which is able to monitor the player state. Monitoring the player state is suitable for detecting a video stalling, but not for predicting a future stalling. Therefore, we focus on the

Fig. 1. YouTube player video buffer

filling of the player's video buffer β depicted in Fig. 1. This buffer fills with the beginning of the video download. As soon as a certain threshold γ is reached, the playback begins. β depends on the download rate and the video rate. As long as the download rate is larger or equal to the video rate, β rises respectively remains constant. If the download rate is smaller than the video rate, β shrinks and the video stalls as soon as β=0.

YoMo is able to calculate β by monitoring the client's network traffic and by using information gained from the YouTube player. Fig. 2 shows YoMo's user interface. The upper display shows the current value of β, the display in the middle the progress of β over time, and the bar at the bottom the download progress of the whole file. β is always displayed in seconds and the displays are divided into three areas colored in green, yellow and red. As long as β is within the green range, there is no need to change the resource allocation of the flow. The yellow range indicates that the video might stall in the near future if no actions are taken. As soon as β drops below 5 s, which means only 5 s of video time remain in the buffer, the video is likely to stall if the resource allocation is not changed. A stalling would result in a degradation of the users QoE. With this knowledge of β, YoMo can not only monitor the current QoE of the user, but it can also predict when the video will stall if the current network state is not changing. This enables us to adapt the available resources in the mesh network to avoid stalling. Theses adaptations might include rerouting, throttling of best effort traffic, or selecting a new mesh gateway to forward the traffic to the Internet.

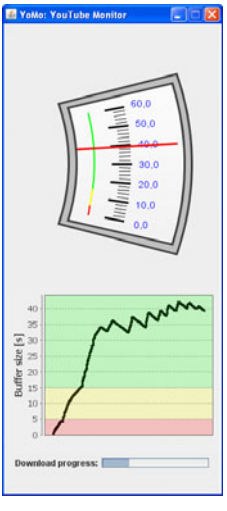


Fig. 2. YoMo GUI

YoMo is a proof of concept which can predict the QoE of a YouTube user and we showed that QoE prediction can be used for radio resource management. Although YoMo is a rather simple QoE prediction tool, the idea can be generalized. QoE prediction can be natively implemented into QoE sensitive applications like streaming players, VoIP, or IPTV software.

References

1. Wamser, F., et al.: On Traffic Characteristics of a Broadband Wireless Internet Access. Special Issue of the Telecommunication Systems (TS) Journal (July 2010)
2. Pries, R., et al.: Dynamic Bandwidth Control in Wireless Mesh Networks: A Quality of Experience based Approach. In: 18th ITC Specialist Seminar on Quality of Experience (May 2008)
3. Raake, A.: Short-and Long-term Packet Loss Behavior: Towards Speech Quality Prediction for Arbitrary Loss Distributions. IEEE Transactions on Audio, Speech, and Language Processing 14(6) (November 2006)

4. Hoßfeld, T., Binzenhöfer, A.: Analysis of Skype VoIP Traffic in UMTS: End-to-end QoS and QoE Measurements. Computer Networks 52(3) (February 2008)
5. Engelke, U., Zepernick, H.J.: Perceptual-based Quality Metrics for Image and Video Services: A Survey. In: 3rd EuroNGI Conference on Next Generation Internet Networks (May 2007)
6. Gustafsson, J., et al.: Measuring Multimedia Quality in Mobile Networks with an Objective Parametric Model. In: 15th IEEE International Conference on Image Processing (December 2008)
7. Staehle, B., et al.: YoMo: A YouTube Application Comfort Monitoring Tool. Tech. Rep. 467, University of Würzburg (March 2010)

COMCON: Use Cases for Virtual Future Networks*

D. Schlosser[1], M. Hoffmann[2], T. Hoßfeld[1], M. Jarschel[1], A. Kirstaedter[3], W. Kellerer[4], and S. Köhler[5]

[1] University of Würzburg, Chair for Communication Networks, Würzburg, Germany
schlosser@informatik.uni-wuerzburg.de
[2] Nokia Siemens Networks GmbH & Co. KG, München, Germany
[3] Institute of Communication Networks and Computer Engineering (IKR), Universität Stuttgart, Stuttgart, Germany
[4] DOCOMO Communications Laboratories Europe GmbH, Munich, Germany
[5] Infosim GmbH & Co. KG, Würzburg, Germany

Abstract. In the Future Internet, a multitude of networks will coexist and complement each other. These networks allow specialization but require isolation of functionalities in order to provide dependable and predictable networks. This allows different networks to run in parallel but isolated from each other. Additionally, network resource scalability is supported to reduce the time and overhead required to introduce new services. The objective of the COMCON (COntrol and Management of COexisting Networks) project is to design novel control and management mechanisms that support the coexistence of networks in a future networking scenario and to illustrate the economic advantages. In this contribution we present three use cases defined in COMCON, which serve as a guideline for our virtual network architecture.

Keywords: Future Internet, Use Case, Virtual Network.

1 Introduction

In the Future Internet, a multitude of networks will coexist and complement each other. These coexisting networks allow specialization but require isolation of functionalities in order (a) to provide dependable and predictable networks (e.g., a banking network), (b) to allow different network technologies to run in parallel, but isolated from each other (e.g., coexistence of 3G and different beyond 3G mobile networks on the same physical infrastructure), and (c) to support network resource scalability to reduce the time and overhead required to introduce new services (e.g., to support the seamless transition from a limited liability beta service to a fully operational resilient high-demand service). Each network should be able to run its own specialized protocols that may fundamentally differ from

* This work was funded by the Federal Ministry of Education and Research of the Federal Republic of Germany (Förderkennzeichen 01BK0917, GLab). The authors alone are responsible for the content of the paper.

T. Magedanz et al. (Eds.): TridentCom 2010, LNICST 46, pp. 584–586, 2011.

today's Internet Protocol (IP) stack. Network virtualization is considered to be a key technology to realize coexisting networks.

The objective of the COMCON project (COntrol and Management of COexisting Networks) is to design novel control and management mechanisms that support the coexistence of networks in a future networking scenario and to illustrate the economic advantages. Virtualization technology is a key component that not only acts as an abstraction layer between services and infrastructure to facilitate innovation, but also is an integral part of the overall design to support the evolution and coexistence of different network architectures. Towards that goal, interfaces between functional roles in coexisting networks, realized by network virtualization, are specified. A provider- and operator-grade management and control function of coexisting virtual networks is built. It comprises of isolation, dynamic reassignment of resources, and efficient and effective monitoring of virtual networks. The requirements for the network reference architecture in the COMCON project is derived from a set of unique use cases. These use cases help to design, evaluate and verify the reference architecture during the design process in an iterative way.

2 Use Cases to Evaluate the Reference Architecture

Among others, we have defined the following use cases: Service Component Mobility, Service Broker, and Beta Slice.

The Service Component Mobility use case considers dynamic migration of service components in a virtual network. Moving or reproducing virtualized components geographically closer to the user enables two kinds of improvement. On the one hand, the relocation of resources might improve the delay, jitter, and other quality of service parameters. The QoE of the user increases accordingly. On the other hand, the relocation can optimize the utilization of network components. If the network is monitored and it is reported that the number of customers using the service from a distant location exceed a certain threshold, the relocation may free capacity on long distant links. This relocation may make sense also from a economic perspective.

The Service Broker is realized as a network component, which knows about the user's needs and selects and bundles services from different providers. Thus, it is the 'single face' of the virtualized networks to the customer and chooses the virtual network costs according to the user's needs in terms of costs and network quality. In this use case we consider a scenario, where the network virtualization is extended to the end-customer. Different virtual network operators (VNOs) compete with their services and the end-customer is free to select a different VNO for each network service he wants to use. For example he might want to watch IPTV using a premium video transfer service provided by one specific VNO. His VoIP and gaming services are delivered by another VNO, which has specialized on low-delay-connections with small bandwidth requirements. However, the peer-to-peer traffic is handled by a VNO that provides only best-effort data transfers with 90% availability, but charges on a cheap flat rate basis.

The Beta Slice enables the creation and testing of new services without the additional cost of setting up a specialized test bed. Often the evaluation of new services in a specialized test bed environment is too expensive. Hence, the new service is never implemented. The Beta Slice is a special purpose virtual network to solve this problem. A new service is launched within a small dedicated virtual network, which restricts the access to a small group of initial users. After the service has been tested successfully, the virtual network can be extended progressively to a full operational network. This way, roll out costs are decreased and expenditures for test bed evaluation are saved. Another aspect is that the time-to-market of the new service may be significantly decreased.

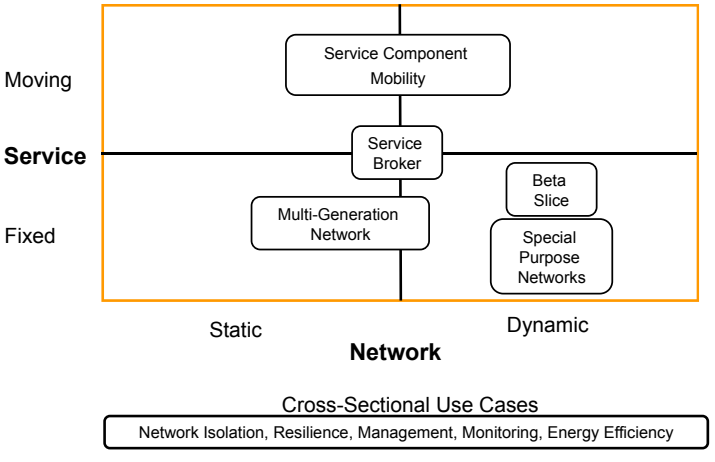


Fig. 1. Clustering of Use Cases

Figure 1 depicts how the use cases differ according to service mobility and network dynamics. The Beta Slice is a good example for a use case, which changes the network size rapidly but does not vary in terms of the service delivery. In contrast, the Service Component Mobility use case is very dynamic in terms of service delivery. The numbers of users is expected to vary over time, but the mean value is considered to be only changing slightly. The Service Broker use case is somewhere in-between. The number of networks attached to the user as well as the the service delivery will change from time to time but not completely.

3 Conclusion and Outlook

Based on these and other defined use cases, the COMCON project will design novel control and management mechanisms for coexisting virtualized networks. We show initial project results derived from the evaluation of the use cases. Moreover, potential business impact will be illustrated.

Evaluation of Future Network Architectures and Services in the G-Lab Testbed

Hans Wippel[1], Oliver Hanka[2], Christoph Spleiß[2], and Denis Martin[1]

[1] Karlsruhe Institute of Technology
{martin,wippel}@kit.edu
[2] TU München
{oliver.hanka,christoph.spleiss}@tum.de

Abstract. Our current Future Internet research in the G-Lab project [1] comprises clean slate network architectures and services. In this vast topic, we focus on two different aspects: (1) Composition as well as adaptation of application and network tailored protocols and (2) novel addressing and routing schemes based on locator/identifier-split. In the following, we describe these aspects including their benefit and usage of the G-Lab testbed. Then we detail our ongoing cooperation and the development of a joint prototype.

1 Composition and Adaptation

In our composition approach, we envision the following development cycle for application and network optimized communication protocols in future networks: First, new protocols are composed by (re-) using units called *Building Blocks* in a design tool. Second, the newly created protocols—called *Netlets*—are evaluated in the G-Lab testbed. Finally after successful evaluation, the newly implemented Netlets can be deployed easily in real networks. The testbed with its real hardware and operating system APIs enables us to use the same Netlet execution framework for the evaluation and the deployment in real networks.

As Netlet framework we use the *Node Architecture* [2] outlined in Figure 1 (left): A requirements-based application interface allows exchanging Netlets without modifying applications since protocol selection and name to address resolution is completely handled by the Netlet framework. A selection algorithm within the *Netlet Selection* component chooses a suitable Netlet based on the requirements given by the application. A generic *Naming and Addressing* component delegates name to address resolution to a component responsible for the current network. This could be, for instance, the mapping service described in the following section. The streams of the respective Netlets are (de-) multiplexed by the *Netlet Multiplexer*. The *Management and Adaptation Component* constantly monitors the conditions of the Netlets, the network, and the applications. If changes are detected, it tunes configuration parameters to adapt Netlets as good as possible to the new conditions. As an abstraction for network connectivity the *Network Access* (NA) is used. Although it can be compared to today's

T. Magedanz et al. (Eds.): TridentCom 2010, LNICST 46, pp. 587–589, 2011.

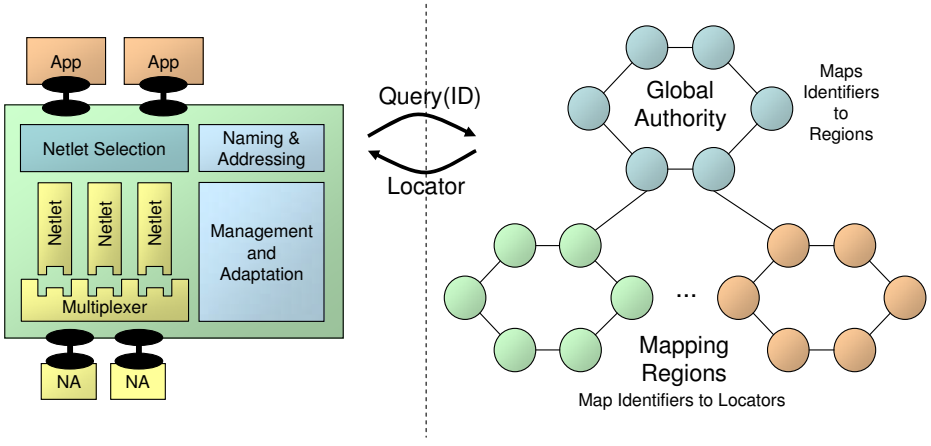


Fig. 1. Separated experiments and their interaction (left: Netlet framework, right: Locator/Identifier-Split Mapping Service)

network interface, it is a more flexible abstraction of any type of network (e. g., physical as well as virtual networks). Especially in virtual networks, we expect more detailed information about the current network condition due to a more sophisticated virtual network management infrastructure.

The combination of the G-Lab testbed and our Netlet framework provides us with an appropriate test environment for our experiments. In a first stage, we want to experiment with various Netlets including transport, routing, and management/monitoring protocols. Thus, we expect to gain better insights on the interaction of Netlets, their possible deployment strategies, signaling mechanisms, and monitoring concepts. In a second stage, we plan to emulate different dynamic network properties (e.g., packet loss, reordering, latency, and data rates) in the G-Lab testbed. Then we will evaluate Netlet adaptation concepts which aim to compensate such network properties and optimize communication.

2 Locator/Identifier-Split

By using the idea of the locator/identifier-split (loc/id-split) for a next generation Internet addressing scheme, we can overcome several problems of today's IP based architecture. The two semantic meanings of today's IP address—who do we want to contact and where can we find him?—are split into two different addresses, the identifier and the locator. The loc/id-split, however, requires a system which is able to map these two addresses before any connection can be established. The so called *mapping service* [3] depicted in Figure 1 (right) can be queried to retrieve the current valid locator for an identifier in order to contact the corresponding node. The mapping service needs to store the identifier/locator tuple for any assigned identifier and has to cope with the burden of frequent locator updates as mobile nodes roam and change their point of attachment

towards the network. Because of the expected high load a centralized approach for the mapping system is not feasible. We therefore suggest the usage of a decentralized system like DHTs. Instead of one large decentralized database, the mapping system is split up in different regions. Each region stores the loc/id-mapping of nodes registered in this region. A region for example is a country or a continent. If the mapping region for a specific identifier is not known, the so called global authority is queried, which maps identifiers to the corresponding regions. The global authority also serves as a single point of trust for a PKI-based security infrastructure. In order to avoid long lookup times in the DHT we use a protocol which resolves a query in only one hop.

To validate our approach, we implemented a prototype of the mapping service and deployed it to the G-Lab experimental facility. As the mapping is based on structured peer-to-peer principles, many nodes are required to instantiate a solid mapping infrastructure. Additionally, measurements only become significant if performed on a large scale.

3 Conclusion and Future Work

The G-Lab experimental platform enables us to connect our prototypes of the two independent research aspects mentioned above. Unlike with two experiments running within separate simulation frameworks, the two prototypes are able to communicate with each other. Therefore, they can utilize the functionality provided by the respective counterpart. In our current setup, the Netlet framework is accessing the information provided by the mapping service to address end-nodes and services by identifiers (cf. Figure 1). In that way, it doesn't have to deal with mobility issues—addressing wise—and end-nodes are able to roam freely. On the other hand, we have the possibility to construct a hybrid mapping service that utilizes peers implemented as Netlets. This is a next step towards a migration of the mapping service to a Netlet based solution. All this wouldn't be possible with separated simulations. The G-Lab experimental platform, therefore, plays a major role in validating theoretical results of the proposals, enables us to gain real-world measurements from our prototypes, and supports our ongoing cooperation.

References

1. G-Lab Project Website, `http://www.german-lab.de`
2. Völker, L., Martin, D., El Khayat, I., Werle, C., Zitterbart, M.: A Node Architecture for 1000 Future Networks. In: Proceedings of the International Workshop on the Network of the Future 2009, Dresden, Germany. IEEE, Los Alamitos (June 2009)
3. Hanka, O., Kunzmann, G., Spleiß, C., Eberspacher, J., Bauer, A.: HiiMap: Hierarchical Internet Mapping Architecture. In: First International Conference on Future Information Networks (ICFIN 2009), Beijing, China, P.R. China (October 2009)

Research and Demonstration of Next Generation Network and Service National Testbed of China

Xiaoyuan Lu, Liang He, Chuan Peng, and Yu Zhang

National Engineering Research Center for Broadband Networks & Applications,
No. 150 Honggu Road, Shanghai, China
{xylu,lhe,chpeng,yuzhang}@b-star.cn

Abstract. This paper introduces the China National testbed and experimental environment and its characteristics. The paper especially presents reconfigurable, service and NGB testbed aspect of their function and value for research and demonstration of China next generation network and service. Furthermore it summarizes research and experimental achievements and resources available for experimentation and formulates suggestions for the future network construction and international cooperation.

Keywords: China National Testbed, Network Architecture, Reconfigurable Testbed.

1 Introduction

Next generation network and Service National Testbed are funded by Ministry of Science and Technology of the People's Republic of China (MOST) and implemented by National Engineering Research Center for Broadband Networks and Applications of China. The objective of this project is to provide innovative technologies and a demonstration environment of new services for National High Technology Research and Development Program of China (863 program) and National Key Special Funds.

China national testbed supports network institutional reformation, tests new networking technologies verifies new network equipments and demonstrates emerging services. Through the above approaches, it provides indoor and outdoor experimental environments, experimental networks in designated areas and a basic environment for large scale experimental demonstration networks. It is able to meet the requirements of experiments and tests of equipments for various types of networks and nodes on different developing stages.

2 Architecture of National Testbed Network

Next generation Network and Service National Testbed of China are classified into three catalogues, namely reconfigurable testbed, service testbed and NGB testbed, according to their functions. China national testbed covers Yangtze River Delta region, Pearl River Delta region, across Shanghai, Zhejiang province, Jiangsu province,

T. Magedanz et al. (Eds.): TridentCom 2010, LNICST 46, pp. 590–593, 2011.

Anhui province, Guangdong province and Hainan province. The total length of the backbone network is more than 4000 km. It serves for more than 1 million users, including residents, enterprises, governments, schools and other subscriber groups. It also has international portals.

The overall architecture of next generation network and service national testbed of China is shown in Figure 1.

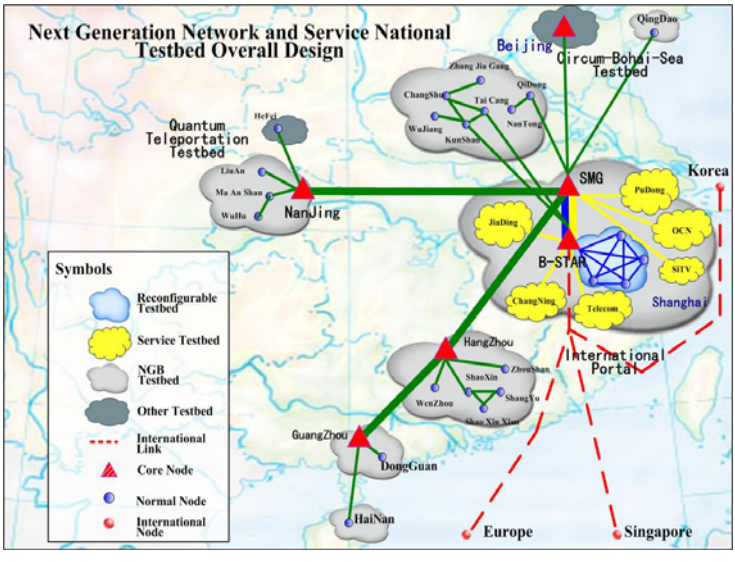


Fig. 1. The overall architecture of next generation network and service national testbed

(1) Reconfigurable Testbed

863 program teams in China have put much effort into exploring the solutions for these problems by constructing reconfigurable testbed, proposing service-oriented technical frameworks, and developing and experimenting reconfigurable equipments.

The basic idea of the technical framework of a service-oriented network is addressed in the following. First of all, network service consists of resource sharing layer, reconfigurable edge layer and logical carrier layer. Secondly, resource sharing layer can provide reconfigurable network services. Logical carrier layer aims to offer bearer services generated from the logical carrier network, according to the individual requirement of different classes of services. While reconfigurable edge layer makes use of the reconfigurable network services provided by the resource sharing layer, which will facilitate the construction of the logical carrier network in the logical carrier layer [1].

The basis of open-reconfigurable routing and switching platform is component-based process technique [2]. There are three aspects of its characters. First of all, a component is the basic process module. The platform provides reconfigurable running support environment for various module, while module provides reconfigurable environment to various components. Secondly, platform, module and component all comply with a serial of common standardized specifications. The standardized module

provided by any third party are able to participate and implement a given task on the same platform and the standardized components provided by any third party are able to participate and implement a particular function on the same platform. Last but not least, the platform level and module level both have the ability to upgrade and reconfigure functions, distribute codes and configure management layer. The component has good maintainability (e.g, install, uninstall, upgrade and update) [3].

The reconfigurable testbed consists of five reconfigurable routers which are connected by optical mesh links which have constructed in Shanghai. Large scale experiment is carried out on it.

(2) Service Testbed

The main function of the service testbed is to carry out new services and set an example. It can implement functions such as replay, VOD, record, search and recommendation etc. The development of the service platform will facilitate new services like interactive program and value-added services. Currently, 3D technique is employed in live broadcasting on this testbed at Shanghai Expo 2010 [4].

One of the challenges with the service testbed is to setup an effective rule for service evaluation in the experiments which can be achieved by embedding evaluation plug-in at the terminals, compiling statistics of Business Operations Support System (BOSS) service behavior and service income and behavior perception of the network devices.

(3) NGB Testbed

NGB is based on the achievements of China Multimedia Mobile Broadcasting (CMMB) and digital cable television. It uses key technologies in "High performance broadband information network 3TNet" to construct the next generation broadcasting (NGB) network which is featured with "Triple-play", wired or wireless and controllable. Compared with traditional Internet, NGB has more advantages such as safe, controllable and reliable. As infomationization promotes industrialization, the public service will be safer and securer in NGB.

The goals of NGB construction are: Transmission bandwidth of 1Gbps access bandwidth of 40Mbps. The digital interactive information consumption on broadband network will become as popular as other utilities services.

3 Conclusion

This paper presents the overall design and network architecture of the ongoing next generation network and service national testbed in China. We illustrate respectively the functions and characters of three types of testbeds, namely reconfigurable testbed, service testbed and NGB testbed. The development and the demonstration of the testbeds are innovative for network system reformation and the planning of next generation network testbed. It will provide a good platform for testing and verifying various techniques and devices. It is the basis in designing and verifying for future networks in China. We welcome oversea visitors and look forward to international collaborations.

Acknowledgment

This project “next generation network and national service testbed of China” is funded by National High Technology Research and Development Program of China (863 program).

References

1. Szypeski, C.: Component Software: Beyond Object-Oriented Programming. Addison-Wesley, Reading (1997)
2. Kozaczynski, W., Booch, G.: Component-Based Software Engineering. IEEE Software, pp. 34–36 (September/October 1998)
3. Bronsard, F., et al.: Toward Software Plug-and-Play. ACM Software Engineer Notes, 19–29 (September 1997)
4. Expo (2010), http://www.expo2010.cn/

G-Lab Deep: Cross-Layer Composition and Security for a Flexible Future Internet

Carsten Schmoll[1], Christian Henke[2], Dirk Hoffstadt[3], Abbas Ali Siddiqui[4], Thomas Magedanz[2], Paul Müller[4], Erwin Rathgeb[3], and Tanja Zseby[1]

[1] Fraunhofer FOKUS Network Research Group
Kaiserin-Augusta-Allee 31, 10589 Berlin
{carsten.schmoll,tanja.zseby}@fokus.fraunhofer.de
[2] Technical University Berlin Next-Generation Networks
Straße des 17. Juni 135, 10623 Berlin
{c.henke@tu-berlin.de,tm@cs.tu-berlin.de}
[3] University of Duisburg-Essen, Computer Networking Technology Group
Ellernstr. 29, 45326 Essen
{dirk.hoffstadt,erwin.rathgeb}@iem.uni-due.de
[4] University of Kaiserslautern, Integrated Communications Systems Group
Erwin-Schrödinger-Straße, 67663 Kaiserslautern
{siddiqui,pmueller}@informatik.uni-kl.de

Abstract. The Internet provides a global communication basis for businesses and communities. But in today's Internet new demands collide with old design principles, resulting in a complex agglomerate of protocols and patches. These makeshift solutions are hard to manage, protect, and extend. The G-Lab DEEP project aims at these challenges with an innovative composition approach with a special emphasis on security. One goal is the dynamic composition of functions from network and service layer based on the requirements of applications. The composition is done by a mediation process that selects suitable function modules and can negotiate whether functions should be positioned on network or service layer. In G-Lab DEEP a prototype for such architecture will be developed.

Keywords: Security, Functional Composition, Cross-Layer, Future Internet, Service Composition.

1 G-Lab Deep Project Description

In the current IP network design, applications and networks are inherently independent. Based on the separation of layers in the current network architecture applications are not able to instruct the network how to handle the applications' traffic (e.g. cannot request QoS or custom routing). The network only offers best-effort packet transport to the overlying applications. Additionally there is no standard way to tell applications the state of the network, so that applications could react and read the network status. Therefore one goal of a future Internet infrastructure is to support

T. Magedanz et al. (Eds.): TridentCom 2010, LNICST 46, pp. 594–596, 2011.

specific demands of applications plus the ability to inform them about the network status. The current paradigm of a layered model for networking is already partially broken up by some cross-layer techniques and a kludge of ad hoc solutions.

Besides incremental solutions for a future Internet new networking paradigms have been considered that follow a "clean slate" approach, i.e. disruptive technologies that develop the future Internet from scratch. One of these approaches is functional composition, which decomposes the network stack in functional building blocks and reorganizes the functionalities in a compositional framework. Functional composition focuses on the flexibility of the network and therefore targets two improvements to the current Internet

1) Ease of management and integration of new functionality
2) Application specific network composition and adaptation based on application requirements instead of using "one-size-fits-all" TCP/IP best effort service

The G-Lab DEEP project takes on the challenges of the current architecture with a modularized functional composition approach that 1) passes application specific requirements to the network layer and 2) uses a cross layer composition technique to allow composition of independent service and network functional blocks in one integrated framework based on the requirements and the network status. A modular solution with loose coupling is desirable to (a) achieve a clean separation of the needed functional blocks without strong entanglement of message passing functionalities, and (b) allow loose binding of functional blocks which are needed for a specific service or application request. Further this breaking down into atomic functional blocks allows for the most flexible combination of functions. This is desirable as each combination of services on the application level may require a different combination of network modules to support it.

In the application layer as well as on the network layer the same derived questions and problems arise for the composition of functions, for example the semantics, description, the management, discovering and the construction of the optimal function chain for the given conditions.

This concept can be visualized well using the example of voice communication in the Internet. The role of voice communication in the future Internet will grow, especially with mobile voice applications, while in parallel new functions are developed and new application requirements, many of them security-related, are evolving.

Consider for example the situation of an emergency call via a mobile voice terminal. Based on the specific user intent to make an emergency call a workflow of services within the service and network layer must be triggered which together form the emergency call. This workflow may invoke auxiliary services like “get location”, “make a call” and “reserve line” to support the nature of the emergency call. Maybe even a voice recording should be added automatically. Such auxiliary or partial services can be provided by service-enablers within the provider network.

While the emergency call is established functional blocks are invoked, but these must also react on requirements specific to this invocation (e.g. prioritization). Therefore the services as well as the network connecting them must treat the call accordingly. Within next generation networks (NGN) such requirements can be

passed along as policies. On the network layer we want to supply such features by functional composition of network blocks.

To secure a reliable provisioning of all of these network and service components, a management solution spanning those layers is needed, which must also include service monitoring as an integral part. This requires having the monitoring itself available on the service and on the network layer, with configuration and data export using standardized interfaces. The overall solution shall also work in an inter-domain environment. Monitoring can then be effectively used to detect anomalies and to trigger corresponding actions based on policies.

Through the introduction of a cross-layer monitoring system the network status is continuously monitored and made available to other network and application services. Based on this cross-layer monitoring service the composition engine becomes situation aware and can automatically compose services based on the network status. In the current IP world network attacks have become a tremendous threat. Besides threats to network elements and end-hosts (e.g. through virus and worms) there also emerged new threats with the development of new applications and services. Voice over IP has been exploited by adversaries for anonymous mass voice calls for commercial purposes (SPIT) and for passing the bill to other users or companies (toll fraud). Therefore the cross-layer composition and monitoring system needs to incorporate these experiences from the current Internet and address these challenges for future application to enable suitable detection and countermeasures in a secure way.

The project G-Lab Deep is funded by the Federal Ministry of Education and Research (BMBF).

G-Mesh-Lab
Wireless Multi-hop Network Testbed for the G-Lab

Mesut Güneş, Oliver Hahm, and Kaspar Schleiser

Distributed Embedded Systems, Computer Systems and Telematics,
Institute of Computer Science, Freie Universität Berlin, Germany
{guenes,hahm,schleiser}@inf.fu-berlin.de

The *G-Mesh-Lab* is a Wireless Multi-hop Network research project of the *Freie Universität Berlin*. As part of the Real World G-Lab Project, the focus of G-Mesh-Lab is to research and develop next-generation multi-hop network algorithms and protocols. It utilizes the *DES-Testbed*[1][2][3], which offers a sophisticated testing and evaluation framework, making it possible to define, schedule, run, monitor and evaluate multi-hop wireless network and wireless sensor network experiments.

The *DES-testbed* consists of 60 *DES-Nodes* spread over two buildings. An extension to 120 nodes, including 13 outdoor nodes, is in the process of being installed. Each *DES-Node* consists of a wireless router equipped with three IEEE 802.11a/b/g transceivers and a sensor node. While the wireless LAN part forms the wireless mesh network *DES-Mesh*, the sensor nodes establish a wireless sensor network called *DES-WSN*. Thus a wireless mesh and wireless sensor network are operated in parallel, making the Testbed one of the largest hybrid networks world-wide. The *DES-Nodes* are deployed in an irregular topology across several buildings on the campus.

The testbed management system *DES-TBMS* supports the definition, scheduling, execution, and evaluation of experiments. As shown in 1, the architecture of *DES-TBMS* consists of five components and four databases. *DES-Cript* is a custom domain specific language based on XML to define experiments either in a textual way or with the help of the web-frontend *DES-Web*. *DES-Exp* is the experiment manager which maintains all experiments stored in the *Experiment Description* database and is responsible for scheduling and executing experiments

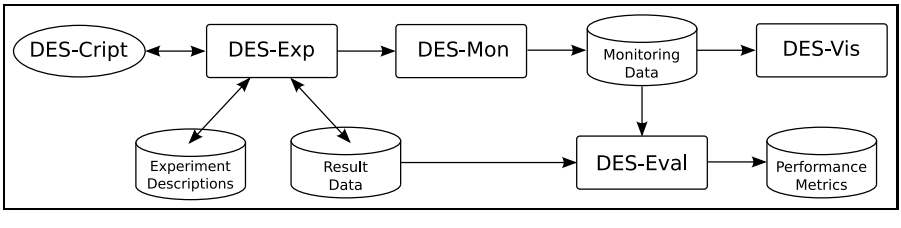


Fig. 1. Architecture of DES-Testbed Management System (DES-TBMS)

T. Magedanz et al. (Eds.): TridentCom 2010, LNICST 46, pp. 597–598, 2011.

and collecting their log files. All raw results of experiments are stored in the *Result Data* database. DES-Mesh is pre- and post-configured by the *DES-Mon* component, which also monitors the testbed during experimentation. The *Monitoring Data* database stores all data required by *DES-Vis* to visualize the network state. Data measured in experiments can be displayed in DES-Vis to get a better insight in the behavior of an algorithm. In a video player like interface any network state during the experiment can be inspected. The *Result Data* and *Monitoring Data* are used by *DES-Eval* for automated evaluation as defined in the experiment description.

In summary, the G-Mesh-Lab provides an easy-to-use wireless multi-hop testbed that enables researchers to evaluate algorithms and protocols, developed only theoretically or by using simulations, on real hardware.

Visit *http://www.des-testbed.net* for more information.

References

1. Günes, M., Blywis, B., Juraschek, F.: Concept and design of the hybrid distributed embedded systems testbed. Freie Universität Berlin, Tech. Rep. TR-B-08-10 (August 2008), `ftp://ftp.inf.fu-berlin.de/pub/reports/tr-b-08-10.pdf`
2. Günes, M., Blywis, B., Juraschek, F., Schmidt, P.: Practical issues of implementing a hybrid multi-nic wireless mesh-network. Freie Universität Berlin, Tech. Rep. TR-B-08-11 (August 2008), `ftp://ftp.inf.fu-berlin.de/pub/reports/tr-b-08-11.pdf`
3. Blywis, B., Günes, M., Juraschek, F., Schiller, J.: Trends, advances, and challenges in testbed-based wireless mesh network research. In: ACM/Springer Mobile Networks and Applications (MONET) (February 2010); special Issue on Advances In Wireless Testbeds and Research Infrastructures, `http://www.springerlink.com/content/hk25051135m36571/`

Emulating Multi–hop Wireless Networks over a Planetary Scale Testbed*

Elio Salvadori[1], Roberto Doriguzzi Corin[1], Roberto Riggio[1], Attilio Broglio[2], Fabrizio Granelli[2], and Andy Bavier[3]

[1] CREATE-NET, via alla Cascata 56/D, 38123 Trento, Italy
[2] DISI – University of Trento, via Sommarive 14, 38123 Trento, Italy
[3] Princeton University, Princeton NJ 08544, USA

1 Introduction

The most commonly used technique to evaluate novel solutions is to leverage on simulation studies which are largely based on a simplified model of the system behavior. Such an approach provides an approximate evaluation of the system's performances, which can be potentially far away from the behavior on a real deployment. Another solution is to exploit real world prototypes and testbeds. Such testbeds are generally based on a limited number of nodes and, due to their experimental purposes, they only partially present the challenges of a real operational environment with hundred or thousand users. Furthermore, the limited number of nodes limits the possibility to study scalability issues.

In this paper a novel architectural framework and its implementation are proposed in order to provide a realistic environment where innovative techniques for multi–hop wireless networks can be tested in a controlled environment. The objective of the proposed solution is to depart from the limitations of simulated or real world testbed, while providing a testing environment which facilitates performing experimental studies of real network nodes firmware at large scale. To this purpose VINI (VIrtual Network Infrastructure) [1], an extension of PlanetLab – a planetary distributed testbed composed of thousand of nodes, has been selected as the basis for the proposed framework.

2 Architecture and Preliminary Results

Figure 1 illustrates the framework that has been set up in order to provide an effective emulation platform on top of VINI. The contribution of this paper is twofold: on the one hand we extend VINI with a *wireless link emulator*, which mimics wireless link conditions in a "wired" environment, and with a *virtual*

* This work was partially supported by the Seventh Framework Programme (FP7/2007-2013) of the European Commission, within the SMART-Net project (grant number 223937) and by the Italian Ministry of Scientific Research (MIUR) within the framework of the Wireless multiplatfOrm mimo active access netwoRks for QoS-demanding muLtimedia Delivery (WORLD) project (grant number 2007R989S).

T. Magedanz et al. (Eds.): TridentCom 2010, LNICST 46, pp. 599–601, 2011.

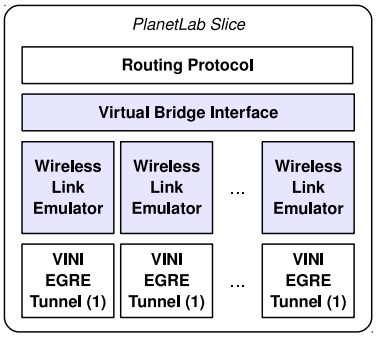
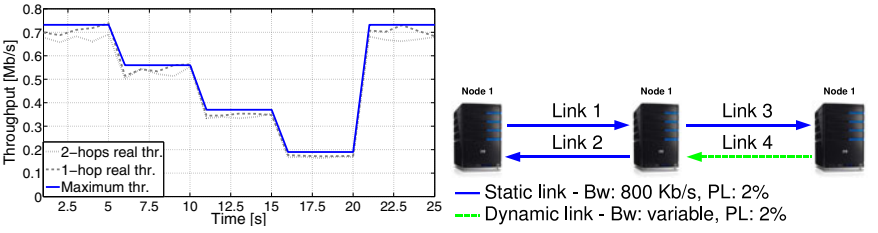


Fig. 1. A VINI "private instance" extended with our modules providing *wireless link emulation* and with *virtual bridging* functionalities

bridge which allows for L2 bridging among virtual interfaces in a slice. On the other hand we assess the viability of our approach by running WING [2], a real–world wireless mesh networking solution [3], on top of a VINI slice. The rationale behind such a choice lies behind the cross–layer techniques exploited by WING in order to perform link–quality routing. Such techniques require access to several MAC-level parameter and are thus very demanding for a network emulator. Even though the results reported in this poster refer to a slice composed by a limited number of nodes, the proposed approach can potentially scale up to thousand of nodes by leveraging on the intrinsic properties of PlanetLab.

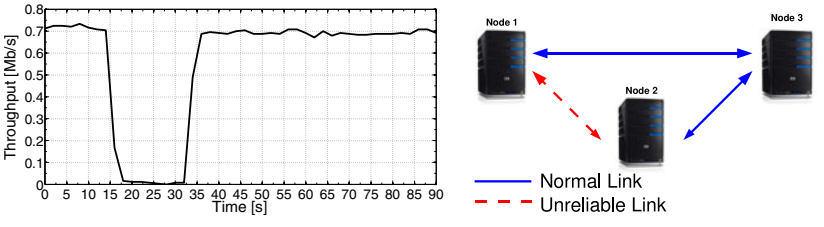
The preliminary experimental activity presented in this poster aimed primarily at assessing the correct behavior of the extension to the standard VINI architecture. Results obtained in both single–hop and multi–hop conditions show that the proposed tool provides an approximated, yet realistic, model for the wireless link conditions; furthermore, it provides researchers with an effective tool to introduce controlled events such as link degradation or disruption. More specifically, Fig. 2a shows the performance of the 2–hops path depicted in Fig. 2b when the available bandwidth at link number 4 is dynamically modified. Link bandwidth and packet loss for link 1 through 3 are respectively 800 kb/s and 2%. The picture shows the currently available bandwidth at link number 4 and the path throughput measured using the *nuttcp* [4] synthetic traffic generator.

Finally, a fully functional Wireless Mesh Network composed of three nodes has been setup using the WING toolkit [2]. It is worth noting that the WING routing protocol that is currently exploited in commercially available platforms has been ran over a PlanetLab slice composed of 4 nodes without any modification to its source code. Figure 3a demonstrate the routing protocol self–healing capability by simulating a link failure in a single–hop scenario in a triangular network topology. The network outage is relative to the time need by the routing protocol in order to switch from a single hop path to a two hops path.



(a) Path throughput. (b) Network topology.

Fig. 2. Controlling a 2-hops path throughput by dynamically reconfiguring a wireless link's parameter (bandwidth)



(a) Path throughput. (b) Network Topology.

Fig. 3. Testing path reconfiguration delay by simulating a link failure

3 Conclusions

In this work a novel wireless network emulation framework built on top of VINI and enabling testing and development of innovative solutions has been presented together with an early assessment of its performance and capabilities. A number of future activities are envisioned, among them one of the most promising is the improvement of the wireless link emulator in order to take into account the broadcast nature of the communication medium, including carrier sensing.

References

1. Bavier, A., Feamster, N., Huang, M., Peterson, L., Rexford, J.: In vini veritas: realistic and controlled network experimentation. In: Proc. of ACM SIGCOMM (2006)
2. Granelli, F., Riggio, R., Rasheed, T., Miorandi, D.: WING/WORLD: An Open Experimental Toolkit for the Design and Deployment of IEEE 802.11-Based Wireless Mesh Networks Testbeds. Eurasip Journal on Wireless Communications and Networking 24 (2010)
3. Riggio, R., Scalabrino, N., Miorandi, D., Granelli, F., Fang, Y., Gregori, E., Chlamtac, I.: Hardware and software solutions for wireless mesh network testbeds. IEEE Communication Magazine 46(6), 156–162 (2008)
4. nuttcp, `http://www.lcp.nrl.navy.mil/nuttcp/`

BERLIN: The Berlin Experimental Router Laboratory for Innovative Networking

Dan Levin, Andreas Wundsam, Amir Mehmood, and Anja Feldmann

Deutsche Telekom Laboratories, TU Berlin, Berlin, Germany
{dan,andi,amir,anja}@net.t-labs.tu-berlin.de

Abstract. Today's disruptive approaches to rearchitecting the Internet, e.g., *Clean Slate Networking* initiatives require testbeds that present unprecedented flexibility to the experimenter. This poster presents BERLIN, a flexible testbed platform designed towards the requirements of Future Internet research. BERLIN combines a diverse landscape of network elements, both software-defined and legacy hardware, and unifies them under a common management interface, presenting them as *pluggable services* to the experimenter.

Keywords: future internet, testbed support, experiment services.

1 Introduction

The Internet's core architecture, including its routing and addressing scheme currently aches under the pressure of the continuing rapid expansion of the net, as well as new applications and usage patterns. Useful incremental improvements have not seen wide adoption in recent years (e.g., IPv6, DiffServ, DNSSec). This has sparked renewed interest in disruptive approaches for re-architecting the Internet core, for instance in a multitude of *Clean Slate Networking* initiatives around the globe.

Due to the diversity of these approaches, Future Internet experimentation requires a flexible testbed that caters to a wide range of requirements. Such experiments may require programmable, virtualized and non-virtualized hardware (e.g., PC servers), software-configurable hardware (e.g., NetFPGAs), and flexibly configurable logical and physical topologies. Experimentation also requires a sound infrastructure of support, including the generation of realistic traffic patterns and proper modeling of user expectations and experiences.

This proposed poster introduces BERLIN, an experiment platform tailored to the requirements of Future Internet research. BERLIN combines a diverse landscape of network elements, both software-defined and legacy hardware, unifies them under a common management interface, and presents them as pluggable services to the experimenter. We first present its architecture and the core services provided, then examine some use-cases of successful Future Internet experimentation in the lab. We believe that our poster can contribute valuable insights to the discussion about the primitives, components, and services required to successfully enable Future Internet experiments.

T. Magedanz et al. (Eds.): TridentCom 2010, LNICST 46, pp. 602–604, 2011.

2 Architecture

BERLIN uses a three-layered architecture with a multitude of different *experimental devices* managed by a unified *management platform*. On this foundation rests the *pluggable service* infrastructure, offering combinations of hardware, software, and configurations as pre-built and customizable services to the user.

BERLIN contains a highly diverse collection of experimental devices including routers, switches, and traffic-generating servers. These devices differ widely in their feature set, performance, and control interfaces. Terminal servers and SNMP controllable power interfaces provide additional *out-of-band management*.

This varied landscape of devices is managed by our custom software management system, the *Labtool*. *Labtool* presents a unified, vendor-agnostic interface to the experimenter for device reservation, configuration, interaction (e.g., console access, power management), and topology management. The Labtool also maintains a complete and historically versioned picture of the physical and logical network testbed topology. Labtool integrates with an automated system configuration and disk imaging tool which allows disk images and router and switch configurations to be deployed quickly onto arbitrary experimentation devices.

This unified management allows BERLIN to offer pre-configured combinations of hardware and software as higher-level *pluggable services* as depicted in Figure 1. These fulfill many common requirements, e.g., traffic generation, monitoring and capturing, network emulation, NetFPGA packet processing, and virtualization services. These pluggable services allow researchers to quickly establish an experimental setup with most of the required services from prebuilt components. For instance, an experimenter may want to evaluate a new router primitive implemented as a *NetFPGA* program, then require *self similar background traffic* to be generated and routed through the *NetFPGA*, apply emulated WAN line delay characteristics, and finally capture packet level traces at several points in the experiment.

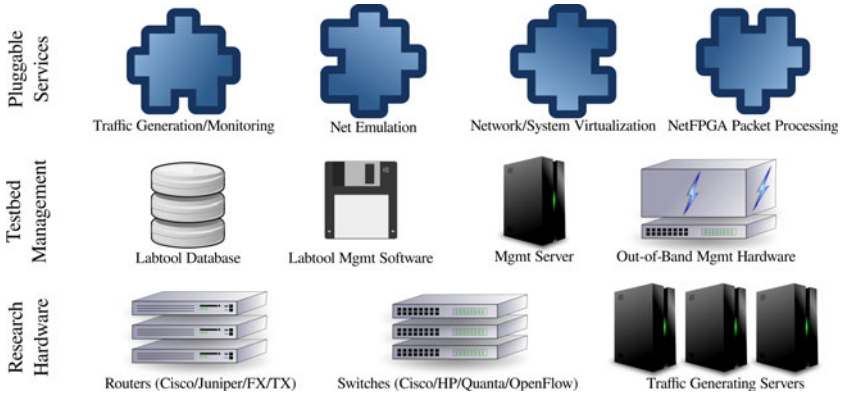


Fig. 1. BERLIN Testbed Architecture

3 Case Studies

We next present cases where BERLIN supports Future Internet experimentation.

HAIR, the *Hierarchical Architecture for Internet Routing* addresses the limits of current Internet routing table growth. It better supports mobility, traffic engineering, and multipath routing by separating the *locator and identifier* elements of the addressing scheme. HAIR places routing elements into *hierarchical name-spaces* to localize the effects of routing changes and improve routing table scalability. BERLIN enables us to devise an experiment to evaluate the HAIR software-based routing elements, all of which utilize special network packet formats. Virtualized hosts with the the Click modular software router as provided by BERLIN can support these special packets. The BERLIN topology management enables simple reconfiguration of the underlying IP network onto which HAIR elements are overlaid. Results from the prototype evaluation show the feasibility of hierarchical routing with mobility utilizing this model of split locator and identifier [1].

QoE/Virt Providing good *Quality of Experience*(QoE) for end users remains a challenging problem on the Internet. Especially *virtualized networks* pose novel challenges, as multiple virtual networks may compete for the same physical resources and the quality of the isolation provided by virtualization platforms varies. Accordingly, we devise a combined experiment on our testbed, confronting *multimedia VOIP traffic* with bursty, self-similar *background traffic* as found on the Internet, and routing both over a *virtualized substrate* based on XEN and OpenFlow with advanced debugging and flow management capabilities. Our results indicate that the advanced troubleshooting and management capabilities in the virtualized setup result in improved QoE, even though the level of isolation provided by the virtualization is limited [2].

4 Summary and Outlook

We have presented the architecture of BERLIN and some case studies underlining its potential as an experimentation platform. In the future, we plan to further extend BERLIN's flexibility, specifically to automatically integrate OpenFlow networks into experiments. To support experiments beyond the size of a single physical testbed we will investigate extending the Berlin testbed architecture to other sites, as well as federating BERLIN with other Future Internet testbeds, while maintaining the architecture of pluggable services that is paramount to its flexibility.

References

1. Feldmann, A., Cittadini, L., Mühlbauer, W., Bush, R., Maennel, O.: Hair: hierarchical architecture for internet routing. In: ACM ReArch Workshop (2009)
2. Wundsam, A., Mehmood, A., Feldmann, A., Maennel, O.: Network troubleshooting with shadow vnets. In: Proc. ACM SIGCOMM Demo Session (2009)

Multiaccess NetInf: A Prototype and Simulations

Teemu Rautio, Olli Mämmelä, and Jukka Mäkelä

VTT Technical Research Center of Finland,
Kaitoväylä 1, 90571 Oulu, Finland
firstname.lastname@vtt.fi

Abstract. This work presents operation and the results obtained in an information-centric multiaccess content distribution prototype using both real laboratory environment tests and simulation experiments with OMNeT++. We present the basic objectives and mechanisms of our solution and evaluate it against a standard BitTorrent. The results show that our prototype based on an information-centric approach can reach high performance gains in both static and mobile scenarios, is scalable and can reduce the amount of inter-network traffic.

Keywords: multiaccess, network of information, content distribution, BitTorrent, information-centric.

1 Introduction

Network of Information (NetInf) [1][2] is an information-centric networking approach being developed by the FP7 4WARD project (see www.4wardproject.eu). The aim of the NetInf project is to provide a communication infrastructure that is more suitable to content distribution applications than the current client-server model. In an information-centric approach the users generally do not care where the information is located, as long as the information is valid. The information should not be tied to specific locations but it could reside anywhere in the network. It is also anticipated that the Future Internet supports mobility and multiaccess.

We implemented and evaluated a new BitTorrent [3] based content distribution system, namely Multiaccess NetInf [4], for showing the benefits of using several network accesses simultaneously (multiaccess). Our prototype provides also a method for downloading content locally from own network. These characteristics should be supported even while moving across several overlapping wireless networks, such as Wi-Fi, 3G or WiMAX. The prototype is evaluated both in a static and mobile scenario and simulation experiments are used to show the scalability of the solution. The content is distributed using the BitTorrent protocol, as a reference, and in the mobile scenario multiaccess NetInf is compared against BitTorrent over Mobile IPv6 [5]. The results show that a multiaccess NetInf solution can reach high performance gains, reduce the amount of cross-network traffic and is also scalable.

2 Operation and Evaluation

Basic scenario of Multiaccess NetInf is illustrated in Fig. 1. The scenario involves three different networks: Core Network and two access networks, which both have

T. Magedanz et al. (Eds.): TridentCom 2010, LNICST 46, pp. 605–608, 2011.

wireless access points (Wi-Fi). In the scenario, Global NetInf name resolution point takes care of local name resolution points, whereas local resolution point maintains a list of local content sources. A NetInf node represents our content downloader and it can be connected to one or more wireless networks. Employment of NetInf Notification Service (NNS) enables indication of new access networks, while NetInf node is moving across different wireless networks. In this prototype implementation NNS was realized by using the trigger management framework [6].

The system operates as follows. NetInf node requests global name resolution from a global resolution point, which responds with the location of local name resolution point(s). When the node makes a content request to local resolution point(s), they report the location of the local sources and the NetInf node can ask content locally from them. After the node detects a new network where it is connected to, it asks for the existence of a local copy of the content from the Global resolution point through NNS. The global resolution point responds with the location of the local resolution point. Node can now request sources from it and also start the content retrieval from the sources located also in the new network.

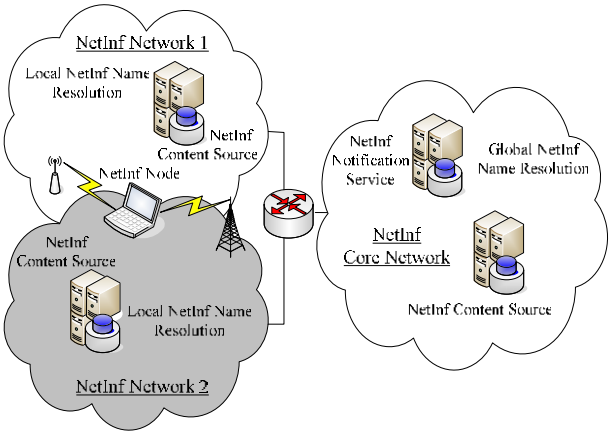


Fig. 1. Basic scenario of Multiaccess NetInf content distribution

We compared single access standard BitTorrent and Multiaccess NetInf in a real laboratory environment and also with simulations. The NetInf node was connected, in this static scenario, to two networks over two interfaces simultaneously. In laboratory experiments, the setup was exactly the same as in Fig. 1, but in simulations the number of access networks and NetInf nodes was larger, as is certain. Simulations were made with OMNeT++, INET framework [7], OverSim [8], and a BitTorrent module [9].

Table 1 shows outperformance of multiaccess NetInf. In laboratory experiments reduction in download durations was 41 %, when comparing multiaccess NetInf to single access BitTorrent. Respectively in simulations, the portion was 32 %. Moreover, we reduced inter-network traffic very drastically: decrease of inter-network traffic was almost 100 % in prototype measurements and 98 % in simulations.

Table 1. Outperformance in download durations and in Cross-Network traffic

Reduction in median (ten runs):	Prototype:	Simulations:
Download Durations	41 %	32 %
Inter-Network traffic / Packets in Backbone routers	100 %	98 %

In addition, we evaluated the performance of our prototype implementation in low mobility scenario with laboratory experiments. Comparison was made between Multiaccess NetInf and BitTorrent over Mobile IPv6. The scenario was the same as in Fig. 1, but the user movement affects to the availability of different access points in the following way. NetInf node is connected the first 15 s only to network 1. From 15 s to 25 s it is connected to both networks and during the rest of the download only to network 2.

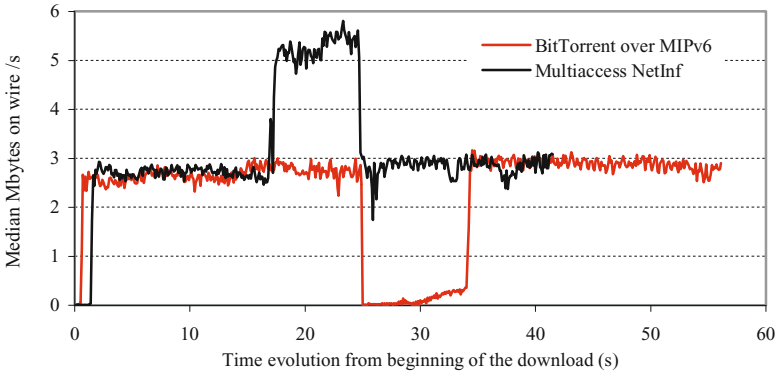


Fig. 2. Throughput in mobile and multiaccess scenario

Again, our system was faster than standard BitTorrent (29 % reduction in median download duration). Also inter-network traffic was decreased almost to none; total reduction was almost 100 %. Evolution of median throughput in both experiments (with BitTorrent and Multiaccess NetInf), represented in Fig. 2, provides an explanation for the superiority of the prototype; it can use two interfaces simultaneously in the middle of the download (from 15 s to 25 s).

Conclusion

This short paper described briefly the results and the testbed of the laboratory experiments and simulations. Results and evaluation showed clearly the outperformance of multiaccess NetInf comparing to single access BitTorrent. In static scenario, reduction of median download duration was 41 % with prototype and 32 % in simulations. Also we reduced inter-network traffic almost by 100 % in both

evaluations. Respectively, our prototype implementation outperformed BitTorrent over MIPv6 in mobility scenario; reduction in median download duration was 29 % and inter-network traffic was reduced almost by 100 %.

Acknowledgement

The work covered in this paper has been sponsored by VTT Technical Research Center of Finland, the ICT 7th Framework Programme Integrated Project 4WARD (partially funded by the Commission of the European Union) and TEKES as part of the Future Internet program of TIVIT (Finnish Strategic Centre for Science, Technology and Innovation in the field of ICT).

References

1. Ahlgren, B., D'Ambrosio, M., et al.: Design considerations for a Network of Information. In: Proc. ACM CoNEXT, Madrid, Spain, Article No. 66, pp. 1–6 (December 2008)
2. Dannewitz, C., Pentikousis, K., Rembarz, R., Renault, E., Strandberg, O., Ubillos, J.: Scenarios and research issues for a Network of Information. In: Proc. Fourth International Mobile Multimedia Communications Conference (MobiMedia), Oulu, Finland (July 2008)
3. Cohen, B.: Incentives Build Robustness in BitTorrent. In: Proc. 1st Workshop on Economics of Peer-to-Peer Systems, Berkeley CA, USA (May 2003)
4. Ahlgren, B., D'Ambrosio, M., et al.: Second NetInf Architecture Description. 4WARD project deliverable D-6.2, Work Package 6, `http://www.4ward-project.eu/index.php?s=Deliverables`
5. Johnson, D., Perkins, C., Arkko, J.: Mobility Support in IPv6., IEFT RCF 3775
6. Mäkelä, J., Pentikousis, K., Kyllönen, V.: Mobility Trigger Management: Implementation and evaluation. International Journal of Communications, Network and System Sciences 2(3) (2009)
7. INET Framework for OMNeT++ 4.0, URL: `http://inet.omnetpp.org/`
8. Baumgart, I., Heep, B., Krause, S.: OverSim: A flexible overlay network simulation framework. In: Proc. of the IEEE Global Internet Symposium, Anchorage, AK, USA, pp. 79–84 (January 2007)
9. Katsaros, K., Kemerlis, V.P., Stais, C., Xylomenos, G.: A BitTorrent module for the omnet++ simulator. In: Proc. 17th Annual Meeting of the IEEE International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS), London, UK, pp. 360–370 (2009)

Prototyping with the Future Internet Toolbox*

C. Dannewitz, T. Biermann, M. Dräxler, F. Beister, and H. Karl

University of Paderborn, Research Group Computer Networks
{cdannewitz,thobie,draexler,azamir,hkarl}@mail.upb.de

Abstract. Prototyping future Internet technologies is an important but complicated task. To ease this process, we have developed the *Future Internet Toolbox (FIT)*. In this paper, we demonstrate how it facilitates prototyping in practice by presenting three scenarios that have been implemented using *FIT*.

1 Introduction

Currently, a lot of research focuses on future Internet technologies. To simplify the process of prototyping and evaluating new concepts and protocols, we developed the Future Internet Toolbox (FIT) – a collection of 4 frameworks covering information-centric networking, data transport, naming, and name resolution.

We have introduced FIT in detail in [1,2]. This paper now focuses on three specific use cases where FIT helped us prototyping new networking features: efficient and secure information-centric data dissemination (Sec. 3.1), rapid information-centric application development (Sec. 3.2), and session mobility for video streaming (Sec. 3.3). We shortly revisit the FIT frameworks in Sec. 2.

2 FIT Frameworks

2.1 Information-Centric Networking and Secure Naming

Several Information-Centric Network (ICN) architectures have been proposed lately (see [2] for details) to overcome the inefficiencies of today's Internet architecture with respect to information dissemination and information handling. The ICN framework supports and accelerates the design and evaluation of ICN architectures. It provides three main aspects: (1) a generic, adaptable *node structure* for building custom information-centric nodes, (2) implementations of various *components* that can be used to compose such a node, and (3) a testbed for interconnecting those nodes into an overall ICN architecture.

The ICN framework is closely connected to the naming framework. The naming framework supports a wide variety of naming schemes, including complex ones involving security-related features like data integrity checking. At the same time, it minimizes the implementation overhead of new naming schemes.

* The research leading to these results has been conducted in the 4WARD project and received funding from the European Community's Seventh Framework Programme (FP7/2007-2013) under grant agreement no. 216041.

T. Magedanz et al. (Eds.): TridentCom 2010, LNICST 46, pp. 609–611, 2011.

2.2 Data Transport and Name Resolution

The data transport framework bases on the Generic Path (GP) architecture that has been developed in the European research project 4WARD. This architecture's goal is to overcome limitations of the ISO/OSI architecture, which is mainly the difficulty to introduce new functionality into today's networks.

This has been achieved by designing the framework for flexibility and extensibility from the beginning. It is service-oriented, i.e., Entities ("layers") can be arbitrarily composed based on their required and provided services (no static stack determined by the technologies anymore). Another key issue are the unified structures. Entities and Endpoints (the termination of a communication path) all have the same interfaces to ease cross-layer networking. Additionally, reusable components like Entities and Endpoints, or mechanisms like name resolution are made explicit to speed up developing new features. As a result, name resolution uses the same mechanisms on all different levels of networking.

3 Use Cases

3.1 Efficient and Secure Information-Centric Data Dissemination

In this scenario, we demonstrate efficient and secure data dissemination mechanisms for the future Internet, based on the FIT-based Network of Information (NetInf) prototype [1]. We used the ICN framework to build information-centric NetInf nodes, both infrastructure and client nodes. Each NetInf node can cache and provide data to other NetInf nodes. In addition, they provide a name resolution service to resolve flat NetInf names into download locations.

A Web browser, extended via a NetInf plugin to allow communicating with the NetInf infrastructure, handles links to flat names instead of URLs by triggering a name resolution. During resolution, an appropriate download location is selected by the NetInf nodes; the browser downloads and displays the requested information. In addition, the client NetInf node checks the data's integrity using the FIT naming framework and caches the data locally. This enables to provide the data to other nodes to increase availability in case of network disruption and to reduce Internet traffic. The whole process is transparent to the user.

3.2 Rapid Information-Centric Application Development

Building information-centric applications involves many similar components like information model, distributed storage, efficient and secure data dissemination, and notification services. Building such applications and the required infrastructure on a global scale based on today's Internet architecture is often cumbersome as most components have to be rebuilt and/or reintegrated for each project.

This scenario illustrates how even complex information-centric applications and the corresponding infrastructure can easily be developed using building blocks (*Information Model*, *Naming Scheme*, *Event/Storage/Search Service*) of the FIT-based NetInf prototype. These blocks enabled us to rapidly develop a

globally scalable collaborative editing and context-awareness application. Specifically, we developed a shopping application that informs the user about available products in near by shops from his shopping list. The shopping list can be edited collaboratively by multiple users. This is implemented by storing the data in a NetInf-specific data structure and utilizes the Event Service to subscribe the applications and a specialized Search Service to any changes performed on the shopping lists, inventory lists, and the user's geo-locations.

3.3 Session Mobility for Video Streaming

In this scenario we focus on FIT's data transport part and show how session mobility can be implemented for a live video stream, i.e., client and server are moved to different machines during runtime.

For this, we first added support for session mobility to the `AbstractEntity`. This covers functions to serialize the Endpoint of an existing GP and to transfer this serialization to another Entity. The GP management functions have been extended to relocate the GP alongside the Endpoint transfer to the new Entity. All these mechanisms are generic, i.e., they can be used for any GP type.

Second, we created a `VideoEndpoint` that is instantiated by a `VideoEntity`. This Endpoint uses the VLC media player to generate and display video streams. In particular, the `VideoEndpoint` contains the current state of VLC (playback position and source video file), which is eventually contained in the Endpoint serialization. This permits session mobility by relocating the video Endpoints on both client and server side. The server-side session mobility is especially useful for dynamic resource allocation for single video streams; a problem that cannot be solved by server migration using virtualization.

4 Conclusion

Experience from implementing the three scenarios with the FIT frameworks shows that FIT simplifies prototyping significantly. Due to its generality, it is useful for many different projects as it reduces redundant implementation of basic testbed functions and provides ready-to-use building blocks. This way, new, small components can be complemented with an overall network architecture.

References

1. Biermann, T., Dannewitz, C., Karl, H.: FIT: Future Internet Toolbox. In: Magedanz, T., et al. (eds.) Tridentcom 2010. LNICST, vol. 46, pp. 444–453. Institute for Computer Sciences, Social Informatics and Telecommunications Engineering (2010)
2. Biermann, T., Dannewitz, C., Karl, H.: FIT: Future Internet Toolbox — extended report. Technical Report TR-RI-10-311, University of Paderborn (February 2010)

Sensei-UU — A Relocatable WSN Testbed Supporting Repeatable Node Mobility

Frederik Hermans, Olof Rensfelt, Per Gunningberg,
Lars-Åke Larzon, and Edith Ngai

Department of Computer Science, Uppsala Universitet
{frederik.hermans,olofr,perg,lln,edith.ngai}@it.uu.se

Abstract. General purpose testbeds for Wireless Sensor Network (WSN) systems are often deployed in lab environments. This means that the testbed environment may differ considerably from the target environment of a system with respect to e.g. radio interference and sensory input. Furthermore, many WSN scenarios include mobile nodes, such as mobile sensors and mobile sinks, whose movements can affect the experimental results considerably. Both these features pose a challenge to achieving consistent and reliable experimental results during the evaluation and testing of a WSN system.

To attack this challenge, we are developing the Sensei-UU WSN testbed. It follows a light-weight, distributed design, so that instances of the testbed can be easily relocated; this enables experimenters to evaluate a system in different environments. To support scenarios including mobile nodes, Sensei-UU uses robots which carry out mobility patterns defined by the experimenter.

Our evaluation shows that our design supports repeatable experiments in various environments, even when the experiments involve mobile nodes.

1 Introduction

WSN systems are deployed in a variety of environments, ranging from e.g. forests, over hospitals, to industrial facilities, whereas many general purpose testbeds for WSNs are deployed in lab environments. In consequence, the behavior of a system in a testbed may be different from its behavior in situ, because of the different radio environments and sensory inputs. From this we conclude that it is desirable to be able to relocate a testbed. The presence of mobile sensor nodes in a WSN scenario further complicates the testing process, as the testbed needs to support node mobility, keep track of mobile nodes, and ensure repeatability of movements of mobile nodes. The latter is necessary to be able to draw conclusive results from repetitions of similar experiments.

We are developing the Sensei-UU Wireless Sensor Network testbed to attack these challenges. Our testbed differs from existing WSN testbeds in two important aspects. First, Sensei-UU may use a wireless control channel. This makes the testbed independent of existing network infrastructure and makes it easy to relocate an instance of the testbed into the intended target environment of the

T. Magedanz et al. (Eds.): TridentCom 2010, LNICST 46, pp. 612–614, 2011.

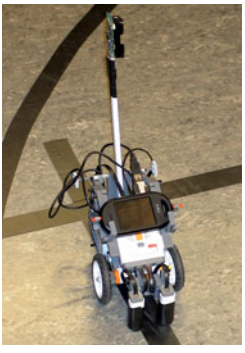


Fig. 1. A mobile node

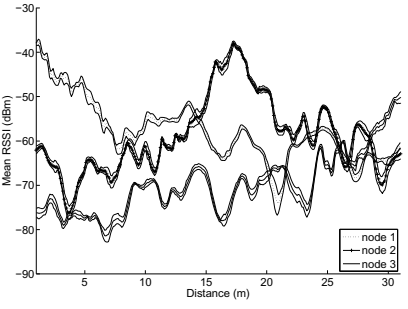


Fig. 2. Received signal strength from stationary nodes to a mobile node, averaged over multiple runs of one experiment

system under test. This flexibility distinguishes Sensei-UU from static testbeds, such as TWIST [2]. Second, Sensei-UU supports experiments involving mobile sensor nodes. Robots carry sensor nodes and follow pre-defined mobility patterns. In contrast to existing WSN testbeds with mobility support, Sensei-UU does not rely on existing infrastructure for node localization and robot navigation, but instead uses a simple line-following approach.

2 Design

Our testbed follows a light-weight, distributed design to make it largely independent of existing infrastructure. Every sensor node – we currently use TelosB [1] nodes – in the testbed is attached to one *sensor host*. A sensor host is responsible for managing the attached sensor nodes; it provides a means to reprogram attached nodes or collect log data from them. We are using WLAN access points as stationary sensor hosts, and OpenMoko FreeRunner [5] mobile phones as sensor hosts for mobile nodes. The FreeRunner's USB host capabilities make it possible to attach one or more sensor nodes to it. Our design is based completely on off-the-shelf components to make our testbed reproducible for other researchers.

Robots in the testbed are based on Lego NXT kits [4]. Robots navigate by following a tape that is laid out on the floor by the experimenter prior to the experiment (Fig. 1). While this approach constrains robot mobility, it increases the independence of the testbed of existing infrastructure such as ceiling-mounted cameras as they are used in, e.g., Mobile Emulab [3]. Each robot carries one mobile phone acting as a sensor host and one or more sensor nodes.

Sensor hosts forward log data from the sensor nodes to a *site manager* over a wireless IEEE 802.11b/g control channel. The site manager in turn delivers the data to an experimenter running a *monitor* software. The monitor also provides the experimenter with an interface to reprogram sensor nodes and control experiment parameters. The use of the site manager as an indirection point facilitates remote access to the testbed.

3 Evaluation

A wireless control channel may potentially interfere with the sensor nodes' radio if both operate in the same frequency band. To ensure that this is not the case in our testbed design, we have measured the interference between IEEE 802.11b/g and IEEE 802.15.4, the radio technology used by TelosB nodes. We have found that if both technologies use different frequencies, and if any two 802.11b/g and 802.15.4 transceivers are separated by at least 1 m, interference is negligible [6].

We further assessed the repeatability of experiments involving mobile nodes. We set up an experiment in which one mobile sensor node constantly measures the quality (in terms of RSSI) of its radio links to three stationary sensor nodes. Figure 2 shows the quality of the respective links to the stationary nodes averaged over multiple runs. For each link, the upper and lower line represent one standard deviation. The standard deviation characterizes the differences in link quality between different runs. From the fact that the standard deviation is reasonably low, we conclude that our testbed allows for repeatable experiments. We have repeated the described experiment in different environments, one of which was an anechoic chamber. There we also found the variance between runs to be low, from which we derive that our testbed itself does not have a substantial impact on the system being tested.

4 Conclusion and Future Work

We have presented our WSN testbed, called Sensei-UU, that is designed to be easily relocatable and to support mobile nodes.

We are currently considering to extend our testbed to incorporate mobile phones as part of the system to be tested. This would allow us to evaluate hybrid networks of sensor nodes and mobile phones. We will release the software components in Sensei-UU under an open source license soon.

References

1. Crossbow Technology, Inc., `http://www.xbow.com/Products/Product_pdf_files/Wireless_pdf/TelosB_Datasheet.pdf`
2. Handziski, V., Köpke, A., Willig, A., Wolisz, A.: Twist: a scalable and reconfigurable testbed for wireless indoor experiments with sensor networks. In: REALMAN: Proc. of the 2nd Intl. Workshop on Multi-hop Ad Hoc Netw. (2006)
3. Johnson, D., Stack, T., Fish, R., Montrallo Flickinger, D., Stoller, L., Ricci, R., Lepreau, J.: Mobile Emulab: A Robotic Wireless and Sensor Network Testbed. In: INFOCOM: Proc. of the 25th IEEE Intl. Conf. on Comp (2006)
4. Lego NXT, `http://mindstorms.lego.com`.
5. OpenMoko Freerunner, `http://www.openmoko.org`
6. Rensfelt, O., Hermans, F., Ferm, C., Gunningberg, P., Larzon, L.-Å.: Sensei-UU: A Nomadic Sensor Network Testbed Supporting Mobile Nodes. Technical Report 2009-025, Department of Information Technology, Uppsala University (October 2009)

A Demonstration of a Management Tool for Assessing Channel Quality Information in Wireless Testbeds*

Apostolos Apostolaras, Vasileios Miliotis, Nikolaos Giallelis, Dimitris Syrivelis, Thanasis Korakis, and Leandros Tassiulas

Department of Computer and Communication Engineering
University of Thessaly
Centre for Research & Technology Hellas (CERTH)
Volos, Greece

Abstract. The gradually growing need for testbed use so as networking algorithms to be validated in real environment, has given rise to optimal utilization of testbed resources. Towards this direction, we present a new management tool that is used for assessing channel quality information in wireless testbed deployments. NITOS Connectivity Tool retrieves data concerning link quality measurements, for providing testbed users with useful information about choosing nodes that occasionally satisfy the requirements (link quality, connectivity) needed, for their experiments. NITOS connectivity tool is a full-fledged managerial tool that exploits testbed utilization by letting testbed users have a complete view about testbed's nodes. This tool allows a more sophisticated way to optimally choose network resources of a testbed.

Lab's website: *"http://nitlab.inf.uth.gr"*[2]

This demo paper is related to the paper entitled *"Towards Maximizing Wireless Testbed Utilization using Spectrum Slicing"* accepted for publication on TridentCom 2010.

Keywords: Testbed, Management, Link Quality.

1 Introduction

A testbed experiment deployment needs node allocation that should satisfy certain topology and link quality requirements. Moreover, the erewhile link quality information gives a testbed user the advantage of properly evaluate the observed experiment/algorithm performance. Since NITOS testbed deployment is not RF isolated, the link quality between any pair of nodes may unexpectedly vary at any point in time due to external interference. For this reason the static distribution approach, that is used in RF isolated wireless testbeds[1][4], is not efficient

* The research leading to these results has received funding from the European Community's Seventh Framework Programme (FP7/2007-2013) under grant agreement n^{o}224263.

T. Magedanz et al. (Eds.): TridentCom 2010, LNICST 46, pp. 615–618, 2011.

for these deployments. Therefore, there is a growing need for updated information in terms of densely measurements of link quality, that will bring in a long term, a more accurate channel quality estimation.

2 NITOS Connectivity Tool

In this demo, we will present a management tool for assessing channel quality information. The tool was developed for NITOS testbed and measures channel connectivity among wifi interfaces. NITOS is a wireless testbed, located in Volos, Greece with 15 nodes, each node equipped with two wifi interfaces. It is deployed on Computer & Communication Dept. University of Thessaly building. NITOS testbed topology is depicted on the left part of Fig. 1. Although the NITOS connectivity tool was developed on NITOS testbed, it can be adapted for use on any wireless testbed, with minor modifications. NITOS connectivity tool is a NITOS scheduler[3] part, which is used for resource allocation on NITOS testbed.

We have implemented NITOS connectivity tool, based on TLQAP [5], which is a protocol, that is used to assess interconnection topology and link quality by estimating packet delivery ratio (PDR) in downlink communication at each node's wifi interface for all requested channel, rate and transmission power combinations. Specifically, TLQAP builds a measurement history log and creates a channel utilization profile, and stores that information in a database that is used for link quality information retrieval by NITOS connectivity tool.

NITOS Connectivity Tool is comprised of three entities: a web interface, a database and a set of .dot scripts. The web interface is the interactive tool that an experimenter uses to choose testbed nodes for some time and it is depicted on the right side of Fig. 1. A user enters the NITOS Connectivity tool through NITLAB's wiki *http://nitlab.inf.uth.gr*. Now, the user has the ability to navigate through testbed's site and select *"Scheduler → Topology-Connectivity"* menu.

Fig. 1. NITOS Connectivity Tool

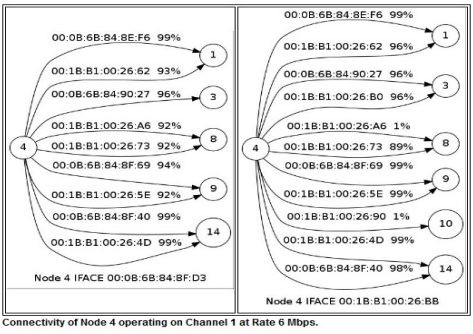


Fig. 2. Link Quality for node 4

Specifically, at first the user selects through web interface, by using a drop-down menu, a sender node that he/she wants to check and might want to use in testbed. Then the user is prompted to select an operating frequency among IEEE 802.11 communication standards 802.11a/b/g and selects the operating rate. Then, he/she goes to the final step where he submits his/her query concerning the link quality of a certain node. In sequence, the tool seeks to a database where the channel quality results are stored and retrieves the information that corresponds to the particular query. This information with the use of .dot files that are used to depict graphs are presented to the users. On Fig. 2, the downlink communication link quality for node 4 among its neighboor nodes is illustrated. Each node is indicated by a circle and the PDR of each link is reported upon edges that indicate link connectivity to certain node's wifi interfaces. Those interfaces are reported with their MAC addresses, with an arrow showing to the node where they belong.

3 Conclusion

In this demo paper we present a wireless connectivity tool that enables better utilization of testbed under effective resource exploitation. In particular, users can select nodes and observe link quality by navigating through a web interface that is built as an extension tool of NITOS scheduler and retrieves information from TLQAP. Thus, users can choose those nodes that satisfy their experiment requirements.

References

1. ORBIT-Radio Grid, `http://www.orbit-lab.org`
2. UTH NITLab, `http://nitlab.inf.uth.gr`

3. Anadiotis, A.C., Apostolaras, A., Syrivelis, D., Korakis, T., Tassiulas, L., Rodriguez, L., Seskar, I., Ott, M.: Towards Maximizing Wireless Testbed Utilization using Spectrum Slicing. In: Magedanz, T., et al. (eds.) TRIDENTCOM 2010. LNICST, vol. 46, pp. 299–314. Institute for Computer Sciences, Social Informatics and Telecommunications Engineering (2010)
4. Raychaudhuri, D., Seskar, I., Ott, M., Ganu, S., Ramachandran, K., Kremo, H., Siracusa, R., Liu, H., Singh, M.: Overview of the ORBIT Radio Grid Testbed for Evaluation of Next-Generation Wireless Network Protocols
5. Syrivelis, D., Anadiotis, A.-C., Apostolaras, A., Korakis, T., Tassiulas, L.: Tlqap: A topology and link quality assessment protocol for efficient node allocation on wireless testbeds. In: Proceedings of the 4th ACM International Workshop on Experimental Evaluation and Characterization (2009)

CentMesh: Modular and Extensible Wireless Mesh Network Testbed*

J. Lim[1], P. Pathak[1], M. Pandian[2], U. Patel[1], G. Deuskar[1], A. Danivasa[1], M.L. Sichitiu[2], and R. Dutta[1]

[1] Dept. of Computer Science,
North Carolina State University, Raleigh, NC, USA
[2] Dept. of Electrical and Computer Engineering
North Carolina State University, Raleigh, NC, USA
dutta@csc.ncsu.edu

Abstract. In this paper we present the design of our wireless mesh network testbed (CentMesh), which facilitates experimentation as a service. CentMesh differs from other testbeds in terms of its modular, flexible and extensible design. The CentMesh software suite provides a modular programming library that can be modified and/or extended by the users of the testbed, allowing them to implement their own modules (e.g., routing, scheduling etc.). The basic services such as transport of control messages, broadcast, etc., are provided to experimenters by a set of system modules. Modularity allows the experimenters to implement only the part of network stack that they are interested in experimenting with, while reusing the other readily available CentMesh modules.

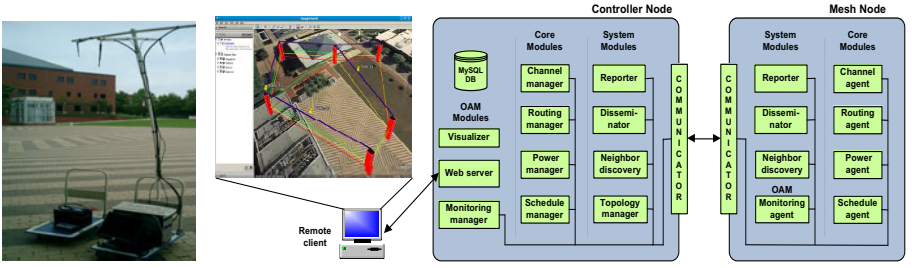
1 Introduction and Motivation

Even though wireless multi-hop networks have been the focus of considerable research efforts in last few years, conducting real-world experiments with wireless devices continues to be difficult. Most of the early efforts [1, 2, 3, 4, 5] in testbed design have focused on simplifying the process of experimentation and facilitating a certain amount of repeatability.

In this paper, we present the design and development of our wireless mesh network testbed *CentMesh*. The fundamental design principle of CentMesh is a clear separation between data transport, signaling and control and management algorithms. The modular structure of CentMesh software allows users to *plug-and-play* existing modules and add new modules. The advantage of such a design is two-fold. First, it allows the researchers to implement only the modules of the stack that they are interested in experimenting with. Second, newer modules needed for various experiments can be developed rapidly and integrated in the testbed independently of all the other testbed modules.

* This work is supported by the U.S. Army Research Office (ARO) under grant W911NF-08-1-0105 managed by NCSU Secure Open Systems Initiative (SOSI). The contents of this paper do not necessarily reflect the position or the policies of the U.S. Government.

T. Magedanz et al. (Eds.): TridentCom 2010, LNICST 46, pp. 619–621, 2011.



(a) CentMesh node (b) CentMesh software architecture

Fig. 1. CentMesh software and hardware

2 CentMesh: Hardware and Software

2.1 Hardware Components

CentMesh uses commodity hardware, each mesh node being a standard desktop computer. The CentMesh software is designed to run on any other hardware platform since we do not use any customized devices. Currently, the testbed consists of 10 mesh nodes deployed inside our department building. Each mesh node contains four Atheros cards, each connected on 4-to-1 miniPCI-to-PCI adapter. While performing the outdoor experiments, we place the mesh nodes on pushcarts (Fig. 1a).

2.2 Software Architecture

The CentMesh testbed uses a centralized control system where one of the mesh node is assigned the role of the controller. Different from other current state-of-art testbeds, we do not deploy any wired backhaul using Ethernet to connect and control the nodes. Instead control messages flow between nodes using one of the radios operating on a fixed channel. The design of CentMesh is modular and clearly separates functionality in control, management, and data planes. The modules are divided into system modules, core modules, and extension modules.

System Modules. System modules are implemented an indispensable part of the CentMesh software that is always present and running on mesh nodes irrespective of the experiment.

The testbed is managed cooperatively by several processes. Control and signaling traffic between processes either local or remote is channeled through a single process called the Communicator. Instead of the client/server model, the Communicator uses a publish/subscribe mechanism, which allows processes to send and receive messages based on the topic they are interested in, while hiding the details of the underlying message processing. Topic-based message filtering in the publish/subscribe model creates logical channels in the network-wide control path, which provides greater flexibility and scalability compared to a typical client-server model. Fig. 1b shows the Communicator as an entry/exit point between various processes running on different mesh nodes. The Communicator supports common multihop operations such as unicast reporting and flooding.

The neighbor discovery process on a mesh node periodically probes its neighbors and collects local neighborhood information. All nodes report the neighbor information to the controller.

For ease of recovery, in case of system crash, every CentMesh node contains three separate Linux installations in three separate disk partitions. The first partition (referred as fail-safe) contains a minimal set of functionality that allows users to remotely access the node and manually recover the node. The second and the third partitions are open to the researchers to perform their experiments.

Core Modules. Core modules are developed by the researchers as part of the experiment. Using the APIs from the CentMesh software, software modules are implemented in general as paired managers and agents; managers contain central intelligence/algorithm of the protocol, and agents perform actuating tasks based on the manager's decision.

We developed two sample core modules: a channel assignment module that uses a greedy edge coloring algorithm and a routing protocol with ETT (Expected Transmission Time) as the routing metric. Both modules can be replaced by experimenters with their own channel assignment and routing modules; however, a channel assignment and a routing module must always be present. An experimenter who is not focused on routing research can reuse the available routing module for the experiments. Similarly, experimenters can swap any other core modules.

Extension Modules. A separate network monitoring module (designed similar to core modules) collects network status information (e.g., installed routes, interface information, data rates, etc.) and reports it to the controller. The visualization program extracts the information provided by monitoring manager and graphically displays the network status using Google Earth.

3 Research Studies and Ongoing Extensions

CentMesh is designed to support many different types of of research projects. Some of the ongoing research projects include coarse-grain TDM scheduling and joint channel assignment-routing protocols. Furthermore, CentMesh also supports security research at various layers as well as mobility management and modeling. In the next phase of deployment, we plan to install the mesh nodes outdoors in our university campus. We soon plan to release the software suite under open source license to make it available to other researchers for their use.

References

1. ORBIT Lab, `http://www.orbit-lab.org`
2. Emulab, `http://www.emulab.net`
3. MIT Roofnet, http://pdos.csail.mit.edu/roofnet
4. TFA Rice Wireless Mesh, `http://tfa.rice.edu`
5. UCSB MeshNet, `http://moment.cs.ucsb.edu/meshnet`

NGN Trial Use and Test Site in Phuket, Thailand by National Telecommunications Commission (NTC) and Chula UniSearch

Supavadee Aramvith, Chaodit Aswakul, Chaiyachet Saivichit,
Prasit Prapinmongkolkarn, and Atiwat Aimdilokwong

Department of Electrical Engineering, Faculty of Engineering, Chulalongkorn University, Bangkok Thailand
supavadee.a@chula.ac.th
National Telecommunication Commission, Thailand
prasit@ntc.or.th

Abstract. National Telecommunication Commission of Thailand has initiated the pilot project which involved the trial and testbed of Next Generation Networks (NGNs) technology in Phuket, Thailand. This project consists of many new technologies implementation in the real environment. In order to implement the NGN technology successfully, we need to be able to summarize the outcomes of these deployment and prepare for improvement in the future. This project also extends the test to the communities in Phuket such as E- Learning applications.

Keyword: Next Generation Networks, NGN, Testbed, E-Learning.

1 Introduction

National Telecommunication Commission (NTC) has long been committed to perform its duty as Thailand's Telecommunication Regulator. Apart from regulating the telecommunication industry in Thailand, NTC has also contributed to various activities involving academics, society and people to tighten the digital divide in Thailand. One of the projects initiated by the NTC is the Next Generation Network Migration Study project which is commissioned to UniSearch of Chulalongkorn University to study the approach and feasibility of deploying NGN in Thailand. After the project has been completed, The commissioner Prasit Prapinmongkolkarn who has overseen this project since the beginning has a vision that this would be a big stepping stone of Thailand Telecommunication. He, then initiated the idea of the NGN Trial and Testbed Project in Thailand.

The NTC has contracted Chula UniSearch, a consulting arm of Chulalongkorn University (CU), led by Dr. Supavadee Aramvith and Dr. Chaodit Aswakul, to operate the project which is participated by CAT Telecom, Samart Telecom, AIS, NEC, Rohde&Schwarz, FORTH and ADC Communications. NGN technologies such as 3G, Wi-Max, VoIP, SIP server/Soft switch, VoD and FTTH are evaluated in the project along with many applications like Tele-Health, Tele- Medicine, Tele-Education, e-Government, NGN Classroom, Mobile TV and Mobile Banking.

T. Magedanz et al. (Eds.): TridentCom 2010, LNICST 46, pp. 622–625, 2011.

NTC On Feb 3rd 2009, the NTC officially opened the Next Generation Network (NGN) trial use and test site in Phuket. The main objective of this project is to study and evaluate the impact and benefits of the migration from legacy networks to NGN which can accommodate a variety of new services. Over 130 distinguished guests from the local administration, academics, healthcare services, as well as executives of telecom operators attended the opening ceremony of the NGN test site presided by General Choochart Promphrasid, the Chairman of NTC.

2 Telecommunications over Next Generation Networks

NGN is a packet switched IP-network that is capable of providing many new converged QoS (quality of service) assured services that users can have access to whether they are stationary or on the move. Convergence is at the core of NGN which delivers data, voice and multimedia applications on the same core network.

The test involved the following technologies and services.

1. High Speed Internet over WiMAX
2. Video Conference over FTTx
3. Voice over Internet Protocols (VoIP)
4. E-Learning

The tests were also concerned with compatibility and interoperability issues in this project between different versions of WiMAX and also between WiMAX and 3G CDMA 1X EVDO or 3G WCDMA 1800 MHz In-band Migration from GSM. In addition, the users' opinions and acceptance of NGN based applications such as VOIP, video streaming and e-learning at two middle schools etc. are also evaluated. Evaluations as well as technical data from this project will be beneficial and valuable to a telecom regulatory agency like NTC for its decision on regulatory policy and framework for NGN and NGN converged services. The NTC would be most happy to share this valuable experience as part of the implementation of the World Summit of the Information Society (WSIS).

3 Test Results

NEC brought their NGN IMS CORE to participate in the project. The Video on Demand (VoD) and Voice over IP (VoIP) on NGN IMS Core were tested. Users' satisfaction level data were then collected. For comparison, the VoD in both standard definition (SD) and high definition (HD) was tested with and without QoS control. The users were very satisfied with VoD with QoS control and VoIP however they evaluated that the quality of VoD without QoS control was quite poor.

FORTH and ADC Communications participated in the project with FTTH, one of the NGN wired access technologies. The throughput and link budget of GEPON technology was evaluated. Interoperability and compatibility were tested using equipments such as optical fiber, splitter module, optical line terminal (OLT) and optical network unit (ONU) from different manufacturers.

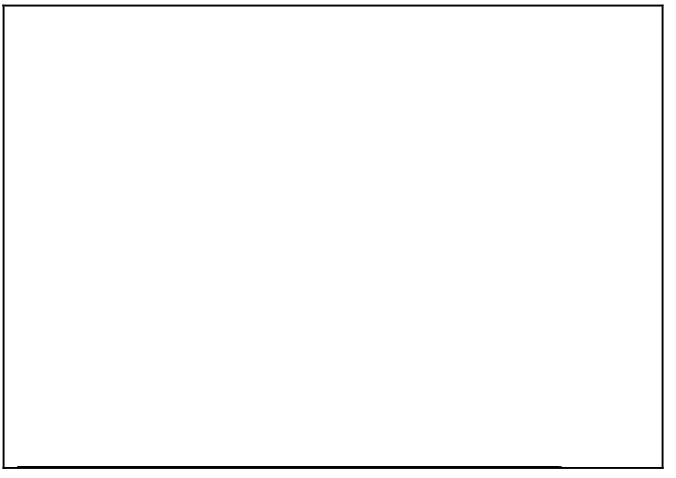


Fig. 1. IMS/NGN Testbed overview in the Project

The equipments from different manufacturers worked together very well and there were no compatibility issues whatsoever. With the OLT and ONU used, the power budget was about 30 dB and this should translate to a maximum distance of 80 km of the G.652D fiber without using any splitter. However the actual tests revealed that the maximum distance was limited to about 20 km due to the dispersion of G.652D fiber at a wavelength of 1,490 nm.

Compatibility tests were conducted between other NGN access technologies as well such as between WiMAX and 3G CDMA IX EVDO. Applications like VoIP, web browsing, FTP and video chat were tested on a notebook equipped with both WiMAX CPE (customer premise equipment) and CDMA 1X EVDO CPE. Users would drive- test these applications as they roamed in and out between WiMAX and CDMA 1X EVDO served areas. The results indicate that a notebook can switch services automatically from WiMAX to CDMA 1X EVDO and vice versa depending on the availability of the signal at the moment. However, all applications tested were disrupted and would not continue the operation. The least affected is the web browsing application because users can click the refresh button on the web page and continue on.

WiMAX, with three base stations operating at a frequency of 2.5 – 2.6 GHz, was also drive-tested to find out the distance that a base station could cover and the download/upload speed. Within the Phuket area, the distance on average from the base station to the CPE is about 2 km maximum but the distance covered was measured at almost 4 km for a line of sight operation or no obstructions in between. However, without having to adjust CPE's antenna orientation every time for optimal reception, the effective distance on average is about 500m in Phuket.

E-learning classrooms and workshops was conducted with ADSL and will be conducted again using NGN access technologies such as WiMAX and FTTH (GEPON). The results will be a comparison of quality of picture and students' satisfaction level data between a base-line case (with old access technologies) and an NGN case.

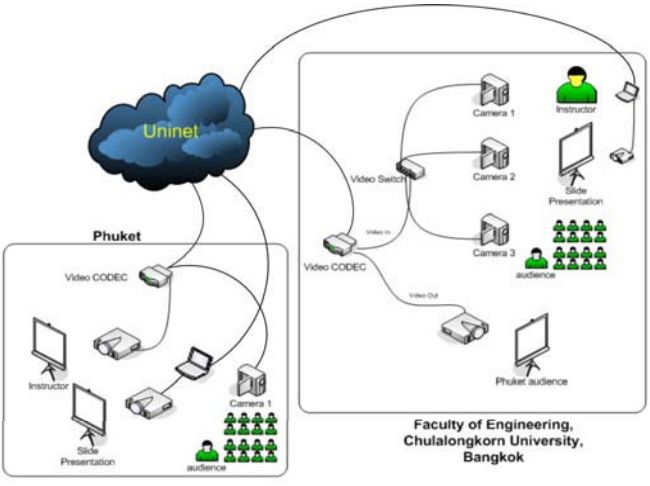


Fig. 2. E-Learning Classroom.

4 Conclusion

At the end of the project, several conclusions have been put together and proposed to the NTC. These suggestions from the study have lead to several urgent issues that need commitment from the NTC, for example, the format and standards of NGN technology to be deployed in Thailand. This requires attention from the regulation body; NTC and operators alike. The migration process requires high investment in terms of people, knowledge, technology and capital. The NTC has to take its role as a regulator as well as an influencer to create a momentum for changes. Furthermore, universal service obligation is one of the issues that could very much be enhanced from the deployment of NGN. Many services could be enabled with the technology. Eventually, the technology and services will be expanded to cover the underprivileged part of Thailand for the benefit all of our citizen.

References

1. ITU-T Y. 2001 General Overview of NGN SERIES Y: Global Information Infrastructure, Internet Protocol: Aspects and Next Generation Networks – Frameworks and Functional Architecture Models (2001), http://www.ictregulationtoolkit.org/en/Document.3243.pdf
2. NGN Proof Experiment in Thailand, 2008 – Study Report, NEC Corporation, March 27 (2009)
3. Final Report, NGN Test and Trial project in Phuket, Chula UniSearch submitted to the National Telecommunication Commission (2009)
4. IEEE 802.16 standard for wireless metropolitan area network, http://www.ieee802.org/16/
5. ITU-T H.323 Recommendation: Packet-based Multimedia Communications Systems, http://www.itu.int/rec/T-REC-H.323/e

A Component-Based Simulation Environment for Large-Scale Simulation of Home Network Systems

Takashi Okada[1,2], Marios Sioutis[1], Junsoo Kim[1], Junya Nakata[1,2], Yasuo Tan[1,2], and Yoichi Shinoda[1,2]

[1] Japan Advanced Institute of Science and Technology, Ishikawa Nomi Asahidai 1-1, Japan
[2] National Institute of Information Communications and Technology, Hokuriku Research Center, Ishikawa Nomi Asahidai 2-12, Japan

Abstract. In this paper, we propose a simulation environment for home network systems. The simulation environment consists of real houses, large-scale simulation testbeds and many simulation components such as environment, home appliances, electric power and human activity. These home simulations can install simulation components with different complexities. It enables the simulation to scale up the number of home on testbeds with limited nodes. As one of the results of the physical environment simulation, we compare the temperature measured in a real environment against that of a simulated. By being executed on large-scale testbeds and cooperating with real houses, the simulation achieves large-scale, realistic and real-time simulation of home network services, such as evaluating effects of an energy consumption service for a city of ten thousands households.

Keywords: simulation, emulation, home network, ubiquitous network, real-time, modeling, CFD.

1 Introduction

In this paper, we propose an ongoing research about realistic simulation environment for home network system, ubiquitous network system and context awareness system. The simulation environment consists of real houses, testbeds and home simulator based on simulation components. Figure 1 describes an overview of the simulation environment. The concept of the simulation environment is an emulation, which enables various home network applications to be executed with a native binary in real-time. The components of the simulation environment can be replaced with other simulation components or real objects.

The experimental environment is assumed to be StarBED[1] which is a large-scale networked testbed and on Home Network testbed. These testbeds provides functionalities to set up experimental configurations with supporting software (SpringOS[2], QOMET[3], RUNE[4]).

T. Magedanz et al. (Eds.): TridentCom 2010, LNICST 46, pp. 626–628, 2011.

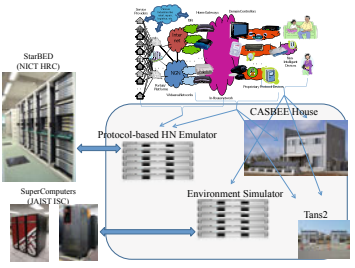


Fig. 1. Simulation Environment Overview

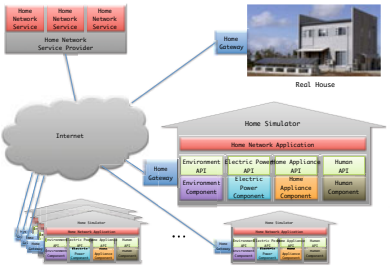


Fig. 2. Home Simulation

2 Home Simulator

The home simulator is a simulation of one house, which includes simulation components such as an environment, home appliances, electric power and human activity.The components are exchangeable for other simulation components. A combination of simulation components affects the complexity of the simulation, the speed of the simulation and the accuracy.

Figure 2 describes a simulation using a real house and home simulators. The home simulator provides common interfaces to access the home network. When an event occurs on the home, the home simulator changes the inner state of simulation components and sends appropriate control messages of home network if necessary.

3 Environment Simulation Component

The environment simulation component provides us for information related to physical space such as location, physical quantities (temperature, humidity, illumination), weather, disaster and so on. This information is measured as raw data by various sensors or is recognized as events by context awareness system.

One of the simulation components is the simulator of the physical environment properties, shown in figure 3. It calculates physical quantities using CFD (computational fluid dynamics). The calculated results can be accessed by the environment API to generate simulated sensor data in real-time.

As an evaluation of the simulation, we compared the simulation results of temperature with real experimental results. Figure 4 describes the comparison of the temperature of one room when air conditioner is cooling the room.

4 Future Works

In this paper, we proposed a simulation environment, which consists of real houses, testbeds and simulation components such as environment, home simulator, electric power and human activity. The simulation system executes in real-time to evaluate large-scale home network services.

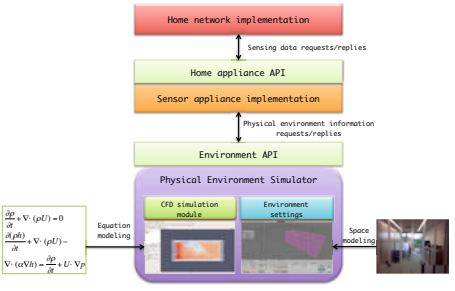


Fig. 3. Physical Environment Simulator

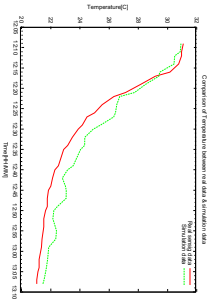


Fig. 4. Comparison of Temperature between real sensing data and simulation result

In future, we keep on developing more simulation components and enhancing the existing one. Currently, we are researching human simulation that models the behavior of residents using three statistical models. The first model is based on a statistical analysis of time schedules of human activities. These time schedules are generated by a profile and user preferences. The second model is a one-human activity model without contradicting behavior patterns. The third model is based on specific activities, which can be defined by the user. These three models can be used to simulate in order to human activity to evaluate home network services.

Finally we accumulate the simulation results to produce knowledge about various home network services.

References

1. StarBED project, `http://www.starbed.org/`
2. Miyachi, T., Chinen, K.-i., Shinoda, Y.: Automatic configuration and execution of internet experiments on an actual node-based testbed. In: Proceedings of the First International Conference on Testbeds and Research Infrastructures for the Development of Networks and Communities (Tridentcom) (February 2005)
3. Beuran, R., Nakata, J., Okada, T., Nguyen, L.T., Tan, Y., Shinoda, Y.: A Multipurpose Wireless Network Emulator: QOMET. In: 22nd IEEE International Conference on Advanced Information Networking and Applications (AINA 2008) Workshops, FINA 2008 Symposium, Okinawa, Japan, March 25-28, pp. 223–228 (2008)
4. Nakata, J., Miyachi, T., Beuran, R., Chinen, K., Uda, S., Masui, K., Tan, Y., Shinoda, Y.: Starbed2: Large-scale, realistic and real-time testbed for ubiquitous networks. In: The 3rd International Conference on Testbeds and Research Infrastructures for the Development of Networks and Communities (TridentCom 2007), Orlando, Florida, USA (2007)

XBurner: A XENebula-Based Native Traffic-Generation Platform

Toshiyuki Miyachi[1], Shinsuke Miwa[1], and Yoichi Shinoda[2]

[1] National Institute of Information and Communications Technology,
Asahidai 2-12, Nomi, Ishikawa, Japan
[2] Japan Advanced Institute of Science and Technology
Asahidai 1-1, Nomi, Ishikawa, Japan
{miyachi,danna}@nict.go.jp, shinoda@jaist.ac.jp

Abstract. There are two types of network traffic in experimental environments. One is traffic which is derived from target elements and the other one is background traffic which is derived from surrounding elements. There is very little knowledge on the relationships between new elements and existing elements; therefore, surrounding elements that seem to have no apparent relationship with the target elements should also introduced into the environmental environments. Therefore emulating background traffic is important to take realistic experimental results.

We propose XBurner—a platform that can be used to generate mass background traffic using a number of actual and native application software on virtual PCs. Driving actual application software is important to introduce real behavior of elements on real environment into experimental environments. The environment in this platform is developed using AnyBed and XENebula and is controlled by SpringOS.

1 Motivations

Background traffic has different characteristics from traffic for target elements. In most cases, one background connection doesn't share wide bandwidth, but many tiny connections totally occupy much bandwidth. Therefore, application software which send out background traffic don't require high performance machines and many of these connections with different source-destination pairs are required to make experimental environment more realistic.

Our approach is to build virtual network consists of many nodes for running many actual network software. Each virtual nodes with its own IP addresses can make many source-destination IP address pairs by their communication using actual network software. By using actual software, experimenters can introduce their own traffic pattern, their own implementations, and several implementations for one specification which have different behaviors.

2 Design and Implementation of XBurner

We use some existing proposals to generate native traffic from remotely-managed actual application software.

T. Magedanz et al. (Eds.): TridentCom 2010, LNICST 46, pp. 629–631, 2011.

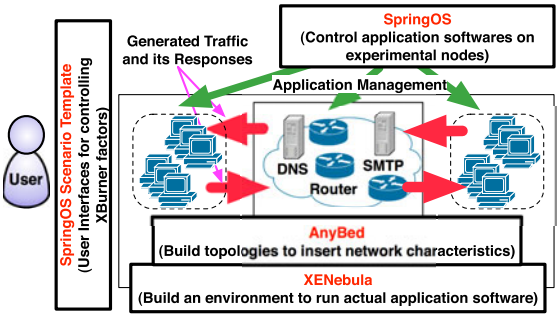


Fig. 1. Traffic Generation with XBurner

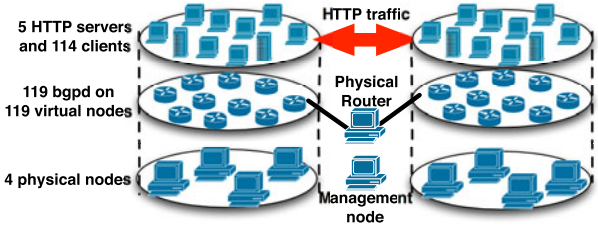


Fig. 2. Experimental Environment

XENebula[1] is a Xen-based platform for building large-scale network experimental environment. It has capability to setup several hundred of virtual machines on a dozen of physical nodes.

AnyBed[2] is a tool that create configurations for the quagga-routing daemon according to topology databases such as CAIDA AS Relationships[3] dataset.

SpringOS[4] is a software suite for controlling experimental nodes to implement user requirements on network testbeds.

Using AnyBed and XENebula, users can run many virtual nodes and emulate a large-scale L3 network that is actually routed by dynamic-routing protocols. SpringOS can trigger application software on the environment to generate native traffic. Its experimental scenarios can be used as templates for other experimenters.

Figure 1 shows the overall schematic representation of XBurner. Common services such as DNS and SMTP which are implicitly used by many applications, may be implemented on the environment.

To evaluate our proposal, we implemented XBurner for simple traffic generation. We built a large dumbbell topology using 238 virtual nodes on 8 physical nodes which consist of Pentium E2200 2.2 GHz CPU, 4 GByte Memory, and 2 NICs; this topology is illustrated in Figure 2. The nodes are connected to a intelligent switch. We emulate two Inter-AS networks on 4 nodes and they were connected via a physical PC router.

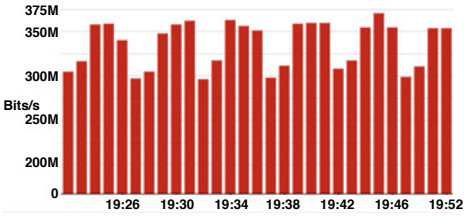


Fig. 3. Measured traffic on the beside of the physical router

To build this environment, we extend XENebula and AnyBed because current implementations of them are focused only to generate one BGP networks. Our extensions cover to run much more application software on virtual nodes and specifying IP address range of emulated network, and so on.

On the building the topology, we generate traffic using our SpringOS scenarios. The server nodes generate "`apache`" configuration and startup it then clients run "`wget`" HTTP clients to get 1M-size file to its servers. After getting the file, clients sleep 1 second and get the file again.

There are many source and destination IP address pairs and the throughput observed beside of the physical router by "`sflow`" is shown in figure 3. This experiment shows that XBurner can run as a platform for traffic generation.

3 Future Works

XBurner is merely a platform for generating traffic using native application software, and the most important point in future is consideration of traffic patterns and packet forms composing them for evaluating typical services and implementations. We'll generate many kinds of traffic and provide them to users as SpringOS scenarios to generate the same traffic patterns. There are still several open issues about XBurner; accuracy of triggering actual software, coverage investigation, preparing kinds of template, and so on.

References

1. Miwa, S., Suzuki, M., Hazeyama, H., Uda, S., Miyachi, T., Kadobayashi, Y., Shinoda, Y.: Experiences in Emulating 10K AS Topology with Massive VM Multiplexing. In: VISA 2009 (August. 2009)
2. Suzuki, M., Hazeyama, H., Kadobayashi, Y.: Expediting experiments across testbeds with AnyBed: a testbed-independent topology configuration tool. In: TridentCom 2006 (March 2006)
3. CAIDA. AS Relationships, `http://www.caida.org/data/active/as-relationships/`.
4. Miyachi, T., Chinen, K., Shinoda, Y.: StarBED and SpringOS: Large-scale General Purpose Network Testbed and Supporting Software. In: Valuetools 2006 (October 2006)

Topology Virtualization for Wireless Sensor Network Testbeds

Daniel Bimschas[1], Maick Danckwardt[1], Dennis Pfisterer[1], Stefan Fischer[1], Tobias Baumgartner[2], Alexander Kröller[2], and Sándor P. Fekete[2]

[1] Institute of Telematics, University of Lübeck, Germany
[2] Dpt of Computer Science, Braunschweig University of Technology, Germany
{bimschas,danckwardt,pfisterer,fischer}@itm.uni-luebeck.de,
{tbaum,kroeller,fekete}@ibr.cs.tu-bs.de

Abstract. In recent years Wireless Sensor Networks (WSNs) have enjoyed a growing amount of attention. One particularly promising prospect is to employ WSNs as an extension of the future internet into the real world; this motivates experimentally driven research to evaluate and benchmark new concepts on WSNs. With our poster we will show our approach to virtualizing Wireless Sensor Network testbeds. With this technique we are able to reconfigure the topology of a WSN testbed without changing the physical location of nodes; it even allows building virtual topologies on top of federated testbeds.

1 Introduction

Testbeds are the natural way for evaluating new algorithms, approaches, or applications after simulations. On the one hand, a testbed with real hardware allows evaluating a system with the hardware restrictions of a WSN, like limited buffer size and battery capacity, variable transmission characteristics, environmental interference, variable time drift, and real-world sensor data. On the other hand, a testbed is expensive to set up and to maintain, hard to reconfigure for a different experiment, and usually features a fixed number of nodes. A possible approach to deal with these disadvantages is to virtualize parts of the testbed. First steps towards virtualization are introduced in [4], where the radio communication in a WSN is virtualized, and in [1], which describes how to run concurrent experiments in one testbed.

In this paper, we present the approach taken by WISEBED, an FP7 EU project, which comprises 9 partners from 6 different countries. Each partner provides a local testbed (i.e., the one presented in [3]) consisting of heterogeneous sensor nodes (such as TelosB, Mica2, iSense or Sun Spot equipped with different sensors) arranged in different topologies with a total of up to 2000 nodes. A user may use the complete WISEBED testbed or only use a subset and can control the experiment via a web-based interface or a piece of software. In the following, we introduce our technique for virtualization (so-called *virtual links*) and their application to create (virtualized) federated testbeds.

T. Magedanz et al. (Eds.): TridentCom 2010, LNICST 46, pp. 632–634, 2011.

2 Virtual Links

Virtual links allow sensor nodes—located in the same or in different testbeds—to communicate with each other even if they are not in communication range or have incompatible radio interfaces. Virtual links are created using a piece of software on each sensor node (a so-called virtual radio), which contains a routing table of the form (SensorNodeID, interface). Upon sending a message to a specific node, the radio knows via which *interface* a message has to be sent. This interface could be the node's hardware radio or the virtual interface, which forwards the message to a testbed server (as shown in Figure 1(b)), which in turn delivers it to the destination node. In addition to adding links to a node, it is also possible to drop messages from certain senders to prohibit communication between neighboring nodes.

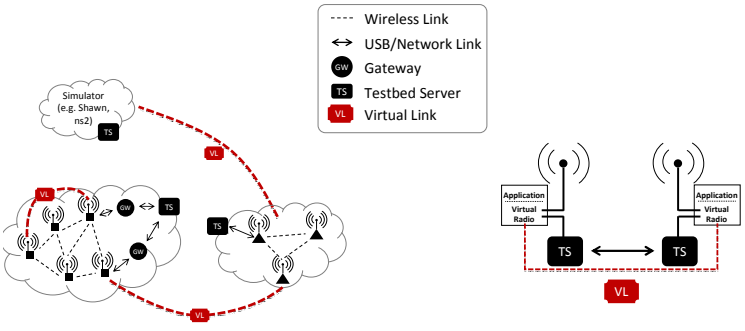


Fig. 1. Architecture of a virtualized testbed (a) and a virtual link (b)

Nodes that are arranged in a grid can be rearranged using this technique, e.g., in a line or using a random topology. This feature allows the specific definition of the desired testbed topology needed for an experiment, similar to simulations, but the experiment is run on real hardware. Furthermore, one can add simulated nodes to a physical testbed and define communication parameters like RSSI, LQI, and message loss. These functions pave the way for building any virtual topology, as it is needed for an experiment. For more details on virtual links, we refer the reader to [2].

3 Federating Testbeds

As shown in Figure 1(a), a physical or simulated testbed is exposed to the internet by a testbed server. Each testbed server provides the functionality to allocate parts of a testbed for a specific period and to manage and control experiments, e.g., flashing and resetting sensor nodes, monitoring the experiment, managing virtual links, etc. The testbed server exposes this functionality through a set of Web Service APIs. In addition to physical and simulated testbeds, we developed a federating testbed server that is able to control different testbeds.

The federator implements the same set of Web Service APIs mentioned above, thereby providing transparent access to the underlying testbeds. Furthermore, the server can control both physical and simulated testbeds, as well as other federated testbeds, allowing for arbitrary hierarchical composition of testbeds.

This means that a user can reserve a local testbed, configure an experiment, log the runtime, and compare the results with experiments on other local testbeds or even federated testbeds, just by changing the connection to the specific testbed server. In a federated testbed, virtual links are used to tunnel messages from one node to nodes in a different testbed, making them virtual neighbors. This allows defining virtual topologies in a federated testbed and hiding boundaries between different testbeds from the application.

4 Poster

We ran a number of experiments, comparing non-virtualized with virtualized testbeds (including flooding of sensor data or time synchronization) to evaluate the impact of virtualization. First results show that our approach is realistic and provides a good performance for experiments. Furthermore, applications cannot distinguish between a real testbed and a partially virtualized one. On our poster we introduce the architecture of virtualized testbeds and the API used to define and establish the virtual topologies, present first results, and give an outlook on future work within WISEBED.

Acknowledgments. This work has been partially supported by the European Union under contract number IST-2008-224460 (WISEBED).

References

1. Aberer, K., Hauswirth, M., Salehi, A.: A middleware for fast and flexible sensor network deployment. In: VLDB 2006: Proceedings of the 32nd International Conference on Very Large Data Bases, pp. 1199–1202. VLDB Endowment (2006)
2. Baumgartner, T., Chatzigiannakis, I., Danckwardt, M., Koninis, C., Kröller, A., Mylonas, G., Pfisterer, D., Porter, B.: Virtualising testbeds to support large-scale reconfigurable experimental facilities. In: Silva, J.S., Krishnamachari, B., Boavida, F. (eds.) ESWN 2010. LNCS, vol. 5970, pp. 210–223. Springer, Heidelberg (2010)
3. Baumgartner, T., Fekete, S.P., Kröller, A.: Hallway monitoring with sensor networks. In: SenSys 2009: Proceedings of the 7th ACM Conference on Embedded Network Sensor Systems, pp. 331–332. ACM, New York (2009)
4. Jayasumana, A.P., Han, Q., Illangasekare, T.H.: Virtual sensor networks - a resource efficient approach for concurrent applications. In: ITNG 2007: Proceedings of the International Conference on Information Technology (2007)

System-Level Service Assurance — The H∀Mcast Approach to Global Multicast*

Thomas C. Schmidt[1] and Matthias Wählisch[2,3]

[1] HAW Hamburg, Department Informatik, Berliner Tor 7, 20099 Hamburg, Germany
[2] link-lab, Hönower Str. 35, 10318 Berlin, Germany
[3] Freie Universität Berlin, Inst. für Informatik, Takustr. 9, 14195 Berlin, Germany
{t.schmidt,waehlisch}@ieee.org

Abstract. The Internet revolution introduced a single, adaptive abstraction layer for global communication. Today, IP interconnects millions of applications, which themselves are bound to the present IP layer via the socket API. After almost 30 years, the time has come to abandon this focus on a single, homogeneously established Internet protocol and thereby release the accumulated needs for innovation on the network layer. H∀Mcast addresses this goal by following the *evolutionary* approach of a hybrid multiservice network layer that decouples service and application development from infrastructure deployment. The objective of this work is a universal, robust service access that allows group applications to run everywhere, no matter what the status of regional technological deployment will be.

Keywords: Internet service architecture, hybrid multicast, multicast mobility management, multicast security.

1 Introduction

Emerging mass applications like IPTV and Massive Multiplayer Online Role Games (MMORGs), but also traditional communication systems such as videoconferencing or newscasts, distribute content to large receiver groups. The most efficient way for group dissemination follows (optimal) distribution trees and is executed on the lowest possible layer [1] available in the network. IP Layer Multicast [2] achieves this goal by providing a corresponding extension of the network socket API and by a dynamic mapping onto the MAC layer. However, a transparent employment of group communication requires a global deployment of IP multicast. Despite of its long-term availability in protocols and implementations, this has not been seen over the years.

The H∀Mcast approach selects the example of group communication to demonstrate how a future multiservice Internet can immediately enable new services,

* This work is supported by the German Federal Ministry of Education and Research. H∀Mcast (http://hamcast.realmv6.org) is part of G-Lab (http://www.german-lab.de).

T. Magedanz et al. (Eds.): TridentCom 2010, LNICST 46, pp. 635–639, 2011.

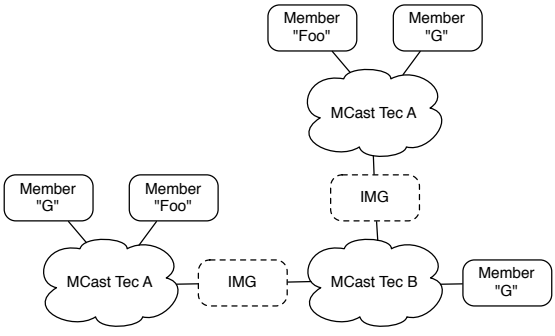


Fig. 1. Reference scenario for hybrid multicast, interconnecting group members from isolated homogeneous and heterogeneous domains

even at a stage of gradual, incomplete deployment. Multicast is to be realized in a concept of multiple, simultaneously operating layers. This will give rise to a continuous *Service Availability*, at first. At second, an adaptive selection will ensure that only the lowest available layer comes into operation and thereby optimize *Service Efficiency*. These functions, combined with further capabilities at end systems, will be hidden behind a uniform programming interface, such that any requirement on network-specific choices or arrangements is taken away from application programmers. A straight forward way to implementing universally deployable products is thus regained.

This paper presents a brief overview of the ongoing project work. In the following section 2, we discuss the core problems and give a conceptual overview of the H∀Mcast system-centric service model. Section 3 introduces the implementation and a new programming interface for group services. A conclusion and an outlook complete this writing.

2 Approaching the Internet Service Problem

During the last decade, the Internet has showed resistance towards deploying new services or technologies, and multicast serves as an excellent example for the underlying deadlock: As group communication is a complex, composite service that can be realized in many flavors and various technologies, individual stakeholders of the Internet community make choices of service design and technological deployment. Since there is no common interface to access these various techniques, applications drive individually implemented solutions that minimize interaction with the network infrastructure and thus act as disincentive towards a universal deployment. Users, programmers and operators not only pay for this with higher efforts, lower efficiency and lack of innovation, but new service requirements such as mobility [3], multi-homing, and security challenge original design parts of the Internet and put additional stress on any rag-type service architecture.

The many flavours and pluralistic technologies of group communication – including IPv4 and IPv6 – require hybrid solutions. As displayed in Figure 1, such approaches need to integrate (1) Multicast domains running the same multicast technology but remaining isolated, possibly only connected by network layer unicast; and (2) Multicast domains running different multicast technologies, but hosting nodes that are members of the same multicast group. Hybrid multicast may be realized with limited efforts and acceptable performance [4,5], but requires enhanced intelligence when accessing the network layer.

Following the end-to-end design principle [1] and inspired by current overprovisioning at end nodes, H∀Mcast allocates this service intelligence at edge systems. The system-centric group communication stack is located in an adaptive, modular middleware that represents an abstraction layer between applications and transport technologies. The middleware is a unique daemon process instantiated once per host at start-up, and includes modules for *service discovery, name-to-address mapping,* and *technology-specific service interfaces.* It provides efficient multicast access for any group application without reimplementation and redundancy, and offers a path to replace current proprietary workarounds deployed in manifold ways. Encapsulated by a high-level API, application-driven service establishment may thus proceed by installing a system service, independent of ISP awareness and without the need of globally upgrading the network.

3 API and Implementation

Multicast application development should be decoupled of technological deployment throughout the infrastructure. It requires a common multicast API that offers calls to transmit and receive multicast data independent of the supporting layer and the underlying technological details. For inter-technology transmissions, a consistent view on multicast states is needed, as well. In contrast to the standard multicast socket interface, the H∀Mcast API abstracts naming and addressing [6]. Using a multicast address in the common socket API predefines the corresponding routing layer. In this approach, the multicast address used for joining a group denotes an application layer data stream that is identified by a multicast URI and without an association to the underlying distribution technology. In addition, a system layer at each device accounts for a *late binding* of names to addresses (i.e., during run-time and not compile-time).

Multicast group names are based on an URI scheme that is defined as follows:

```
scheme "://" group "@" instantiation ":"
             port "/" sec-credentials
```

The *scheme* refers to the namespace of the assigned identifier (e.g., ip, or sip), *group* denotes the group ID, *instantiation* identifies the entity that generates the instance of the group, *port* identifies a specific application, and *sec-credentials* are optional to implement security. Valid group IDs will be ipv://224.0.1.1:4000 or sip://snoopy@peanuts.com, for example. This identifiers are passed to directly to a socket, using high-level group calls.

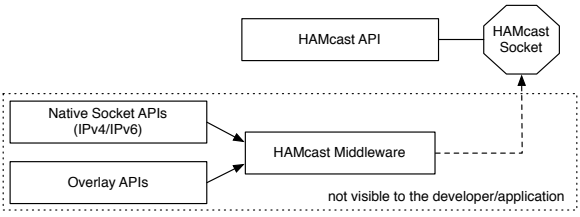


Fig. 2. The H∀Mcast service architecture composed of an abstract socket that grants service access as a facade to one or several network service interfaces via an intelligent middleware

The H∀Mcast socket describes a group communication channel composed of one or multiple interfaces. A socket may be created without explicit interface association by the application, which leaves the choice of the underlying forwarding technology to the group communication stack. However, an application may also bind the socket to one or multiple dedicated interfaces, which predefines the forwarding technology and the namespace(s) of the Group Address(es). All of this happens at a system-layer as visualized in Figure 2 that concentrates hybrid multicast complexity and simplifies application and network duties.

4 Conclusions and Outlook

In this paper, we argued that a system-centric deployment in parts is a promising way to overcome the current Internet feature freeze. For the example of hybrid adaptive multicast, we presented an architecture and prototypic realization of a group communication service that met the requirements by two core contributions. First the various service instantiations and deployments are subsumed below an abstract service entry point implemented as a middleware at end systems. More importantly, a universal service access is granted to applications following a refined general multicast concept and a technology-agnostic API. Thereby and for the first time, group applications can be written once, but run everywhere.

In future work, we will extend implementations and large-scale testing. In particular, support for different programming languages waits to be added. We further plan to intensify standardization work of the API to make these achievements valuable for a wider public.

References

1. Saltzer, J.H., Reed, D.P., Clark, D.D.: End-to-End Arguments in System Design. ACM Trans. Comput. Syst. 2(4), 277–288 (1984)
2. Deering, S.: Host extensions for IP multicasting. IETF, RFC 1112 (August 1989)
3. Schmidt, T., Waehlisch, M., Fairhurst, G.: Multicast Mobility in Mobile IP Version 6 (MIPv6): Problem Statement and Brief Survey. IETF, RFC 5757 (2010)

4. Wählisch, M., Schmidt, T.C.: Between Underlay and Overlay: On Deployable, Efficient, Mobility-agnostic Group Communication Services. Internet Research 17(5), 519–534 (2007)
5. Wählisch, M., Schmidt, T.C.: An a Priori Estimator for the Delay Distribution in Global Hyrid Multicast. In: Proc. of the ACM SIGCOMM CoNEXT.., pp. 19–20. ACM, N.Y (December 2009)
6. Wählisch, M., Schmidt, T.C., Venaas, S.: A Common API for Transparent Hybrid Multicast. individual, IRTF Internet Draft – work in prog. (March 02, 2010)

ONIT Workshop 2010

Session 1: Services and Architectures

Emergency Services in IMS

Andreea Ancuta Onofrei[1], Yacine Rebahi[1], Thomas Magedanz[1],
Fernando López Aguilar[2], and José Manuel López López[2]

[1] Fraunhofer FOKUS Institute,
Kaiserin-Augusta-Allee 31, 10589 Berlin, Germany
{andreea.ancuta.onofrei,yacine.rebahi,
thomas.magedanz}@fokus.fraunhofer.de
[2] Telefonica Research & Development,
Emilio Vargas 6, 28043 Madrid, Spain
{fla,josemll}@tid.es

Abstract. There is an increasing demand of multimedia and location aware services when an accurate and immediate response is needed, for example in the case of emergency services. In parallel, the technology has evolved from Circuit Switch (CS) to Packet Switch (PS) networks requiring backwards compatibility. As a result the emergency services support for NGNs, in particular for IMS, is being standardized by 3GPP, NENA, ETSI. In this paper we present architecture for Emergency Services support in IMS, starting from the one from 3GPP. We have implemented this architecture as the Emergency Branch of the open-source project Open IMS Core. The available methods for personalizing the testbed and validation scenarios are described here. More important, the management of resource allocation and prioritization of emergency calls is also discussed in the context of the emerging EPC.

Keywords: IMS, emergency services, location aware, LOCSIP, LoST, SIPp.

1 Introduction

Telecommunications play a major role in speeding response and minimizing loss of life and property. Communications systems can help, for instance, when making a daily emergency call to police, ambulance and fire brigade. For example, if the location of the caller is determined the call can be forwarded by the network to the nearest Public Safety Answering Point (PSAP), including mobile rescue teams, eliminating the delay for a central call center to transfer the call. At the same time the PSAP can be instantly informed about the accurate location of the caller leading to reduced necessary call duration and problems related to pronunciation or human error. When the caller is moving, the PSAP can monitor the current caller location.

IP Multimedia Subsystem (IMS) [0] is the result of the standardization effort of 3GPP for a platform offering Voice over IP (VoIP) and other multimedia services supporting multiple access network technologies from Public Switched Telephone Network (PSTN) to 3G mobile. Other standardization bodies, like ETSI, are also

T. Magedanz et al. (Eds.): TridentCom 2010, LNICST 46, pp. 643–652, 2011.

defining interfaces towards IMS. Next Generation Networks (NGNs), in particular IMS, are certainly the future replacement of the current telecommunication networks; therefore the current emergency systems need to be upgraded in order to fulfill the NGNs requirements.

The need for Emergency Services support in the emerging platforms has been recognized also by 3GPP that initiated the standardization of context aware emergency services architecture in IMS. In order to validate the specifications and to find the possible breaches, there is a need for a testbed. This is the reason we have developed one that enables not only the industry to have a testbed for emergency services, a standardization starting point or an evaluation testbed for potential commercial solutions, but also the academia in the path for future research.

The open-source Open IMS Core project [3] was released in 2006 by the Fraunhofer FOKUS Institute and has been recognized as a reference implementation and testbed for IMS. The emergency branch of the project [2] can be used as emergency services in IMS testbed. It has been developed during the collaboration with Telefonica Research and Development (TID) under the umbrella of the project IP-based Emergency Application and serviCes for nExt generation networks (PEACE) [4] partly financed by the European Commission.

The central functionality of the testbed, dispatching a call to the nearest required service center, has a more general appliance in the field of context aware services: it can be used for other services than emergency ones like taxis, pharmacies, restaurants or gas and electric stations.

In this paper we present the implemented testbed. The document is divided as follows: section 2 for the state of the art in the emergency services support for IMS and some of the protocols related to location information defined by IETF, section 3 for the design and implementation of the testbed as well as the general functionality and methods for personalizing it, section 4 for the most representative validation scenarios are presented and section 5 for the roadmap of the solution and conclutions.

2 State of the Art

In this section we describe the architecture for emergency services support, as specified by 3GPP and the IETF standardization effort for emergency services, including: the emergency URNs that a caller can use when generating an emergency call and some of the current formats and protocols related to location information.

2.1 3GPP Architecture of the IMS

The IMS framework [Fig. 1] is the result of the 3GPP standardization group and includes three layers: the *Access Layer*, the *Control Layer and* the *Service Layer*.

The *access layer* consists of IP routers and legacy PSTN switches that provide access to the IMS network both from contemporary IP telephony devices and older circuit switched devices respectively. IP devices compatible with IMS incorporate a SIP user agent that can be used, for example, to place voice or video calls toward the network.

The *control layer* of the IMS network manages (amongst others): the subscriber authentication and call establishment and release using components with Call Session Control Function (CSCF): Proxy (P-CSCF), Interrogating (I-CSCF) and Serving (S-CSCF) and the subscriber profile and the interface to the service layer at the Home Subscriber Server (HSS).

The applications are hosted in the *service layer*. This layer provides the end user service logic and consists of SIP Application Servers (AS). An AS executes IMS applications and services by manipulating SIP signaling and interfacing with other systems.

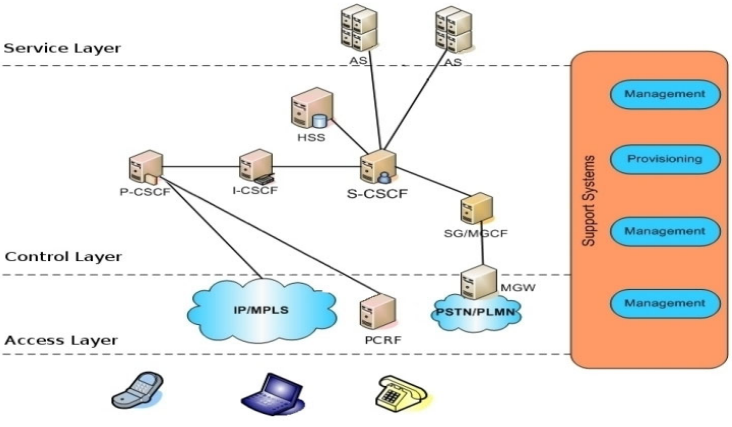


Fig. 1. IMS Architecture

The 3GPP architecture for the Emergency Services support in IMS is defined in TS 23.167 [5]. For routing the emergency call (E-Call) to the nearest PSAP, e.g. Police station new components have been added in the *Control layer*.

The *Emergency-CSCF (E-CSCF)* – retrieves the PSAP URI for the E-Call from the LRF and forwards the call accordingly. If the PSAP URI is a TEL URI [6] the call will be sent to the gateway towards the PSTN. In the case that the URI is an Internet protocol URI, .e.g. SIP [7], the call will be routed using the interface to the Internet.

The *Location Retrieval Function (LRF)* – receives the caller location from the E-CSCF if this information is present in the initial request of the E-Call. Otherwise it will query the network about the caller location. It maps an E-Call to the nearest PSAP of the required service by having an interface with or including the *Routing Decision Function (RDF)*.

The *emergency registration* (E-Reg) was introduced in TS 23.167 [5]. It can be used by the user when roaming or not yet registered to its home network; otherwise the existing non-emergency registration to its home network is enough for making an E-Call. The E-Reg is used to allocate the network resources to the subscriber, independently of its home network by ignoring the roaming restrictions (see TS 29.228 [8]). At the SIP level, the E-Reg is a registration where the REGISTER request contains a “sos” URI parameter in the Contact header as defined in [9].

The P-CSCF recognizes *emergency calls* (E-Call) as the calls of which the initial INVITE request contains a request URI that matches one of the emergency numbers or an emergency services URN (see subsection 2.2). When the caller does not have enough credentials to authenticate, e.g. no available SIM, based on local policies the caller can still make emergency calls, known as *"anonymous" E-Calls.*

2.2 IETF Standardization Effort for Emergency Services

Currently the operator recognizes as emergency calls the calls addressed to emergency numbers from the country the caller is located or emergency numbers from networks that have roaming agreements with the operator.

Some of the countries even have a unified emergency number, e.g. 112, for multiple emergency services. One advantage of using a single emergency number is that it is easier for the citizens to remember it. The disadvantage is that a call taker has to identify which services (e.g. Police or Ambulance) are required in every case and forward the call accordingly. This operation implies a delay which can be critical in most of the situations, especially when there is congestion at the call center.

In order to unify the emergency number based on the type of service and independently of the country, *emergency URNs* were defined in [9], e.g. *"urn:service:sos.police"* for referring to the Police Department. They enable the caller to make E-Calls independent of its roaming state. For example, emergency URNs would be the solution for a roaming caller that does not know the emergency number from the attached network and the emergency number from its country/home network is not recognized by the attached one or, even worse, it has a different meaning leading to the call being redirected to a different service.

Location information can be conveyed directly, in a "location-by-value" format or indirectly, in a "location-by-reference" one. The recipient of a location by reference has to "dereference" it by interrogating a location server using the data from the reference thus obtaining a location by value [11].

Presence Information Data Format (PIDF) objects [12] consist of XML encapsulated information that can be carried in the payload of the protocols meeting the requirements from [12]. SIP is one of these protocols and at the same time can be used to establish a multimedia session. Based on the requirements regarding location objects [13], the **PIDF-Location Objects** (**PIDF-LO**) were defined as an extension of PIDF objects for "location-by-value" in [15]. SIP messages convey PIDF-LOs as body parts with the Content-Type "application/pdf+xml". A PIDF-LO can encode *geodetic* as well as *civic* location information, timestamps, and privacy requirements [16, 17] and can be used for routing the call to an appropriate service instance by context aware systems.

Following the same principle of presence services using SIP, Open Mobile Alliance (OMA) [18] has defined the protocol Location in SIP/IP Core (**LOCSIP**) [19] that can be used for retrieving the location of a certain target identified by the SIP URI, IP, IMSI or other characteristics. The protocol consists of a SIP subscription [20] to the new introduced event package "location". If the information is available, the notifications will include the location information for the corresponding target in the PIDF-LO format. This protocol also supports subscription filters [21] to request

automatic generated notifications when a parameter of the subscription's target has changed, in our case when the user's location is modified, useful for monitoring a moving target.

The Location to Service Translation protocol (**LoST**) [22] is an XML-based protocol which can be used for mapping a tuple (location, service URN) to a service URI responsible for that type of service in an area that includes the given location. The LoST payload can use protocols such as HTTP or HTTPS as carriers. In particular, it can be used to decide which PSAP is the one in charge of a specific emergency service in an area.

3 Open IMS Core Emergency Services Testbed

Open IMS Core Emergency Testbed implements a set of standard compliant components, easily adapted to further evolution of the standards and other research topics. Following the emergency services architecture from TS 23.167 also described in the section 2.1 of this paper, the components E-CSCF and LRF have been added to the original Open IMS Core project. The architecture can be seen in Fig. 2.

The P-CSCF is the first node in the SIP signaling path from the IMS network. It recognizes E-Calls and forwards them to the E-CSCF. The latter will query the LRF for the PSAP URI. In the query the E-CSCF will insert the requested service URN and, if present, the location information of the caller from the SIP INVITE request.

When the query includes location information, the LRF can proceed immediately with mapping the pair (location, type of service) to a PSAP URI using the interface to the RDF. We have designed the *RDF as a LoST server*, because the LoST protocol suites the role of the RDF. Another reason was that both SIP and LoST use the PIDF-LO format for encoding location information eliminating the delay introduced by a format translation when creating a LoST request.

In order to support *E-Call without location information* we have designed the interface between the LRF and the network based on the LOCSIP protocol. The reason for choosing this protocol was that it is SIP-based and is using the same location information format as the initial INVITE request of the E-Call. This means that LRF will process the location information received from both the E-CSCF and the LOCSIP server without any format translation. The LRF has to inform the E-CSCF about the acquired location data. The E-CSCF includes the location information received from the LRF in the initial request and forwards it to the PSAP URI, without any translation of the location format.

The specification of the interface between E-CSCF and LRF is still an open issue as Release 9 of IMS [24] does not state which protocol, type of messages and how their content should be encoded. The design has to support callback and location update, which have an impact on the information stored in the E-CSCF and LRF components and depend on: the registration state of the caller and the type of PSAP to which the call was forwarded: VoIP-based or legacy one (PSTN telephones).

We have identified two types of messages that the interface between the E-CSCF and LRF should support:

1. **Mapping Request.** when a new E-Call is about to be established, sent from the E-CSCF to the LRF for mapping a location, if available, and service type to a PSAP URI. The reply contains the PSAP URI and the retrieved location information, if not included in the initial INVITE request.
2. **Release Request.** when an E-Call is released: the E-CSCF alerts the LRF that it should free any resource allocated to the emergency call. This is important as it keeps the LRF clean and able to better use its resources.

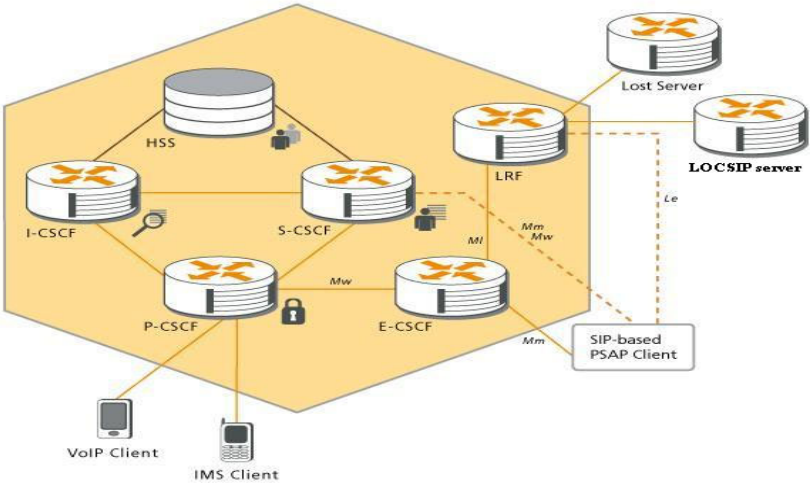


Fig. 2. Architecture of the Emergency Services Support of the Open IMS Core

We have designed this interface *based on SIP*, as this protocol can be used to carry the location information of the caller, is easily extensible and, according to our design, a SIP stack is already included in the implementation of the two involved components: E-CSCF and LRF.

The next step was defining which SIP messages to use between the E-CSCF and the LRF. We have considered using the OPTIONS request to for the *mapping request* and its response to transport the PSAP SIP URI and the caller location information if returned from the LOCSIP server. An OPTIONS request with a marker for terminating the processing of the emergency call would also cover the *release request*.

The implementation of our testbed is based on TS 23.167 [5], TS 29.228 [8] and TS 24.229 [22]. Open IMS Core has a module dedicated for each component.

The support for emergency registrations was added by enhancing the modules pcscf, icscf and scscf.

Then the *engine for routing the emergency calls* was implemented. The **pcscf module** was enhanced to recognize E-Calls. For an easily personalization of the list of the supported emergency numbers an XML file with *(emergency numbers, emergency URNs) associations* is loaded when the pcscf module starts.

The **ecscf module** and the **lrf module** were added as instances of SIP Express Router (SER) [25] written in the C language, just like the other CSCFs.

The interface between the P-CSCF and E-CSCF has been *secured* using the Path header to eliminate one of the possible methods for a third party to impersonate the PSAP. More details about the possible attack and the solution can be found in [26].

When processing the E-Call, the location information included by the caller or acquired by the LRF is considered to be "Location-by-value" as defined in [16, 17]. Both E-CSCF and LRF expect the PIDF-LO in the civic or geodetic formats mentioned in subsection 2.2. Otherwise the E-CSCF will reply with a 424 (Bad Location Information) error message as defined in [11]. The support for the PIDF-LO formats was implemented in a *pidflo library* based on the library *libxml* [28], shared by both modules.

When no caller location information was provided by the E-CSCF but the caller contact address is known, the lrf module will try to retrieve acting as a *LOCSIP client*. The *LOCSIP server acts as* ***a Global Location Enabler*** that can access several location databases, based on the type of the access network technology and the received information about the caller, e.g. PSTN number, IP. For mapping the emergency call to an appropriate PSAP, the module can also act as a *LoST client* using HTTP as transport for the LoST payload. For this purpose a *lost library* was developed based on *libcurl library* [27] for HTTP messages handling.

4 Validation Scenarios

For confirming the capabilities of our emergency services testbed support for multiple scenarios was added, from which the following were selected as the most representative. All the scenarios describe the case when the user makes a call to the Police Department, but the method or status of the caller location retrieval is different. The first one covers the case of *Location acquired by the user or caller device* and the second one *Location acquired by the LRF*. These are presented here as validation scenarios in order to prove the practical importance of the current implementation.

To generate the scenarios an enhanced Monster the IMS client [31] able to store location information and recognize and generate emergency calls. It has dedicated buttons for calling the Police Department, the Fire Brigade and Ambulance. For backwards compatibility the user can also dial emergency numbers, e.g.112. For testing GPS location, a GPS-enabled Location Agent was developed that connects to the IMS client using the NMEA0183 protocol. Two modes of operation exist: stand-alone GPS or Assisted-GPS (to speed up position acquisition) over OMA Secure User Plane Protocol (SUPL) [32].

Location acquired by the user or caller device. The first scenario represents the case when the calling IMS client is capable of determining the current location (geodetic or civic) (Fig. 3). Alice is registered to her home network using the SIP URI sip:alice@open-ims.test. She sets the IP and port of the of the NMEA server running on the GPS-Location Agent terminal. The location Germany, Berlin is acquired and the user is alerted about it. Then Alice calls the Police Department. The LRF has data for the area that includes Alice's location and type of service and maps it to the PSAP URI sip:police_berlin@open-ims.test. The call is forwarded by the E-CSCF accordingly. The PSAP can extract the location of the caller from the received INVITE request and show it on a map.

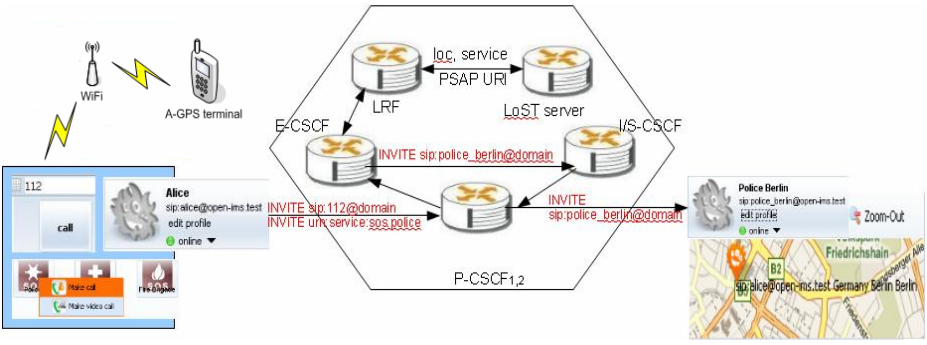


Fig. 3. Emergency call when the caller or caller's devide is able to acquire the lcoation information

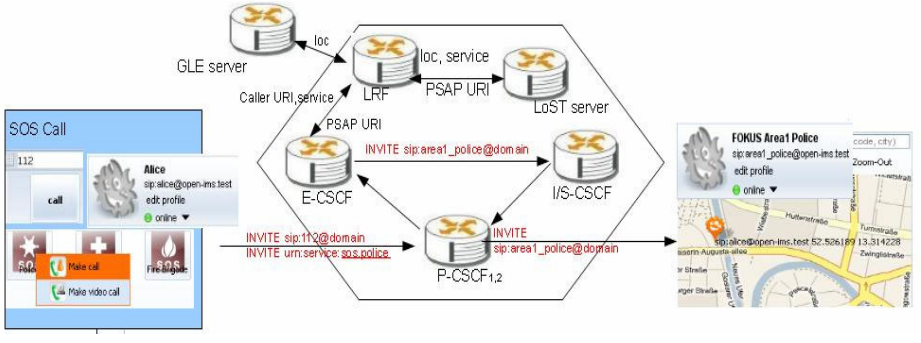


Fig. 4. Emergency call when the caller or caller's devide is not able to acquire the lcoation information and the LRF succsessfully retrieves it from the GLE

Location acquired by the LRF. The second scenario represents the case when the caller device is not able to acquire the current location and Alice make a call to the Police Department and this time no location information is included in the INVITE request (see Fig. 4). If the GLE is aware of the caller location, geodetic or civic, the LRF will be able to retrieve it. We consider also that the LoST server has data about the Police Department in an area that includes the caller location. The LRF maps the call to PSAP SIP URI "sip:area1_police@open-ims.test". As a result the E-CSCF will forward the call to the designated PSAP after including the location information in the body of the INVITE request. In this case, the PSAP can extract and show the caller location on a map.

The scenarios have been presented as validation scenarios to the European Commission during the First Year Review of the PEACE project.

We have also created SIPp [33] scripts as a lightweight and public method for emulating: the caller, the PSAP and optionally the LOCSIP server.

5 Conclusions and Further Work

In the present paper we have presented an emergency services testbed implemented following the standards from 3GPP and developed as an extension of the Open IMS Core project.

One of the features not exploited yet in our implementation is a cross-layer prioritization of the emergency calls over the non-emergency ones. We mean by cross-layer prioritization a mechanism for prioritizing both the signaling and the data flow. The prioritization can be very important especially if we have a real situation where the network resources are limited or the network is congested. We intend to develop an algorithm for the P-CSCF to prioritize the signaling flow of emergency services over the one of non-emergency services. With the emerging Evolved Packet System (EPS) [34], which is being standardized by 3GPP as a policy based framework between multiple access network technologies and managed multimedia services, we intend to interface our emergency services IMS testbed with an EPC testbed as can be seen in [35].

Another useful feature is to enable the PSAP to update the caller location information, optionally using time and boundary filters. This can be achieved using the Le interface between the PSAP and the LRF, which needs further specification. The privacy and security issues related to this field are also an important research topic for our team. Following this goal, the testbed will be enhanced with a security framework for validating the location of the user and as well one for protecting the privacy of the user.

References

1. IP Multimedia Subsystem (IMS), http://www.3gpp.org/article/ims
2. Emergency Branch of the Open IMS Core, http://openimscore.org/emergency
3. Open IMS Core project, http://openimscore.org
4. Peace European Project, http://www.ict-peace.eu/
5. IP Multimedia Subsystem (IMS) emergency sessions, TS 23.167
6. Schulzrinne, H.: The tel URI for Telephone Numbers. RFC 3966 (December 2004)
7. Rosenberg, J., Schulzrinne, H., et al.: Session Initiation Protocol (SIP). RFC 3261
8. IP Multimedia (IM) Subsystem Cx and Dx Interfaces; Signalling flows and message contents, TS 29.228
9. Patel, M.: SOS Uniform Resource Identifier (URI) Parameter for Marking of Session Initiation Protocol (SIP) Requests related to Emergency Services, draft-patel-ecrit-sos-parameter-07.txt (October 26, 2009)
10. Schulzrinne, H.: A Uniform Resource Name (URN) for Emergency and Other Well-Known Services, RFC 5031 (January 2008), http://tools.ietf.org/html/rfc5031
11. Polk, J., Rosen, B.: Location Conveyance for the Session Initiation Protocol., draft-ietf-sip-location-conveyance-13.txt (July 13, 2009)
12. Sugano, H., Fujimoto, S., Klyne, G., Bateman, A., Carr, W., Peterson, J.: Presence Information Data Format (PIDF), RFC 3863 (August 2004)
13. Day, M., et al.: Instant Messaging / Presence Protocol Requirements. RFC 2779
14. Cuellar, J., Polk, J., et al.: Geopriv Requirements. RFC 3693 (February 2004)

15. Peterson, J.: A Presence-based GEOPRIV Location Object Format, RFC 4119 (December 2005), http://tools.ietf.org/html/rfc4119
16. Winterbottom, J., Tschofenig, H., et al.: GEOPRIV Presence Information Data Format Location Object (PIDF-LO); Usage Clarification, Considerations, and Recommendations, RFC 5491 (March 2009)
17. Thomson, M., Winterbottom, J.: Revised Civic Location Format for Presence Information Data Format Location Object (PIDF-LO), RFC 5139 (February 2008)
18. Open Mobile Alliance (OMA), http://www.openmobilealliance.org/
19. OMA Location in SIP/IP Core (LOCSIP), http://www.openmobilealliance.org/Technical/release_program/locsip_v1_0.aspx
20. Roach, A.B.: Session Initiation Protocol (SIP)-Specific Event Notification, RFC 3265 (June 2002)
21. Rosen, B., et al.: Filtering Location Notifications in the Session Initiation Protocol (SIP), October 26, draft-ietf-geopriv-loc-filters-07.txt (2009)
22. Schulzrinne, H., et al.: LoST: A Location-to-Service Translation Protocol., RFC 5222
23. Internet Protocol (IP) multimedia call control protocol based on Session Initiation Protocol (SIP) and Session Description Protocol (SDP); Stage 3, TS 24.229
24. 3GPP Release 9, http://www.3gpp.org/Release-9
25. SIP Express Router (SER), http://www.iptel.org/ser/
26. Rebahi, Y., Onofrei, A.A., Magedanz, T.: Security in the emergency services support for the IP Multimedia Subsystem. In: MANWEEK 2009 (2009)
27. LibCurl, http://curl.haxx.se/
28. The XML C parser and toolkit of Gnome, libxml, http://xmlsoft.org/
29. CU LoST Reference Implementation, http://honamsun.cs.columbia.edu/index.html
30. PostgreSQL, http://www.postgresql.org/
31. Monster the IMS client, Fraunhofer Institute FOKUS, http://www.monster-the-client.org/
32. OMA Secure User Plane Location V2.0 (SUPL), http://www.openmobilealliance.org/Technical/release_program/supl_v2_0.aspx
33. SIPp, http://sipp.sourceforge.net/
34. Evolved Packet System (EPS), http://www.3gpp.org/ftp/Specs/html-info/29273.htm
35. EPC interface with IMS, http://www.openepc.net/en/openepc/use_cases/ims/index.html

BIQINI – A Flow-Based QoS Enforcement Architecture for NGN Services

Christoph Egger, Marco Happenhofer, Joachim Fabini, and Peter Reichl

Vienna University of Technology
Institute of Broadband Communications
Favoritenstr. 9/E388,
A-1040 Vienna, Austria
Telecommunications Research Center
Vienna (FTW) Donau-City-Straße 1,
A-1220 Vienna, Austria
{christoph.egger,marco.happenhofer,joachim.fabini}@tuwien.ac.at,
reichl@ftw.at

Abstract. Novel mobile cellular access network technologies like Long Term Evolution (LTE) promise capacities exceeding the ones of existing 3G networks by at least one order of magnitude. This evolution will enable the deployment of services which, due to their capacity requirements, are currently restricted to fixed access networks. On the other hand, packet-switched-only architectures raise the need for a reliable and accurate management of these high access capacities, particularly service-specific Quality of Service (QoS) enforcement, in order to prioritize real-time (voice) services and safeguard a satisfactory Quality of Experience (QoE) to the user.

In this paper we present the concept and architecture of a flow-based QoS enforcement architecture called BIQINI which has been developed at the Telecommunications Research Center Vienna (FTW). It consists of a standard-compliant Policy and Charging Rules Function (PCRF) which is supported by an emulated Policy Enforcement Function (PCEF). Extending the FOKUS OpenSource IMS testbed as well as other session-based signaling frameworks, BIQINI's emulated enforcement component enables inexpensive but highly realistic tests on real-time voice and -video traffic, supporting impairments like delay, jitter, loss, and link capacity limitation out-of-the-box. In addition, BIQINI can interface with external policy repositories, thus providing a versatile playground for testing rules and policies in an emulated, realistic environment for real media streams.

Keywords: QoS, IP Multimedia Subsystem, Policy, Policy Enforcement.

1 Introduction

Driven mainly by the high capacities available in fixed access networks, the imminent end-of-life of circuit-switched equipment, and the huge expectations concerning OPEX (operational expenses) reduction when operating one single IP- based platform, the replacement of circuit-switched voice networks by packet- switched data networks is currently experiencing a strong progress. However, the architectural requirements of

T. Magedanz et al. (Eds.): TridentCom 2010, LNICST 46, pp. 653–667, 2011.

such large-scale carrier-grade IP-based telecommunication systems exceed the complexity of plain IETF SIP networks by orders of magnitude in order to enable satisfactory user experience already for the most important basic service, which is still voice. In this context, the IP Multimedia Subsystem (IMS), standardized by the 3rd Generation Partnership Project (3GPP), has become an important candidate architecture for the access-agnostic covering of all relevant aspects from signaling to media, from QoS reservation to security, charging and billing. Recently, several standardization bodies, namely 3GPP, 3GPP2, ETSI TISPAN and PacketCable, have joined their forces to define an interoperable Common IMS platform, agreeing to maintain one single set of 3GPP standards starting with 3GPP Release 9. Moreover, recognizing the need for interoperability and lower IMS complexity, main IMS vendors and operators have engaged in initiatives like One Voice[11] and Rich Communication Suite Initiative (RCSI)[12], defining minimum mandatory sets for IMS system- and service-level capabilities and features.

Released under the GNU Public License in 2006, the Fraunhofer FOKUS Open Source IP Multimedia Subsystem Core (OpenSource IMS) implementation has become a cornerstone of the scientific and industrial IMS research community. OpenSource IMS offers a generic, extensible 3GPP Release 7 IMS core network reference implementation, including signaling and security features as well as modules supporting the extension by means of additional interfaces (reference points). However, today's fixed and upcoming Next Generation Mobile Network (NGMN) access technologies and –services, which require network capacities exceeding the ones of existing 3G networks by an order of magnitude, mandate the use of adequate service-specific QoS enforcement mechanisms to maintain a Quality of Experience (QoE) similar to the one guaranteed by circuit-switched voice services. Despite of this urgent need, this function is not implemented by OpenSource IMS.

Therefore, this apparent gap has been addressed within the application-oriented project BACCARDI (Beyond Architectural Convergence: Charging, SeCurity, Applications, Realization and Demonstration of IMS over fixed and wireless networks) which has been conducted at the Telecommunications Research Center Vienna (FTW) during the years 2008 and 2009. As a result, the BACCARDI IMS QoS Implementation Initiative (BIQINI) has designed and implemented a QoS enforcement function which extends the OpenSource IMS by means of a generic, extensible, 3GPP Release 7 conforming Policy and Charging implementation and the corresponding interfaces. Main parts of the BIQINI concept and implementation have been contributed by the Institute of Broadband Communications (IBK), Vienna University of Technology, with support of FTW and the associated industry project partners Alcatel-Lucent Austria, Kapsch CarrierCom, mobilkom austria, and Telekom Austria.

The main aim of BIQINI is to provide a highly flexible QoS playground and multi-purpose plug-in for policy repositories, implementing a complex, stateful rules function, supporting active network capacity management as well as the PCC push model. With respect to this feature, BIQINI's Policy Enforcement component extends OpenSource IMS to become a reliable testing platform for Quality of Experience (QoE) for real-time multimedia streams. Note, however, that BIQINI does not depend on OpenSource IMS, and instead supports integration with other SIP or non-SIP session-based signaling protocols as well.

In this paper we argue that BIQINI provides a clear advancement compared to the current state of the art, most notably the open source Policy and Charging Control Framework (PCC) published by a group of the University of Capetown (UCT) [9][10] in 2007. In contrast to BIQINI, the UCT PCC operation relies on stateless gate opening and closing, moreover its architectural framework is relatively limited with respect to scalability, extensibility and active link capacity management.

Within the open source community, in November 2009 the NGN working group of Fraunhofer FOKUS has announced an Evolved Packet Core (EPC) implementation which is supposed to include a PCC. However neither the detailed concept nor the implementation of OpenEPC has been released so far. Likewise, Fraunhofer FOKUS' Policy and Charging Control Architecture (PoCCA) is maintained as closed source, being presented briefly in [3] which focuses mainly on rule processing.

To the best of our knowledge, these three contributions already conclude our account of current related work. In contrast, commercial PCC solutions are offered by various IMS vendors. However, two factors are prohibitive in deploying these implementations in applied IMS research at universities or other non-profit institutions: from a technical point of view, these implementations are closed source, which hinders additions and modifications at source code level, whereas the cost factor of PCRFs and particularly of PCEFs provides a huge barrier for IMS-related research activities.

The remainder of this paper is structured as follows: in Section 2, a brief survey of related architectures and open source routing and traffic control tools is provided. Section 3 describes the fundamental concepts and the architecture of the BIQINI implementation, whereas in Section 4 we present some results for selected traffic scenarios. Section 5 concludes the paper with a brief summary and outlook.

2 QoS Enforcement Architectures and Tools

Having recognized that the promotion of three diverging IMS standards for mobile, fixed and cable networks, respectively, entails the severe danger of an overall IMS failure, 3GPP has agreed in 2007 together with other involved standards organizations, notably ETSI TISPAN for fixed networks and PacketCable for cable networks, to harmonize their corresponding standardization activities. Starting with 3GPP Release 8, the 3GPP therefore develops and promotes a Common IMS architecture which conforms to the requirements of all three standardization bodies, whereas in 3GPP Release 7, main QoS-related interfaces (reference points), particularly Rx/Gx for 3GPP and Gq'/Re for the Resource and Admission Control Subsystem (RACS), are not yet harmonized.

As a consequence, the 3GPP Release 7 PCC compliant BIQINI architecture aims at merging the commonalities of the 3GPP and TISPAN architectures towards the framework for policy based admission control [4], which has been defined by the IETF as shown in Figure 1. Here, the Application Function (AF) is positioned within the SIP signaling path, having access to all requests for certain services along with their detailed media descriptions. The AF is responsible to query the Policy Decision Point (PDP), which decides if a specific request is granted or rejected, depending on policies, rules, request information and user profile(s). In case the service request is granted, a request with rules that should be activated is sent to the Policy Enforcement Point (PEP). The PEP enables the requested service flow according to the specifications sent by the PDP.

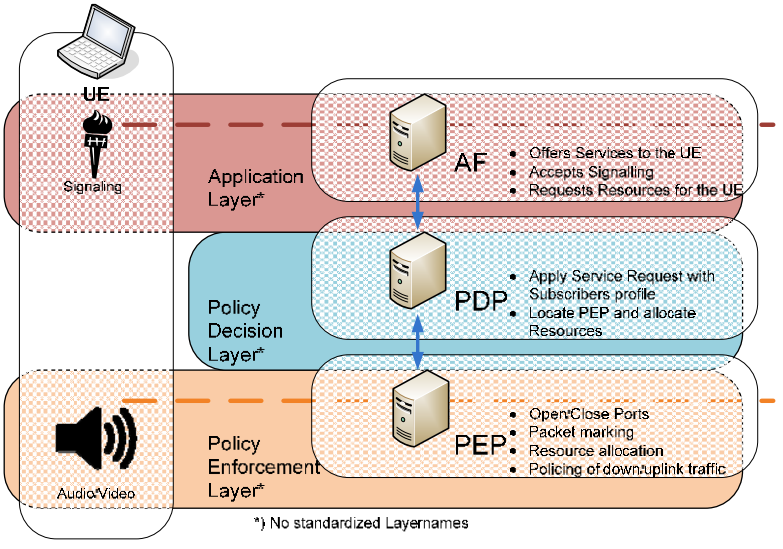


Fig. 1. IETF Architecture of Policy-based Admission Control

As far as NGN QoS enforcement is concerned, the standardization bodies 3GPP, (targeting mobile networks) and ETSI TISPAN (focusing on fixed NGN networks) have designed their own architectures, depending on the particular access network requirements. In the case of 3GPP, this architecture is called Policy and Charging Control (PCC) [6]. Figure 2 depicts main functions in this architecture which can be easily mapped to layers and functions in the previously presented IETF architecture.

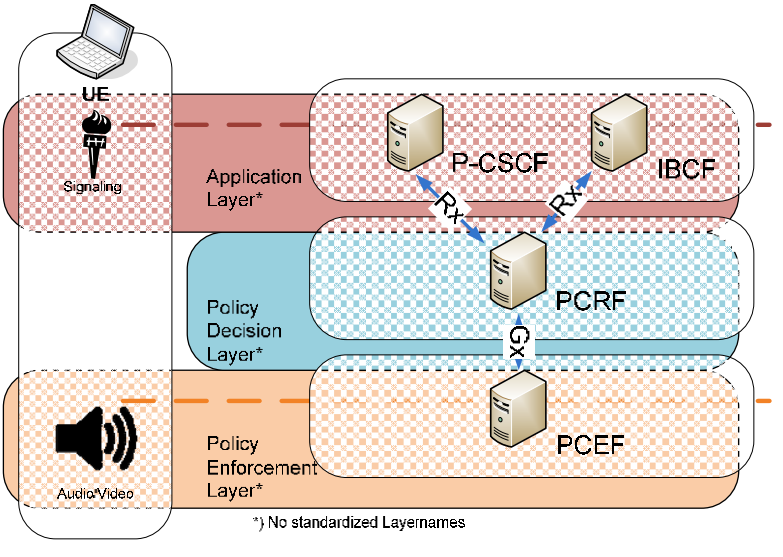


Fig. 2. 3GPP PCC Architecture

On the other hand, as mentioned earlier, ETSI TISPAN has developed its own architecture for QoS enforcement which is called Resource and Admission Control Subsystem (RACS) [7] and illustrated in Figure 3.

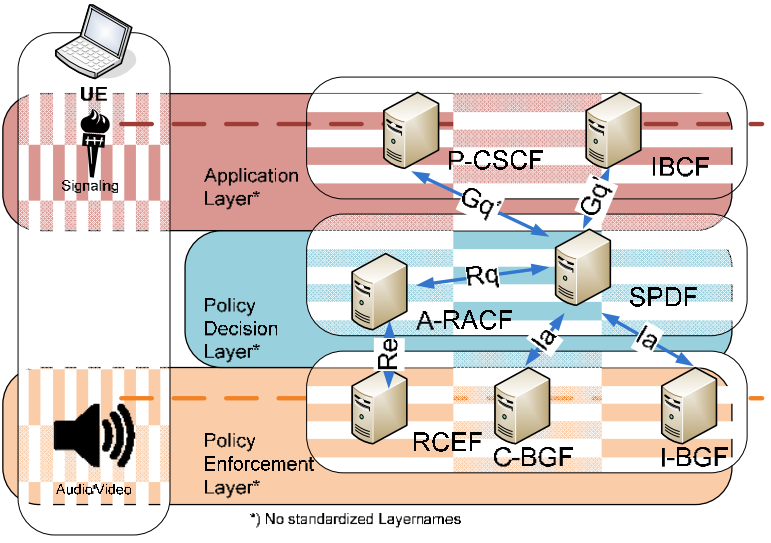


Fig. 3. ETSI TISPAN RACS Architecture

In order to merge these two architectural approaches into a common open source framework and considering the focus on access networks, BIQINI does currently not include an Interconnection Border Control Function (IBCF), nor a Core or Interconnect Border Gateway Function (C-BGF or I-BGF). Charging functionality has been included but interfaces have not been implemented yet. Furthermore we have decided to integrate and harmonize PCRF, Service Policy Decision Function (S- PDF) and Access Resource and Admission Control Function (A-RACF) into one component, namely the Policy Decision Point (PDP). ETSI-defined Resource Control Enforcement Function (RCEF) and 3GPP-specific PCEF functionalities have been merged into a Policy Enforcement Point (PEP) component. The resulting architecture is aligned to the IETF recommendation [4] based on three components: AF, PDP and PEP.

BIQINI is heavily relying on advanced routing and traffic control tools provided by the open source operating system Linux, most notably several system tools which support IP traffic queuing configuration. A detailed description of routing, switching, bandwidth management, queuing and IP security functions in Linux systems can be found in [5], in the rest of this section we will only provide an in-depth view on the default Linux command-line application for queuing configuration which is called traffic control (tc) and supports the modification of the queuing strategies to be used for outgoing and incoming IP packets on specific network interfaces.

In tc, the different queuing types and –strategies are denoted as queuing disciplines. After setting up new queuing disciplines, IP packets must be assigned to certain queues

using so-called filters which match the IP packets against specific patterns, corresponding to certain fields or byte sequences in the packet. Examples of these patterns include, e.g., source IP address, ports or any other field in the IP header. Upon successful match, the specific IP packet will be assigned to the corresponding queue. Note that a queuing discipline can, for instance, realize bandwidth management, reorder packets, delay packets, modify packets, etc., depending on the selected queuing discipline. A list of supported and implemented queuing algorithms can be found in [5].

The BIQINI implementation combines several queuing disciplines to realize QoS enforcement. The classful Hierarchical Token Bucket (HTB) algorithm manages the reserved bandwidth by allocating requested bandwidth to microflows. The queuing discipline dsmark manipulates the DSCP field of IP packets, marking all IP packets queued in a specific dsmark queue using a specified DSCP value.

BIQINI uses the queuing discipline netem for implementing realistic access network emulation. The netem algorithm can delay, reorder, drop, and duplicate IP packets. By setting up several netem queuing disciplines, with different delay and loss values for distinct DSCP marking emulates a DiffServ[2] network.

Finally, selective rejection or dropping of IP packets can be configured using the ipTables utility. A rule in ipTables describes traffic patterns and defines corresponding actions for this traffic, e.g., drop, reject, or accept. The traffic can be categorized by means of ports, addresses, protocol numbers, flags, etc. Similar to common firewalls, ipTables supports default rules for packets that cannot be matched to a rule. In most cases it is preferable to drop or reject all traffic that is not explicitly accepted by a rule.

3 BIQINI – Architecture and Basic Concepts

Based on the survey of related architectures provided in the previous section, we will now present the key concepts and the resulting architecture adopted for the BIQINI QoS enforcement framework in detail.

3.1 Architecture

The basic architecture of the BIQINI implementation is sketched in Figure 4. In the depicted scenario, the QoS enforcement is applied on an access link which on both ends is protected by respective PEPs. Whereas PEP 1 on the user side ensures that the micro-flow from the user agent to the core is scheduled correctly such that the service requirements (e.g. bandwidth) are fulfilled, PEP 2 on the core side is performing the same task for the opposite traffic direction. Note that, if PEP 1 were not installed, the user could use the access link excessively with other services, for instance sending extremely large emails or uploading huge amounts of data. Such bandwidth consuming services could then severely reduce the quality of real time services like voice communication. Thus, PEP 1(included in e.g., Customer Premises Equipment (CPE)) ensures proper bandwidth usage in the uplink direction. The finding that a layer-3-QoS enforced access links must be protected on both ends (therefore extending the 3GPP and TISPAN architectures) is an essential outcome of the BIQINI project.

Otherwise QoS enforcement cannot protect uplink and downlink simultaneously, bearing the risk of QoS degradation. This architecture is similar to [1], which covers a study of enforcing QoS on Customer Premise Networks (CPN) and supports the realistic asymmetrical modeling of impairments on access links.

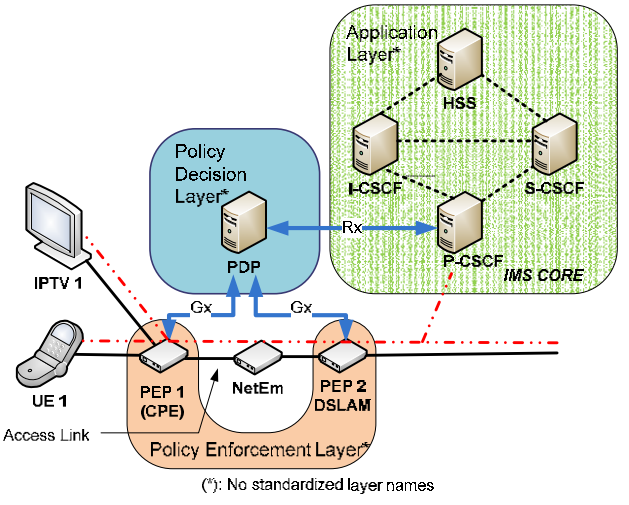


Fig. 4. Architecture of BIQINI QoS Enforcement for Access Links

When no GGSN or DSLAM is available, the characteristics of a specific access network (e.g., ADSL or 3G) can be accurately emulated from the point of view of a layer 3 protocol with the NetEm instance depicted in Figure 4 [14]. This is done by means of the Linux netem queuing discipline.

In order to differentiate between different service classes, we also have installed a netem instance on the PEPs. To be more specific, a microflow of the class "realtime conversational" needs low delay values and loss rates. In this case, our PEPs have to guarantee that the service receives the correct QoS on the access link. To realize this, the PEPs have to mark the IP packets with the correct DSCP value corresponding to this service class and handle it correspondingly. When the packets traverse the core network, DiffServ enabled router can determine which service class is to be used for a specific packet. As an example, we suggest to use the DSCP class 0x03 for realtime conversations. Thus, both PEPs have to mark packets of this class with the DSCP value 0x03, and at the same time the netem at the access link has to assign packets with DSCP value 0x03 to the queuing system handling realtime conversation, which, on its part, must realize low delay values and loss rates. On the other hand, best effort traffic could receive for instance the DSCP marking 0x00, which causes netem to treat such IP packets with a delay of several hundreds of msec and loss rates of 2% and beyond.

In our overall QoS architecture, the PEPs are responsible for realizing bandwidth management. To this end, each incoming flow is shaped according to the installed

rules. Flows that utilize too much bandwidth are queued at the PEP, thus increasing the corresponding delay value. In the worst case, this may lead to dropping the packets as soon as the queue is filled.

Our PEPs can be configured with or without ipTables (see section 2). In the case of a configuration without ipTables, any traffic is admitted but marked at the PEP as best effort traffic. Therefore, netem treats this traffic with lowest quality. However, if services are enforced through the PDP, they are marked at the PEP with a different ("better") DSCP value and are treated with higher priority. Additionally, the PEP also ensures that the enforced services can utilize their required bandwidth.

The communication between AF, PDP and PEP is realized using the Diameter [8] protocol. More specifically, the AF utilizes AA-Requests (AAR) and AA Answer (AAA) messages over the Rx interface to transport authorization requests to the PDP, whereas a Session Termination Request (STR) terminates a session. The communication between PDP and PEP is realized over the Gx interface and uses Re- AuthRequests (RAR) and Re-Auth Answer (RAA) messages. Figure 5 illustrates the Diameter messages employed.

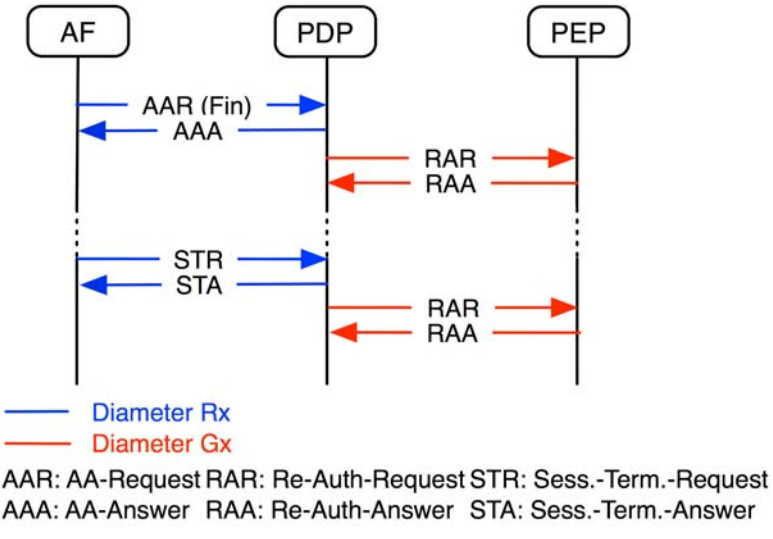


Fig. 5. Message Flow between AF, PDP and PEP

3.2 PDP – Policy Decision Point

As already mentioned previously, the PDP is the specific component that is responsible for translating the media-specific data of a service request (like codec and media type) to QoS-specific parameters (like bandwidth and delay requirements). In order to realize this task, the PDP uses rules from an external rules repository, which allows deriving the appropriate QoS parameters from service specific and user specific data.

Our implementations of the Rx and Gx interfaces are based on the jDiameterPeer due to the Fraunhofer FOKUS group. On top of it we have built a basic logic, which is able to handle incoming messages, to keep pointers to the corresponding state machines and to forward messages to the Rx and Gx interfaces. We have implemented the resulting state machine (depicted in Fig. 6) as well as "dummy" interfaces to a Subscription Policy Repository (SPR), which handles user and domain policies. Additionally, our implementation provides references to the PEP instances which have to be used for each subscriber.

One of the main tasks of the PDP consists of mapping all requests to sessions, thus enabling the storage of a consistent state for each session. In this context, messages received by the Policy Decision Point (PDP) can be subdivided into preliminary service information messages and final service information messages. Whereas final service information messages install rules at the PEP, preliminary information messages only check if a requested service meets the corresponding policies and if there are enough resources available.

The resulting state machine includes therefore the following set of states:

- *Receiving*: waiting for incoming AAR message
- *Accepted PRE*: received a AAR with Service-Type AVP set to PRELIMINARY_SERVICE_INFORMATION
- *Rejected*: Service information received with not acceptable content (either due to policy or insufficient resources)
- *Accepted Final*: received AAR with Service-Type AVP set to FINAL_SERVICE_INFORMATION
- *Committed*: enforcing the rules at the PEP successful

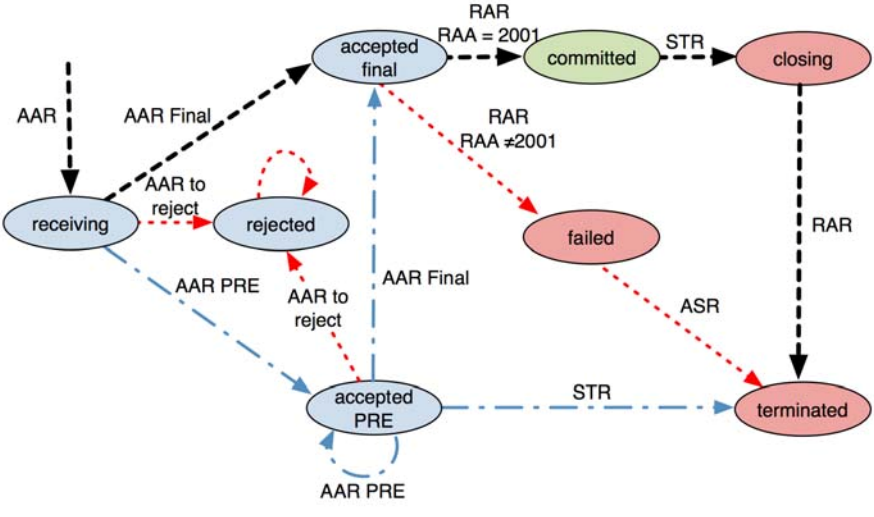


Fig. 6. PDP State Machine

- *Closing*: received STR from the AF, trying to terminate session at the PEP
- *Failed*: enforcing the rules at the PEP failed
- *Terminated*: Session successfully terminated

Note that in Fig. 6, a sequence of black (dashed) arrows depicts the traversal through the states, if the initial request already contains the final service information and if the installing of the rules at the PEP works properly. Blue arrows (dash-dotted) indicate that the initial request contains only preliminary service information, and a second request is sent to install the rules at the PEP. Red arrows (dotted) are used if either the request coming from the AF is not acceptable or the installation of rules at the PEP has failed. In order to maintain the clarity of presentation, the three states necessary for a session update are not shown in the diagram. Note that for each session handled at the PDP, a new state machine is created in order to deal with the corresponding messages.

3.3 PEP – Policy Enforcement Point

The Policy Enforcement Point (PEP) is responsible for detecting microflows and processing them according to the configured QoS parameters. Both processes (detection and processing) are defined by rules which are received via the Gx interface from the PDP.

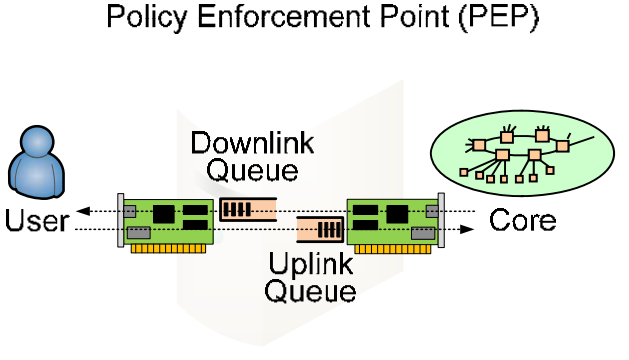


Fig. 7. Interface Cards and Queuing at the PEP

Our implementation is realized with Java, reusing the jDiameter stack already introduced. The detection and enforcement process is realized by Linux tools like tc and ipTables. The PEP stores the current state and configuration of each rule and tracks the bandwidth consumption in the PEP Rules Management module for charging. For a comprehensive illustration of the main PEP components we refer to Figure 7.

The PEP works an intermediate component between the communicating hosts. In order to be fully transparent for IP traffic, it is essential to configure the PEP as a bridge. With Linux, this requires two interface cards at the host, one towards the user agent and one towards the core network. The queuing is always realized at the

sending interface, whereas QoS enforcement for traffic directed to the user agent can only be realized on the interface card towards the user agent (i.e. downlink). Similarly traffic directed to the core network can only be treated at the interface card towards the core network (i.e. uplink). In our implementation, we use the Linux system tool brctl to configure a Linux host with two interface cards as a bridge.

Additionally, we have decided to reuse the concepts for installing, modifying and removing of rules by means of the Diameter protocol from the 3GPP PCC architecture, where these rules are called "charging rules" and include four main aspects:

- Each rule has a unique identifier (Charging Rule Name), which is mainly necessary to be able to remove a certain rule.
- Each rule is responsible for a certain microflow characterized by a flow description using the following parameters: source IP address, destination port, destination IP address and protocol number. In the case of a bidirectional flow, two such flow descriptions are required.
- The third part of a rule contains QoS information which is described in terms of the guaranteed bit rate, the maximally requested bandwidth and the traffic class (like streaming, realtime conversational, etc.). The traffic class is used to set the DSCP value properly.
- The last part in the charging rule concerns charging information. As our implementation currently is prepared for charging but does not implement charging interfaces, any information contained in this part is ignored.

Altogether, these four parts are encoded as so-called Charging Rule Definition AVPs, see Figure 8.

```
[Charging-Rule-Install]
        [Charging-Rule-Definition]
              [Charging-Rule-Name]                Video-Rule;12345
              [Flow-Description]
                    permit out 17 from 10.0.0.1 to 10.0.0.6 10001
              [Flow-Description]
                    permit in 17 from 10.0.0.6 to 10.0.0.1 3400
              [Flow-Status]                       ENABLED (3)
              [QoS-Information]
                    [QoS-Class-Identifier]        1
                    [Max-Requested-Bandwidth-UL]  175000
                    [Max-Requested-Bandwidth-DL]  175000
                    [Guaranteed-Bitrate-UL]       150000
                    [Guaranteed-Bitrate-DL]       150000
              [Online]                            DISABLE_ONLINE (0)
              [Offline]                           ENABLE_OFFLINE (1)
              [Metering-Method]                   DURATION (0)
              [AF-Charging-Identifier]            chargingid987654321
```

Fig. 8. Example of a Charging Rule Definition

After starting the PEP, some generic rules have to be installed first (for example signaling traffic should be able to pass under all circumstances). Such predefined rules are stored in the configuration file of the PEP and are activated automatically. As an alternative option, a Charging Rule Command triggered by the PDP can also install these rules.

After receiving an installation request via the Gx interface, the PEP has to configure the bandwidth management (realized with the HTB queuing discipline, see section 2) and activate the firewall for the respective microflow. The rate and ceil parameters of the tc command are configured by the guaranteed bit rate and Max-Requested Bandwidth value of the QoS Information AVP, which has to be completed both for the uplink and downlink direction in the case of a bidirectional microflow. To realize DiffServ marking, we add a dsmark queuing discipline and configure a DSCP value that fits to the traffic class. In the next step, the firewall is informed by the ipTables command that this microflow has to be allowed to pass through. As mentioned before, it is also possible to run the PEP without gate blocking. By using the tc queuing system HTB, it is guaranteed that a certain microflow cannot exceed the reserved maximum requested bandwidth. If too much traffic should be injected into the network, the PEP will shape the traffic according to the installed rules. Additionally, the currently used bandwidth and the total amount of sent bytes are observed for each microflow. These data could be used in later releases to realize flow-based charging. Note that we do not use these data to infer a bearer loss, as individual services could generate traffic patterns with no data sent over a long period of time that would wrongly be detected as bearer loss.

4 Results

We have tested the BIQINI QoS enforcement framework for various typical scenarios including voice and triple play applications in combination with an Open Source IMS testbed. For background load generation, we have used the Jperf frontend, which relies on the functionality of the Iperf traffic generator [13]. The chosen scenarios illustrate the reduction of best effort Constant Bit Rate (CBR) Jperf UDP traffic due to a realtime session, which has been signaled using IMS and enforced using BIQINI. Figures 9, 10 and 11 depict the goodput and jitter of a Jperf best effort stream for three typical cases:

- Scenario 1: 400 kbps best effort traffic is sent over the bridge, while after 15 sec the PEP is activated and consequently throttles down the traffic to 200 kbps, see Fig. 9 top.
- Scenario 2: The 200 kbps traffic is running in the background, while after 60 sec a CBR G.711 call starts for a limited duration of 12 sec, which needs about 82 kbps (excluding IP headers), see Fig. 9 middle. Note that the maximal bandwidth has been chosen in order to illustrate the impact of the G.711 call, which needs nearly half of the bandwidth available at Jperf.
- Scenario 3: A Variable Bit Rate (VBR) HD video stream consumes varying bandwidth depending e.g. on the amount of motion within subsequent frames and thus reduces the Jperf best effort traffic accordingly. In contrast to constant bit rate (CBR) voice streams, allocation of the maximum required bandwidth for VBR video streams leads to a significant waste of resources. This demonstrates one important BIQINI feature: due to the use of tc's HTB queuing discipline, the PEP does not reserve the maximum bandwidth required for the video stream. However, it adapts dynamically and automatically to the video stream's effective bandwidth consumption up to a specified maximum limit of the prioritized traffic.

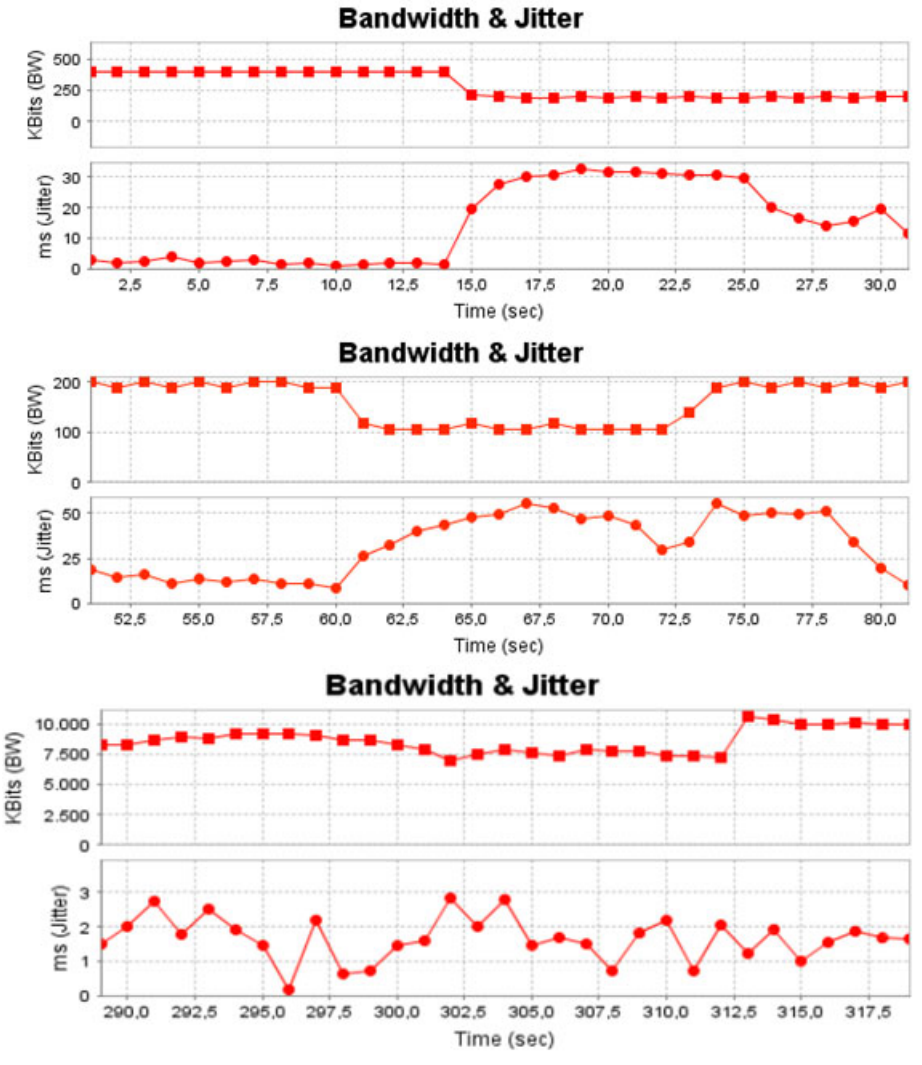


Fig. 9. Three Typical JPerf Scenarios: Reserved Bandwidth (Sc. 1, top), CBR voice call (Sc. 2, mid), VBR HD video call (Sc. 3, bottom)

5 Conclusions and Future Work

This paper presents the fundamental concepts and the architecture of the BIQINI QoS framework. Intended to serve as an add-on to the FOKUS Open Source IMS testbed, it is well suited for use also with other session-based signaling protocols like plain IETF SIP. The implementation comprises a Policy Decision Point (PDP) as well as an emulated Policy Enforcement Point (PEP) enabling transparent, realistic and automated QoS and QoE trials. BIQINI realizes the reference points Rx and Gx specified

by 3GPP and provides a policy interface which can be used to experiment with various policy repositories.

One important limitation of the emulated PEP concerns the delay between activation and reaction (i.e. QoS enforcement). In our experience, this delay can amount up to one second which we, however, consider to be acceptable for a prototype implementation.

As future work we plan to publish the BIQINI source code under the GNU Public License as a relevant enhancement of the current functionality of the Open Source IMS testbed. Moreover, recent projects at the FTW Vienna with specific focus on policies and services in IMS and non-IMS environments have started to extend BIQINI's policy repository by interfacing with semantic policy engine implementations.

Acknowledgement

Part of this work has been funded in the framework of the Austrian COMET competence program supported by the Austrian government and the City of Vienna. The authors would like to thank their colleagues Hannes Weisgrab, Ronald Fischer, Joachim Zeiss (FTW), Franz Edler (Alcatel-Lucent Austria) and Wolfgang Brandstätter (Telekom Austria) for their invaluable help with the implementation and all BACCARDI colleagues for their continuous support and enthusiasm. We gratefully acknowledge the tremendous support of our industrial project partners in BACCARDI, namely Alcatel Lucent Austria, Kapsch CarrierCom, mobilkom austria, and Telekom Austria.

References

1. Telecommunications and Internet converged Services and Protocols for Advanced Networking (TISPAN); Remote CPN QoS Control; Study on CPN - RACS Interaction, ETSI TR 182 031 (Draft version)
2. Blake, S., Black, D., Carlson, M., Davies, E., Wang, Z., Weiss, W.: An Architecture for Differentiated Services. IETF, RFC 2475 (1998)
3. Diez, A., Gouveia, F., Corici, M., Magedanz, T.: The PCC Rule in the 3GPP IMS Policy and Charging Control Architecture. In: IEEE Globecomm, New Orleans (December 2008)
4. Yavatkar, R., Pendarakis, D., Guerin, R.: A Framework for Policy-based Admission Control. IETF, RFC 2753 (2000)
5. Linux Advanced Routing and Traffic Control, `http://www.lartc.org`
6. 3rd Generation Partnership Project; Technical Specification Group Services and System Aspects; Policy and charging control architecture, TS 23.203 Release 8.0.0
7. Telecommunications and Internet converged Services and Protocols for Advanced Networking (TISPAN); Resource and Admission Control Sub-System (RACS): Functional Architecture, ETSI 282 003 V2.0.0
8. Calhoun, P., Loughney, J., Guttman, E., Zorn, G., Arrko, J.: Diameter Base Protocol. IETF, RFC 3588 (2003)

9. Waiting, D., Good, R., Spiers, R., Ventura, N.: Open Source Development Tools for IMS Research. In: Proceedings of the 4th International Conference on Testbeds and Research Infrastructures for the Development of Networks & Communities (2008)
10. Good, R., Ventura, N.: An end to end QoS management framework for the 3GPP IP multimedia subsystem. In: IEEE International Conference on Communications and Telecommunications 2007, ICT-MICC 2007, Malaysia, pp. 605–610 (2007)
11. One Voice Initiative, Version 1.0.0, http://news.vzw.com/OneVoiceProfile.pdf
12. The GSM Association: Rich Communication Suite Initiative, http://www.gsmworld.com/our-work/mobile_lifestyle/rcs/index.htm
13. National Laboratory for Applied Network Research (NLANR). Iperf Performance Measurement Tool, http://dast.nlanr.net/Projects/Iperf
14. Fabini, J., et al.: Generic Access Network Emulation for NGN Testbeds. In: Proceedings of the 4th International Conference on Testbeds and Research Infrastructure for the Development of Networks & Communities (TRIDENTCOM 2008), Innsbruck, Austria, Paper ID 3135 (2008)

An Identity Management Infrastructure for Secure Personalized IPTV Services

Daniel Díaz Sánchez[1,*], Florina Almenárez[1], Andrés Marín[1], Eugen Mikoczy[2], Peter Weik[3], and Thomas Magedanz[3]

[1] Telematic Engineering Department, Carlos III University of Madrid
Avda. Universidad, 30, 28911 Leganés (Madrid), Spain
[2] Slovak University of Technology, (Bratislava), Slovakia
[3] Technische Universität Berlin, (Berlin), Germany

Abstract. This article focus on IPTV security and IPTV service personalization by the introduction of an Identity Provider as new participant in IPTV service provision that deals with authentication, user profile and device profile management. The Identity Provider, integrated as part of the Telco operator, would provider user profiles with a wider scope than application specific profiles, enabling high personalization of services and improvement of user experience. Paper gives overview about existing IPTV security technologies but also describe novel architecture for secure personalized NGN based IPTV services.

Keywords: IPTV security, IdM, NGN based IPTV.

1 Introduction

IPTV is envisioned as the next step in user's TV experience with a provision of highly personalized services ranging from linear television, video on demand (VoD), near video on demand (n-VoD), personal video record (PVR) to advance blended services as messaging, chatting, presence and web 2.0 mashups.

Traditional broadcast-only content protection, as DVB Conditional Access [1], can not authorize users before they acquire the signal so contents are protected before delivered over the air. In this scenario, user authentication and content protection are performed entirely by the customer's hardware. DVB's major drawback is the absence of the concept of "user" that is substituted by the concept of customer (subscription). Customers are associated with hardware and their profiles are handled by the provider. Every user under a subscription receives the same service, thus personalization, if exists, is often poor.

IPTV provides a return channel that enables interaction as well as the separation of users and subscriptions can be provided. IPTV must be able to maintain protected contents within the boundaries of the subscription (contract) during the entire content lifecycle regardless the user equipment. In some cases, IPTV can prevent

* This work has been partially supported by "Jose Castillejo" mobility grant (Daniel Díaz) and Netlab project both financed by Spanish Ministry of Education.

T. Magedanz et al. (Eds.): TridentCom 2010, LNICST 46, pp. 668–683, 2011.

unauthorized users to acquire protected contents on demand since it can authorize them beforehand. However, some scheduled contents as linear television or n-VoD, which are broadcasted over IP using DVB-CA technology for efficiency, still require strong hardware protection.

In the article we summarize how the most relevant IPTV standardization bodies approach service and content protection, user identification and user profiling. The most innovative IPTV platforms allow users to consume IPTV services from several Content Providers; through managed and unmanaged networks; using different devices. The summary signs the fragmentation concerning IPTV content protection and user management. So, in order to achieve the provision of highly-personalized IPTV services while maintaining security it would be necessary to have a separate profile for every principal involved in the service provision. For instance, a content profile, which describes every detail about the content lifecycle; a subscription profile, managed by the IPTV provider, that express the rights a customer pays for and how he delegates them to users; a user profile for personalization; and a device profile containing protection capabilities, identifiers and cryptographic information.

It will be shown that IPTV user profiles are usually application specific, covering only IPTV related information, since IPTV users profiles are highly related to Telco-originated identities. These Telco-originated identities are based on *identifiers* and *authentication* mechanisms where the original scope is much narrower than in the case of Internet-originated Identity. We expect a user profile to be a user's Digital Identity available to many services, so rich in personal information, whose disclosure to the different services is handled by the user. In this way the user obtains the desired personalization from any service while respecting its privacy.

The article proposes the introduction of a Identity Provider (IdP) as new participant in IPTV service provision. The IdP deals with authentication and user profiles on behalf of different services including IPTV. Thus, IPTV services can be highly personalized by the use of an enriched user profile. Moreover, users are expected to store the profiles of their preferred devices under their user profile. In this way, the user can access IPTV services through several devices selecting the preferred one in every interaction. Device profiles would be used by IPTV service providers to check if a device is adequate for accessing a given content, to adapt the content protection to the selected hardware or to suggest any other user device as an alternative.

2 IPTV Security

Security topics in IPTV are: *service protection*, *content protection*, *key distribution, rights expressions, user management*, *device protection* and *network protection*. This section we will define the objectives of these security topics, how they are traditionally grouped together and how IPTV security technologies handle them. A service is a collection of video/audio contents bundle together in a package. *Service protection* ensures that subscribers are only able to gain access to services that are part of their subscription thus it governs the acquisition process. Once acquired, contents must remain under the agreement the user maintains with the content provider. *Content protection* techniques protect contents against unauthorized copy, distribution or manipulation.

Security solutions for IPTV must respect *user privacy*. Information about users as personal information, payment data or addresses must be protected by encryption and policy enforcement; also traceable information, as identifiers that might reveal service type preferences or habits, must be obfuscated. The user equipment as visualization devices, set top boxes, home gateways are part of the security infrastructure protecting contents. *Device protection* aims on avoiding attempts to hack devices, distribute virus or perform Denial of Service attacks (targeting the user or the provider network). Devices rely on cryptographic material stored in tamper proof hardware to perform some security tasks. In DVB the majority of security functions are delegated to devices. Devices are also highly related to *Content export* technologies that allows to move a content from one device to another preventing piracy.

The aforementioned security topics are grouped together in three major functional groups with some overlap among them: *Conditional Access Systems*, *Digital Rights Management* and *Copy Protection*. However, the practical realization of those security functions leads to two different scenarios, ruled by different content protection technologies, known as *acquisition and post-acquisition*.

DVB Conditional Access (CA) Systems [1-3], Marlin [4] and OMA BCAST [5] are security technologies governing acquisition. Content protection technologies might require dedicated hardware. In DVB it is necessary a combination of a descrambler, a Conditional Access Module and a smart card in every visualization device. OMA BCAST supports a smart card or DRM (smartcard less) profile. ETSI TISPAN has utilized NGN security mechanisms also for NGN based IPTV and trying to reuse the existing service protection and content protection standards. Once contents have been acquired, the post-acquisition scenario starts. Contents must remain within the bounds of the contract until the content lifecycle ends. Contracts can be enforced using *Digital Rights Management* and Copy Protection techniques as CSS (used in DVDs) or Advanced Access Content System (AACS). These specifications dictate how a legally acquired content may be converted to other Codec or format, edited, redistributed, or stored in other devices. The foundations for any copy protection system are rights expression languages. These languages have evolved from the simplest expression, as copy control indicator (CCI) field, to the complexity of MPEG21 Rights Expression Language (REL) [6], Usage State Information (USI) described in DVB-CPCM [7], Octopus DRM [8] used in Marlin (Open IPTV forum) or OMA DRM.

2.1 DVB IPTV Security

DVB specifications have been adopted by the majority of broadcasting systems during the last decades to distribute contents over satellite (DVB-S), terrestrial (DVB-T) and mobile networks (DVB-H). IPTV is expected to reuse DVB Conditional Access (DVB-CA) seizing already deployed head ends and consumer hardware. DVB-CA defines a holistic approach standardizing content format, metadata and protection procedures from the head end to the user equipment involving content providers, network operators and consumer electronics manufacturers. Fig. 1 shows the structure.

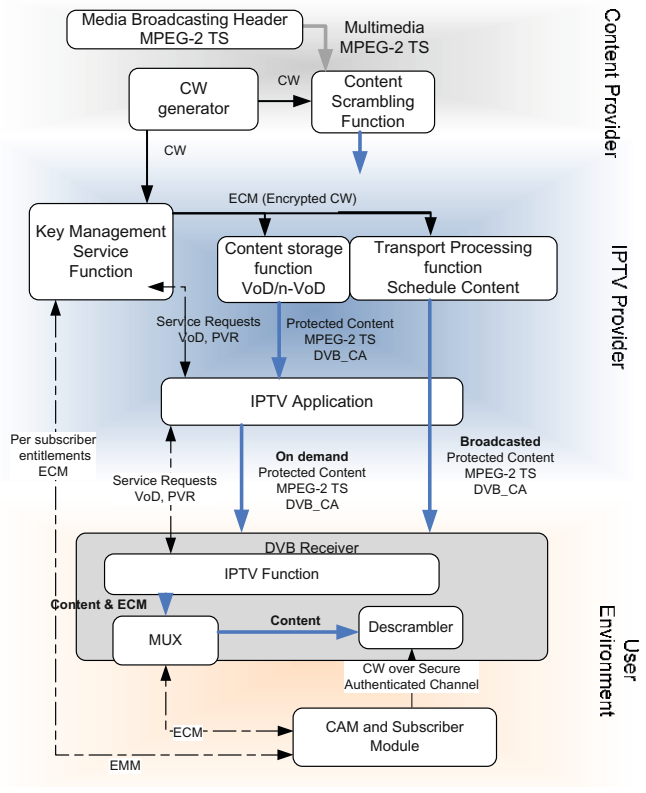


Fig. 1. Distribution networks. IPTV provides bidirectional communication and access control to the network whereas traditional broadcast networks cannot.

DVB Conditional Access (CA) Systems are defined across several specifications as DVB-CA (Conditional Access), DVB-CSA (Common Scrambling Algorithm) [1] and DVB-CI (Common Interface and CI+) [3][9].

DVB uses MPEG-2 *Transport Streams* (TS) as the preferred ***content format***. DVB IPTV encapsulates MPEG-2 Transport Streams in IP packets, reusing thus the traditional Conditional Access infrastructure and hardware. MPEG-2 TS is very useful for broadcasting over networks where errors might occur since it combines data streams with audio and video. MPEG-2 defines a *Program* as a set of *Packetized Elementary Streams* (PES) containing audio, video and clock references. It includes also data tables describing the relationships between the streams and data called Program Specific Information (PSI). There are three PSI tables related with Conditional Access: the Program Association Table (PAT), the Program Map Table (PMT), and the Conditional Access Table (CAT). The last points to *entitlement management messages* (EMMs) and *entitlement control messages* (ECMs).

Content acquisition in DVB broadcasting only networks is entirely managed by the end user hardware. DVB relies on SimultCrypt [2] which separates content encryption, content delivery and key distribution. Audio and video is scrambled with

a hardware-generated unpredictable session key called Control Word (CW) that changes frequently. DVB traditionally used Common Scrambling Algorithm (CSA) [1] for scrambling but new algorithms, based on Advanced Encryption Standard (AES), are under development, as ATIS CSA, DVB-CSAv3 or DVB-CPCM Local Scrambler Algorithm. These algorithms are inefficient in software implementations to prevent the development of software cracks.

The key distribution system is not standardized except for the messages used to convey that information to customers and the interface between hardware. CWs are encrypted with a *Service key* (SK) and distributed using ECMs. Providers send EMMs contain the SK and DRM information, encrypted with a *customer key* CK, to update SK. In broadcasting only networks ECMs and EMM are broadcasted together with the content in MPEG-2 CAT tables and repeated frequently to deal with transmission errors. Nevertheless, in IPTV the reception of ECMs and EMMs can be acknowledged thus can be sent directly to users.

DVB manages i*dentification*, *authentication* and *authorization* in user equipment in cooperation with CA ***hardware***. DVB Common Interface (CI/CI+) [3][9] defines the communication interface that every Conditional Access Module (subscriber module) must fulfil to communicate with a standard descrambler (decryption system). A CAM implements the key distribution protocol (EMM/ECM) for a given CA system provider. The most advanced version of the Common Interface specification, CI+, defines how to use the descrambler's public key to open a Secure Authenticated Channel between the CAM and the descrambler for CW delivery. As the user might infer, the CAM must be collocated with the descrambler so in order to use a different visualization device it is necessary to move the CAM from one device to another. Fortunately, some works propose to place the CAM in a gateway in order to share it with several descramblers through IP [10] or DLNA [11].

Post-acquisition process starts after the content is descrambled. During the acquisition process, decrypted contents never go out of tamper proof hardware so the final destination of the content (a digital video record or a TV) must satisfy several requirements. To protect descrambled contents from being accessed once acquired, the decryption hardware, if not integrated in the visualization device, should export contents through a High-Bandwidth Digital Content Protection [12] (HDCP, HDMI, GVIF) or a similar secure interface. Moreover, DVB defines also some specifications, as DVB-CPCM [7], to allow contents to moved, copied or exported. To identify authorized devices, DVB-CPCM supports the definition of *authorized domain*: the set of DVB-CPCM compliant devices within a household among which contents can be moved.

The reader must note that DVB lacks of ***user management*** so there is neither user authentication nor profile. *Identification, authentication* and *authorization* are performed by the user hardware thus the customer is identified by its equipment. In broadcast only networks, to demand authorization for accessing new contents, a customer uses a modem, integrated in the user equipment, or calls to the customer service. In DVB IPTV this can be handled by the return channel.

2.2 Open IPTV Forum Security

Open IPTV Forum (OITF) has developed an end-to-end solution to access enriched and personalized IPTV services that can be accessed through either managed or

unmanaged networks [13]. It aims on standardizing the user-to-network interface (UNI).The architecture of the system is depicted in Fig. 2.

The OITF content protection supports three media ***formats***: OMA DCF, Marlin IPMP and MPEG2-TS. Regarding ***content protection***, OITF describes its architecture in [14] with two different approaches: the terminal centric approach (CSP-T) and the gateway-centric approach. OITF defines three different keys for content encryption that are provided by the Content and Service Key Management Function. The Content Key is used for Marlin Content encryption and both Service Key and Program Key are used as described in section 2.1 to generate the ECMs and EMMs to cope with MPEG-2 TS content delivery. In OITF, MPEG-2 TS contents can be delivered protected as stated in DVB specifications with either a Marlin CA protection (identified with the appropriate CA descriptors) or with any other DVB CA.

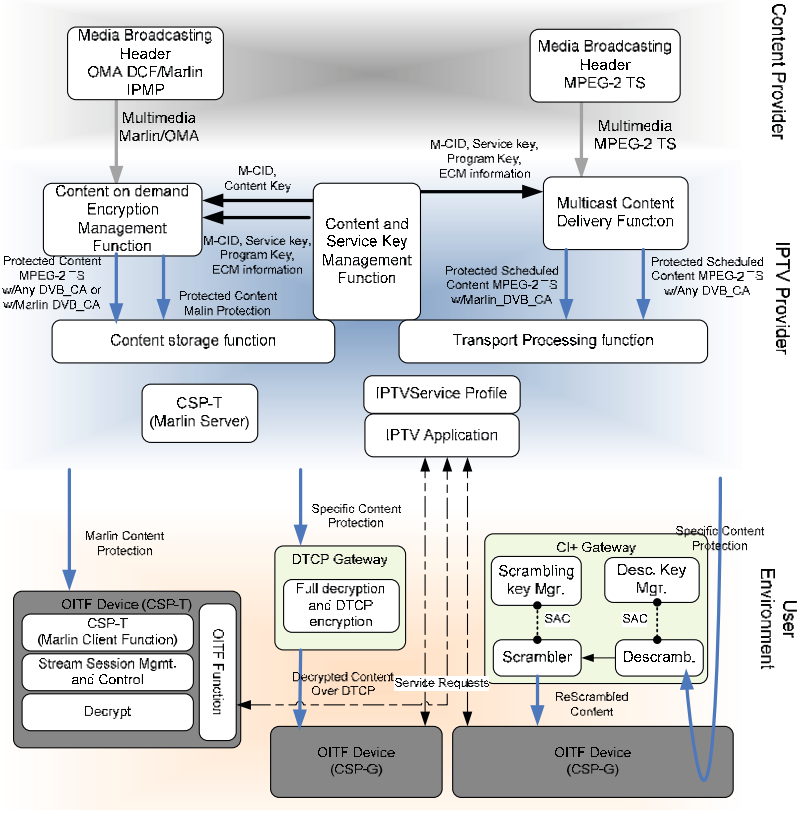


Fig. 2. Open IPTV forum functional architecture of content protection

CSP-T is based on Marlin Broadband [13] defined by the Marlin Developer Community. The CSP-T client in the OITF interacts directly with the CSP-T server function in the network to acquire protected content. The OITF function (OITF device) requests contents through the IPTV application located it the IPTV provider

network. A protected content acquisition involves the IPTV application and IPTV Service Profile function. The acquisition starts when the OITF detects the protection by the execution of Node acquisition, Link acquisition and License Acquisition Marlin protocols [14]. Marlin uses an Octopus-based DRM system such that uses Nodes and Links objects to express relationships among principals within the system (e.g., users, devices, and subscriptions). Once the OITF device obtains a Node and Link represented by a Business Token it requests a license from the IPTV application. The IPTV application requests the user profile and the business token from the IPTV Service Profile and issues a license bound to the Node that represents the user. If the user is allowed to access the requested content, it will be able to request the corresponding Content Key to the CSP-T Server so the CSP-T server can request the key in behalf of the user to the Content and Service key Management function.

The gateway-centric approach is optional in OITF. The gateway acts as a bridge between the network and the OITF device. The content protection is terminated in the gateway and a local protection system is used between the gateway and the OITF device. The OITF, upon the reception of content protected with a *CA descriptor* that it can not handle, performs discovery to find a CI+ gateway to decrypt the content. If the OITF device finds an appropriate gateway it authenticates to the gateway and redirects the content for decryption. The gateway is equipped with a DVB descrambler that opens a Secure Authenticated Channel with the CAM to receive key material. The descrambler outputs the descrambled content to a scrambler that encrypts the content into a compatible format. Finally, the gateway sources the protected content to the OITF device.

Regarding ***user management***, traditional DRM systems, as DVB, licenses are directly bound to the device that is used to obtain the rights and also to a customer identity. In Marlin, a License is typically bound to a user (more precisely, to an Octopus Node representing the user), and relationships between users and devices, or users and subscriptions, are maintained separately [14]. User management is handled differently for managed and unmanaged networks. In unmanaged networks OITF proposes, in [16], the use of HTTP Digest Authentication. The user must authenticate with the Service Access Authentication located in the IPTV provider network in order to be identified and authorized to access IPTV services. In managed networks, user identification and authorization is based on either 3GPP IMS AKA or SIP Digest. The authentication in this case is triggered when either the Internet Gateway is switched on or the user demands personalized services. Moreover, in some scenarios, Generic Bootstrapping Architecture (GBA) [17] and SAML [18] Web-based Single Sing on techniques can be also used.

The user profile is handled by the IPTV Service Profile. OITF specifications separate Subscription Profiles from User Profiles. Moreover, it manages the links between the principals as subscriptions, users and devices in a comprehensive way. Nevertheless, the user profile is still an application specific profile limited to the IPTV context; thus, it is hard to achieve a high degree of service personalization.

2.3 ETSI TISPAN IPTV Security

ETSI TISPAN works on NGN based IPTV as part of TISPAN NGN release 2/3 [18] on two IPTV architecture: NGN dedicated/integrated IPTV subsystem (non-IMS) [19] and NGN IMS based IPTV [20].

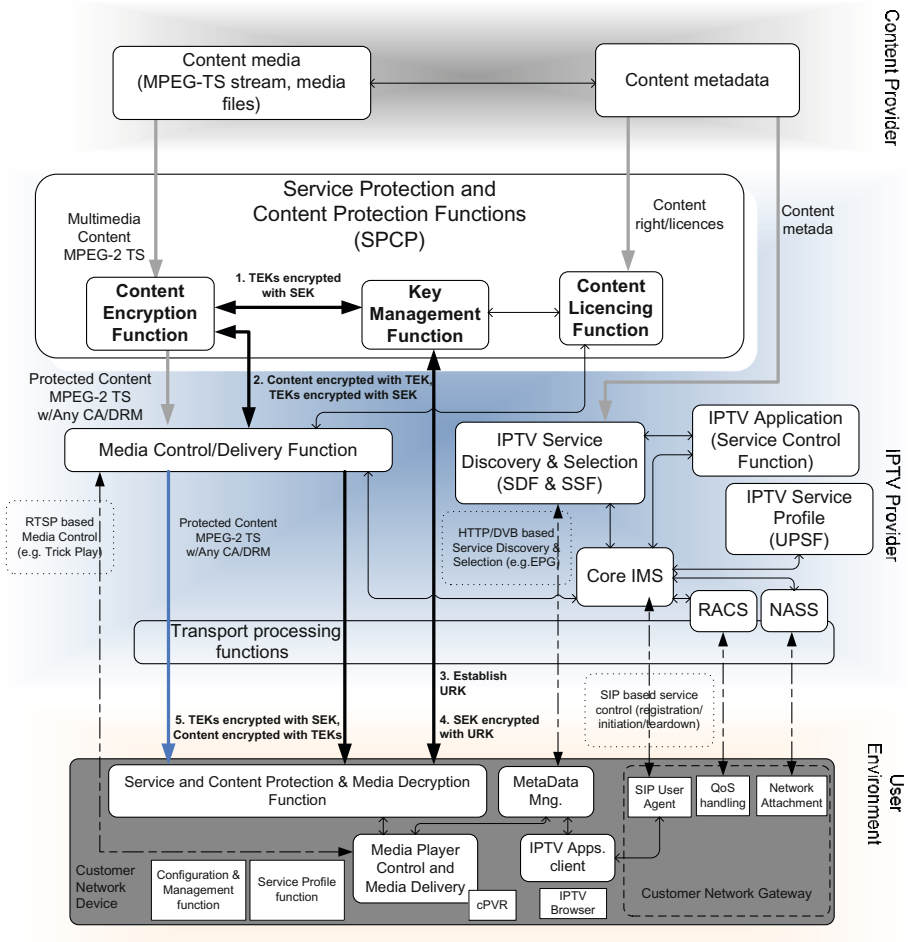


Fig. 3. TISPAN IMS based IPTV functional architecture with service and content protection

IPTV security has been part of NGN Security architecture specification [21] and TISPAN also provides deeper analyses in technical report for IPTV security for NGN release 3 [22].

TISPAN focus on two aspects of IPTV security service protection and content protection (SPCP): ***Content Protection,*** that assures a protection of content or content assets during its entire lifetime and ***Service Protection,*** that have to provide the protection of content (e.g. files or streams) and IPTV related service information during delivery which may include content already protected and meta data that the service provider adds to the content.

The generic model for service protection of IPTV as identified in [22] is based on well defined key hierarchies (3 or 4 layer hierarchies) uses a set of keys that provide cryptographic isolation of services and content for both unicast and multicast distribution of IPTV content:

User Root Key (URK) - A symmetric key used for the protected transfer of SEK in multicast service, or for protection of TEK in unicast service. This key is known only to the IPTV user and the SKMF (Service Key Management Function) and should be derived as part of an authentication and authorisation service (e.g. bootstrapping by GBA and/or IMS-AKA).

Session Encryption Key (SEK) - A symmetric key used for the transfer of traffic encryption keys on a multicast service. This key is known to the session members and to the the SKMF.

Traffic Encryption Key (TEK) - A short lifetime symmetric key used to encrypt the IPTV media within the NGN. This key is shared with all IPTV users for a specific channel or programme and with the MDF containing the CEF (Content Encryption Function).

ETSI TISPAN is re-used existing NGN security architecture and security mechanisms like NASS bunded authenification on transport layer and IMS AKA on service leayer and also bootstrapping mechanisms like GAA/GBA.

There are analyzed several candidate solutions for service protection and content protection (SPCP) for TISPAN NGN based IPTV [22]. Generally there are identified several candidates: OMA BCAST [5][23], DVB CSA/ SimultCrypt [24], 3GPP MBMS [25], OIPF Marlin [4].

TISPAN IPTV service protection model for both multicast and unicast services may be based on the 4-Layers or 3-Layers Key Hierarchy. Figure 4 shows IPTV service protection model based on 4-Layers Key Hierarchy and IPTV functional entities that are tightly related to service protection [22].

OMA BCAST solution addresses service protection and/or content protection and can take into account already deployed service protections can support DVB SimultCrypt and also MBMS. OMA BCAST DRM Profile provides solution for equipments without presence of a smart card. OMA BCAST Smartcard Profile supports using IMS UICC.

2.4 Other IPTV Solutions

A large number of IPTV providers have created and deliver own services through commercial proprietary-based vertical solutions such as Microsoft IPTV Edition (MSN TV 2 set-top box and Xbox console), Apple TV, Intel, Real player, etc.

Although Apple TV platform is not quite IPTV yet, it is just a link from the PC to the television[1] for content transfer. These solutions focus on the ***content protection*** using fully proprietary DRM solutions; for instance, Windows Media DRM [26] from Microsoft TV, or DRM Plus from Verimatrix VACS (Video Content Authority System) software for Intel.

They use DRM and PKI keys (i.e. X.509 certificates) for access and authentication stored in a dedicated chip (i.e. smart cards). The authentication required for VoD and PVR is based on network identifiers (i.e. through regional clusters) or serial numbers in the deco without tamper proof ***hardware***. So, authorization can follow two approaches. Firstly, server-side using unicast connections through Access Control

[1] http://www.iptv-watch.co.uk/2007/06/10/apple-tv-could-add-iptv-capability-in-2008/
http://www.iptv-watch.co.uk/2009/08/25/zte-and-apple-get-into-iptv/

Lists (ACLs), the subscriber record is looking up in a database before granting a given connection (i.e. VoD, PVR). Secondly, using broadcast or multicast applications by providing a subscriber's device(s) with cryptographic keys necessary to access through IGMP protocol to join or leave an IP multicast (e.g. Pay TV channels). The distribution of these keys is not specified, it can be performed as part of setting up a unicast connection for SSL, or alternatively, keys can be sent to the receiver in advance. Thus the user profile is not managed adequately.

On the other hand, 'over the top' services such as Youtube, Megavideo, Joost are using the Internet as a bidirectional channel to provide global reach. These solutions do not offer security neither services nor QoS. The access is usually anonymous.

3 Improvements for Secure Personalized IPTV Services

3.1 Motivation

The majority of user management functions as authentication attribute exchange and user profile management have been already addressed by the Internet community and standardized by relevant organizations. Identity Management technologies support the concept of a user-centric Digital ID as a set of attributes. The criteria for selecting attributes for an identity matches, among others, technical needs, roles intended to be played by the user, privacy concerns and legal constraints. Thus, an Identity Management System can be used for many purposes as *authentication, authorization, verification, uniqueness, linkage, preferences/attribute exchange, and reputation* [27] by using a set of protocols, languages and processing rules.

The integration of traditional content distribution networks with Internet Identity Management systems was inconceivable due to the lack of a return channel. Nevertheless, Next Generation Networks break this tendency allowing this integration. It can be considered one hot topic in current security and NGN research; in fact, there are several works that proposes such integration. In [28], the authors establish the requirements for the integration of Identity Management technologies in Next Generation Networks. Moreover, describes how the central notions in NGN (Telco-originated) identity management solutions are *identifiers* and *authentication* where the original scope is much narrower than in the case of Internet-originated Identity hampering the access to services when they are provided outside Telco domain. Regarding workgroups, for instance, the Focus Group on Identity Management (FG IdM) [29] or Kantara Initiative, with a broaden objective, are contributing to this integration.

For the purposes of IPTV security we concentrate in what is known as Identity Federated model. In a Telco federated model, users' data are in an Identity Provider (IdP) located in the operator domain with interfaces to Internet so the information can be easily accessed by Relying Parties (or external services). Authentication is handled by the operator or a third party depending on where the source of authentication is. This is known as a meta Identity Provider since implements many different interfaces for Relying Parties, as SAMLv2.0 [18], ID-WSF (Identity Web Services Framework) and WS-Federation; and also several authentication mechanisms around a user profile.

In section 2, we described IPTV security architectures where the concept of user is either missing or associated with a subscription or dedicated hardware. Nevertheless, user profiles are still managed by the IPTV provider.

Our meta-IdP aims on providing services for user authentication, user profile management and device profile selection. User authentication can be performed by either third parties (PKI, user/password, OpenId) or the Telco provider (GBA, IMS AKA). The user profile (Digital ID) links IPTV identifiers under a subscription to user preferences. Thus, IPTV providers can deliver highly personalized services to the user relying not only on IPTV user preferences but also on information collected by user profiling during IPTV service consumption or provided by third party Internet services through the user profile. Device profiles are stored in a list of preferred devices under the user profile. A device can be explicitly selected by the user or automatically according to the context. The subscription, user and device profile are associated under an IPTV session with specific user (user identity) and should be used for accomplish service personalization, content casting and content protection casting. The following sections describe the relation among principals, the architecture, interface and services of the proposed IPTV security infrastructure.

3.2 Relation among Principals

The Fig. 4 shows the relation among principals. The subscription profile is managed by the IPTV provider. The subscription profile stores customer's information as: payment, contract details, rights, delegation policies and authorized users. The delegation policies express how subscription rights are delegated to users (e.g. parental control, content types, playback time window…). Users are managed by the Telco meta-IdP system. The meta-IdP provides user authentication and profiles. A user profile is a collection of attributes or claims about service preferences, relation with Internet services (as social networks or communities) and preferred devices.

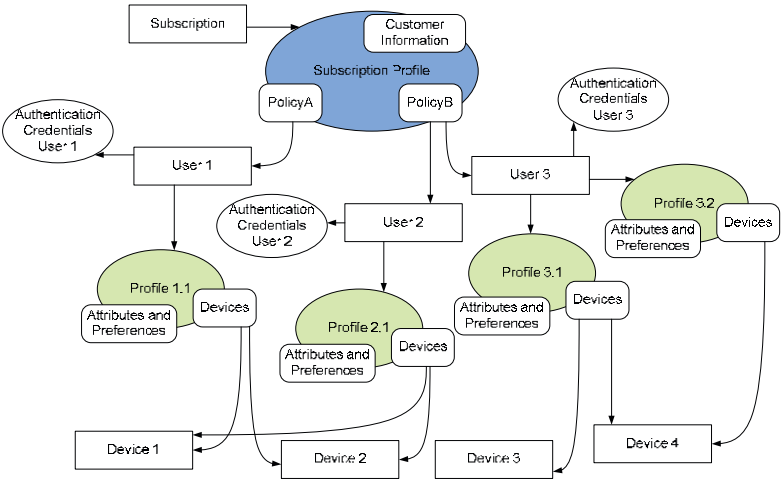


Fig. 4. Relations among principals in our Telco meta Identity Provider

A user might have more than one profile in its identity portfolio in order to separate duties or roles. The device profile contains a detailed description of the supported formats, content protection technologies, identifiers and credentials (i.e. descrambler public key).

As shown in Fig. 5, an IPTV session is characterized by a subscription profile, a user profile and a device profile. Under the scope a session a user will be able to access personalized contents if his subscription rights are enough and the device meets the requirements of the content provider. The session information will be used for adapting the content to the device (content casting) and/or to deliver the content using the most appropriate content protection technology (content protection casting).

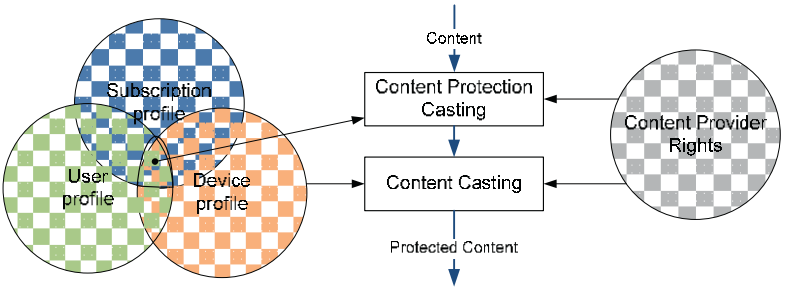


Fig. 5. IPTV Session characterization for content and protection casting

3.3 Architecture and Interfaces

In this section we describe the functional architecture of the proposed IPTV security infrastructure. Our proposal does not substitute but complements existing IPTV security solutions. The core element of our architecture is the meta-IdP located in the Telco operator domain. This element is not part of the IPTV security infrastructure itself since handles user authentication and profiles on behalf of any service including IPTV. Fig. 6 shows the architecture.

The meta-IdP exposes the Authentication Service and the Attribute Exchange and Assertion function. The Authentication function is intended to serve as the authentication endpoint for the majority of the services used by an average user (including Internet services). It must support multiple authentication mechanisms (and credentials) including Telco (GBA, IMS-AKA) and Internet (i.e. PKI, username and password or OpenId) authentication. The security assertions and attribute exchange interface conveys authentication decisions, profiles and attributes to third party services, Internet or Telco services. This interface must support many security assertion languages and protocols (SAML, WS-Federation, WS-Security, Oauth...). The idea is to facilitate authentication and user management to users and services improving user experience while reducing management costs.

The IPTV provider can take profit from the Identity Provider by using several functions as session management, service personalization, content casting and content protection casting. These functions might be implemented by IPTV platforms as DVB IPTV that lacks from an appropriate user management or mapped to existing IPTV

functions, for instance, TISPAN SPCP or Open IPTV Forum node and link acquisition. The Session Management function is in charge of retrieving the subscription profile, the user profile and the device profile. The Session Management function binds those profiles to a session identifier and may check the profiles against the license of the requested content determining the most appropriate content format (include encoding, resizing or aspect ratio modification.) and content protection .

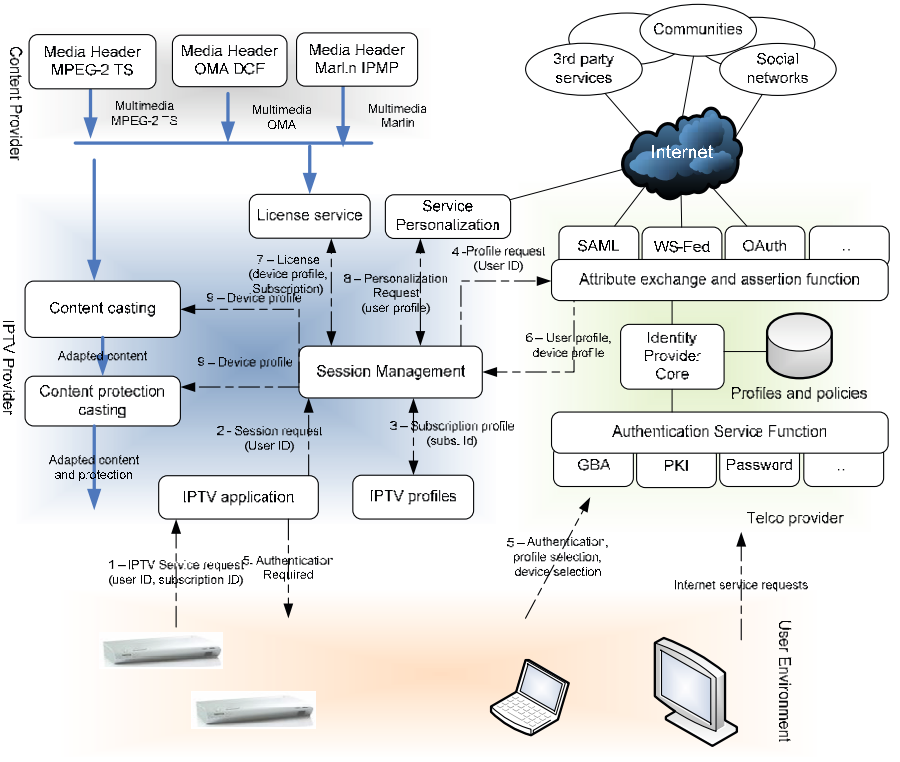


Fig. 6. Security infrastructure for service personalization, content and protection casting

The Service Personalization function receives the user profile from entities like UPSF/HSS, Session Management function IPTV application and uses it for retrieving information from service/user data and other services to personalize the IPTV service. Generally, in IPTV architecture can exist the hierarchical model of relations (as shown in Fig.7) between the internal service states of functional elements, the IPTV Service State information and Presence information and all these information could be used for user profiling and flexible IPTV service personalization [30];

- the internal service state information can be aggregated in IPTV Service State, and IPTV Service State data can be used to update Presence.
- IPTV presence may be related also to User Action Data (e.g. bookmarks) as well as used for updating user's Service Access History (for user profiling).

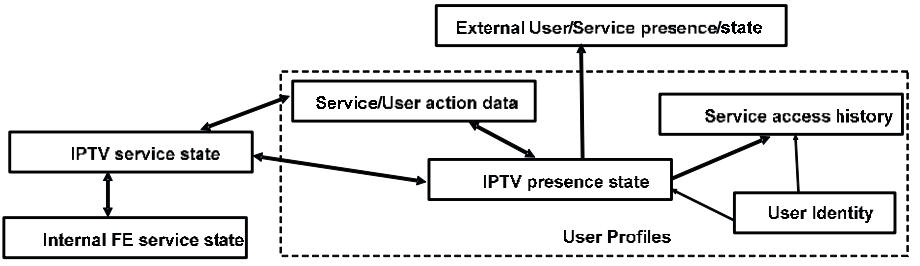


Fig. 7. Using service state and user profile and presence for IPTV service personalization

- User profile (presence state) could be updated from/to external application (including Internet social communities statuses and profiles).

The degree of personalization depends on the information disclosed by the user in his profile. Advanced IPTV services require mechanisms to personalized content and user interaction in all phases of IPTV service (service attachment, service discovery and selection, services initiation/modification/teardown and last but not least for any service interaction).

3.4 Content Acquisition

A content acquisition in the proposed architecture starts with an IPTV service request through a managed or unmanaged network. The user requests a service to the IPTV application providing his user and subscription identifier. The IPTV application sends this information to the Session Management function (e.g. core IMS). If the user has no valid session the Session Management function resolves the user identifier and finds the Identity Provider authentication function. The Session Management sends this information to the IPTV application, which redirects the user to the Identity Provider authentication function using, for instance, an HTTP redirection to an Identity Provider HTML or CE-HTML (for consumer electronics hardware) page.

Then, the user authenticates, selects the attributes to disclose in his profile (or a predefined user profile) and selects the device (or leaves it to the default). The Identity Provider asserts the user's identity to the Session Manager through the appropriate attribute exchange and assertion protocol. After that, the Session Manager requests the user profile and the device profile to the Identity Provider and matches user identity with the subscription profile delegation policy. If the user is entitled to access this content, the Session Manager checks the device profile with the content license to find out if the device fulfils content provider's requirements.

If the device is able to cope with the acquisition and post acquisition content protection requirements, the Session Manager selects the most appropriate content format and protection. Then the Session Manager sends the user profile to the Service Personalization function. The Service Personalization function might rely on user profile, service policies, presence/service state and aggregated user data from different sources as NGN Telco, third party and Internet services. The degree of personalization depends on the information disclosed in the user profile.

Finally, according to the session, the Session Manager starts the content adaptation. It triggers the Content Casting function to adapt the content to device‘s requirements as format, size, quality, bitrates... Moreover, it also triggers the Content Protection Casting function to protect the content with the appropriate technology.

4 Conclusions

This article goes through the most important efforts on IPTV content protection showing the fragmentation of the market. Every standardization organization has chosen its own content protection leaving user management and device management under a non interoperable silo that hampers personalization and scalability. This article proposes the introduction of an Identity Provider as new participant in IPTV service provision that deals with authentication, user profile and device profile management. The Identity Provider, integrated as part of the Telco operator, would provider user profiles with a wider scope than application specific profiles, enabling high personalization of services and improvement of user experience. Moreover, it can be also critical for content protection assurance since it can manage device profiles; thus helping IPTV providers to adapt the content to the device’s specific hardware. In that way, users can be able to select the device to be used, breaking the traditional tendency of binding users to devices; and content providers can be sure that their contents are under control during the entire content life cycle.

References

1. Support for use of scrambling and Conditional Access (CA) within digital broadcasting systems, ETR 281 V1. Technical Report, Digital Video Broadcasting (1996)
2. Implementation Guidelines, of the DVB Simulcrypt Standard, TR 102 035 V1.1.1. Digital Video Broadcasting (2004)
3. Common Interface Specification for Conditional Access and other Digital Video Broadcasting Decoder Applications, EN 50221. Technical Report, CENELEC (1997)
4. Marlin Broadband Architecture Overview for Marlin Adopters. Intertrust (2007)
5. Mobile Broadcast Services Architecture, Candidate Version 1.1, OMA-AD-BCAST-V1_1-20091013-C. Technical Report, Open Mobile Alliance (2009)
6. Wang, X.: MPEG-21 Rights Expression Language: enabling interoperable digital rights management. IEEE Multimedia 11(4), 84–87 (2004)
7. Digital Video Broadcasting Content Protection & Copy Management (DVB-CPCM), DVB Project Bluebook Document A094R2 (2008)
8. The Role of Octopus in Marlin. Technical Report, Marlin Developer Community (2006)
9. CI Plus Specification, Content Security Extensions to the Common Interface V1.2. Technical Report, CI Plus LLP (2009)
10. Díaz-Sánchez, D., Marín, A., Almenárez, F., Cortés, A.: Sharing conditional access modules through the home network for Pay TV Access. Transactions on Consumer Electronics 55(1), 88–96 (2009)
11. Díaz-Sánchez, D., Sanvido, F., Proserpio, D., Marín, A.: Extended DLNA protocol for sharing protected Pay TV contents. In: IEEE International Conference on Consumer Electronics, Las Vegas, USA (2010)

12. High-Bandwidth Digital Content Protection System Revision 1.3, Technical Report (2006)
13. OITF Release 1: Vol.1 Overview. Technical Report, Open IPTV Forum (2009)
14. OITF Release 1: Vol.7 Authentication, Content Protection and Service Protection. Technical Report, Open IPTV Forum (2009)
15. Open IPTV Forum, Functional Architecture V1.2. Technical Report, Open IPTV Forum (2008)
16. Generic Authentication Architecture (GAA), Generic bootstrapping architecture, TS 33.220 V8.7.0. Technical Report 3GPP, ETSI (2009)
17. Mishra, P.: SAML v2.0. OASIS Standard. Technical Report SAML v2.0, OASIS Security Services TC (2005)
18. ETSI ES 282 001, TISPAN; NGN Functional Architecture (2009)
19. ETSI TS 182 027, TISPAN; IPTV functions supported by the IMS subsystem (2009)
20. ETSI TS 182 028, TISPAN; NGN Integrated IPTV Subsystem in NGN (2009)
21. ETSI TS 187 003,TISPAN; NGN Security; Security Architecture (2009)
22. ETSI TR 187 013, TISPAN; Feasibility study on IPTV Security Architecture (2009)
23. OMA-TS-BCAST_SvcCntProtection – v1_0: Service and Content Protection for Mobile Broadcast Services", version 1.0, Open Mobile Alliance
24. ETSI TS 103 197,DVB; Head-end implementation of DVB SimulCrypt
25. 3GPP TS 26.237, IP Multimedia Subsystem (IMS) based Packet Switch Streaming (PSS) and Multimedia Broadcast/Multicast Service (MBMS) User Service. Release 8
26. Leung, Y., Peinado, M., Strom, C.: Binding Digital Content to a Portable Storage Device or the like in a Digital Rights Management (DRM) System, U.S. Patent 7010808. Microsoft Corporation (2006)
27. Palfrey, J., Gasser, U.: Digital Identity Interoperability and eInnovation. Retrieved from Case Study (2007),
    `http://cyber.law.harvard.edu/interop/pdfs/interop-digital-id.pdf`
28. Subenthiran, S., Sandrasegaran, K., Shalak, R.: Requirements for identity management in next generation networks. In: 6th International Conference on Advanced Communication Technology, pp. 138–142. IEEE, Los Alamitos (2004)
29. ITU-T Focus Group on Identity Management. Report on Identity Management Use Cases and Gap Analysis. ITU-T (2008)
30. Schumann, S., Mikoczy, E., Podhradsky, P., Muruchi, F., Maruschke, M.: Presence management and merging presence information for NGN services" on "Wireless and Mobile Networking", WMNC 2009, Gdansk, Poland, September 9-11. Springer, Heidelberg (2009) ISBN: 978- 3-642-03840-2

Framework for IMS Service Scenario Implementation

Andrey Krendzel[1], Jawad Hussain[2,*],
Josep Mangues-Bafalluy[1], and Marc Portoles-Comeras[1]

[1] Centre Tecnològic de Telecomunicacions de Catalunya (CTTC), IP Technologies Area
PMT, Av. Carl Friedrich Gauss 7, B4, 08860 Castelldefels – Barcelona – Spain
{andrey.krendzel,josep.mangues,marc.portoles}@cttc.cat
[2] Royal Institute of Technology (KTH),
Kungl Tekniska Högskolan, SE-100 44 Stockholm-Sweden
jawad.kth@gmail.com

Abstract. This paper presents an experimental framework for implementation of an IMS/NGN reference service scenario by means of open source software. Multiple service enablers are deployed to build this service scenario. Interoperability tests between the deployed IMS entities and user equipment are carried out, as well as performance measurements of signaling overhead and delay of different IMS procedures involved to support the service scenario. Then, some preliminary results are obtained. After that, the IMS prototype is integrated with in-lab UMTS/HSDPA and WLAN networks to test IMS procedures in more close-to-real environment. Additionally, practical experiences with the IMS testbed deployment are discussed.

Keywords: IMS, testbed, open source, service scenario, service enablers, interoperability, performance evaluation, decomposition.

1 Introduction

The IP multimedia subsystem (IMS) represents a uniform open architecture platform for a managed IP-based infrastructure that will enable the deployment of both basic calling services and an unlimited number of wireless-enhanced rich multimedia services that mix telecom and data services. In this context, rich means bundling multiple service enablers (e.g., voice/video connectivity, presence, instant messaging, conferencing, gaming, TV broadcasting) [1].

By itself, IMS does not specify any new service. Instead, it provides flexible tools and a unified platform for network operators and service providers to build and create their access-agnostic service scenarios by means of reusable service enablers [2].

To identify the problems related with the deployment of an IMS platform as well as to evaluate the performance of IMS-related components and different IMS/NGN services, there is the need for an IMS testing infrastructure [3]. One of ways to build an IMS testbed is to use open and vendor-independent source code that is available for free.

* This work was developed during an internship of Jawad Hussain at the CTTC.

T. Magedanz et al. (Eds.): TridentCom 2010, LNICST 46, pp. 684–702, 2011.

There have been some publications concerning IMS testbed deployment based on open source tools and technologies. For instance, references [4] and [5] describe the experience of the authors when implementing IMS core components consisting of control functions and a subscriber database. On the other hand, references [6] and [7] deployed prototypes for interoperability testing and critical performance evaluation of a presence service, respectively. Additionally, reference [8] focuses on the implementation of a location service enabler on top of an open source IMS core. Thus, most of the above papers are focused on studying a certain IMS enabler (e.g., presence, XDMS, or location) or on studying the IMS core.

In this paper, we present an open IMS testbed deployment in the context of the implementation of a reference service scenario that integrates multiple service enablers. The main contributions of the paper may be summarized as follows:

- We develop a framework for reference service scenario implementation by previously decomposing the whole implementation process into four steps (planes). In the first plane, we describe the service scenario itself. In the second plane, we define the service enablers that the implementation of such a scenario requires. In the third plane, we consider the functional entities that these service enablers involve. In the fourth plane, we map all these functionalities into some IMS physical entities.
- We deploy these IMS entities using different open source packages and we discuss about our practical experience when development the IMS testbed.
- We carry out interoperability tests between the IMS entities and a client in our testbed. We also evaluate the performance of the IMS procedures of the tested service scenario by characterizing their delay and signaling overhead.
- We integrate the IMS prototype with the EXTREME Testbed® [32] to validate it in close-to-real environment. In particular, we test IMS procedures through UMTS/HSDPA and WLAN access technologies.

The rest of the paper is organized as follows. In Section II, a brief description of the IMS concept from the viewpoint of service provisioning principles is given. In Section III, an approach to an IMS service scenario implementation based ondecomposition is considered. In Section IV, the testbed architecture to support this service scenario is presented and practical issues concerning the testbed deployment are discussed. Section V focuses on test cases with regard to interoperability issues. Section VI presents the preliminary results of the measured performance metrics. In Section VII, aspects concerning integration of IMS testbed with in-lab UMTS/HSDPA and WLAN networks are considered and delay that these technologies bring in IMS procedures is discussed. Section VIII concludes the paper.

2 Main Principles of the IMS Concept

The IP multimedia subsystem (IMS) concept has been developed for the provision of IP multimedia services by means of IP multimedia sessions to users. While the IETF has standardized Internet protocols (e.g., SIP, Diameter, etc.), the 3GPP has defined the IP Multimedia Overlay platform (the IMS specification began in 3GPP Rel'5 in 2002) over different underlying technologies (e.g., GPRS/UMTS,

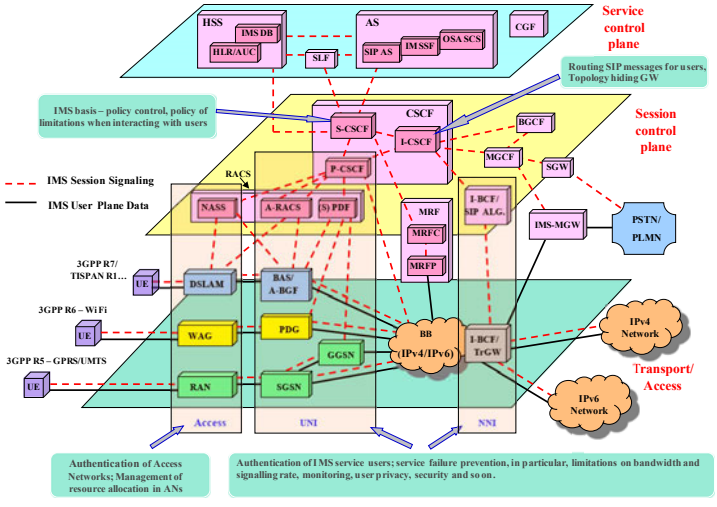


Fig. 1. IMS functional architecture overview

WLAN, DSL) for session control based on Internet protocols as well as the procedures to support generalized mobility across these technologies. IMS also inherits the traditional telecommunications experience concerning guaranteed QoS, flexible charging mechanisms, etc. [9]. An overview of the IMS functional architecture based on 3GPP, ETSI TISPAN, and ITU specifications is presented in Figure 1.

By analyzing Figure 1, one may observe that IMS integrates and develops some ideas from the Intelligent Network (IN) concept. For instance, in the IN concept, the logic of the service (e.g., a toll free or 800 number) was separated from the core switching system (TDM switches) by means of an external node called the service control point (SCP). There is also a triggering point, called the service switching point (SSP), that was added to TDM switches to forward a call related to an IN service towards the SCP by means of the Intelligent Network Application Part (INAP) protocol. The INAP protocol allows the SCP to control and monitor the switch [1]. Since services are no longer developed in the TDM switch, service providers enable developing different value-added services (VAS) for their networks without submitting a request to the core switch manufacturers and waiting for the long development process [10].

In the same manner, in IMS, service-related functions are independent of the underlying transport-related technologies [11]. Roughly speaking, the Call State Control Functions (CSCF) of IMS correspond to the functionality of the TDM switch/SSP in the IN concept, and a SIP Application Server (AS) in IMS corresponds to SCP in IN. Both concepts have triggering criteria to invoke a service, but the respective triggering mechanisms are completely different.

Besides, the IN offered an idea of service independent building blocks (SIBs) for reusable service functions. In accordance with this idea, a service is built as a composition of various SIBs.

In the same manner, in accordance with the IMS concept, a service itself is not standardized, but service building blocks that are reusable by various services. These

building blocks are called service capabilities by 3GPP, service support capabilities by ITU-T, and service enablers by OMA [1].

The objectives of IN were not fully reached because of the lack of independence from the INAP protocol, the lack of software reusability, and the lack of openness by manufacturers and operators [1]. The approaches developed within the IN concept cannot be directly applied to IMS because of the service, signaling, and architectural differences between IN and IMS.

However, some principles of the IN concept can be useful in the context of IMS service provisioning. In particular, in IN there is a conceptual model defined in ITU-T recommendations (Q.1211, Q1213-Q.1215, Q.1218, Q.1219). In accordance with these recommendations, a service implementation process includes several planes. In the first plane, the service and its features are described; the second plane is the global functional plane, where SIBs and the global service logic are defined; the distributed functional plane defines functional entities that are involved in the implementation of a service and the relationship between them; the last plane deals with physical entities and protocols.

It is worthwhile to consider in a similar way the issues concerning the implementation of IMS services. It will help to provide a detailed view of service modeling in IMS. One such approach in the context of the deployment of an IMS reference service scenario is introduced in the next section. In this case, and based on decomposition, the whole implementation process is divided into some steps (planes).

3 Decomposition of the IMS Service Scenario Implementation Process

Similar to the IN conceptual model, our approach to the deployment of an IMS reference service scenario includes four planes, namely, service description plane, service enablers plane, functional plane, and implementation plane. The proposed decomposition into planes is illustrated in Figure 2. Each of these planes in the context of a service scenario is described below.

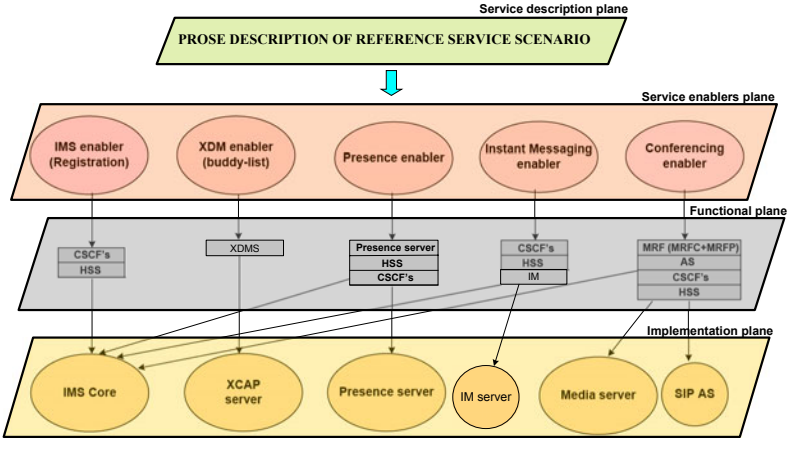


Fig. 2. Decomposition of the IMS service scenario implementation process

3.1 Service Description Plane

The service description plane (see Figure 2) deals with the description of a service scenario to be realized by means of the IMS platform. Any network issues needed for the service implementation are not considered in this plane.

As a case study we consider an abstract simple service scenario that describes a real life situation. Several similar scenarios can be developed and implemented by means of IMS to satisfy the different needs of subscribers.

The prose description of the reference service scenario used throughout this paper follows. There are three close friends from childhood time: Mark, Bob, and Nicholas living in one neighborhood. Now they are businessmen working at different places during the day. They are quite busy, but when all of them are free, they like to occasionally go to a night party together by previously discussing where it is better to go. Thus, they are interested in being subscribed to a service that from a user (a "friend") perspective may be described as follows. Mark creates his contact list by adding the contacts of Bob and Nicholas in it. He is able to see in the contact list whether his friends are available and willing to communicate or not. Mark also allows spreading information about his availability to Bob and Nicholas. When all his friends have as status "available for a party", Mark sends them in parallel a short message like "we have a conference call in 10 minutes to discuss where to go". In 10 minutes, Mark initiates a conference (e.g., audio) with his friends.

This service scenario includes some procedures that must be supported to eventually deploy this service scenario. The procedures and service enablers that are needed to implement it are considered in the next plane, called service enablers plane.

3.2 Service Enablers Plane

Since it is assumed that the scenario is deployed by means of IMS, there is the need for a registration procedure in the IMS platform. Additionally, the following procedures must also be supported: the procedure to maintain a contact list, the procedure to handle information to be aware of contact availability and willingness to communicate (presence information), the procedure to send short messages, and the procedure to arrange a conference call between multiple participants. Note that some other procedures (e.g., those concerning policy and charging issues) should also be involved to support this service scenario, but they are out of the scope of the paper.

Thus, the service scenario requires some service enablers that perform the above procedures. In particular, the user registration procedure deals with mutual authentication between the user equipment (UE) and the IMS platform by using the 3GPP Authentication and Key Agreement (3GPP-AKA) mechanism. This process is carried out by means of the IMS core infrastructure (IMS enabler) including control functions and the subscriber database.

The procedure to maintain a contact list (buddy-list) including friends, family members, colleagues, or other groups with whom an individual may want to communicate is carried out by the XDM (XML document management) enabler. This enabler is in charge of making user-specific service-related data available to other services and service enablers within the IMS network [12]. The buddy-list is the type of data that is mainly associated with the XDM enabler.

The procedure to handle presence information is supported by the Presence enabler, which spreads user-related presence updates (i.e., current status of a user) throughout the network to other users who register to receive notification of changes of a user's presence and availability state [13].

The procedure to send short messages is fulfilled by means of the Instant Messaging (IM) enabler, which allows engaging two or more users in a real-time text messaging.

Finally, the procedure to arrange a conference call between two or more users is carried out by the Conferencing enabler, which supports communication between multiple participants. It allows a user to initiate, modify, and terminate media sessions. Conferencing applies to any kind of media stream by which users may want to communicate [14].

The above enablers belong to the plane of service enablers shown in Figure 2.

3.3 Functional Plane

Each service enabler is characterized by a set of functional entities (FEs) and relationships between them, as specified in the IMS standards. These functional relationships between FEs are usually illustrated by information flow diagrams [15].

The IMS enabler involved in carrying out the IMS registration procedure includes Call State Control Functions (CSCFs), namely, Proxy-CSCF (P-CSCF), Interrogating-CSCF (I-CSCF), Serving-CSCF (S-CSCF), and Home Subscriber Server (HSS). The CSCFs are responsible for routing signaling and managing sessions [33]. The HSS is a database including user identities (both public and private) and service-related information. It is responsible for Authentication, Authorization and Accounting (AAA) [33].

The Presence enabler deals with the widespread publication and subscription of presence information [15]. A user can subscribe to presence information for his/her contacts. If the contact accepts his request, the subscriber (watcher) will be registered for presence notification. Whenever a friend (presentity) publishes presence information, the IMS presence framework will notify the subscribed users. This enabler involves the IMS core functional entities (CSCFs) and the Presence server, which manages presence information uploaded by a presentity UE and handles presence subscription requests.

The XDM enabler deals with documents that are stored in (logical) repositories in the network, generically referred to as XML Document Management Servers (XDMS). Each repository is associated with a functional entity that uses the data in its associated repository to fulfill its functions [17]. Thus, the XDMS has a service- independent functionality that can be used in a variety of person-to-person or group communications [17].

The Conferencing enabler involves the CSCFs, the HSS, the Multimedia Resource Function (MRF) (consisting of Media Resource Function Controller (MRFC) and Media Resource Function Processor (MRFP)), and the Application Server (AS). The MRF provides media related functions, such as media manipulation (e.g., voice stream mixing) and playing of tones and announcements [16]. A user initiates a conference by means of the MRFC/AS entity of the user's home network that assigns a conference URI (Uniform Resource Indicator) to the conference and configures the

MRFP. The conference call is established and the RTP data begin flowing between the UE initiating the conference and the MRFP [15]. The conference initiator then uses the refer procedure to add more users to the conference. The new users establish a call to the conference URI included in the refer message. When the conference is in progress, RTP media streams are mixed and propagated to all participants [15].

The Instant Messaging (IM) enabler is the well-known instant messaging paradigm adopted in the IMS framework [18]. It involves the CSCFs, the HSS, and the IM server. An instant message (IM) is transferred by using a SIP MESSAGE, which is a SIP extension defined in IETF RFC 3428 [14]. Messages are directly sent out to the destinations through the IM server. Upon receipt of a MESSAGE request, a SIP UA (user agent) will reply with a 200 OK or a 202 Accepted response, which indicates that the SIP UA has received the SIP request message [14].

By focusing on the functional plane represented in Figure 2, one may notice that the CSCFs and the HSS functional entities are common to several enablers. Note that the "servers" (i.e., Presence, XDM, IM, and AS) in context of the plane are functional entities.

3.4 Implementation Plane

In this plane, all functional entities defined in the previous plane are mapped into the corresponding physical entities that have to be deployed in the IMS prototype. In fact, one or more functional entities may be mapped into the same physical entity. In particular, P-CSCF and I-CSCF are SIP proxy servers [7]. S-CSCF plays the role of SIP proxy and SIP registrar [7]. All these components together with the common database (HSS) are mapped into one physical block called the IMS core.

The presence functionality defined by the Presence enabler requires a SIP Presence server. The instant messaging functionality is supported by means of the SIP IM server. The conferencing enabler functionality requires a SIP media server to support the media resource function and a SIP AS to support the end-user service logic. An XCAP server is needed to provide the ability to query, modify, or delete data stored in XML.

Thus, all these physical entities (the IMS core, the Presence server, the XCAP server, the IM server, the Media Server, the SIP AS) must be deployed to support the above service scenario (see the implementation plane in Figure 2). Practical issues concerning the deployment of the testbed, including the above entities, are considered in next session.

4 Testbed Architecture and Deployment

The architecture of our IMS prototype to implement the reference service scenario is presented in Figure 3. In accordance with the service scenario, all friends live in one neighborhood and it is assumed that they belong to the same IMS network. Thus, we initially develop the testbed for intra-domain test cases. For other service scenarios, the testbed can be extended to inter-domain test cases.

We have analyzed available open source packages to build this testbed since they are free, they are not identified with a particular vendor, and they supply codes. We have installed and configured the above-defined IMS entities slightly modifying codes to support interoperability between them.

For the testbed deployment, we use a Linux desktop machine, Genuine Intel Pentium (R) Dual–Core CPU 2.50 GHZ with 4GB RAM, and Ubuntu 9.04 OS.

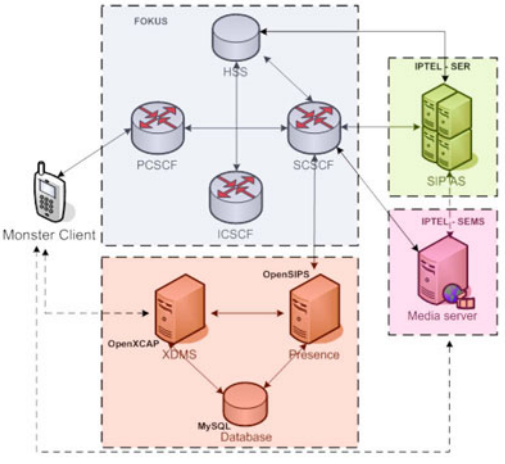


Fig. 3. The IMS testbed architecture

We have started with the development of the IMS Core. The four components (P/S/I-CSCFs and a lightweight HSS) are deployed by using open source code developed by FOKUS [19] as extensions to the SIP Express Router (SER) [26]. It is available through svn repositories at [19]. The lightweight HSS supports the diameter protocol and uses the Cx interface for diameter signaling with the S/I-CSCF [19]. User data is kept inside a MySQL database. We installed this open source (revision 778) and we found that it presents a very efficient and reliable implementation from the viewpoint of operations, administration, and maintenance (OA&M). All IMS core components run on one IP (on loopback), but using different ports.

The simplified flow sequence diagram captured in the testbed for the UE registration procedure is presented in Figure 4.

Our Presence server implementation is based on Open SIP Server (OpenSIPS) [20], which is a mature open source implementation of a SIP server. OpenSIPS (previously called OpenSER) is more than a SIP proxy/router, as it includes many application functionalities. It unifies voice, video, IM, and presence services. We have tested OpenSIPS as a Presence server and we have integrated it in our IMS core. For this purpose, we used OpenSIPS v.1.5, which is the last available version in svn repositories. The presence service based on the open source of this version demonstrates good interaction with the implemented IMS core entity in our testbed. It handles SIP SUBSCRIBE, PUBLISH, and NOTIFY methods. Presence data from subscribers and publishers are saved in a MySQL OpenSIPS database (see Figure 3) in

the corresponding presentity and watcher tables. The simplified sequence diagrams for IMS presence subscription/notification and publication procedures based on the collected traces are shown in Figures 5 and 6, respectively.

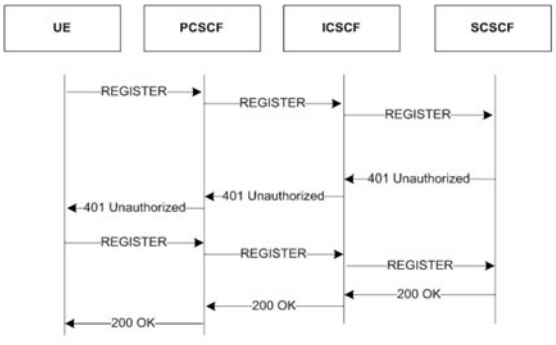


Fig. 4. Testbed registration procedure

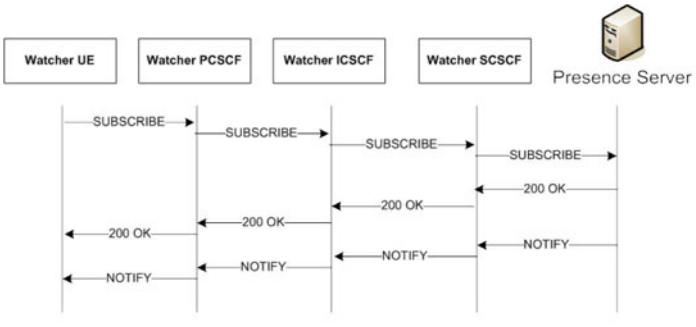


Fig. 5. Testbed presence subscription/notification sequence diagram

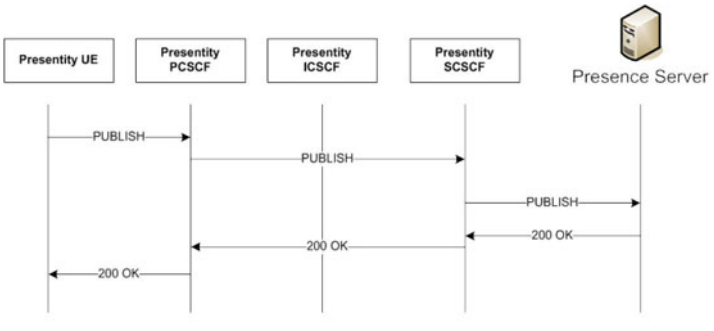


Fig. 6. Testbed presence publication sequence diagram

For the XCAP server implementation, the OpenXCAP source (v.1.1.2) has been used [21]. It is an open source package supporting a full-featured XCAP server. An XCAP server is used by SIP SIMPLE clients and servers to manage buddy-lists and policy for subscriptions to the presence service and it can support other types of published events by using SIP.

We have tested the interaction between the UE and the OpenXCAP server to manage buddy-list by means of the XCAP protocol (see Figure 7). Moreover, we made some configuration changes in OpenSIPS to support intercommunication between the Presence server and the OpenXCAP server. The OpenXCAP server uses the XML-RPC interface to interwork with the presence server, as it is illustrated in Figure 7. For this purpose, the xmlrpc_url port must match the OpenSIPS mi_xmlrpc port. The OpenXCAP backend is used for storage and authentication purposes, and data may be saved either in its own database or in the OpenSIPS database. We configured OpenXCAP to use the OpenSIPS database for service subscribers and XCAP resources.

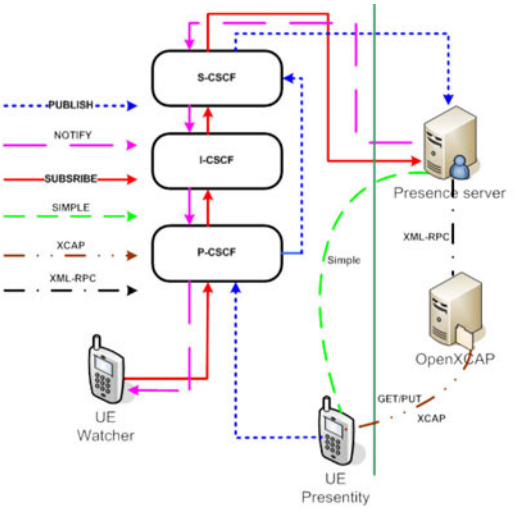


Fig. 7. Interaction diagram between the UE, Presence server, and XCAP server

As media server, we used SIP express media server (SEMS) v.1.1.1 developed by IPTEL [22]. It provides audio centric MRF core functionalities. We have tested its basic applications, like announcement and conferencing. SEMS runs on loopback, having the open-ims.test domain name. We used the SEMS implementation with unixctrl interface and configured it to handle the announcement and conferencing module support.

To support end user service logic, we deployed a SIP AS. There are many SIP application servers available, like Sailfin [23] or Mobicents [24] based on SIP servlet technology. However, we preferred to deploy a SIP AS based on the SIP express router (SER) [25], since the IMS core is also based on SER. In our IMS prototype, the server is located between the IMS core and SEMS entities. SER is a high-performance and configurable SIP server that can act as SIP registrar, proxy, or

redirect server [25]. We used the SER 0.96 version [26] available at svn repositories that is able to interact with SEMS. We used SER in re-direct server mode to forward SIP INVITE, ACK, BYE, and CANCEL methods to SEMS. We configured SER on loopback and it has the same domain name (open-ims.test). SER writes the received SIP messages and forwards them to SEMS by using the unix socket server. The testbed conference call initiation diagram is shown in Figure 8.

Our IMS user agent is based on an open environment developed by FOKUS and called MONSTER (The Multimedia Open InterNet Services and Telecommunication EnviRonment) [27]. We used the most recent release, i.e., version 0.9.8. It consists of a suite of integrated applications, such as voice, video, photo sharing, contacts, etc. [27]. However, the MONSTER public version has limited functional features that can be extended by the use of the MONSTER API. In particular, we observed that it does not support video conference.

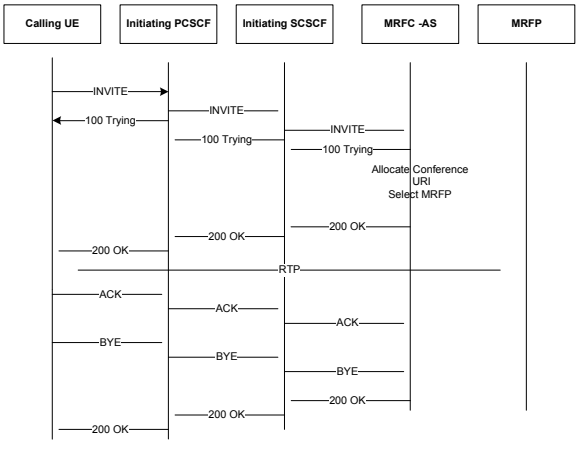


Fig. 8. Testbed conference call initiation sequence diagram

Besides, we also tried to deploy an Instant Messaging (IM) server. OpenSIPS used before to deploy the presence server can be also configured to support end-to-end IM. We installed and integrated an external e-jabbered IM server v.2.1.0 [28] by using the XMPP OpenSIPS module. However, later, we found that interoperability between the UE and the IMS server is not supported, since the current public MONSTER version does not provide IM functionality, though it can be developed and added by means of the MONSTER API. At the same time, the MONSTER client allows supporting the SIP MESSAGE method, and in the service scenario this feature can be used by friends instead of instant messaging. For this reason, currently, we do not use the IM server in our testbed.

Thus, we have deployed the following IMS components in our IMS testbed: the IMS core, the Presence server, the XCAP server, the Media server, the SIP AS, and the UE. The results of some relevant interoperability tests between the components for our reference scenario are considered in the next section.

5 Interoperability Tests

To check interworking between the UE and the Presence, XCAP, and Media servers, interoperability tests have been carried out. Table 1 presents all test cases run. Test cases 1-5 are related to the Presence server, test cases 6-8 are related to the XCAP server, and test cases 9-10 are related to the Media server, respectively.

Table 1. List of interoperability tests carried out over the testbed

TEST CASE #	*TEST TITLE*	*TEST CASE DESCRIPTION*
1	Publication of presence information	Presence information initially published by IMS UE must be correctly received by IMS subscribers that are eligible for receiving it
2	Publish modification	Published information modified by the UE must be correctly received by the presence subscribers
3	Subscription removal	If a watcher removes its presence subscription, a presentity must update presence information
4	Subscription refresh	Presence server must continue to send presence information to a watcher in case presence subscription is refreshed
5	Notification of presence information from multiple presentities	This case is similar to test case 1, but it is extended to multiple presentities
6	Group-list XDMS document creation	UE must be able to successfully create its buddy-list in the XCAP server by using http PUT/DELETE methods
7	Group-list XDMS document retrieval	By means of the http GET method, an IMS user can successfully download its buddy-list from the XCAP
8	Group-list XDMS document validation and deletion	The UE must be able to manipulate its buddy-list
9	Playing media announcements	The Media server must be able to play an audio announcement
10	Supporting conferencing application	The media server must support conferencing with two or more UEs

We have repeated each of the above test cases 15 times. Test results are presented in Table 2.

Table 2. Results of the interoperability tests

# OF TEST CASES	*# OF TRIALS*	*# OF PASSES*	*# OF FAILS*
10	*15*	*134*	*16*

As one can observe in Table 2, 134 out of 150 trials have been successful. All 16 failures have arisen in test cases related to the Presence server, when the UE of a

presentity is being switched off. In this situation, the MONSTER UE is not always able to send a publish message to the Presence server. As a result, watchers can see the status of the presentity as “on-line” instead of “not available”. The same observation has been obtained in [6], when testing the SIMPLE presence protocol implemented by the UCT client [29] and the IMS Communicator [30], which were developed before MONSTER. Both clients display the last presence information published by the UE [6]. Thus, this problem still exists in the public available version of the MONSTER client as well.

We also verified that all testbed sequence diagrams (see Figures 4-6 and Figure 8) are in conformance with the message sequence charts provided by the IMS standards.

6 Preliminary Results

Some preliminary results were obtained in our IMS testbed. In particular, we evaluated two performance metrics that characterize the delay and overhead of the IMS signaling procedures deployed to support the reference service scenario. In particular, the first metric accounts for the time elapsed since the UE sends a signaling message to the IMS server until it receives a response message back from the server according

Table 3. Test cases for evaluating the round trip signaling time of various procedures

TEST CASE #	*IMS PROCEDURE*	*TEST CASE DESCRIPTION*
1	IMS REGISTER	RTST between sending the initial REGISTER request to the IMS core and reception of the 200 OK message (total delay for registration and de-registration procedures)
2	IMS INVITE (IMS to IMS call)	RTST between sending the initial INVITE message to the IMS core and reception of the 200 OK
3	IMS watcher REGISTER/SUBSCRIBE	RTST that a watcher spends sending REGISTER request to the IMS core, SUBSCRIBE-ing to presence information about presentity and receiving NOTIFY message from presence server (PS)
4	IMS presentity REGISTER/PUBLISH	RTST that a presentity spends sending the REGISTER request to the IMS core, PUBLISH-ing to PS and reception of the 200 OK
5	IMS presentity PUBLISH	RTST that a presentity spends when Presence state changes (PUBLISH) for the Watched User and reception of the 200 OK message from the PS
6	IMS INVITE for Media server	RTST between sending the initial INVITE message to Media server and reception of the 200 OK message from the server
7	IMS HTTP PUT for XCAP server	RTST between the UE PUT-ting its buddy-list in the XCAP server and reception of the HTTP 200 OK message from the server
8	IMS HTTP DELETE for XCAP server	RTST between the UE DELETE-ing its buddy-list from the XCAP server and reception of the HTTP 200 OK message from the server

to a certain IMS procedure. We call it the round trip signaling time (RTST) below. The second performance metric is the signaling overhead (which accounts for the application layer payload) generated by SIP messages for a certain IMS procedure. We performed each test case for these metrics ten times and captured the associated traces by using Wireshark [31].

The description of test cases to evaluate the RTST for different IMS procedures is presented in Table 3.

The box-plot presented in Figure 9 shows the results of the signaling time evaluation of the different IMS procedures.

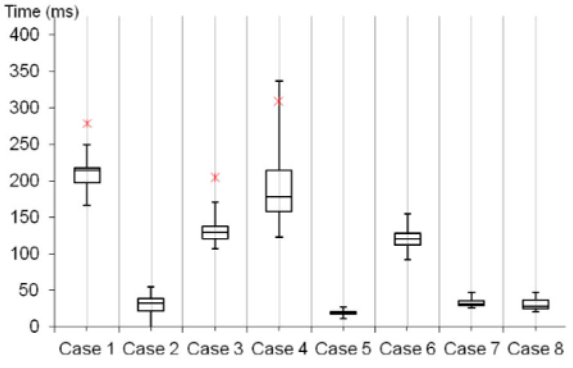


Fig. 9. RTST evaluation of different IMS procedures

The results of the signaling overhead generated by SIP messages in the above IMS procedures are presented in Table 4.

Table 4. Overhead (application layer payload) associated to each of the deployed procedures

TEST CASE #	*IMS PROCEDURE*	*AV. NUMBER OF BYTES*	*ST. DEVIATION*
1	IMS REGISTER	21.320	89.77
2	IMS INVITE (IMS to IMS call)	19487	12.98
3	IMS watcher REGISTER/SUBSCRIBE	23141	79.96
4	IMS presentity REGISTER/PUBLISH	15990	82.41
5	IMS presentity PUBLISH	4724	20.69
6	IMS INVITE for Media server	13130	5.61

The analysis of the results presented in Figure 9 (RTST) and Table 4 (overhead) brings the following observations. The total time that both registration and de-registration procedures take is quite high (the first test case). SIP messages between CSCFs are transmitted and processed in a small amount of time. On the other hand,

the main component of the delay comes from the communication with the HSS when it is needed to retrieve information (e.g., when the S-CSCF downloads the authentication vector) from the MySQL database that stores user-related data. This observation is further confirmed by the second test case, in which just CSCFs are involved in routing and processing all signaling messages to establish an IMS to IMS call. As it may be observed, this procedure takes much less time, although it generates almost the same signaling overhead.

Test cases 3 and 4 do not contain the de-registration procedure, but the subscription and publication procedures initiated after the registration are considered, respectively. The procedures are also time-consuming. Especially, average RTST and dispersion are significant for the publication procedure. This is because of the authorization process of the watcher (subscriber) or publisher (presence entity) with the Presence server, which requests user presence data stored in the MySQL openSIPS database. Test case 5 further confirms this observation, as it considers a case for which presence state changes, hence sending PUBLISH messages to the Presence server, but without requiring the publisher to run an authorization process. As a result, the delay in this case is very low compared to the previous one. Note that in test case 3, more signaling overhead is generated than in test case 4 due to the additional payload bytes devoted to the NOTIFY message (see Figure 5 and 6).

The delay in test case 6 is caused by the interaction with the media server to assign a conference URI to the conference, to determine media capabilities, etc. The INVITE procedure for media server (test case 6) contains less message exchanges between IMS entities than the INVITE procedure for IMS to IMS call establishment (test case 2). As a result, the first one generates less overhead, as seen in Table 4.

Additionally, the RTST values obtained for test cases 7 and 8 (PUT/DELETE procedures of the HTTP protocol) show that the average time required to put/delete a buddy-list in/from the XCAP server is approximately of 25 ms, hence much lower than other more complex procedures.

Finally, if we consider the obtained results for our reference service scenario then from a user (a “friend”) perspective it generates an average of 40 Kbytes of signaling overhead and lasts for 280 ms, in our IMS prototype. The sequence of actions included in this calculation are: Mark’s *registration* to IMS core, *subscription* to the presence service to get *notifications* on the presence status of his friends, *publication* of his presence information so that it is available to his friends, and *initiation* of the conference call with them.

7 Integration of the IMS Prototype with the EXTREME Testbed®

To better validate IMS operation in close-to-real environment, we have started the integration of the IMS prototype with the EXTREME Testbed® [32]. In particular, we connected the UE to the deployed IMS platform through two transport technologies, namely UMTS/HSDPA and WLAN IEEE 802.11, as illustrated in Figure 10.

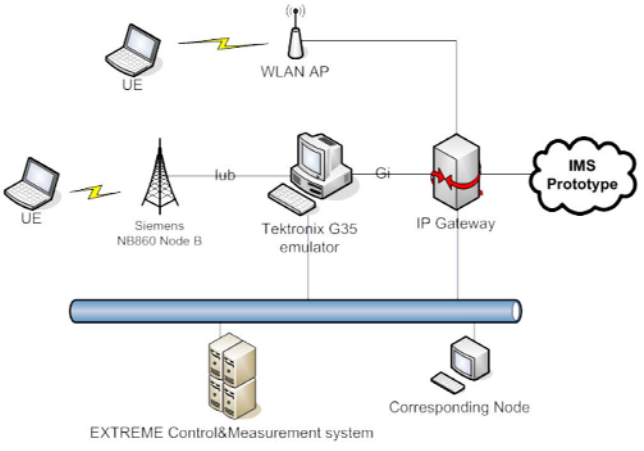


Fig. 10. Integration of the IMS prototype with EXTREME testbed ®

The in-lab real-time UMTS network consists of a real Node B (Siemens NB8860) with WCDMA and HSDPA functionalities, a protocol emulator (Tektronix K1297-G35) of the RNC and the PS core network (including SGSN, GGSN, and a subset of the functionalities of the HLR), and a PC acting as IP gateway to external networks [33]. An ATM optical interface connects the Node B to the G35 emulator (featuring a real Iub interface). The G35 communicates to the EXTREME platform via a Gigabit Ethernet interface (through the Gi interface) allowing interworking between UEs and nodes in the Internet.

The WLAN access point (AP) is based on IEEE 802.11g, configured in "Infrastructure BSS" mode, and connected to the IP Gateway. UMTS/HSDPA and WLAN wireless cards are used to connect to the UMTS and WLAN networks, respectively.

We integrated our IMS prototype through the IP gateway with the UMTS and WLAN networks. To check conformance, the IMS registration/de-registration procedures have been run (10 times) through both transport technologies. The results of the registration/de-registration delay for these procedures obtained in the IMS EXTREME testbed are presented in Table 5. It is interesting to compare them with the values of delay for the same procedures with the same transport technologies measured in the "IMS experience centre" [3] for a real-life test network using real UMTS/HSDPA equipment. The results got in the IMS experience centre testbed are also presented in Table 5.

Table 5. Comparison of IMS registration/de-registration delay in different testbeds

Testbed name	Network	Registration delay, ms (min/max/avg)	De-registration delay, ms (min/max/avg)
IMS EXTREME tes	WLAN (802.11g)	100/137/120	80/114/99
	UMTS/HSDPA	288/332/309	242/290/263
IMS experience centre testbed	WLAN (802.11g)	9/89/22	12/28/16
	UMTS/HSDPA	460/840/520	470/750/520

By analyzing the results obtained in the IMS EXTREME testbed we have observed that registration and de-registration delay over the WLAN network is composed mainly of the above-mentioned delay when retrieving data from the MySQL database (twice for each procedure, each time taking around 40-60 ms). The total registration and de-registration delay over WLAN almost coincides (difference is a few ms) with values obtained for both procedures in the IMS testbed (see test case 1 in Figure 9). Thus, the delay that the WLAN network itself brings is negligible.

On the other hand, the delay that the UMTS/HSDPA network brings is higher. It is almost three times more than in the IMS testbed (~570 ms vs. 210 ms). The results got in [33] in the UMTS/HSDPA EXTREME infrastructure observed an average downlink delay of 40 ms and uplink delay of 60 ms. For two message exchanges in both directions (see Figure 4) the delay takes around 200 ms plus the delay (ranging from 80 to 120 ms) caused by twice as much requests to the MySQL database. Thus, the total delay is around 300 ms for the registration and around 260 ms for de- registration, which is coherent with the results that we obtained (see Table 5).

By analyzing the results obtained in the IMS experience centre testbed one may observe that the delay measured in that case through a WLAN network is very small compared with our results. A potential hypothesis is that they use caching to minimize the delay caused by the MySQL database, since maximal registration delay is 89 ms (probably when retrieving data from the MySQL) which is close to our values, but the average registration/de-registration delay is 22 ms and 16 ms respectively (caching might be used).

On the other hand, the delay experienced through their UMTS/HSDPA network substantially exceeds that obtained in our testbed. A potential explanation may come from the differences in delay (around 90/100 ms downlink/uplink) observed in [33] between a commercial UMTS/HSDPA network and the UMTS/HSDPA EXTREME infrastructure, in which the UMTS core network (SGSN/GGSN) is emulated by means of the G35 test equipment. Given that some IMS procedures require messages to traverse this UMTS/HSDPA infrastructure multiple times, the difference in terms of total signaling delay is substantial increased.

Conclusions

A framework for deployment of an IMS prototype to implement a reference service scenario that involves multiple service enablers has been considered in the paper. The whole service implementation process has been decomposed into four planes, namely, service description, service enablers, functional, and implementation. As a result of the decomposition, we found that for the service scenario, one must deploy the IMS core, the presence server, the XCAP server, the Media Server, and the SIP application server. We installed various available open source packages featuring these IMS entities and we modified and configured them so that they can interact.

To evaluate the interworking between the IMS client [27] and the deployed IMS components, interoperability tests have been conducted. The UE was not always able to send a publish message to the Presence server when the equipment was being switched off. All the rest tests were successful. Besides, we found that the current public version of the client does not support instant messaging and video conference.

We verified that the IMS procedures (registration, subscription, etc.) in our testbed are in conformance with the message sequence charts provided by the IMS standards.

Then, preliminary results concerning the signaling overhead and delay of the different IMS procedures of the service scenario have been obtained. The reference service scenario generates an average of 40 Kbytes of signaling overhead and lasts for 280 ms. We observed that the main delay is caused by retrieving data from the MySQL database. We suppose that caching may reduce essentially the delay. To get access to the deployed IMS platform through in-lab UMTS/HSDPA and WLAN access technologies we integrated our prototype with the EXTREME Testbed® [32]. We observed that the additional delay caused by the WLAN network in IMS registration/de-registration procedures is very low. On the other hand, the delay that UMTS/HSDPA network infrastructure brings in the IMS procedures is essential. It may be several times more then the delay in the IMS platform itself for the procedures.

In future work, we are going to extend our IMS prototype by adding more functionality in the deployed framework as well as in the IMS client to support more complex service scenarios over heterogeneous access technologies. Besides, we are planning to conduct a more exhaustive performance evaluation of different IMS components by using a signaling traffic load generator.

Acknowledgement

This work was supported in part by the Spanish Ministry of Science and Innovation under grant number TEC2008-06826 (ARTICO), by the Catalan Regional Government under grant 2009SGR-940, and by the Spanish Ministry of Industry, Tourism and Commerce under grant number TSI-020301-2008-13 (WIMSAT).

References

[1] Bertin, E., Crespi, N.: IMS Service, Models, and Concepts. In: IP Multimedia Subsystem (IMS) Handbook, ch. 1. CRC Press, Boca Raton (2009)

[2] Sardella, A.: Building IMS-Capable Core Network. White paper, Juniper network (March 2006)

[3] Balakrishna, C.: IMS Experience centre. A real-life Test Network for IMS services. In: Proc. 5th Int. Conference on Testbeds and Research Infrastructures for the Development of Networks & Communities (2009)

[4] Anwar, B.P., Singh, K.: IMS SIP core server test bed. In: Proc. Int. Conference on IP Multimedia Subsystem Architecture and Applications (2007)

[5] Tang, J., Davids, C., Cheng, Y.: A study of an open source IP Multimedia Subsystem test bed. In: Proc. Qshine 2008, Hong Kong, China (July 2008)

[6] Maarabani, M., et al.: Interoperability testing of presence service on IMS platform. In: Proc. 5th Int. Conference on Testbeds and Research Infrastructures for the Development of Networks & Communities (2009)

[7] Lin, L., Liotta, A.: A critical evaluation of the IMS presence service. In: Proc. MoMM 2006 (2006)

[8] Reichl, P., et al.: Practical experiences with an IMS-aware location service enabler on top of an experimental open source IMS core implementation. Journal of Mobile Multimedia 2(3), 189–224 (2006)

[9] Magedanz, T.: IP Multimedia System (IMS)-Principles, Architecture and Applications. In: 2nd IEEE Workshop on Mobility Aware Technologies and Application, Montreal, Canada (October 2006)
[10] http://en.wikipedia.org/wiki/Intelligent_network
[11] ITU-T. Y.2001, Next Generation Networks – Frameworks and functional architecture models (December 2004)
[12] Al-Begain, K., et al.: IMS: A Development and Deployment Perspective. Wiley, Chichester (September 2009)
[13] 3GPP TS 23.141, Presence service; Architecture and functional description; Stage 2
[14] Ahson, S.A., Ilyas, M.: IP Multimedia Subsystem (IMS) Handbook. CRC Press, Boca Raton (2009)
[15] http://www.eventhelix.com/ims/
[16] http://en.wikipedia.org/wiki/IP_Multimedia_Subsystem
[17] http://www.openmobilealliance.org/Technical/release_program/xdm_v1_0_1.aspx
[18] Mikka, P., Georg, M., Hisham, K.: The IMS IP Multimedia Concepts and Services. John Wiley & Sons, Chichester (2006)
[19] Open IMS Core's Homepage, http://www.openimscore.org/
[20] OpenSIPS project, http://www.opensips.org/
[21] OpenXCAP, http://openxcap.org/
[22] SIP Express Media Server, http://www.iptel.org/sems
[23] Project Sailfin, https://sailfin.dev.java.net/
[24] Mobicents, http://www.mobicents.org/
[25] SIP Express Router, http://www.iptel.org/ser/
[26] SER and SEMS work together, http://ftp.iptel.org/pub/sems/doc/current/Configure-Sems-Ser-HOWTO.html
[27] MONSTER – the client, http://www.monster-the-client.org/
[28] Ejabberd, http://www.ejabberd.im/
[29] http://uctimsclient.berlios.de/
[30] http://developer.berlios.de/projects/imscommunicator/
[31] http://www.wireshark.org/
[32] http://www.cttc.es/en/project/EXTREME.jsp
[33] Dini, P., et al.: A real-time cellular system architecture to experiment with UMTS/HSDPA in a laboratory. In: Proc. of TRIDENTCOM 2009, Washington, USA (April 2009)

ONIT Workshop 2010

Session 2: Architectures and Services Convergence

Optimization of Network Redundant Technologies Collaboration in IMS Carrier Topology

Filip Burda, Peter Havrila, Marián Kněžek, Klaudia Konôpková, Ivan Kotuliak, Ján Murányi, and Juraj Nemeček

Slovak University of Technology, Faculty of Informatics and Information Technologies, Ilkovičova 3, 842 16 Bratislava, Slovakia
{filip.burda,phavrila,marian.knezek,klaudia.konopkova, jan.muranyi,juraj.nemecek}@gmail.com, Ivan.Kotuliak@stuba.sk

Abstract. The Traffic Engineering in IMS network is a hot topic as operators require extensive QoS management in their network. A combination of IP SLA with the Object Tracking and MPLS Traffic Engineering can create automatic solution for applying new rules to the ISP carrier topology. IP SLA provides an opportunity to track specified parameters of the links and devices. In this article, we optimize convergence and load distribution among existing links in the network in automated way. Nowadays, similar solutions work mainly manually. Innovative solution, which finds suboptimal bandwidth utilization automatically, without requirement of the network administrator involvement, is described also.

Keywords: MPLS, Traffic Engineering, Object Tracking, Redundancy.

1 Introduction

IMS (IP Multimedia Subsystem) architecture is creating momentum in the research of telecommunication technologies and data networks. As these two previously separate worlds are fusing into the one converged environment, there are more than enough issues that operators would like to resolve for smooth incorporation of the IMS into their core networks. In our research, we have focused on a fundamental operation of the underlying data routing around the IMS core. In the last generations of telco networks, the quality of service and load-balancing could be native to the whole network. In IP networks, such things are hardly native as data networks and particularly the IP networks are routed in the shortest path first manner. This approach creates limitations on the ability of these networks to utilize the bandwidth of routes other than those declared as shortest paths to the destination. This limitation is currently being focus of world-wide research from which a new concept called the Traffic Engineering is rising as an old solution for new environments, particularly MPLS (MultiProtocol Label Switching). Put together, Traffic Engineering is the manipulation of traffic to fit our network [1].

T. Magedanz et al. (Eds.): TridentCom 2010, LNICST 46, pp. 705–712, 2011.

This article focuses on MPLS Traffic Engineering (TE) technology and its usability in the IMS environment. In our approach, we combine Cisco solutions IP SLA with the Object Tracking and MPLS TE to create unique automatic solution for applying new rules to a changing topology (e.g. in cases of link failures or over utilizations). IP SLA gives us an opportunity to track specified parameters of the links and devices. Consequently, these results can be applied to the Object Tracking for creating entries that will be applied to the routing table upon specified event. The aim is to optimize convergence and load distribution among existing links in the network. Our approach increases availability of services, overall quality of services and tries to easily satisfy SLA (Service Level Agreement – in terms of an actual agreement for quality of service) demands between service provider and customers [2].

The article is organized as follows: Section 2 describes the problem in details and provides existing solutions. Section 3 presents our approach of applying IP SLA to the network. Section 4 describes our lab environment. Section 5 contains obtained results. Concluding results and ideas for future are given in Section 6.

2 State of Art

Traffic Engineering is used to solve a fundamental problem as displayed in Fig 1. For this example of the IP network, all links are OC-3 links with the bandwidth roughly 150 Mbit/s. Now, let us assume that we know that the router R1 sends 90 Mbit/s of data to the router R6 and router R7 sends another 80 Mbit/s to router R6. In the classical shortest path first manner, R2 has link to R5 as next hop towards R6. This will simply result in congestion on the link between routers R2 and R5 and obviously, alternative link through the path R2-R3-R4-R6 remains under full utilization. The possibility of using Traffic Engineering is by manipulating costs. This results in costs equilibrating of all alternative paths and then load-balance between these paths. This solution is usable in small networks, but large scale deployment can be problematic. More sophisticated approach is the Load Sharing, which can better reflect the available resources (e.g. bandwidth) along paths. An alternative is Asynchronous Transfer Mode (ATM) networks, where Permanent Virtual Circuits (PVCs) can be constructed between the end-points and load can be shared between these PVCs and no detrimental manipulation to the link costs is necessary.

IP SLA is Cisco specific function for supporting monitoring of specific parameters. IP SLA can monitor different constraints of the node, link or path from the routers and taking appropriate actions and informing administrator through SNMP protocol (Simple Network Management Protocol). IP SLA currently monitors various types of delay, jitter, RTT, number of dropped packets, latency, etc. IP SLA is a tool to satisfy the defined constraints in SLA [3].

Object Tracking is another Cisco specific feature. In Object Tracking we are monitoring an object, which is i.e. IP SLA object, status of an interface, status of IP address, presence of the destination network in the forwarding table, metric of the path, etc. Composite objects can be created, where other objects are put together through boolean logic or threshold system [4].

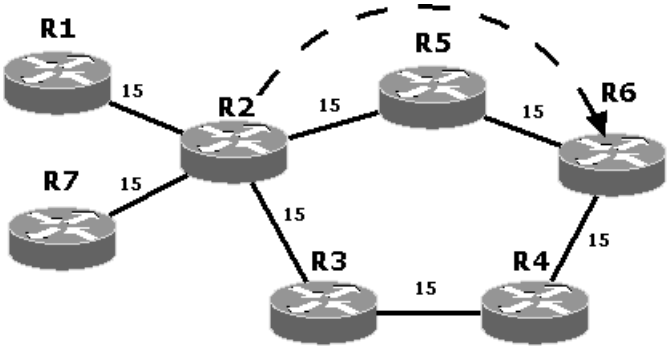


Fig. 1. Example of the IP network with the potential for Traffic Engineering use

Currently, a combination of ATM facilitates management through PVCs and scalability of the IP infrastructure resulted in the MPLS networks as MPLS TE. MPLS enables chaining of the labels in Protocol Data Unit (PDU) and thus the ability of non-bottom labels to have other than routing purposes. The two most common uses for these labels in one PDU are VPN (Virtual Private Networks) and TE tunnels identifications. However, tunnels are still created mostly manually as a part of the network design. Adding new TE tunnels to the network can be accomplished either by the strategic approach by creating full mesh of TE tunnels in parts of the network or by the tactical approach by the monitoring link utilizations and by adding TE tunnels when they are required [5, 6].

There is also an approach based on enabling the premium service classes in the DiffServ over MPLS-enabled network [7]. The advantage of this solution is the implementation as framework. However, DiffServ is mandatory in this case and only chosen parameters are measured. Contrary to DiffServ over MPLS, we would like to track the tens of different parameters [8]. Furthermore, the dependence on the DiffServ is not suitable and we would not like to be limited to some QoS model.

There is also work based on the delivering QoS in the Next Generation Network (NGN) [9]. This paper presents usage of the QoS for NGN end-user applications. Several concepts for allowing the control of QoS levels are discussed. We would like to present the automatic approach without fixed QoS classification and marking.

There is also work based on the modeling and simulating of traffic aggregation over MPLS networks [10]. This work is focused on SIP call setup and SIP operation. We focus not only on SIP signaling, but also on media delivery.

3 Proposed Solution

Basically, monitoring the links and the devices is done by specialized applications. Monitoring is done mostly by the Simple Network Monitoring Protocol (SNMP). There is also possibility of using IP SLA to track some parameters of the links, which are informing the administrator via SNMP. There is a huge variety of parameters, which can be actively monitored from the routes and many ways of how to and

when to inform the administrator. However, time between sending the trap message, receiving and reading it by the administrator and taking the appropriate action is too long. If there is no backup plan created, minutes can pass. Automation of this SNMP based process is part of our future efforts.

Our current solution, described in Fig. 2 combines IP SLA with Object Tracking [7, 8] and Object Tracking with the static routes defined in the routing table. We are focusing on the MPLS TE tunnels. The only prerequisite is that the TE tunnels are defined statically in the routing table. In our solution, monitored parameter with IP SLA is mapped one to one with Object Tracking. We are also creating composite tracked object, which changes its state after several conditions are met in the other tracked objects. This composite object is mapped with the static route. After some critical values are detected by the IP SLA, the tracked objects change their state automatically. So when the composite object will change its state, the static route will change its state as well. When the static route goes down, the other static routes with worse preference will take place or the dynamic routes. This is a unique automatic solution. Backup plan takes place automatically. Problems are that increasing number of the IP SLA objects increase also bandwidth and CPU utilization. Even more, also the backup plan must be prepared as a part of the network design. If needed, SNMP messages can be still sent to the administrator and administrators can manually change the policy if necessary.

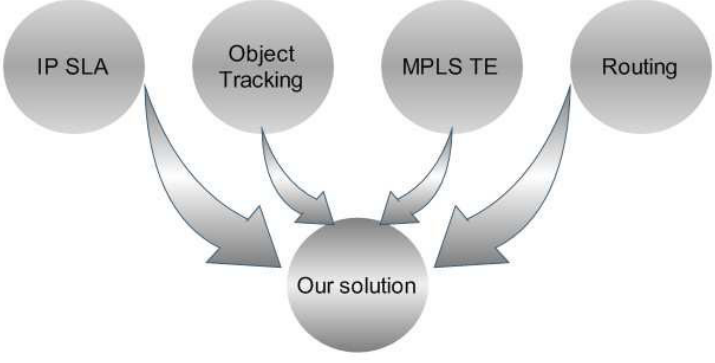


Fig. 2. Proposed approach: combination of IP SLA, Object Tracking, MPLS TE and routing

4 Test-Bed Setup

In Fig. 3, the test-bed of our IMS core with the surrounding redundant carrier network is presented. The carrier network is composed of eight routers with multiple redundancies. We have two exit points in our network to simulate transit ISP network. One exit point is on the far left side, and one on the far right side. IMS core is situated in the network center.

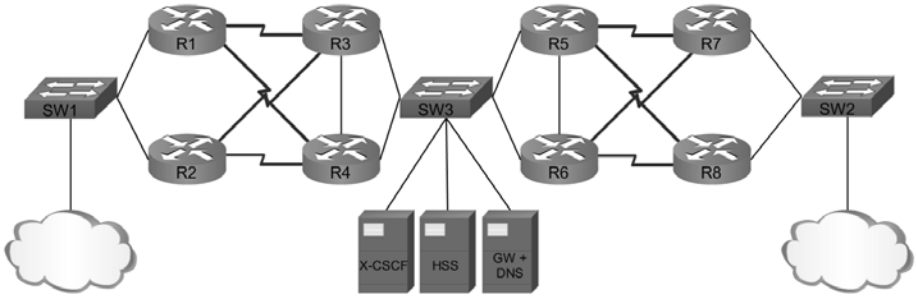


Fig. 3. Testbed scheme

Two MPLS TE tunnels are present on routers R1, R2, R7 and R8. These TE tunnels are created across the whole network, from the one exit point to the other exit point. One TE tunnel on the router R1 is placed across routers R3, R5 and R7 and the second TE tunnel from R1 through R4, R6 to the R8. On the other routers, TE tunnels are placed similarly to this. The first TE tunnel is called primary and the second is called secondary TE tunnel. Primary tunnels are placed among "upper" routers (R1, R3, R5, R7). We assume that on the exit points, the External Border Gateway Protocol (eBGP) will be configured.

The primary TE tunnel is placed in the routing table statically with some preference. The primary and the secondary TE tunnel are learned via dynamic routing protocol, in our case via Integrated Intermediate System to Intermediate System (IS-IS) protocol with worse preference. The static primary TE tunnel is tracked by an object. All the traffic destined for the networks behind the other exit point is going through this static TE tunnel. After some conditions are met, tracked object goes down, which leads to removing this static route from the routing table. After removing this static route, the same dynamically learned primary TE tunnel and also the secondary TE tunnel are placed into forwarding table. The traffic is now load balanced.

For our measurements, the network traffic will enter only the router R1 and is destined to the exit point behind the routers R7 and R8 (c.f. Fig. 3). We assume that customers are already registered and they are initiating voice or video calls. One of the customers is located behind the left exit point and one behind the right exit point. After initiating calls, the bandwidth utilization is rising, which leads to increased Round Trip Time (RTT). Because of 128 kbit/s link between routers, only one call can be established with no quality penalty. After initiating the second call, increased RTT, jitter and also packet loss is observed. Both calls have equally penalized quality.

For our second measurement we will configure IP SLA objects. We have chosen RTT and average jitter for IP SLA tracking. In the IP SLA object 1 we configure the icmp-echo type of packet with the Type Of Service (TOS) decimal value of 184, which is the decimal representation of EF class. Threshold is configured to the 20 ms for our test purpose. Frequency of sending these packets and controlling the quality of the link is 1 second. The second IP SLA object is configured in the same way as the IP SLA object 1. The threshold value is set to 4 ms, again, only for test

purposes. The reaction is configured for an average jitter with the upper threshold of 4 ms and lower threshold of 3 ms. If threshold limits are exceeded, immediate action is taken. Each and every IP SLA object is mapped to its own unique object in Object Tracking (1 and 2). One composite tracked object is created with the boolean logic, designated as object 3. If any object is down, the whole composite object is down. This composite object is used for the static route configuration. Tracked objects 1 and 2 are delayed. If they are not delayed and if one of the IP SLA object fail their test, immediate action is taken. We are delaying the "down" and the "up" state three times the frequency of IP SLA object, which is 3 seconds. If three times in a row the test fails, tracked object is considered to be down. The same rule applies for the "up" state.

5 Performance Results

During measurement, we established the first call. The TE tunnel bandwidth was sufficient for exactly this one call as indicated by acceptable RTT and Jitter characteristics for 128 kbit/s A/S interfaces in Table 1. Next we have established second call. After the second call was established, the RTT measurements started to constantly rise (c.f. Fig. 4). After 4 seconds of both calls in place, call quality deteriorated beyond acceptable threshold (c.f. Table 1). However with our optimization, network was able to detect degrading call quality and dynamically switch traffic patterns in Traffic Engineering manner to accommodate rising demands for network throughput. In optimized environment, after the second call was placed and call characteristics worsened beyond specified threshold, corresponding IP SLA measurement bound to object 1 failed immediately within the next testing period. For the next 3 seconds object 1 was forced to be delayed before changing its state, as a protection against premature backup TE tunnel activation. After this timer expired, object 1 has changed to the down state, also forcing object 3 to go down. This has led to deletion of the corresponding static route in the routing table. The dynamic routes took place immediately, resulting in the creation of the same primary TE tunnel and additional secondary TE tunnel. In this setup, the routers could begin to load balance between these two TE tunnels. In the next half second, given the load balanced environment capable of sustaining two concurrent calls, the quality for the both calls returned to acceptable levels. IP SLA object still hold two TE tunnels up, because with load-balanced solution, jitter characteristics remained above normal in node (c.f. Table 1). After termination of one call, jitter characteristics returned to acceptable values. Therefore IP SLA realized that it is possible to return to a single TE tunnel solution. All the objects changed their state to up and static route was placed back to the routing table. There were no negative effects observed during our experiments.

In Fig. 4, graph of the RTT measurement in time is depicted. Recording of values began when second call was established. It is obvious that our optimization system needed roughly 4 seconds, to detect, propagate, compute, and update routing forwarding information base for rising throughput demands via load sharing TE tunnels.

Table 1. RTP behavior with and without pro-active backup MPLS TE tunnel utilization

	RTT [ms]		Jitter [ms]	
	Without IP SLA	*With IP SLA*	*Without IP SLA*	*With IP SLA*
1 call	51,4	51,4	1	1
2 calls (0-4 s)	~305	~305	1	1
2 calls (after 4 s)	>2000	51,4	4-6	4-6

Consequently, after roughly 4 seconds, our system was able to dynamically achieve sustained acceptable call quality. In comparison to static TE tunnel, that simply became congested. Note that for half a second interval after 4th second, RTT is in a great variation after applying our configuration. During this time, out of sequence packets are arriving, causing these values "jumping" from the upper to the normal RTT level. These packets were discarded by the VoIP clients. Clients were able to communicate after the 4.5 seconds with its expected quality. Without IP SLA, RTT was constantly rising up to the 2000 ms (c.f. Table 1).

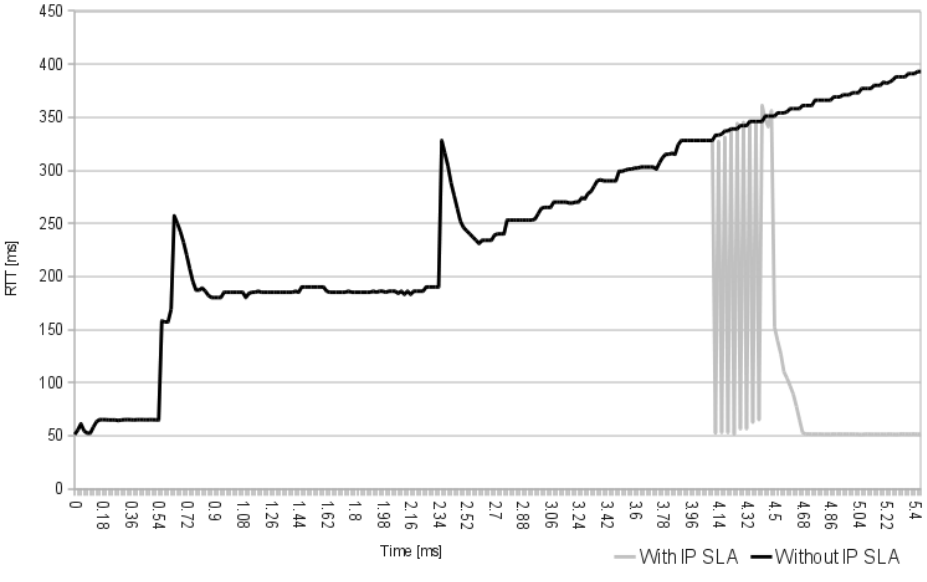


Fig. 4. Comparison of the RTT evolution in time for system with and without IP SLA

6 Conclusion

Nowadays, there are some approaches for optimizing the cooperation of the network IMS core. These approaches work manually, no automatic solution is available. We have proposed innovative solution, which finds suboptimal bandwidth utilization automatically, without requirement of the network administrator involvement. However in this stage, some backup plans are required for immediate response.

We have demonstrated that the IP SLA with the Object Tracking can be effectively combined with the other technologies like MPLS TE. Especially, after link failure or link overload, it can reroute traffic in fully automated way to alternative routes. Even that this approach eliminates need for administrator action, it is still rather slow. Therefore, the open problems for the next research are the optimization of TE tunnel creation and the strategies for their deployment.

Acknowledgments. This work was supported by the Grant No.1/0243/10 and 1/0649/09 of the Slovak VEGA Grant Agency and by the project ITMS 26240120005. We would like to thank Tatrabanka foundation grant for supporting this project.

References

1. Osborne. E., Simha, A.: Traffic Engineering with MPLS. Cisco Press (2002) ISBN 1-58705- 031-5
2. Evans, J., Filsfils, C.: Engineering a multiservice IP backbone to support tight SLAs. Computer Networks 40(1), 131–148 (2002)
3. Cisco Systems, Inc.: Cisco IOS IP SLAs Configuration Guide. Technical report (August 2008)
4. Cisco Systems, Inc.: Configuring Enhanced Object Tracking. Technical report (March 2009)
5. Awduche, D.O., Jabbari, B.: Internet traffic engineering using multi-protocol label switching (MPLS). Computer Networks 40(1), 111–129 (2002)
6. Szviatovszki, B., Szentesi, A., Juttner, A.: Minimizing re-routing in MPLS networks with preemption-aware constraint-based routing. Computer Communications 25(11-12), 1076–1084 (2002)
7. Alshaer, H., Elmirghani, J.: Enabling novel premium service classes in DiffServ over MPLS-enabled network. International Journal of Network Management 18(5), 447–464 (2008)
8. Chen, H., Cheng, B., Su, H.: An objective-oriented service model for VoIP overlay networks over DiffServ/MPLS networks. Computer Communications 30(16), 3055–3062 (2007)
9. Mustill, D., Willis, P.: Delivering QoS in the next generation network – a standards perspective. BT Technology Journal 23(2), 48–60 (2005)
10. Rong, B., Lebeau, J., Bennani, M., Kadoch, M., Elhakeem, A.: Modeling and Simulation of Traffic Aggregation Based SIP over MPLS Network Architecture. In: Proceedings of the 38th Annual Symposium on Simulation, pp. 305–311 (2005) ISBN, ISSN:1080-241X, 0-7695- 2322-6

The Potential of Consolidating SIP and XMPP Based Communication for Telecommunication Carriers

Sebastian Schumann, Michael Maruschke, and Eugen Mikoczy

Slovak University of Technology (STU),
Ilkovicova 3, 81219 Bratislava, Slovak Republic
{schumann,mikoczy}@ktl.elf.stuba.sk
Deutsche Telekom AG, Hochschule für Telekommunikation Leipzig (HfTL),
Gustav-Freytag-Str. 43-45, 04277 Leipzig, Germany
maruschke@hft-leipzig.de

Abstract. This paper describes an approach for combining Session Initiation Protocol (SIP) based voice communication with Extensible Messaging and Presence Protocol (XMPP) based presence enhancements. The actual role of SIP and XMPP in the Internet Protocol (IP) based communication was analyzed, especially from the telecommunication carrier (Telco) point of view.

The proposed infrastructure extends a typical SIP infrastructure with XMPP for presence status integration. XMPP will be used as instant messaging and presence (IM/P) service infrastructure, the presence information will be extended with SIP phone status information of telecommunication endpoints. A first prototype has been developed and tested successfully.

Keywords: SIP, XMPP, Presence, Federation, Collaboration, Telco, Web 2.0, NGN.

1 Introduction

This paper discusses various approaches to overcome the current lack of sophisticated and field-tested presence-enabled communication infrastructures in Telco environments. The target is to provide a service environment for both multimedia sessions and IM/P.

The use of instant messaging has recently become popular not only in private, but also in the business segment. When people communicate through instant messaging systems, one major service enabler must be included: Presence. Without knowing the current state of the other endpoint (i.e., its presence information), the *instant* message would be nothing more than an ordinary E-mail. Knowing that the other party is actually online enables small ad-hoc text-based dialogs and other instant services.

So far, instant messaging and presence infrastructures are usually provided by Internet Service Providers (ISP). Presence states are exchanged between clients

T. Magedanz et al. (Eds.): TridentCom 2010, LNICST 46, pp. 713–726, 2011.

that users have running on their computers or notebooks. If a Telco would like to implement the presence service in its role as a service provider, its first major function would be enabling instant messaging. In order to not just be yet another IM/P provider, Telco's have to think of ways to provide a unique presence service. They can do this by enriching the presence states of their users with telephone state information. This makes their presence system unique amongst providers of the current ISP world[1].

2 Status Quo

Within this paper, the "telecommunication world" will be used as a summary of Telco's and smaller providers that have specialized in offering their customers VoIP based telephony, almost entirely using SIP for session signaling. To date, this telecommunication world itself has not been pushing forward instant messaging and presence.

In contrast to facts mentioned above, the internet community has been using instant messaging and presence for many years. Together, they became part of many Web 2.0 web sites to offer real-time communication to their users.

The "internet world" will be used as a synonym for companies that host web pages or web services and offer communication services – mainly through instant messages.

Some instant messaging clients may already provide some form of voice communication. The protocols used for those clients are either proprietary or they use another important protocol specified by the Internet Engineering Task Force (IETF) exactly for this purpose: XMPP. The use of instant messages and presence is very mature and well implemented already. However, the voice transmission has so far either been proprietary, or is not at a highly developed stage for voice service providers.

This paper summarizes both service protocols mentioned above – SIP and XMPP – and proposes a federated architecture for current Telco's to extend their multimedia infrastructure with mature instant messaging and presence.

While some existing research approaches (e.g., [1], [2]) address the interworking function for converged IP messaging services such as E-mail, Short Messaging Service (SMS) or XMPP instant messaging, the primarily focus of this paper is the integration of SIP telephony states in an XMPP framework and building a converged SIP/XMPP based communication infrastructure. It is not desired to have users from the SIP domain communicating with users from the XMPP domain through gateways. Those gateways have already been realized (e.g., [3]) and they are working fine already. In the context of this paper however, it is assumed instead that *a single user* is present in *both domains* (i.e., in the SIP domain for voice communication and in the XMPP domain for messaging and presence

[1] It is well known that Skype is an ISP client/system that combines presence and telephony. In this paper, the term "Telco" refers to fixed-line telecommunication providers with a Voice over IP (VoIP) infrastructure. Customers use the Telco service in the classic way through hardphones.

exchange). The target of our research is a mature communication infrastructure that can be implemented *today* in the telecommunication world.

3 Protocols and Their Context

3.1 SIP and Presence

SIP is a well-established protocol for the telephony over IP based networks, which has been standardized by the IETF. Many Telco's have already implemented it in their network in order to provide VoIP telephony. The market knows a variety of mature server implementations (proprietary and open-source) end devices and many softphones. [4] defines SIP as "an application-layer control protocol for creating, modifying, and terminating sessions with one or more participants".

The basic SIP methods only enable registration and session establishment. Several extensions exist, which enable the protocol to handle presence. The following possibilities are required for using SIP as presence protocol:

- Means for subscribing to presence information of users
- Means for being notified with presence information of users
- Means for publishing own presence information
- Means for managing watcher authorization

With the CPP (Common Profile for Presence) specified in [5], an abstract model for delivering presence information is available. This model describes the messages between the client presence application and the presence management server. SIP based presence uses the general event notification framework from [6], which defines `SUBSCRIBE` and `NOTIFY`. The particular SIP presence functions for subscriptions and notifications are defined in [7]. This extension of the event notification framework describes basic means to retrieve presence information.

[7] mentions three possible methods showing how the presence service can obtain presence states:

- Analyzing the SIP `REGISTER` messages[2]
- Co-location of the Presence User Agent (PUA) with the presence server
- The client uploads the presence information to the presence server

For the third method, [8] extends SIP with another method: SIP `PUBLISH`. This completes the SIP methods necessary for both sending and retrieving presence information.

An event template package for management of watchers (acc. [9]), and even more importantly for the handling of authorization with the Extensible Markup Language (XML) Configuration Access Protocol (XCAP) (acc. [10,11]), complete the complex framework for SIP based presence.

[2] Here the gathering of advanced important information is already mentioned: The indication of instant messaging support by the client (e.g., SIP `Allow` header for `MESSAGE`). As in our concept, instant messaging as a very important service to be enabled by presence and should be available at the endpoint.

For quite some time, SIP servers have implemented SIP extensions mentioned to handle the exchange of presence information between users. A few of these have even started to integrate or offer interfaces for some XML Document Management Servers (XDMS). However, currently there are only a few – if any – SIP clients that support SIP and XCAP at a satisfactory level. Interoperability between clients is usually not provided yet. One reason for this is that the IETF has defined SIP and XCAP, but does not consider an overall view regarding a presence architecture with all its functions and tasks.

An integrated SIP/XCAP based infrastructure that provides a framework to deliver the presence service with major required functionalities as mentioned within this paper has been standardized by the Open Mobile Alliance (OMA) in [12]. The approach will also be used in the IP Multimedia Subsystem (IMS). For the extensions of a currently SIP based architecture, the extension with presence requires far more than only adding a presence server and an XDMS. The internal interfaces (some of which are even out of scope of the current standard) have to be implemented, services such as resource lists require a tight integration of resource list server, presence server and XDMS and finally clients have to fully support the standards to guarantee a proper handling and satisfying all expectations the users have on privacy.

In summary, Telco's that want to implement the presence services now, and also remain end device vendor independent and close to standard compliant in their implementations, will immediately be confronted with many tasks, if they decide building the service on SIP. Both the scope and complexity of the problems have been shown in this section.

3.2 XMPP and Presence with Messaging

XMPP is standardized by the IETF and mainly designed "for the purpose of building instant messaging and presence applications" (from [13], [14]). While SIP is similar to the Hypertext Transport Protocol (HTTP)[3], XMPP is based on XML.

Knowing that SIP was extended with IM/P, it is not surprising that XMPP has been extended towards session establishment. The extension is called Jingle, standardized in [15], [16]. It is currently implemented in some clients, but not considerable for any larger Telco environment[4]. Despite this fact, all IM/P related advanced functions like

- Authorization handling
- Group chat/multi-user chat
- Server-side storage of the buddy list
- Remote control of clients

are really well implemented in XMPP.

[3] It has for example header/body or header fields on single lines with value and attribute separated by colon.

[4] No hardware endpoints exist, SIP is already well established within the Telco network.

The server as well as client software is very mature and many large companies[5] are successfully using XMPP as their choice for offering IM/P based services or products. Implementing XMPP would leave many customers the free choice in their client and the ability to re-use their currently installed application.

For a Telco operators network, XMPP would be the number one choice at this time to

- provide instant messaging and presence service
- offer a variety of existing clients (the users do not have to get used to yet another client)
- go for standardized protocols, also to decrease the internal implementation efforts necessary.

As discussed within this section, there is no first choice in only one protocol for telephony and presence management. At the moment, the best option would be to keep SIP for session signaling and extend the network with XMPP for instant messaging. Users could use multi-protocol clients to access both services and use the benefits from both. This would however not be without problems.

From the user point of view, multi-protocol clients are currently used because their contacts use different instant messaging systems that do not share a common standard. The multi-protocol clients help the users to unite their contacts in one program. From the operator point of view, this merging at the client-side cannot be controlled. If the operator provides SIP based presence for telephony status information and an XMPP network for their customers to immediately start to chat[6], it is up to the customer and their software to merge the states of their contacts.

An alternative approach is to provide a single infrastructure and to federate the presence states server-side. By federating the existing SIP infrastructure (that exists and inherits all mentioned benefits for multimedia communication) with an XMPP environment (to take advantage of the mature XMPP implementations and large internet community using it), the operator can easily extend its current infrastructure with IM/P. The customers would only need an XMPP client for desktop messaging in addition to the SIP softphones or hardphones already in use. The XMPP states would include the presence states of their VoIP communication devices, hence providing extended presence information that would help the Telco to separate from the current ISP presence.

4 Consolidated SIP and XMPP Architecture

4.1 Focus

The main concept that this paper proposes can be derived by taking all considerations and observations from the previous sections into account. The proposal

[5] Cisco, Google, HP or Sun.

[6] Providing an XMPP access allows the customers also to use their account for integration with the large XMPP clouds in the existing internet world.

assumes that a Telco wants to extend its SIP based telephony infrastructure with presence. The Telco wants to offer a mature infrastructure to its customers and allow them an easy and fast use of its new service. The customers should not necessarily require any new software. They should, however, benefit from call state information the Telco can provide.

The operator has generally two choices, if standardized protocols[7] are mandatory:

- SIP based IM/P
- XMPP based IM/P

The main advantage of SIP based IM/P is that only one protocol will be used within the Telco's signaling network. In the end, it follows existing next-generation network (OMA/IMS) approaches using only SIP for all parts of communication. As mentioned in section 3.1, the OMA concept provides further concepts of inter-domain and inter-service communication. The SIP based concept is however still considered immature (regarding exact specifications that cover *all* required parts for presence) and not yet sufficiently interoperable (regarding available clients). It still takes time for vendors to catch up.

As the target presence service should be mature as well as a solution for offering the service immediately, the paper proposes to base it on XMPP. A positive side-effect is that the solution is *at this very moment* already interoperable with existing IM/P infrastructures and well established in client desktop messaging.

4.2 Concept

The typical user that is discussed within this paper uses primarily a SIP hardware phone for voice communication. A small number of users is also using software based SIP phones. The user is most likely using XMPP instant messaging clients to connect to an IM/P service infrastructure.

The concept will be based on a typical communications set-up as depicted in Fig. 1. The existing SIP communication infrastructure contains all components in a simplified way required to provide the customer the basic telephony service (e.g., proxy, registrar, media resources, gateway). The existing IM/P service infrastructure consists of XMPP clients and servers. Both infrastructures are currently isolated.

To extend this SIP infrastructure depicted in Fig. 1, a transparent proxy has been placed between the signaling endpoints (User Equipment (UE), SIP server). This proxy is aware of all signaling traffic of the UE. From the session related exchanged SIP messages, a co-located PUA can generate SIP `PUBLISH` messages. The generation of publications makes general sense for the following events:

- A SIP UE is registering (i.e., the user can be reached for voice communication)

[7] Both the SIP extensions for presence and XMPP, follow the general model for presence and instant messaging standardized by the IETF (acc. [17], [18]).

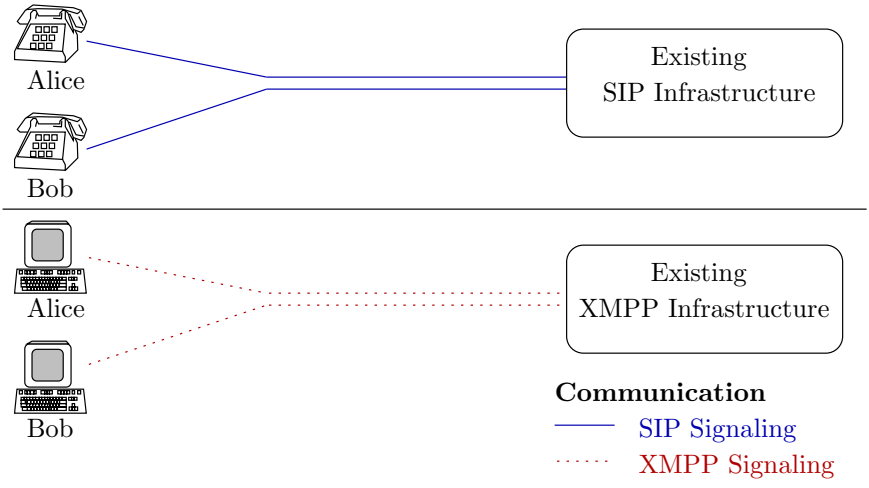


Fig. 1. Current SIP and XMPP isolated communication infrastructures

- A SIP UE is establishing a dialog (in-bound or out-bound call, i.e., the user is busy)
- Further events accessible[8]

In order to represent the SIP UE state in a user's XMPP presence state, it does not make too much sense to show that it is registered. In a typical environment, hardphones are registered all the time. This presence information is not helpful, as it is not conductive to a potential improvement in communication. The most relevant event is call state information. The off-hook event (knowing that a user is in a telephone call) is a typical busy state. The user is busy and does not want to instantly respond to messages.

The generated SIP `PUBLISH` events for a UE with an open dialog will be transferred into the XMPP world as presence state. This process will be explained in detail in the following subsections. For integrating the published information into XMPP, a gateway is required. This gateway will be referred to as XMPP Publisher in the following.

The tasks of this XMPP Publisher are:

- Creating XMPP `<presence>` stanzas from received SIP `PUBLISH` messages
- Map the SIP user identification to an XMPP user identification
- Provide authentication for that user towards his XMPP server and publish his presence state

[8] There are no means in SIP to signal certain changes like setting Do Not Disturb (DND) at a hardware endpoint. However, if the user is performing those changes with telephony call "star codes", they are signalled to the SIP system and the proxy could evaluate this to generate events. As this is usually not the only way to perform those settings, generating presence events from them is not considered within this paper.

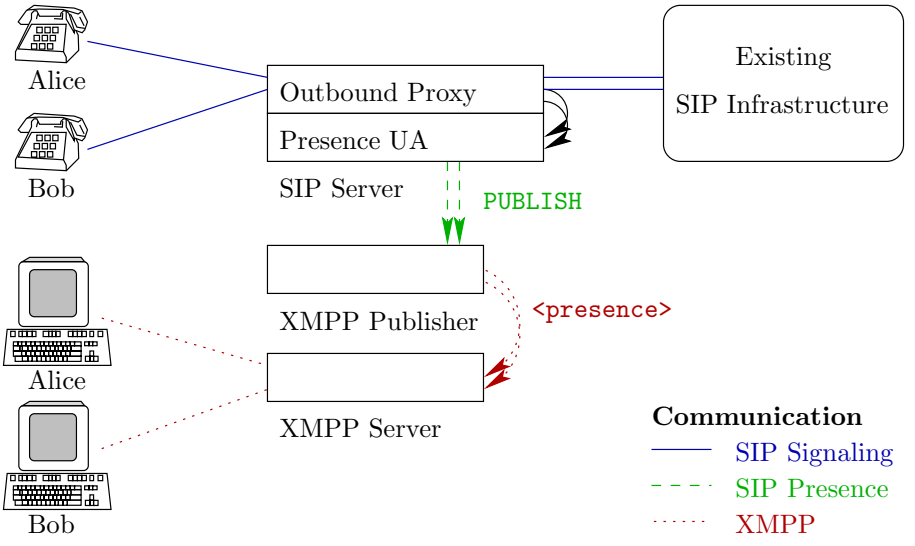


Fig. 2. Implemented federated SIP/XMPP communications infrastructure

This paper will focus on an XMPP based presence system as discussed. The proposed design can be either simplified (e.g., by co-locating the XMPP Publisher with the SIP proxy and skipping the PUA) or made more complex (e.g., in forwarding SIP `PUBLISH` messages from clients to the XMPP Publisher).

Fig. 2 shows the implemented extension of the previously shown isolated SIP and XMPP infrastructures as a prototype. It contains all components that have been introduced within this subsection. The proposal of this paper is based on the assumption that the SIP and XMPP server infrastructures are both managed by the Telco.

At this point, all SIP related presence information (call state information transformed to presence events and user generated presence events) is available within the XMPP network. The XMPP clients, which are used by the user for the exchange of messages in the "internet world", are enhanced with the information from the "telecommunication world" and change their states accordingly.

4.3 XMPP Publishing

Publishing the state for XMPP only makes sense, if a user has an account in this system. The presented concept can be used for both users that have accounts in the Telco's own XMPP network as well as users with an existing account in another domain.

For the publication of the presence state and merging it with the existing XMPP resources of the user, two basic use cases have been elaborated:

a) Publish a separate resource (e.g., "sip-phone") with the telephony state of the user
b) Modify the presence state of an existing resource

Case a) has been the most obvious case in the beginning of the research. The SIP presence state is presented as a separate, additional resource of the user. This resource can be either online (i.e., the user is registered), busy (i.e., the user is in a SIP dialog) or offline (i.e., user not registered). Publishing this resource can generally happen independent of whether the user is logged in with another resource or not. The technical realization is rather obvious: The XMPP Publisher must open an instance with a certain resource (e.g., "sip-phone") on behalf of the user to change the state accordingly.

Case b) is not so obvious in the beginning, but has several advantages in comparison with a). Although this case makes only sense when the user is already online with one or more resources, the approach supports the concept more than approach a). The primary instant messaging happens within the XMPP world; the SIP state is only supporting and refining. Hence, a single SIP state of a hardware endpoint is not necessarily required, as no IM conversation can happen with this endpoint anyway. On the other hand, when an XMPP resource is online, it makes perfect sense of refining its state and letting potential contacts know that the user is in a SIP conversation.

The second case has another advantage: XMPP users can only reach resources they can actually communicate with. In case a), the resource "sip-phone" is addressable; hence, users might send messages to it – only receiving an error message as response. By modifying the state of an existing resource as b) indicates, the user keeps control over it and has this resource available for messaging as well. The realization of case b) requires the used client to support [19]. During SIP registration, the XMPP Publisher will log on with a resource that has a negative priority on behalf of the user. This online resource will know the other resources. If the user makes a phone call, it will change the presence state of the client and set it to something pre-defined (e.g., "do-not-disturb" with the status "I am currently on the phone").

For both cases, resources are very important. Either a new resource or an existing resource will represent the users SIP state. Another not less important point to make the concept work is the priority that is assigned to a certain resource[9]. XMPP handles priorities as follows:

Resources with positive priority can receive messages. XMPP servers send messages to the resource with the highest priority. Some clients only show the state of the resource with the highest priority as user state, some show all.

Resources with negative priority must not receive messages. If only one resource is online, and this resource has a negative priority, XMPP servers handle the message as if the user was offline.

Case a) would conceptually work fine, if a resource would get a negative priority: The XMPP server would never deliver messages to it. If the user would not be online with another client, no messages would be lost. The major drawback here

[9] Although this concept would work regardless of the priorities, many clients will only present the overall state from the resource with the highest priority.

is that now client would indicate at first sight, that the user is in an actual conversation. All other states would have a higher priority and not be reflected in almost any client at all. Case b) on the opposite would perfectly work. The XMPP Publisher would modify the state of an existing resource with the highest priority and the state would be reflected immediately. A positive side effect is that the user could immediately change the state back, if desired[10].

4.4 Addressing

Several drafts (e.g., [20], [21]) discuss the general possibility to signal the XMPP Address of Record (AoR) in SIP (or vice versa), so the servers can make a correct assignment. The purpose of those drafts however differs from the approaches of this paper. While those drafts focus on an SIP/XMPP gateway for users from one world communicating with users from the other (as in [20]) or on converged SIP/XMPP clients and the signaling of the AoR for mappings between SIP and XMPP "communication streams" (as in [21]), this paper focusses on a single user in both worlds that wants to use solely XMPP for instant messaging and presence information exchange.

In the presented concept, an actual XMPP client instance is running on the XMPP Publisher on behalf of each user. This concept requires the provisioning of the Jabber ID (JID) *and* the password; hence, this additional signaling for assigning publications is not required. For each SIP user that has assigned an identity in the XMPP Publisher, the XMPP JID and password are provisioned separately. This allows the system to work with the Telco's XMPP infrastructure (pre-provisioned system), but also with separate XMPP servers.

In a provider pre-provisioned system, the SIP provider has also complete supervision over the XMPP server. The XMPP user account is provisioned at the same time as the SIP account. The XMPP Publisher has access to the XMPP user database to successfully authenticate against the XMPP system to modify the resource as a user.

For a separate XMPP server, the user has to provide his credentials to the XMPP Publisher. In both cases, the signaling of the XMPP AoR is not required as the assignment between SIP user and XMPP user AoR cannot be automatized.

4.5 Implemented Prototype

The currently implemented prototype uses several simplifications:

- The proxy is not using a PUA. The open-source SIP application server OpenSIPS [22] has been configured to recognize events in SIP that signal off-hook and on-hook. For both cases, the server calls the XMPP Publisher directly.
- The prototype is not using [19] yet, but a separate resource (as in case a) sec. 4.3). The resource increases its priority if the user is off-hook to be sure it is signaled in the client. The mentioned problems appear, but the concept is visible.

[10] If the XMPP resource that represents the SIP state in case a) supports [19], a client could change its state back as well. It could also increase the priority of its active resource. The authors are aware of this but consider it as too complex for a "quick change-back".

- The XMPP Publisher knows certain events (online, still online, on the phone, still on the phone, offline):
  - Online if the user first registers
  - Still online if the user re-registers
  - On the phone if the user initiates or receives a call
  - Still on the phone for re-`INVITE` messages
  - Offline if the user de-registers or the registration expires.
- The publishing towards XMPP is done using Bidirectional-streams over Synchronous HTTP (BOSH). The persistent Transport Control Protocol (TCP) connections (each published user towards the server) were not implemented, BOSH has rather been chosen with a certain timer (450 seconds). This allows the sending of "events" over non-persistent HTTP connections. The registrations and SIP session timer are forced to 300 seconds. This is the reason for the "still"-events (to allow a "refreshing" the timer).

A prototype according to the concept presented within this paper is currently being implemented and will be tested in practical scenarios.

5 Future Scope and Use Cases

The presented XMPP Publisher can even be of use when the Telco decides in the future to migrate to IMS or centralized SIP based presence management (IMS includes this). While the customer might be able to use sophisticated hardware or software for connecting his co-located SIP PUA, he will still most likely use the XMPP infrastructure. Fig. 3 shows that the XMPP Publisher can still be used to bring the centralized SIP presence information (that is stored in the presence server) easily into the XMPP network, by simply relaying the SIP publications to the XMPP Publisher.

The IMS user that might be using an IMS SIP compliant hardware endpoint still benefits from the proxy and connected PUA (to generate presence events from the SIP communication). Those events might be stored in the presence server of the IMS in the future. SIP/XCAP compliant IMS clients publish their presence states directly to the presence server. The presence server collects all those events and the co-located XDMS infrastructure (omitted in Fig. 3) handles privacy and XCAP documents the client provides. As clients in this IMS-Telco infrastructure will still have *and actively use* XMPP clients and their existing infrastructure, the XMPP Publisher will enrich all of these with IMS presence information.

The previously described scenario can further be improved by developing the XMPP Publisher into an "SIP/XMPP Publisher". The integrated XMPP client side, which is connected on behalf of the SIP user, is active as one of its resources. This also means that it is aware of the presence states of all its XMPP contacts. If some mapping can be implemented in the future (e.g., if SIP and XMPP belong to one operator, the username-part of SIP and XMPP AoR is unique and belongs to a single customer) this XMPP presence states can be "back-published" towards the SIP system as well. Both systems would then be aware

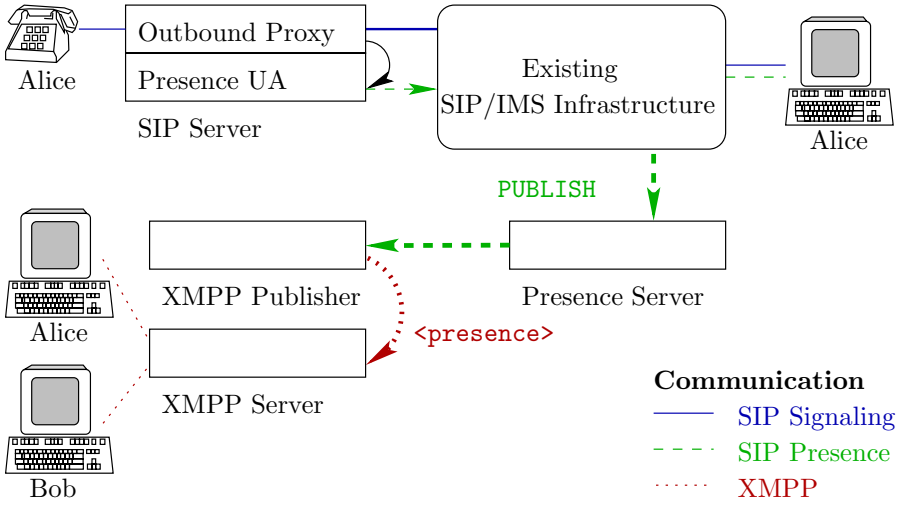


Fig. 3. Future integration of the XMPP Publisher

of the presence states of all users – at least where a proper mapping could be made. Converged messaging would furthermore even allow the communication throughout the "protocol borders".

Another future target is to improve the identity mapping and the related authorization (pre-provisioned user names and passwords). The authors are currently analyzing whether XMPP provides means and can be extended with a useful mechanism respectively to connect the XMPP Publisher similar to components with the XMPP server. A "trusted component" could then modify presence states of the served users. Moreover, the provisioning efforts would be minimized and the simplicity of the concept improved.

Additional use cases have been identified by the authors for the proposed federation of SIP and XMPP architectures. The interworking can be extended from presence-based services also to messaging/notification or location-based services. The information just needs to be mapped in a meaningful way. The near future may include also the initiation of voice/video communication or content streaming initiation. Some drafts are working on that as well.

The main advantage of both protocols is their extensibility and capability to transport XML encapsulated information in the protocol body. Finally, the same XML body should be used in the same manner and enable convergence of both worlds and easier interworking. The most relevant scenario is using this convergence to integrate some of the Web 2.0 technologies based on XML or XMPP for extension of integrated concepts with SIP based architecture. The goal is not replace SIP protocol technology with XMPP protocols but used each of their strengths and benefit from their interworking for Telco and Internet convergence. This wider scope of proposed federation should also extend the applicability and Web 2.0 service accessibility in the future. This is also valid for

NGN/IMS based architectures, as it would make them more attractive for application developers and internet users. In these cases the same services will be available on NGN networks, but also on the internet and most importantly providing the same user experience and large user community.

6 Conclusion

The presented concept is a good compromise between immediately starting to introduce presence in the telecommunication world. XMPP as widely used and adapted concept has been chosen.

The authors are aware and support the moves from the standardization committees to push SIP based presence management forward. IMS is from the author's perspective clearly the future; its presence management concepts included. The Telco's are however reminded that integration with the existing Web 2.0 infrastructure is important and the presented concept can highly increase the user experience. Customers will choose the Telco that does not only offer new converged services on their platform, but also respect the existing communication infrastructure of the clients and integrate it as good as possible.

Acknowledgments. The STU and the HfTL are actively cooperating since 2006. The universities are working together on common topics in the area of Next Generation Networks and participate in ongoing projects involving their core network interconnection.

The authors Maruschke and Schumann are members of the 'Institut für Telekommunikationsinformatik' at the HfTL. Working mainly in the area of Next Generation Networks, their focus is on migration of core networks of global telecommunication operators towards packet oriented networks as well as their interconnection.

This paper also presents some of the results and acquired experience from various research project such as NGNlab project [23], European Celtic-EURECA project Netlab[24], Leonardo da Vinci projects InCert [25] and Train2Cert [26], AV project: Converged technologies for next generation networks (NGN), Slovak National basic research projects VEGA No. 1/3094/06 and VEGA 1/4084/07.

References

1. Blum, N., Lampe, S., Magedanz, T.: Design of a Message Interworking Function for Converged IP Messaging in Next Generation Networks. In: Computers and Communications, ISCC 2009, pp. 80–85. IEEE Press, Sousse (2009)
2. OMA Converged IP Messaging Architecture, System Description System Description-V1 0-20091024-D, Open Mobile Alliance (2008)
3. OpenSIPS XMPP and PUA_XMPP module, `http://www.opensips.org/Resources/PuaExtensions#pu_xmpp`
4. Rosenberg, J., et al.: RFC 3261: SIP: Session Initiation Protocol. IETF (2002)
5. Peterson, J.: RFC 3859: Common Profile for Presence (CPP). IETF (2004)

6. Roach, A.B.: RFC 3265: Session Initiation Protocol (SIP)-Specific Event Notification. IETF (2002)
7. Rosenberg, J.: RFC 3856: A Presence Event Package for the Session Initiation Protocol (SIP). IETF (2004)
8. Niemi, A.: RFC 3903: Session Initiation Protocol (SIP) Extension for Event State Publication. IETF (2004)
9. Rosenberg, J.: RFC 3857: A Watcher Information Event Template-Package for the Session Initiation Protocol (SIP). IETF (2004)
10. Rosenberg, J.: RFC 4825: The Extensible Markup Language (XML) Configuration Access Protocol (XCAP). IETF (2007)
11. Rosenberg, J.: RFC 5025: Presence Authorization Rules. IETF (2007)
12. OMA Presence SIMPLE, Architecture Document OMA-AD-Presence_ SIMPLEV1_1-20080627-A, Open Mobile Alliance (2008)
13. Saint-Andre, P.: RFC 3920: Extensible Messaging and Presence Protocol (XMPP):Core. IETF (2004)
14. Saint-Andre, P.: RFC 3921: Extensible Messaging and Presence Protocol (XMPP): Instant Messaging and Presence. IETF (2004)
15. Ludwig, S., et al.: XEP-0166: Jingle. XMPP Standards Foundation (2009)
16. Ludwig, S., et al.: XEP-0167: Jingle Audio Media Description Format. XMPP Standards Foundation (2009)
17. Day, M., et al.: RFC 2778: A Model for Presence and Instant Messaging. IETF (2000)
18. Day, M., et al.: RFC 2779: Instant Messaging/Presence Protocol. IETF (2000)
19. Troncon, R., et al.: XEP-0146: Remote Controlling Clients. XMPP Standards Foundation (2006)
20. Sparks, R.: Internet Draft (expired): Establishing jabber Messaging Sessions with the Session Initiation Protocol. IETF (2002)
21. Veikkolainen, S., et al.: Internet Draft: Guidelines and Protocol Extensions for Combining SIP Based Real-time Media Sessions With XMPP Based Instant Messaging and Presence Service. IETF (2009)
22. OpenSIPS (Open SIP Server), a mature Open Source implementation of a SIP server, `http://www.opensips.org`
23. NGNlab - NGN laboratory at STU in Bratislava, `http://www.ngnlab.eu`
24. NetLab - Use Cases for Interconnected Testbeds and Living Labs, `http://www.celtic-initiative.org/Projects/NETLAB/`
25. InCert Next Generation Network Protocols Professionals certification in InCert, International Certificates of Excellence in Selected Areas of ICT, `http://incert.eu`
26. Train2Cert, Vocational Training for Certification in ICT, `http://train2cert.eu`

Deploying IP Multimedia Subsystem (IMS) Services over Next Generation Networks (NGNs): The IU-ATC Integrated Test Bed

A. Oredope[1], M. Dianati, B. Evans[1],
Rohit Budhiraja[2], and Bhaskar Ramamurthi[3]

[1] The Centre for Communication Systems Research (CCSR),
Faculty of Engineering and Physical Sciences,
University of Surrey, Guildford
GU2 7XH, United Kingdom
{A.Oredope,M.Dianati,B.Evans}@surrey.ac.uk
[2] CeWIT, India
rohit@tenet.res.in
[3] Indian Institute of Technology, Madras
I.I.T. Post Office Chennai - 600 036 India
bhaskar@tenet.res.in

Abstract. The IP Multimedia Subsystem (IMS) is the Third Generation Partnership Project's (3GPP) standardized service platform that enables the deployment of rich and personalized services over fixed and mobile networks whilst allowing end-users ubiquitous access to services such as voice, video, presence and online gaming anytime and anywhere. However, the delivery of these services to the end-users is highly dependent on the available or preferred access network which could range from fixed broadband access to mobile 4G connections. Although the IMS was initially developed as the core network for Third Generation (3G) systems, it has now been adopted as the service platform for the Long Term Evolution (LTE) and System Architecture Evolution (SAE). As this transition of 3G to 4G and beyond evolves, there is an immediate need for a research testbed that facilitates the research, development and early trials of the integration of these technologies. This has motivated us to integrate the IMS based Advanced Next Generation Network (ANGN) testbed at the University of Surrey (UniS), U.K. with the 4G Access Network Testbed at IIT Madras, India via an academic transnational network link to form a fully functional telecommunications mobile network. In this paper, we discuss the rationales, motivations and objectives behind the integrated testbed whilst also investigating how it can be extended to support 4G and future technologies such as LTE/SAE and WiMAX. The testbed as a whole plays a key as role in the future of IMS development as it provides a fully functional platform similar to commercial networks for researchers to investigate and demonstrate the feasibility of their proposal in a realistic environment.

Keywords: IMS, LTE, SAE, NGN, 4G, 3G, WiMAX.

T. Magedanz et al. (Eds.): TridentCom 2010, LNICST 46, pp. 727–735, 2011.

1 Introduction

Next Generation Networks (NGN) provides the long awaited convergence of the Internet and the cellular worlds, providing ubiquitous cellular access to all the services that the Internet offers and vice-versa [1]. Examples of these services are Voice over IP (VoIP), Instant Messaging, online gaming, video conferencing and presence services. This is achieved by a combination of state-of-the-art access and core technologies providing an access-independent all-IP platform that enables the interconnectivity and interoperability of mobile networks to other heterogeneous networks via packet switched technologies. The development of such an end-to-end all-IP platform for research purposes will facilitate the early trials and development of various new and emerging technologies aimed at NGN platforms. Although various IP Multimedia Subsystem (IMS) testbeds already exist [2-7], our aim in Theme-10 of the IU-ATC UK-India project, is the development of an international research testbed interconnecting key NGN elements in order to emulate an operational and production-level NGN platform. This includes the access, core-network and service platforms.

Such an integrated testbed will help researchers to investigate and demonstrate feasibility of their proposals in a realistic environment. The test-bed will also complement the other analytical and simulation based studies that may be performed in any individual study. In other words, the test-bed will not be considered as a replacement to analytical or simulation based studies but rather to aid the researchers in investigating the practical challenges. Our work in the Theme 10 of the IU-ATC UK-India project will use the existing testbed developed by IITM and UniS as launching pads. This includes the ongoing project in IITM to develop a testbed for research in radio access techniques by deploying a small radio access network with three SDR- based Base Stations (BS) supporting MIMO/OFDM based physical layers [8]. In addition, UniS has an all-IP Advanced NGN (ANGN) testbed [9] that can be configured to provide core service and network infrastructures using both carrier grade IMS platform and the OpenIMS [10] core. Thus, combining these two testbeds, we aim to build a complete platform that can contain different type of virtual mobile networks on a shared physical platform. For example, the physical infrastructure may contain two concurrent virtual WiMAX and LTE mobile networks. A unique feature of the test-bed will be its distributed processing property. While the radio part of the network will be developed and deployed by IITM, the service platform and core network will be hosted by UniS. The two parts of the network will be connected by various academic networks as shown in Fig. 1.

There will be sufficient flexibility in the topology of the network by means of using configurable switches in both the radio and core sections of the network. In addition, some flexibility will be possible by allowing remote researchers to upload their physical layer and Radio Resource Management (RRM) schemes to the IITM radio network. The core component will be deployed according to the experiment. For instance, an experiment may require LTE network components, whilst another experiment may need a WiMAX network. These experiments can concurrently co-exist on the same physical infrastructures. Both core and radio networks will provide appropriate interfaces for remote researchers and network administrators. Thus, the other partner will be able to remotely deploy their experiment and get back the measurements that they need. As an example, some partners may only need to do experiments on physical and

radio parts. Such partners will be able to access the corresponding APIs at the IITM side. Some other partners may be interested in studying some application performance; thus, they could access the application server in UniS to deploy their applications.

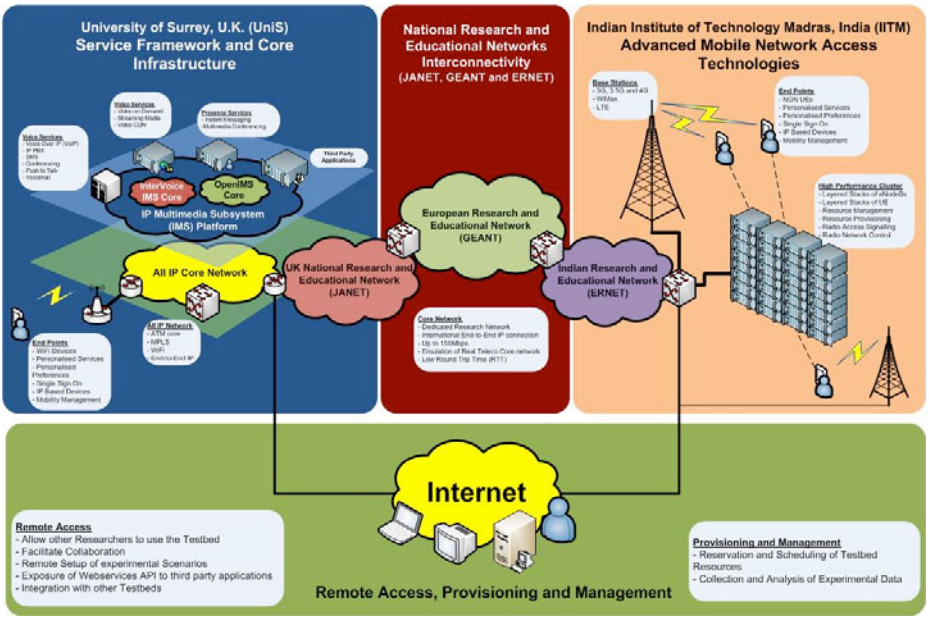


Fig. 1. An Overview of the IU-ATC Integrated Testbed

The rest of this paper is organised as follows: In Section 2, we provide the rationale, motivations and research objectives for the testbed looking at the roles that the testbed can play in the development of the integration of IMS and the advanced access networks. In Section 3, we describe the integrated testbed while pointing out some key features in the respective UniS and IITM the testbed. Finally, conclusions and future work are presented in Section 4.

2 Background

The IMS has been developing rapidly over the years, offering a solution to the all- IP vision of rich, multi-access multimedia services accessible anywhere, at any anytime, with the required quality, and at the right price [11,12]. The IMS is now widely considered as the future service-centric platform of reference, although several researchers are still arguing that the benefits of IMS come with a high cost linked to its layered approach (complexity). Only a large-scale deployment of the IMS and a wide adoption by operators and service providers, will allow a full assessment of its benefits and shortcomings. The enormous interest surrounding the IMS indicates that this may really become the standard framework for deploying advanced, ubiquitous services over converged, all-IP networks, i.e the “Domain of Services”. However, not all

the research areas have been fully explored. One of the areas is the deployment of IMS over next generation 4G networks, which explores access technologies such as the LTE and WiMAX. There have been various concerns on the integration of the LTE core, known as the System Architecture Evolution (SAE) with the IMS [13]. Although the functionalities that are proposed in the IMS can be achieved in the SAE, the integration of IMS with SAE provides the major advantage of being a domain of services in which operators can easily bring new services to the market quickly while also exposing various functionalities to third party developers and also charging appropriately for these services. The integration of both the IMS and SAE however currently raises various concerns and it posses as an area worth investigating.

This has been one of the key motivations to combine the existing test-beds of IITM and UniS into an integrated test-bed that can be used to prototype and test advanced protocols developed within the framework of the IU-ATC project. This will help researchers and the IMS community as a whole to investigate/demonstrates feasibility of their proposals in a realistic environment. The test-bed will complement the other analytical and simulation based studies that will be performed in individual areas. In other words, the test-bed will be one of the many tools for IMS based analytical or simulation studies. It will also help the researcher to investigate the practical challenges as we have outlined in the following objectives.

First, a fully operable end-to-end system and corresponding performance- monitoring tools will be developed. In order to achieve this, the Indian portion of the test-bed will be dedicated to the radio-related technology aspects of the next generation of mobile networks, from the air interface, physical, Multiple Access scheme. For research on higher level networking features at the network level and above, e.g., Mobility support for IP networks, Multicast support, Service platform for future mobile networks, and IP Multimedia Subsystem, the UniS segment of the testbed will be used. The most powerful capability of the integrated test-bed will come into play when research is carried out employing both segments simultaneously, interconnected across the continents, using the transnational academic network, which provides higher bandwidth and fewer routes as compared to the public Internet. As an example, a scenario for establishment of an end-to-end multimedia service session between two User Equipments (UE), one in the UniS and other in IITM segments of the testbed. The IMS core and required CSCF (Call Session Control Functions) are provided and maintained within the UniS ANGN, and radio part research scenarios are tested within IITM testbed. This will allow possible interoperability issues to be identified and solutions developed. Key performance indicators (KPI) encompassing application, radio and network protocols will also be identified. Software modules for capturing of KPIs and their post-processing will also be developed.

Another key objective that would be facilitated by the testbed will be the ability to develop, analyse and optimise novel mobility management protocols enabling fast and reliable horizontal and vertical handover. In order to achieve this the UniS ANGN will be enhanced with development and evaluation of new advanced mobility management and network/user security protocols. The advanced mobility management will be based on multi-layered mobility management protocol concepts which require cooperation between application layer, network layer and handover at layer 2 using respectively SIP, MIPv4/6 and layer 2 handover algorithms respectively. The multi-layered mobility management protocol will be optimized with a fully integrated context transfer

protocol for fast and reliable and secure handover involving AAA for user authentication. We will also be developing and analyzing various sets of cross-layer link adaption, resource management, admission control, packet scheduling and routing algorithms to provide QoS guarantees for a variety of services and application. We also aim to investigate

Other research objectives that can be investigated on the testbed include Advanced MIMO Schemes including various BS co-operation techniques and establishing their respective performance parameters, NGN charging mechanisms and inter-domain presence management to mention a few.

3 The Integrated Testbed

The "End-to-End Transnational Wireless Network Testbed" is based on Software Defined Radio (SDR) network elements, a High Performance Computing (HPC) cluster, and an IP-based network nodes with IP Multimedia Subsystem (IMS) service development platform. In this section we provide a brief overview of the existing facilities and their current status, at both IITM and UniS, an interworked architecture together with selected advanced research ideas to be tested and optimized utilizing the testbed. The testbed will provide real-time and non real-time (depending on algorithm complexity) emulation of a 4G network, providing researchers with an insight into how their ideas perform and how different techniques interact. Part of the testbed is proposed to be located on the campus of Indian Institute of Technology, Madras in order to leverage the significant physical resources already available and interconnected through various academic networks with niversity of Surrey (UniS) existing Advanced Next Generation Network (ANGN) testbed. The capabilities of the UniS platform will be significantly augmented with all-IP based mobility management protocols with context-transfer techniques for fast and reliable handover employing a novel multi-layered mobility management technique. It will also be enhanced with new integrated Resource Management, MAC and routing protocol as well as a new cross- layer control plane functionality enabling cross-layer operation. The capability of the IITM platform will be significantly enhanced with proposed advanced MIMO and Air- interface techniques enabling full cross-layer operation between Physical layer and the higher layers in the protocol stack, resulting in power and bandwidth efficient air- interface solutions for the future 4G systems. This cooperative experimental based research project aims at leveraging on existing testbeds in the UK and India and enhance them with advanced Physical layer techniques, MAC, RM, MM and Network layer and SIP protocols together with cross-layer control plane techniques for efficient 4G all-IP network.

The testbed at IITM will have four BS radio-nodes and 12 UE radio-nodes in the coverage area of the BS nodes. Each BS node will have four antennas while a UE node will have two antennas. BS and UE nodes will have transmit power of 5 W and 250 mW respectively. Testbed is designed for a transmission bandwidth of 20 MHz. All the protocols of MAC and PHY (as well as applications, in other layers) will be run in the cluster and the digitized samples of the signal that has to be transmitted/received on air will be communicated to/from BS nodes as Ethernet packets over optical fibre links. Cluster has 16 nodes and each node has 8 processors. Applications

could be run in real-time or near real-time depending on the degree of parallelisation achieved. BS as well as UE nodes will be synchronised to the GPS clock to enable co-ordinated transmission. Ethernet packets will be time stamped to ensure transmission in the intended frame.

The test bed at the University of Surrey provides an end-to-end all-IP platform for the delivery of NGN services using the standardised IP Multimedia Subsystem (IMS) service framework. These services include Voice over IP, video streaming and multimedia conferencing to mention a few. The testbed is also made up of state-of-the- art network and service elements that allow various forms of research to be carried out on both fixed and wireless network domains. The service domain is made up of an open source IMS core and also an industry standard IMS platform harnessing the advantages of IMS SIP profiles that are used in live environments by various telecommunications providers. The UniS testbed is developed using a layered approach allowing for either each layer to be individually extended or allowing for cross layer research and extensions to be achieved. The layers are mainly divided into the access technologies, the IP core and the service domain.

At the access level, the testbed provides a platform to study concepts such as mobility management, vertical and horizontal handovers, new communication technologies, signalling effects on the integration of mobility and security and secured communications over public infrastructures to mention a few. Although converged networks already have some mechanisms in place to enhance the deployment of mobile services [2,11,14], the test bed provides the opportunity to research these technologies in real environments using standardized protocols and applications. Furthermore, based on the technologies available on the testbed, significant attention has been devoted to individual studies of mobility at the access network level in converged networks investigating specifically the integration of mobility and security. This is has led to the development of a wide range of access technologies and protocols such as RMA controlled Soft Radios, Mobile IP and Mobile IPv6. Other research areas that a closely related with the access layer include AAA Context Transfer, Dynamic Configuration of Access Networks and Sensor Environments.

The IP core network provides a flexible, re-configurable network platform, capable of supporting an extensive range of networking and service provisioning scenarios. It can also be used to set up protocol stack functionalities in a flexible manner in order to support adaptive quality of service for multi-media communications in mobile environments. The IP core also contains almost all the essential constituents of the public Internet. It can be configured to support both IPv4/IPv6 and has the ability to undertake research in every area of fixed and mobile communication such as Mobility support for IP networks, Service Discovery, Location and Routing in MANETs/PANs, Terminal and network reconfigurability, Quality of Service (QoS), Network and data security and Multicast support.

The Service domain on the test bed is made up of a back-end infrastructure enabling a converged architecture which can provides personalized services to end users and devices. It uses the service framework known as the IP Multimedia Subsystem (IMS), which is the Third Generation Partnership Project (3GPP) standardised core network for the all-IP convergence of fixed and mobile networks. The core signalling protocol in the IMS is the Session Initiation Protocol (SIP), a lightweight standardised protocol for creating, modifying and terminating multimedia sessions

over the Internet and also in converged networks. The UniS testbed is made up an open source IMS core known as the FOKUS OpenIMS platform and also a carrier-grade IMS core developed by InterVoice, which is used in live telecommunication networks. Based on the two IMS core, services such as Voice over IP (VoIP), Video on Demand (VoD), Presences Services and a combination of these services can be provided to both fixed and mobile end users using the testbed. The service domain has motivated various research in the field of Sensor Networks, Charging and Billing in converged networks, Service Management and Quality of Experience (QoE) to mention a few.

4 Future Work and Conclusions

The IMS as a whole has great potential and has been developing rapidly over the years, offering a solution to the all-IP vision of rich, multi-access multimedia services accessible anywhere, at any anytime, with the required quality, and at the right price. The IMS is now widely considered as the future service-centric platform of reference, although several researchers are still arguing that the benefits of IMS come with a high cost linked to its layered approach (complexity). Only a large-scale deployment of the IMS and a wide adoption by operators and service providers, will allow a full assessment of its benefits and shortcomings. These will hopefully be unveiled in the next few years. The enormous interest surrounding the IMS indicates that this may really become the standard framework for deploying advanced, ubiquitous services over converged, all-IP networks, i.e the "Domain of Services". However, not all the research areas have been fully explored as yet especially in the area of distributed session management. Many approaches have been developed to eliminate (or reduce) the number of centralized servers from the IMS, aiming at a better level of scalability.

Furthermore, as the IMS is paving way for the introduction of the next generation 4G networks, which explores access technologies such as the LTE and the WiMAX. There have been various concerns on the integration of the LTE core, know as the System Architecture Evolution (SAE) with the IMS. As most of the functionalities that are proposed in the IMS can be achieved in the SAE, the IMS still provides the major advantage of being a domain of services in which operators can easily bring new services to the market quickly while also exposing various functionalities to third party developers while also charging appropriately for these services. The integration of both the IMS and SAE at the moment raises various concerns and it posses as an area worth investigating.

Another promising area of research is that of the deployment of peer-to-peer services in the IMS. This requires coming up with solutions as to how SIP sessions can be managed via distributed signalling protocols. Before this vision can become a reality, a number of issues still need to be tackled. P2P overlays do not map well onto physical networks, since P2P systems optimize IT resources but neglect the network. There is a fundamental clash between current P2P and network architectures. P2P architectures were designed to bypass the network operator, limiting its ability to control and influence the P2P application overlay. On the other hand, fundamental operations such as charging, security, quality control, and location management are hard to realize without the operator's collaboration. P2P services in the IMS, bring them into the operator's realm, creating the preconditions for a richer P2P service. If

placed in perspective, P2P services in the IMS have the potential to address problems that have so far remained unresolved. Current P2P systems are increasingly faced with the problem that they do not offer a sound digital rights management solution. Privacy and data retention legislation is also bound to curb the further development of current P2P systems. There is also the major issue of how to cater for national security requirements, given that it is not currently possible to perform legal intercepts on P2P communications and data flows. P2P is now a service in high demand which requires an immediate, fundamental redesign. The IMS may be the next P2P provisioning platform, provided that the IMS itself evolves towards a more decentralized architecture. All in all, facilitating P2P services in the IMS will enable the deployment of new and exciting services, achieving what the IMS was initially designed for i.e. a platform of services.

Finally, the UK-India transnational testbed is just one many steps in the various approaches in the IU-ATC to continually contribute the development of the digital economies of both countries specifically focussing on the rural and remote areas. As the testbed evolves, various technologies and concepts will be developed and added to it in order to continually support the research aims and challenges faced by the IU-ATC research Themes and applications as well as a facility for industry to showcase and test future products.

References

[1] Camarillo, G., Garcâia-Martâin, M.A.: The 3G IP multimedia subsystem (IMS): merging the internet and the cellular worlds. Wiley, Chichester (2006)

[2] Balakrishna, C.: IMS experience centre A real-life test network for IMS services. In: 5th International Conference on Testbeds and Research Infrastructures for the Development of Networks & Communities and Workshops, TridentCom 2009, pp. 1–8 (2009)

[3] Mecklin, T., Opsenica, M., Rissanen, H., Valderas, D.: ImsInnovation - experiences of an IMS testbed. In: 5th International Conference on Testbeds and Research Infrastructures for the Development of Networks & Communities and Workshops, TridentCom 2009, pp. 1–6 (2009)

[4] Al-Hezmi, A., Friedrich, O., Arbanowski, S., Magedanz, T.: Provisioning of an Open NGN/Triple Play Toolkit and Testbed. In: 3rd International Conference on Testbeds and Research Infrastructure for the Development of Networks and Communities, TridentCom 2007, pp. 1–6 (2007)

[5] Magedanz, T., Witaszek, D., Knuettel, K.: The IMS playground @ FOKUS-an open testbed for generation network multimedia services, pp. 2–11 (2005)

[6] Mullins, R.: A framework for the creation of end user and enabling services on IMS. In: 5th International Conference on Testbeds and Research Infrastructures for the Development of Networks & Communities and Workshops, TridentCom 2009, pp. 1–3 (2009)

[7] Gonguet, A., Durecu, O., Gaste, Y.: Exoticus: An IMS experimentation testbed Experimentation methodology and environment for IP applications. In: 5th International Conference on Testbeds and Research Infrastructures for the Development of Networks & Communities and Workshops, Washington, DC, USA, pp. 1–8 (2009)

[8] James, J., Ramamurthi, B.: A two-antenna two-tone space-frequency code using reduced channel feedback. In: IEEE 8th Workshop on Signal Processing Advances in Wireless Communications, SPAWC 2007, pp. 1–5 (2007)

[9] Akhtar, N., Georgiades, M., Politis, C., Tafazolli, R.: SIP-based end system mobility solution for all-IP infrastructures. IST Mobile & Wireless Communications Summit (2003)
[10] Vingarzan, D., Weik, P., Magedanz, T.: Design and Implementation of an Open IMS Core. In: Magedanz, T., Karmouch, A., Pierre, S., Venieris, I.S. (eds.) MATA 2005. LNCS, vol. 3744, pp. 284–293. Springer, Heidelberg (2005)
[11] Östergaard, S.: Extending IMS specifications based on the charging needs of IPTV
[12] Mikoczy, E., Sivchenko, D., Xu, B., Rakocevic, V.: IMS based IPTV services-Architecture and Implementation
[13] Sauter, M.: Beyond 3G-Bringing Networks, Terminals and the Web Together: LTE, WiMAX, IMS, 4G Devices and the Mobile Web 2.0 (2009)
[14] Magedanz, T., de Gouveia, F.C.: IMS–the IP Multimedia System as NGN Service Delivery Platform. e & i Elektrotechnik und Informationstechnik 123, 271–276 (2006)

ONIT Workshop 2010

Session 3: Prototyping and Interoperability

Prototyping of an Interconnection Border Gateway Function (IBGF) in Next Generation Networks (NGN) Using Open Source Software Tools

Stephan Massner and Michael Maruschke

Institute of Telecommunication and Information Technology
Hochschule für Telekommunikation Leipzig (HfTL) Gustav-Freytag Str. 43-45,
Leipzig, 04277, Germany
Phone: +49 (0) 341 3062-238; Fax: +49 (0) 341 3062-245
{massner,maruschke}@hftl.de

Abstract. This paper illustrates both the detailed theoretical setup of the Interconnection Border Gateway Function (IBGF) and its technical implementation using Open Source Software components. Moreover, a model is proposed to handle different traffic classes according to their corresponding Quality of Service (QoS) requirements while also forwarding "best effort" data traffic. The functionality is verified by evaluating data streams with varying QoS parameters, which are identified by an adequate packet marking method.

Keywords: Next Generation Network (NGN), Interconnection Border Gateway Function (IBGF), IP-Multimedia Network, Resource and Admission Control Subsystem (RACS), IP-Multimedia Subsystem (IMS), Quality of Service (QoS).

1 Introduction

In Next Generation Networks (NGNs), IBGFs are used to connect IP-Multimedia Subsystem (IMS) networks and IP Multimedia networks on the transport layer, utilizing transport functionality at network borders. Its tasks are specified in standards by the European Telecommunications Standards Institute - Telecoms & Internet converged Services & Protocols for Advanced Network (ETSI-TISPAN) [1][2][3]. These may vary in descriptions by the 3rd Generation Partnership Project (3GPP). The IBGF has a horizontal interface (Ds) to the transport layer below the IMS network, and another to the transport layer of other IP-Multimedia networks (Iz). Between both transport layers the IBGF manages media streams to meet given QoS parameters of forwarded data. Utilizing the vertical interface (Ia), the IBGF is managed by transport control functionalities, like the Service Policy Decision Function (SPDF) located in the Resource and Admission Control Subsystem (RACS).

T. Magedanz et al. (Eds.): TridentCom 2010, LNICST 46, pp. 739–752, 2011.

2 General IBGF Description

2.1 IBGF Tasks

The standardized[3] tasks an IBGF fulfills are:

- opening/closing gates
- allocating/translating IP addresses/port numbers
- IPv4/IPv6 interworking
- hosted NAT traversal
- packet marking
- resource allocation/bandwidth reservation
- policing
- QoS/usage metering
- transcoding
- detection of inactive bearer connections
- specific call-independent procedures
- BGF overload control

With its functionality, the IBGF represents a domain with integrated service classes, due to the fact that QoS resources have to be provided for transmitted data streams (See Figure 1). The control of these QoS resources is managed by functionalities on the Transport Control Sublayer, in this case the SPDF[1]. It sends information to the IBGF about the allocation of QoS resources, data stream handling (classification, Topology Hiding, marking, policies,...), and rules about traffic classes. The IBGF on the other hand informs the SPDF about currently used, reserved, and free resources (QoS/usage), based on individual or bundled streams. Furthermore, it provides information about currently configured as well as policies used.

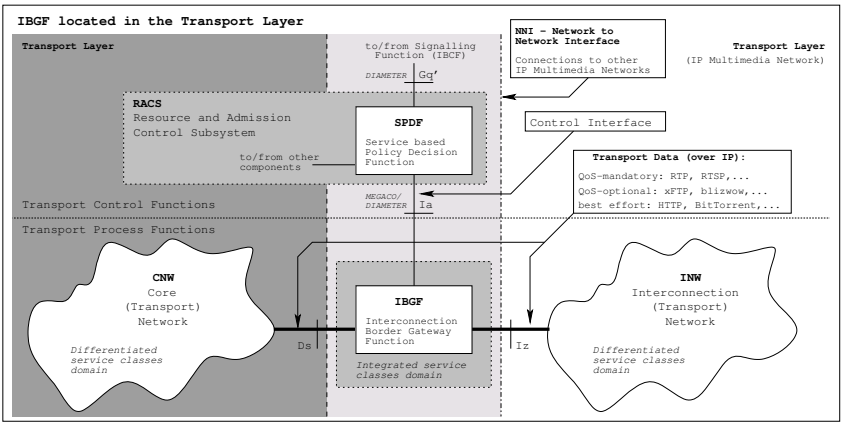


Fig. 1. IBGF located in the Transport Layer

The IBGF is located between domains with differentiated service classes, because it handles data streams according to their classification, identified by their individual marking. The latter is also edited by the IBGF according to guidelines.

2.2 IBGF Interfaces

Vertical Interface to the Transport Control Sublayer (Ia). The connection to the RACS[2] can basically be divided into two different types. In the first case, a direct connection can exist between the SPDF and the IBGF[4]. In the second case, multiple SPDFs and IBGFs can communicate with each other. The latter requires a reporting framework as specified in ETSI TR 182 022[5].
If a point to point connection exists between the SPDF and the Transport Control Sublayer (first case), information to be exchanged can generally be categorized into:

- Policy Push, Audit Request (SPDF to IBGF) and
- Usage Metering, QoS-Reporting (IBGF to SPDF)

However, in the second case, this data transfer is realized by introducing a sublayer and thus changing the topology. A functional division into Reporting Source, Reporting Collector, and Reporting Sink is necessary, as illustrated in Figure 2. Thereby, entities appear both as transport sources and sinks in the Transport Control Sublayer and Transport Process Sublayer, depending on the flow of information. The entity inside the sublayer acts as a reporting controller, which aggregates, filters, and delivers received data to the destination. Moreover, it can introduce modifications, if this meets the destination's requirements.

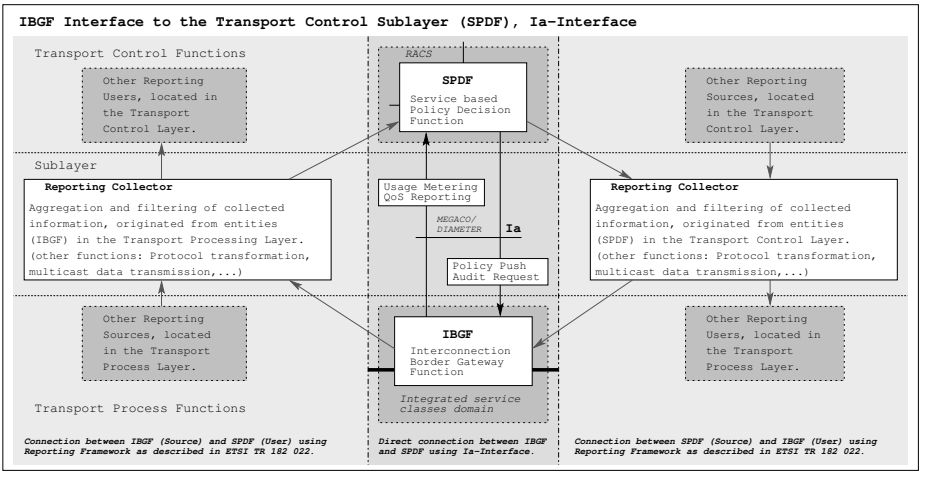


Fig. 2. Ia-Interface

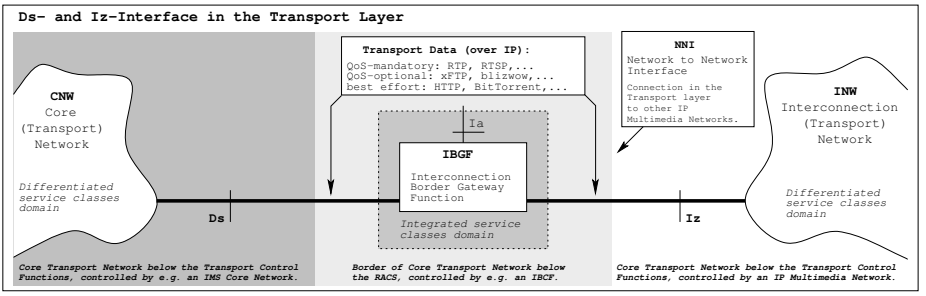


Fig. 3. Ds- and Iz-Interface

Horizontal Interface to the Transport Process Sublayer (Ds and Iz). Both interfaces terminate traffic from IPv4 and/or IPv6 networks (See Figure 3). Here, data streams from different traffic classes are received and sent. Upon receiving, data packets are sorted into traffic classes, which are distinguished by QoS characteristics (roughly: QoS, non-QoS). Packets, for example, can be classified in the following way:

- non-QoS: "best effort" - e.g.: Hypertext Transfer Protocol (HTTP), BitTorrent, ...
- QoS: "QoS-mandatory" - e.g.: Real-Time Transport Protocol (RTP), Real-Time Streaming Protocol (RTSP), ...
- "QoS-optional" - e.g.: x File Transfer Protocol (xFTP), blizwow, ...

QoS characteristics for classifications can be structured into single criteria (parameters) like jitter, delay, bandwidth, packet loss or any combination of the above. Moreover, a dynamic division of different classes into subclasses is performed, depending on respective parameter values as deciding criteria. Therefore it is possible to allocate resources dynamically, which affect a respective stream dedicated to a QoS class. Policing is applied depending on each QoS class and is not further divided into subclasses (e.g.: Stochastic Fairness Queuing (SFQ), Class Based Queuing (CBQ) with / without Token Bucket Filter (TBF), etc.). QoS marking is also executed according to corresponding QoS classes, which is identical for respective subclasses.

2.3 Detailed Description of IBGF Function Blocks

The IBGF functional blocks (see Figure 4) are divided into:

① **IPv4/IPv6 termination** of the inbound data stream, including the termination of layers 1-3.
② **Gate** for the inbound stream and its examination according to **filter** criteria like origin (source address : source port) and destination (inbound interface; destination address : destination port). For "best effort" data streams, dedicated port ranges can be reserved (e.g. HTTP) which are not managed

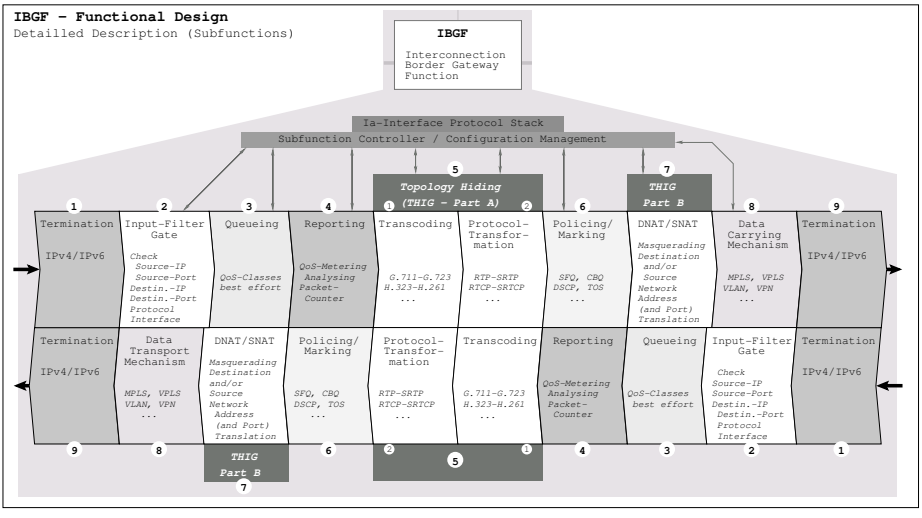


Fig. 4. IBGF - Functional Design

separately by the Transport Control Sublayer, but instead follow common rules. Thus, "best effort" traffic is treated with network neutrality principles. Both the process of blocking and releasing the corresponding gate is subject to policy guidelines (bandwidth limitation etc.) set by the SPDF.

③ **Queueing**, i.e. classification of inbound data streams into corresponding QoS classes and their respective subclasses according to QoS requirements.

④ **Reporting and measurement** of current QoS values, including jitter, packet loss, used bandwidth, current delay, packet counting (data volume) for accounting purposes, and determining the current duration of an existing connection. Moreover, measurement results and connection information can be divided into general and QoS connection data.

⑤ **Topology Hiding** (part A) includes both **transcoding** (5.1) and **protocol transformation** (5.2). For optional transcoding, the IBGF acts as a media gateway, whereas it translates e.g. the media codec G.711 to G.723, or H.323 to H.261. Optional protocol transformations include the conversion from RTP to Secure Real-Time Transport Protocol (SRTP) or Real-Time Transport Control Protocol (RTCP) to Secure Real-Time Transport Control Protocol (SRTCP).

⑥ **Policing and marking** is applied by the IBGF by first reading all queues of the corresponding QoS classes and subclasses with their respective adjusted algorithms. These different algorithms can be applied in a parallel and/or serial, as well as interdependent and/or independent fashion. Markings applied to fields like the Differentiated Service Code Point (DSCP)[6], the Type of Service (TOS)[7] etc. depend on their corresponding classes.

⑦ **Topology Hiding** (part B) covers procedures for Destination Network Address (and Port) Translation (DNA(P)T) and/or Source Network Address

(and Port) Translation (SNA(P)T), to hide data stream origins and set the next route as the destination accordingly.

⑧ **Data carrying mechanism** describes the grouping of similar streams, for example with similar destinations and QoS parameters, into data stream bundles. These can then be transmitted through Multi Protocol Label Switching (MPLS), Virtual LAN (VLAN), Virtual Private Network (VPN) or with Virtual Private LAN Service (VPLS).

⑨ **IPv4/IPv6 termination** of the outgoing data stream, including layer 1-3 termination.

3 Prototyping and Implementation

The IBGF implementation (see Figure 5) covers the following function blocks:

❶ The position of interfaces, whether considering the Ds- or Iz-interface, is irrelevant to this case. Here, the IPv4/IPv6 termination is applied to an inbound unidirectional data stream of any arbitrary session, consisting of a RTP and RTCP data stream.

❷ Thereafter, individual data packets are checked by examining the origin (source IP : source port), destination (destination IP: destination port), receiving interface, and the protocol used. In case the determined data matches the filter rules, the specified data packet is allowed to pass the gate. This is realized by utilizing the netfilter framework[8].

❸ For further processing, data packets are sorted into their corresponding QoS classes. In this case, a higher prioritized class (Subclass A2) is used for the RTCP data stream and a lower prioritized class (Subclass n1) is used for the RTP stream. The RTCP stream's packet loss should be close to "0" at the expense of the RTP data stream. This was realized by using functionalities already integrated inside the Linux Kernel.

❹ The measurement of bitrate, delay, jitter, and packet loss of both data streams is executed separately and independently for each subclass utilizing functionalities from the netfilter framework. If a substantial bitrate exceedance is detected, by overstepping a predetermined bitrate level, foremost the RTP data stream will be limited.

❺ Topology Hiding (Part A) is implemented separately. For this purpose, the RTP data stream is transcoded first (part 5.1), by converting the media codec type from G.711 to G.723. This requires the termination of the received RTCP data stream and subsequent generation of a new RTP data stream (outgoing termination). Moreover, the RTCP data stream has to be modified. The protocol transformation of the RTP and RTCP which follows (part 5.2) includes embedding in Secure Shell (SSH).

❻ Policy enforcement for data streams of different traffic classes is carried out with different algorithms, which are adjusted to their corresponding QoS. The subclass of a certain traffic class and the "best effort" class are treated with SFQ. Traffic classes amongst each other are processed by CBQ, whereas

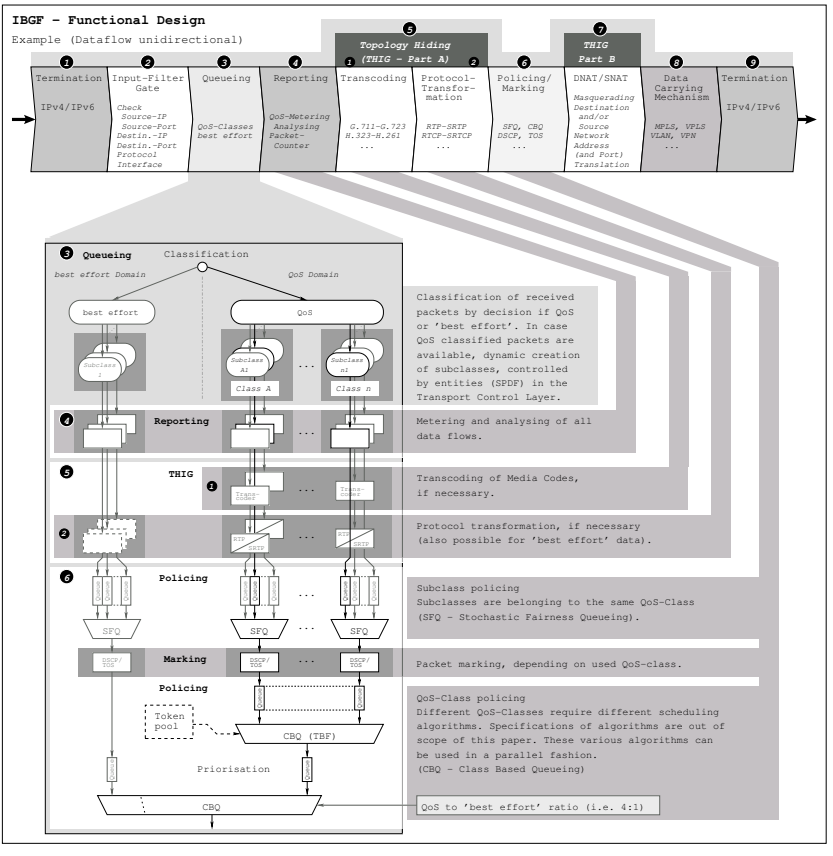


Fig. 5. IBGF - Functional Design Example

a TBF is implemented for quantity management of outgoing QoS data traffic. A final utilization of CBQ considers the division of 20% of all resources for 'best effort', and 80% for QoS data. The marking of data streams of a traffic class is accomplished after combining subclasses by manipulating the Differentiated Service (DS) and TOS fields respectively. Therefore, data packets can be treated according to their QoS in the subsequent differentiated service classes domain. Also, the implementation is carried out with functionalities of the netfilter framework.

❼ At this point, masquerading of both data streams is realized, using the netfilter framework. For this purpose, source and destination descriptions are translated.

❽ In a separate implementation, traffic grouping is carried out for the mutual transport over a MPLS network. To accomplish this, data streams with similar QoS and a common destination are combined.

❾ Finally, the outgoing IPv4/IPv6 termination is executed for the unidirectional data stream.

4 QoS-Measurements

4.1 Standardization

Several parameters have been defined by the International Telecommunication Union - Telecommunication Standardization Sector (ITU-T) describing boundaries acceptable by real time media transport in IP networks. Definitions mainly focussed on are:

- **IPPR** IP Packet Rate (proportional to Bandwidth): IPPR (Packet · s-1), specifies the total number of IP packets during a specified time interval divided by the time interval duration[9].
- **IPTD** IP packet Transfer Delay: IPTD (s), specifies the time between two events at time t_1 (ingress event) and t_2 (egress event), where both events corresponds to one IP packet. Two conditions have to be fulfilled, where T_{max} defines the maximum Transfer Delay Time[9]: $(t_2 > t_1)$ and $(t_2 - t_1) \leq T_{max}$.
- **IPDV** IP packet Delay Variation (Jitter): IPDV (s), specifies the variation of the IPTD. Different methods of calculating IPDV are known and described in[9][10][11]. It is irrelevant which method will be choosen, because the range given by absolute values between min. and max. IPTD is independent.
- **IPER** IP packet Error Ratio: IPER (1), specifies the number of total errored IP packets divided by the sum of total successful transfered IP packets and the errored IP packets. Both numbers must refer to the same time interval[9].
- **IPLR** IP packet Loss Ratio: IPLR (1), specifies the number of total lost IP packets divided by the total transmitted IP packets. Both ratios must refer to the same time interval[9] [12].

4.2 Measurement

The measurement of achievable QoS parameter demands on an arrangement in several parts providing a set of functions needed by described measuerement methods (see Figure 6). In the following, all measured QoS parameters of the described prototypical IBGF are named and exemplified:

- IPPR: Throughput using an increasing bitrate and Type of Service (TOS) marked IP packets
- IPTD: Delay between each incoming and outgoing IP packet, using an identifier contained in the payload
- IPDV: Delay variation between each measured delay of incoming and outgoing IP packets
- IPER: Comparision of header and payload of sent and received IP packets including the same content
- IPLR: Counting dropped IP packets by checking if sent unified identicable IP packet was received during the measure time or not

As described in IPPR measurement, an increasing bitrate IP flow was used to test several conditions: underload, limit load and overload. Achieved values have been selected to these operating states and evaluated as limited to underload and limit load condition. These different operating states can be defined as follows:

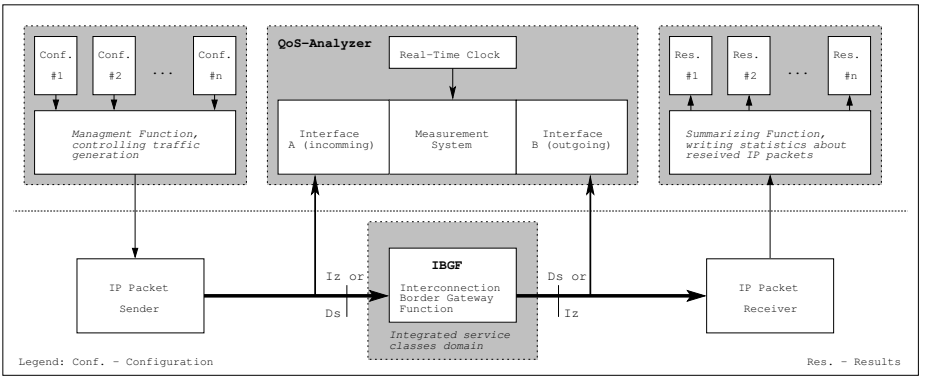


Fig. 6. IBGF - Test Setup for QoS Measurements

- Underload: Incoming IP traffic (λ) is lower than max. outgoing IP traffic (μ); $\lambda < \mu$
- Limit Load: Incoming IP traffic (λ) is equal to max. outgoing IP traffic (μ); $\lambda = \mu$
- Overload: Incoming IP traffic (λ) is higher than max. outgoing IP traffic (μ); $\lambda > \mu$

Each particular measurement point has been calculated using five measurement points preventing random errors. The results for these five related test series were obtained from similar test executions. Furthermore all tests have been done under the condition that impacts based on processing power, memory, operating system etc. have no bearing on IP packet processing. Based on five specified traffic classes in ITU-T standard a combination into three traffic classes has been done on condition that lower QoS classes are included in next upper QoS class, that means:

- class 0 contains class 0 and class 1 (Upper bound on the mean IPTD delay of class 0 is $100ms$; class 1: $400ms$),
- class 2 contains class 2 and class 3 (Upper bound on the mean IPTD delay of class 2 is $100ms$; class 3: $400ms$),
- class 4 contains class 4 and class 5 (Upper bound on the mean IPTD delay of class 4 is $1s$ and upper bound on the packet loss probability of class 4 is $1 * 10 \exp -3$; class 5 has not specification for IPTD, IPDV, IPLR adn IPER).

4.3 Measurement Results

All named parameters like IPPR, IPTD, IPDV, IPLR and IPER have been measured. Due to the fact that all tests resulted in an IPER of zero, no further diagrams containing the error ratio are figured out. In terms of comparision different tests related to various physical links, i.e. 10/100/1000MBit/s bandwidth

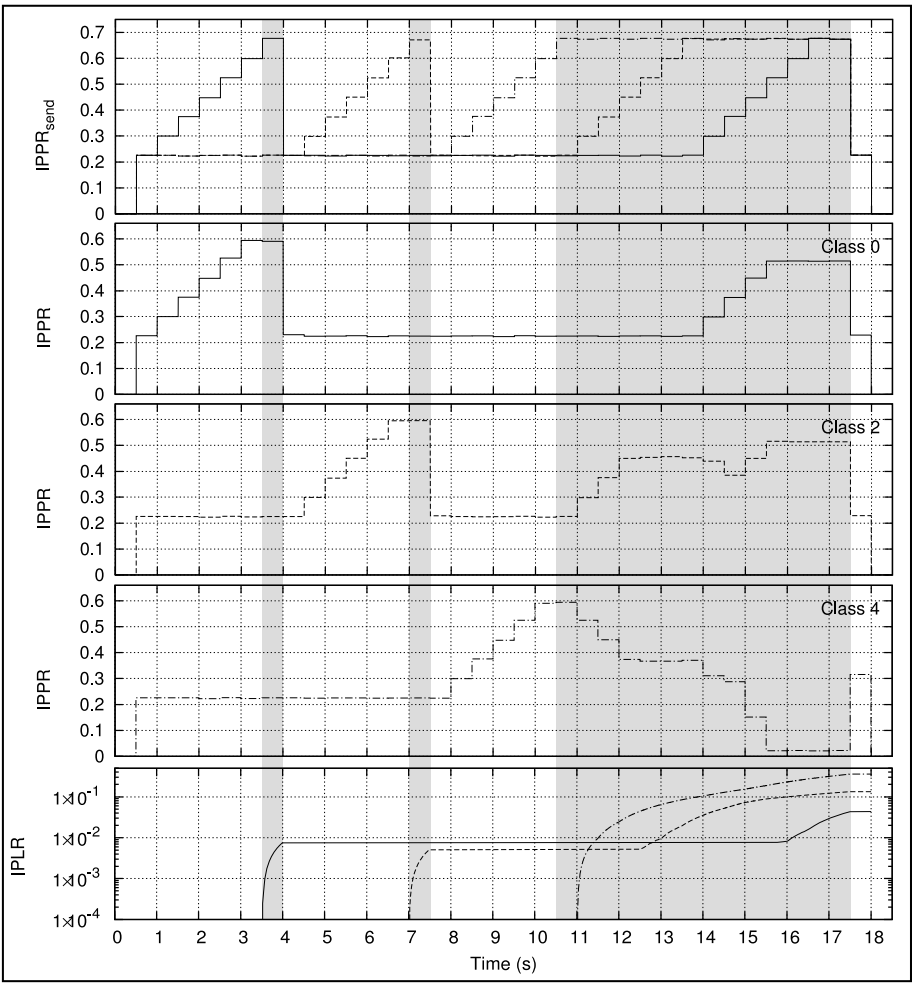


Fig. 7. IPPR and IPLR Measurements (overload condition in gray highlighted parts) Note: IPLR results are cumulated

and diverse logical links from 1MBit/s to 250MBit/s bandwidth no dependencies have been identified. Seeing that a detailed diagram for each test does not offer more information, all IPPRs have been normalized. All QoS classes named class 0, 2 or 4 are exactly the same as described in ITU-T standard[13]. Based on used IP traffic flows, test results are drawn seperatly and combined, distinguishable in line types. In addition all sent IP traffic flows ($IPPR_{send}$) are shown above in Figure 7. For all diagrams, time intervals are the same and overload conditions are marked by gray highlighted areas.

Using the normalized view of incoming and outgoing IP traffic limit load and subsequently overload conditions are readily identifiable using IPLR by evaluating too (see Figure 7). While operating condition is below overload status,

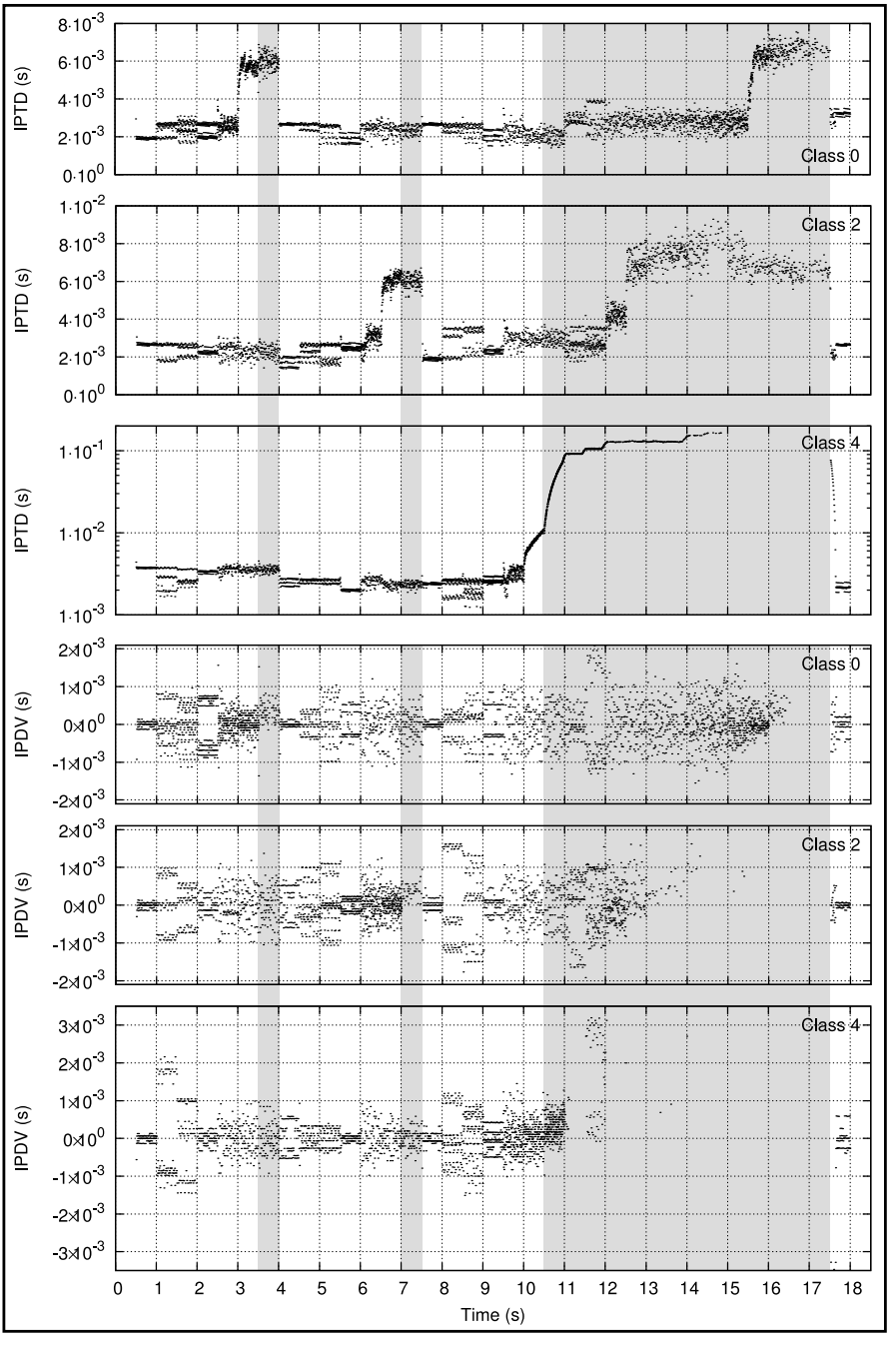


Fig. 8. IPTD and IPDV Measurements (overload condition in gray highlighted parts)

Table 1. QoS Class 0 - Measurement results (underload and limit load)

Parameter	Maximum Value from Standard	Measured Value	Evaluation
IPTD	3ms	$< 3.5ms$ (underload) $< 6.9ms$ (limit load)	(pass)
IPDV	3ms	$< 2.1ms$	pass
IPER	$1 * 10 \exp -4$	0	pass
IPLR	$1 * 10 \exp -3$	$0 * 10 \exp -3$	pass

Table 2. QoS Class 2 - Measurement results (underload and limit load)

Parameter	Maximum Value from Standard	Measured Value	Evaluation
IPTD	3ms	$< 4.0ms$ (underload) $< 6.8ms$ (limit load)	(pass)
IPDV	unspecified	$< 2.1ms$	pass
IPER	$1 * 10 \exp -4$	0	pass
IPLR	$1 * 10 \exp -3$	$7.3 * 10 \exp -4$	pass

Table 3. QoS Class 4 - Measurement results (underload and limit load)

Parameter	Maximum Value from Standard	Measured Value	Evaluation
IPTD	64ms	$< 12ms$	pass
IPDV	unspecified	$< 3.2ms$	pass
IPER	$1 * 10 \exp -4$	0	pass
IPLR	$1 * 10 \exp -3$	$9.2 * 10 \exp -4$	pass

all IP traffic flows will be scheduled as given by traffic policing rules related to each IP traffic class. In doing so, no packet loss has been detected exceeding maximum values from ITU-T standardisation[13]. In case an overload condition occurs, packets have been lost. On the basis of recorded IPLR, it is evident that class based queueing and sceduling algorithms have been implemented as well as configured policing rules in terms of traffic class hierarchy.

Commonly obtained results about IPTDs shows the IP packet processing time does not exceed considerable given maximum values in underload condition. Other operating states cause in significant exceedings of IPTD maximum values compared with underload condition (see Figure 8)[13]. Minor deviations about $0.5ms$ to $1.0ms$ have to be evaluated with reference to IPDV results. Based on the worst case both values, maximum IPTD ($3ms$) and maximum IPDV ($3ms$) results in an interval of $3ms + 3ms = 6ms$, a differentiated evaluation should be done. So an under-usage of IPDV enables an exceeding of IPTD in theory provided that in doing so the sum of IPTD and IPDV does not exceed $6ms$. Regarding measured results this theoretical fact can be applied here. All results marked by brackets in table 1, 2 and 3 are evaluated involving the limitation described above. Further collected data about IPDV results have been analysed. Thereby an under-usage for all traffic classes has been found evidently (see

Figure 8). Summarized for all measured traffic classes as defined in ITU-T table 1, 2 and 3 contains maximum values and measured values added by an evaluation. Contained maximum values related to given QoS class are assigned to a Border Gateway Function equates to the presented IBGF prototyp. Partially, table elements are splitted to seperate different operating states like underload and limit load condition. Only underload operation state has been evaluated expecting work load states were been lower than limit load state.

5 Summary

By evaluating different processed data streams, the prototypical correctness was proven by using a range of Open Source Software tools (NetFilter, TrafficControl). Therefore, basic functions required by the standard are realized. Moreover, by implementing points 5.1 and 5.2, special functions of the IBGF for converting transported data were successfully demonstrated. Further studies and research are still necessary for the data carrying mechanism illustrated in point 8. The communication over the Ia-Interface was realized by means of an Extensible Markup Language Remote Procedure Call (XML-RPC) on a functional level, and terminated by a custom replication (protocol generator and receiver) instead of the SPDF. Achieved measurement results shows detailed that the presented IBGF prototyp meet the requirements given by ITU-T standardisation related to the Border Gateway Function. Minor changes in results of IPTD are acceptable due to under-usage of IPDV because the interval sum does not exceed theoretical maximum value.

6 Future Prospects

The implementation for the illustrated structure of the IBGF was carried out using Open Source Software components. However, implementing the Ia-interface according to standards[3] for the communication with the SPDF is still outstanding. A focus for further studies is benchmarking the IBGF with respect to the expected data throughput of each corresponding operation and subsequent optimization of function blocks, which were illustrated here. Moreover studies related to optimizing control parameters of traffic control should be aimed from the viewpoint of existing media codecs capsulated in IP packets. With the prototypical implementation of the IBGF it is possible, in principle, to realize a carrier grade conform interconnection of IMS networks and IP Multimedia networks on the transport layer.

Acknowledgement

Supports from students have contributed to prepare this presented IBGF prototype. The authors would like to thank Stephan Händel for implementing the IBGF using OSS, Peter Hermann for testing performance and desired policing

rules related to QoS parameters and Michael Erler for supporting test tools. We would like to thank the Institute of Telecommunication and Information Technology at the HfTL for making test and measurement equipment available to us.

References

1. ETSI-TISPAN ES 282 001: NGN Functional Architecture v3.3.0 (February 2009)
2. ETSI-TISPAN ES 282 003: Resource andAdmission Control Sub-System (RACS): Functional Architecture v3.3.0 (February 2009)
3. ETSI-TISPAN ES 283 018: Resource and Admission Control: H.248 Profile for controlling Border Gateway Functions (BGF) in the Resource and Admission Control Subsystem (RACS); Protocol specification v2.7.1 (September 2009)
4. ETSI-TISPAN TS 183 048: Resource and Admission Control System (RACS); Protocol Signalling flows specification; RACS Stage 3 v1.4.0 (June 2008)
5. ETSI-TISPAN TR 182 022: Architectures for QoS handling v2.0.0 (December 2007)
6. IETF RFC 2474 (Kathleen Nichols and Steven Blake and Fred Baker and David L. Black): Definition of the Differentiated Service Field (DS Field) in the IPv4 and IPv6 Headers (December 1998)
7. IETF RFC 1349 (Philip Almquist): Type of Service in the Internet Protocol Suite (July 1992)
8. netfilter Framework (October 2008), `http://www.netfilter.org/`
9. ITU-T Y.1540: Internet protocol data communication service IP packet transfer and availability performance parameters (November 2007)
10. IETF RFC3393: IP Packet Delay Variation Metric for IP Performance Metrics, IPPM (2002)
11. IETF RFC3550: RTP: A Transport Protocol for Real-Time Applications (2003)
12. IETF RFC3357: One-way Loss Pattern Sample Metrics (2002)
13. ITU-T Y.1541: Network performance objectives for IP-based services (February 2006)

Interoperability among Different IMS Cores

Fernando Ortigosa[1], Sergio Fernandez[1], Andres Marin[2], Davide Prosepio[2], Borja Iribarne[3], Itziar Ormaetxea[4], Ivan Kotuliak[5], Tomas Kovacik[5], Eugen Mikoczy[6], Timo Lahnalampi[7], Timo Koski[8], Piritta Hakala[8], Janne Ilitalo[9], and Juha Teräslahti[9]

[1] Ericsson Network Services S.L. c/ Hungría, 8
28943, Fuenlabrada, Madrid, Spain
{Fernando.ortigosa,Sergio.Fernandez}@ericsson.com
[2] Universidad Carlos III de Madrid, Spain
amarinit.uc3m.es, dade.pro@gmail.com
[3] Telefonica Investigacion y Desarrollo, Spain
iribarne.borja@gmail.com
[4] Software Quality Systems, Innovalia Group
iormaetxea@sqs.es
[5] Slovak University of Technology in Bratislava, Slovakia
ivan.kotuliak@stuba.sk, tokosk16@gmail.com
[6] Slovak Telekom, Slovakia
eugen.mikoczy@t-com.sk
[7] DIMES, Finland
timo.lahnalampi@dimes.fi
[8] University of Turku, Finland
{timo.koski,piritta.hakala}@utu.fi
[9] Octopus, Finland
{Janne.ilitalo,juha.teraslahti}@octo.fi

Abstract. The IMS as core for operators' network is imminent. However, the interconnection is still an issue on basic call layer, but also on services layer. In this article, we present the NetLab project approach, which has the aim to create a testing environment for research and development of new services and applications based on the IP Multimedia Subsystem (IMS). The results of interconnection tests between several IMS platforms are also provided here.

Keywords: IMS, IPTV, NGN, IM.

1 Introduction

NetLab project aims to sustain research and experimentations that will ascertain the convergence and interoperability of different test beds, protocol variants and services based on IP Multimedia Subsystem (IMS). The choice of IMS is considered a key factor by the project for obtaining a sustainable and generic interconnection proof testbed where the user can discover and select the network and services used to perform his

T. Magedanz et al. (Eds.): TridentCom 2010, LNICST 46, pp. 753–764, 2011.

tests. Particular emphasis is placed on interoperability of the test beds, the interconnection and sharing of software tools, the experimentation and validation of protocols and services, and in providing trusted access to services.

IMS was originally designed by the standardization body "3rd Generation Partnership Project" (3GPP). However standards may slightly differ in their implementation, causing some interoperability issues among different manufactures.

In order to be able to accomplish these objectives, various IMS cores have been used for testing. The tests have been done from different locations at different countries. Each partner has set up a lab environment to connect to the Netlab network. Two technologies have been used in order to deploy this network:

- A virtual private network (VPN) which does not need any special infrastructure but a simple connection to the Internet
- GEANT. This is a high bandwidth network for research and education purposes.

As long as Netlab project pretends to create a testing environment for research and development of new services and applications based on the IMS the first set of tests have been oriented towards the interoperability of different IMS cores and IMS clients.

Netlab test bed is composed of three vendor-different IMS cores from:

- Nokia IMS core
- Ericsson SDS
- Fokus OpenIMS core

Netlab consortium has dedicated their efforts to study these multimedia platforms and their interoperability. Every partner has been able to connect to all the available IMS cores and test different services from them.

The article is organized as follows: Section 2 describes interconnection of IMS cores and provides further details. Section 3 deals with test detailed description and also with their evaluation. This is the core section of whole article. Concluding remarks and open question for future work are given in Section 4.

2 Network Interconnection

The GÉANT project was a collaboration project between 26 National Research and Education Networks representing 30 countries across Europe, the European Commission, and DANTE. Its principal purpose was to develop the GÉANT network - a multi-gigabit pan-European data communications network, reserved specifically for research and education use. The project also covered a number of other activities relating to research networking. These included network testing, development of new technologies and support for some research projects with specific networking requirements. This European Network is continuously being increased on size and bandwidth features.

The VPN is a network that is constructed using the Internet as the medium for transporting data. This system uses encryption and other security mechanisms to ensure that only authorized users can access the network and that the data cannot be intercepted. The technology we are using for implementing this virtual private network is called “transport layer security (TLS)” an application layer technology which allows tunneling an entire network's traffic over the Internet. In particular we are using “openVPN” which is an open source solution available for general usage.

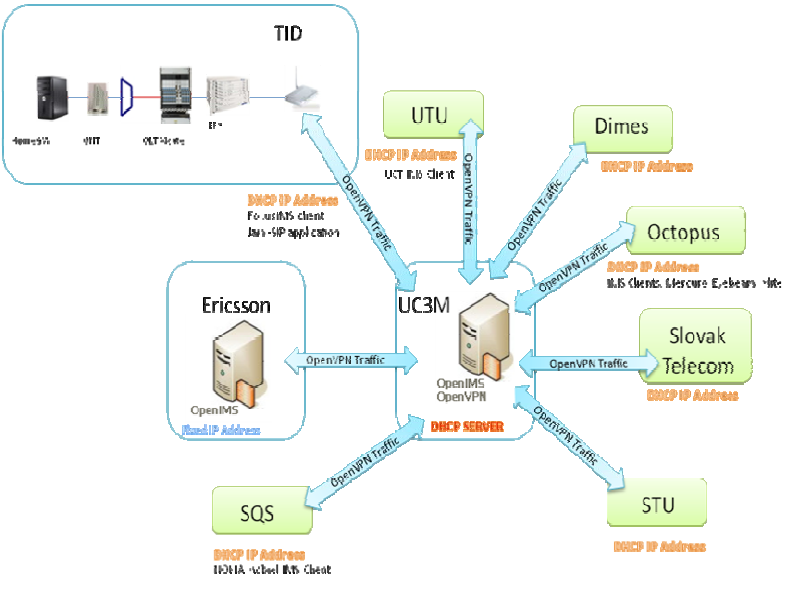


Fig. 1. Netlab interconnection

2.1 Netlab IMS Cores

Netlab project includes international partners participating in testing IMS cores interconnection in all network layers. IMS networks cores are deployed by different vendors (OpenIMS, Ericsson and Nokia).

OpenIMS. The Open IMS Core is an Open Source implementation of IMS core functions. It has been developed by the Fraunhofer institute and it is composed of IMS Call Session Control Functions (CSCFs) and a lightweight Home Subscriber Server (HSS), which together form the core elements of all IMS/NGN architectures as specified today within 3GPP, 3GPP2, ETSI TISPAN and the PacketCable intiative. The four components are all based upon Open Source software (e.g. the SIP Express Router (SER) or MySQL). This core is deployed at UC3M and STUBA cores. Slovak partners provide their NGN lab for Netlab tests. OpenIMS core is installed in a virtual machine (Sun Virtual Box environment). It has 512 MB of RAM memory reserved for the virtual machine and a Pentium IV at 2.4 Ghz. The operating system is Ubuntu Jaunty.

Ericsson SDS (Service Development Studio)**.** This is a emulation for the Ericsson IMS core. SDS is a tool for development and end-to-end testing of both the client and server side of new convergent all-IP (IMS) applications. SDS contains a standards- based IMS network simulator with communication services (CoSe) emulators. SDS supports clients and devices for Mobile, Fixed Broadband, and WLAN access. It uses Java community common practices and de-facto standards and provides high-level APIs to hide the network and terminal device complexity from the designer. This core is provided by Ericsson. This IMS core is installed in a virtual machine (VMware environment). It has 512 MB of RAM memory reserved for the virtual machine and a Pentium IV at 2.4 Ghz. The operating system is Windows XP.

Nokia IMS CORE. Nokia release 2.0 is based on Nokia FlexiServer Platform. The platform supports high availability and load balancing for applications and services running on the platform. Nokia IMS has a static subscriber capacity of 500 000 and is able to process dynamically 10 000 SIP messages per second. The Nokia IMS solution is implemented according to 3GPP/3GPP2 Rel.6 IMS specifications and ETSI TISPAN architecture. Software used: Nokia release 2.0 IMS consists of Connection Processing Server (CPS) and Nokia IP Multimedia Register (IMR). Nokia CPS provides the Call State Control Function (P/I/S-CSCF) functionalities and IMR provides HSS functionality. Hardware used: Both CPS and IMR have been built on top of fault tolerant FelxiServer 3 platform. FlexiServer 3 platform uses FlexiServer Blade hardware which provides flexible and scalable platform for achieving carrier grade availability. Each server blade has two 1.6 GHz Intel Pentium 4 Xeon processors, memory and back panel interfaces like Ethernet and FC-AL (Fiber Channel Arbitrated Loop). The PMC card has two 1 Gbps Ethernet ports for communication to the outside.

3 Tests

Tests described in this article have been completed in environment of interconnected IMS cores using VPN technology. However, involved universities have been interconnected through GEANT network. OpenIMScore was located at Leganes, Madrid and in Bratislava, Slovakia. SDS (Ericsson IMS core) was located at Fuenlabrada, Madrid. Nokia IMS core was located at Oulu, Finland.

The tester has remote access to all of the IMS cores in order to be able to retrieve log information. The remote access is provided by Remote Desktop connection for Windows XP operating systems and VNC technology in case of Linux based cores.

The proof of correct interconnection of IMS cores has been done using following scenarios:

- Registration of client in different labs
- Voice session establishment
- Instant messaging service provisioning
- Presence service provisioning
- Roaming scenario.

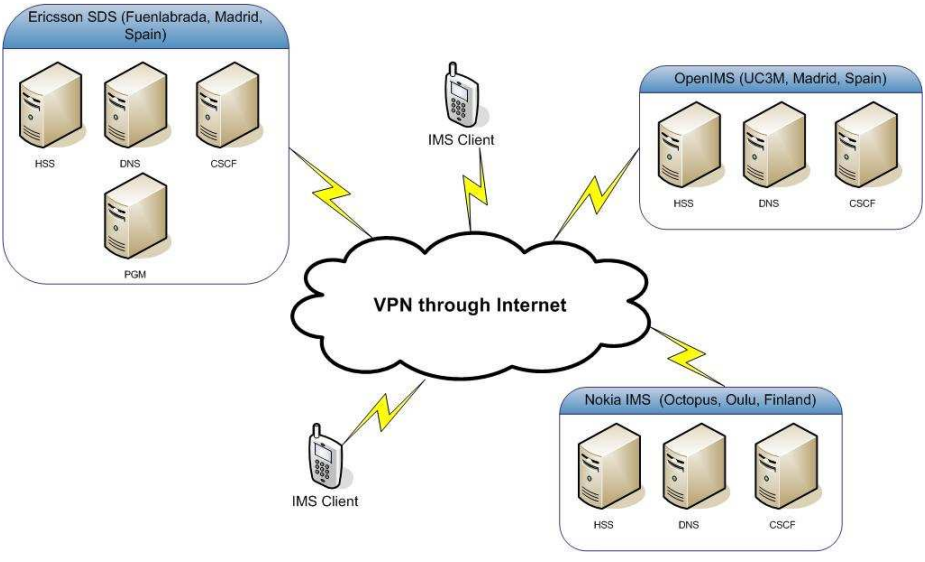


Fig. 2. Netlab laboratory

3.1 IMS Registration

3.1.1 Test Objective

IMS-level registration is the procedure where the IMS user requests authorization to use the IMS services in the IMS network. The IMS network authenticates and authorizes the user to access the IMS network.

IMS-level registration is accomplished by a SIP REGISTER request. A SIP registration is the procedure whereby a user binds his public URI to a URI that contains the host name or IP address of the terminal where the user is logged in. Unlike regular SIP procedures, registration with the IMS is mandatory before the IMS user's terminal can establish a session.

The IMS registration procedure uses a SIP REGISTER request. However, this procedure is heavily overloaded in the IMS, in contrast to SIP registration, and this overload is for the sake of fulfilling the 3GPP requirement of a minimum number of round trips.

Two algorithms have been used for registering: AKAv1-MD5 and basic username/password.

3.1.2 Test Procedure

To test registering process in Netlab we propose to register a user in IMS cores included in Netlab project – NGNLAB (SK), Octopus (FI), Ericsson (ES), UC3M (ES) and SQS.

User registers from different locations (from different labs of Netlab partners) using different IMS User Agent applications: UCT IMS client, X-lite, Monster and Mercuro.

3.1.3 Test Results

Following table shows the fact that most of the registration tests have been successful. We can see the different IMS clients and the different cores tested.

Table 1. IMS registration results

Client/Core	*OpenIMS*	*SDS*	*Nokia*	*Remarks*
X-lite	Ok (basic user/pwd algorithm)	Ok (basic user/pwd algorithm)	Ok (basic user/pwd algorithm)	Registration without two round-trip authentication.
UCT	Ok (both algorithms)	Ok (basic user/pwd algorithm)	Ok (both algorithms)	
Monster	Ok (both algorithms)	Ok (basic user/pwd algorithm)	Ok (both algorithms)	
Mercuro	Ok (both algorithms)	Ok (both algorithms)	Ok (both algorithms)	

X-lite application is pure SIP client. It supports only registration without two round-trip authentication while it does not support MD5 authentication. X-lite is not prepared for two round trip authentication which can be accomplished only by an IMS client.

UCT and Monster do not support MD5 authentication in Ericsson SDS core. The reason is that the UCT and Monster do not include *cnonce* in the authenticated request for the qop reply. qop support is optional and should not be forced by the core, as UCT and Monster simply ignore it and this backward compatibility is aspired by RFC 2617.

In case that we want to use different IMS/SIP clients for registration tests, we should use simple authentication method without two round-trip algorithm as some clients do not support complex algorithms.

3.2 Voice Call

3.2.1 Test Objective

The objective for this test is an establishment of a voice call between two IMS clients registered IMS cores deployed at different labs in different countries. We use the infrastructure being given by the VPN and the GEANT networks.

Test prerequisites

The clients need to be registered at IMS cores before starting the voice call.

3.2.2 Test Procedures

The main idea for this test is setup of a voice call through IMS cores included in Netlab project. At this testing stage a call is established always only through one of involved IMS cores (to involve several cores for voice calls we propose different test). An objective of this scenario is to repeat the voice call establishment process from different locations (different labs of different partners) using different IMS/SIP clients (UCT ims client, X-lite, Monster or Mercuro).

3.2.3 Test Results

The process of the voice call establishment through one core has been successful for all Netlab project partners' cores. The tests were successful for all chosen IMS/SIP clients. There were no problems while we tried to setup a voice call between X-lite client on both communicating sides.. Same result was achieved while using UCT IMS client, Monster and also Mercuro.

However some problems came up when we tried to establish a voice call using two different IMS clients. We could successfully test a voice call between X- lite and Monster client and vice versa. Also calls from X-lite to Mercuro and vice versa were successful.

3.3 Instant Messaging

3.3.1 Test Objective

There are two modes of operation of the instant messaging (IM) service - stand-alone instant messages or messages which are part of a session of instant messages.

Pager mode instant messaging is IM mode where message is sent as a stand-alone message not having any relation with previous or future instant messages. The name of the mode comes from the way a two-way pager works. The model is also similar to the SMS (Short Message Service) in cellular networks.

Session-based instant message is IM mode where message is sent as part of an existing session, typically established with a SIP INVITE request.

Both models have different requirements and constraints; hence their implementation is different.

We focus on pager mode instant messaging.

3.3.2 Test Procedure

At this testing stage we send and receive instant messages through each one of the IMS cores included in Netlab project. The messages are sent from one IMS core to another one.

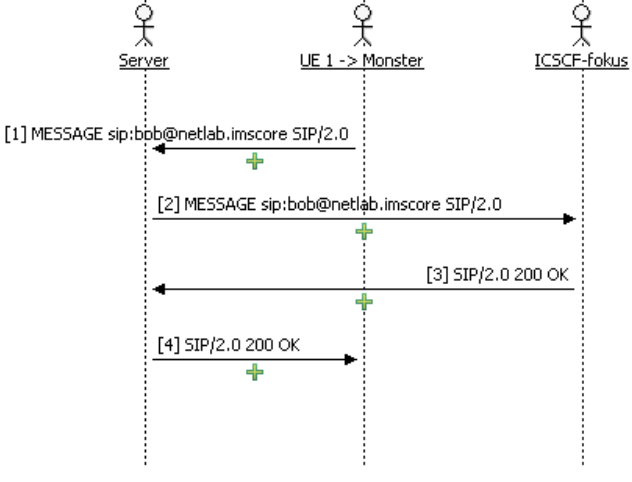


Fig. 3. IM flow diagram

Instant messages are are sent from different locations (different labs from different partners) using different IMS clients.

Test prerequisites
The clients must be registered to the IMS core before starting the session.

3.3.3 Test Results

The following table of tests results shows that all of IM tests have been successfully passed. We can see the different IMS clients and the different cores tested:

Table 2. IM results

Client/Core	*OpenIMS*	*SDS*	*Nokia*	*Remarks*
X-lite	Ok	Ok	Ok	
UCT	Ok	Ok	Ok	
Monster	Ok	Ok	Ok	
Mercuro	Ok	Ok	Ok	

Some problems were raised when trying to execute instant messaging with different IMS clients at the end of the communication. Probably because of internal implementations on the IM service within the IMS clients.

UCT crashed occasionally when messaging with Monster or Mercuro. UC3M modified the UCT IMS client code in order to avoid this kind of crashes.

3.4 Presence Service

3.4.1 Test Objective

The presence service enables any user to subscribe to the presence information of his friends, publish his own presence information or decide if he wants to provide others with his presence information. The presence interconnection assures that the presence information is exchanged through different operators' networks, assuring rich communication between end users. The objective of this test is to demonstrate how presence service works through different IMS cores.

3.4.2 Test Procedure

For this test we use the following functional entities:

- PGM (Presence and group management) - presence application server provided by Ericsson.
- Two different IMS cores - Ericsson SDS and OpenIMS.
- Two different IMS clients

- X-lite which subscribes to Ericsson SDS
- Monster which is registered at OpenIMS core

Both cores are configured to point to the same presence server.

3.4.3 Test Results

We proved that presence service worked between chosen IMS cores and clients. Following figure shows the flow of SIP messages exchanged.

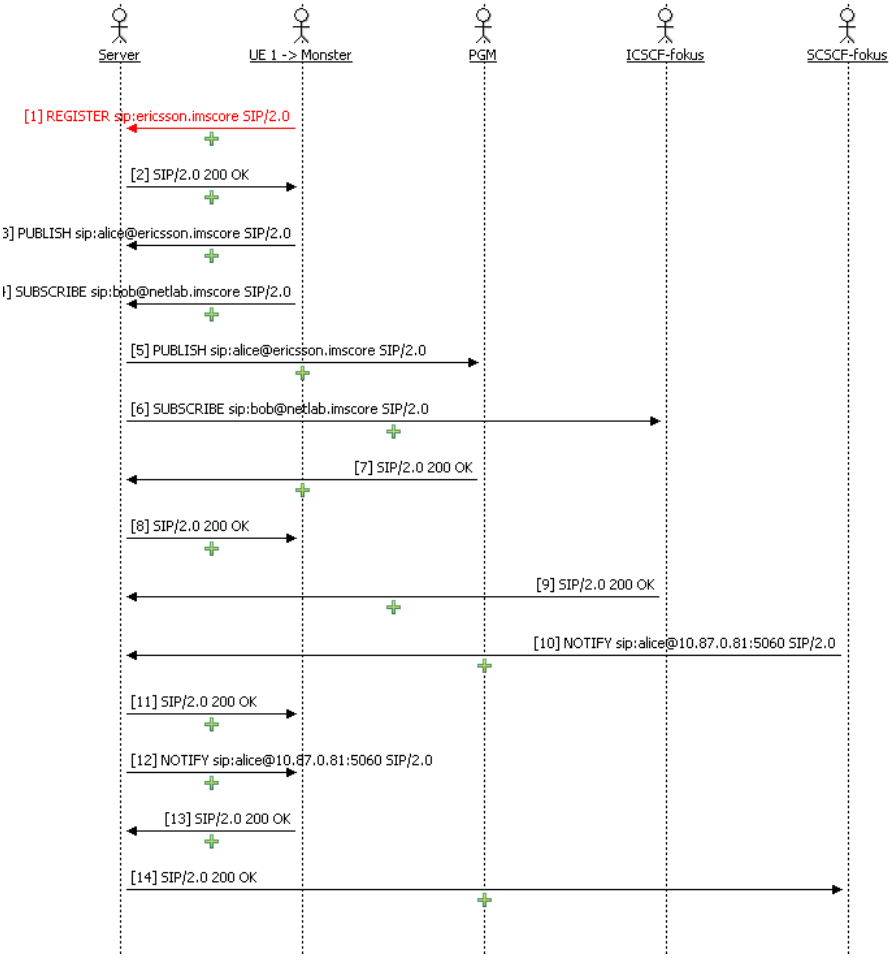


Fig. 4. Presence test flow of SIP messages

In Figure 4 we can see how a client (Alice) registered into SDS core, published her status and subscribed to the presence status of Bob (which was registered into the OpenIMS core) . At that moment Bob notified Alice about his status.

3.5 Voice Call among Different Cores

3.5.1 Test Objective
The aim of this test is to demonstrate the interconnection among different IMS cores. In order to accomplish this test we have to configure the IMS cores to be able to reach each another. We also had to configure the DNS servers from every IMS core so that they were able to reach the I-CSCF of other cores.

3.5.2 Test Procedure
In this test we use following infrastructure:

- Two different IMS cores - Ericsson SDS and OpenIMS.
- Two different IMS clients
  - X-lite which is registered at Ericsson SDS
  - Monster which is registered at OpenIMS core.

The purpose of this test is to test a voice call in both directions, from OpenIMS to SDS and from SDS to OpenIMS.

3.5.3 Test Results
The voice call was successfully established in both directions.

3.6 Roaming

3.6.1 Test Objective
The purpose of this test is to allow a user to register at visited network. This means that the subscriber could travel and connect to a different access network but he still would be able to access his home network although the operator which provides access network to him is not the one which he has contract with. We had to enable roaming and configure allowed visited networks to allow registration from a particular visited network.

3.6.2 Test Procedure
We use the Ericsson SDS core and the OpenIMS. We try to register at OpenIMS core configuring the Ericsson SDS as our P-CSCF. In this architecture the Ericsson SDS is the visited network and OpenIMS is customers' home network.

At the same time we register at Ericsson SDS using OpenIMS as P-CSCF. In this case we have the opposite configuration - Ericsson SDS is customers' home network and OpenIMS is the visited network.

3.6.3 Test Results
The test successfully proved that roaming scenario works in Netlab testing environment. Following figure shows the flow of SIP messages from the point of view of the visited network (the server is the SDS IMS core and the visited network). In the opposite direction (SDS as the home network and OpenIMScore as visited network) the test was also successful.

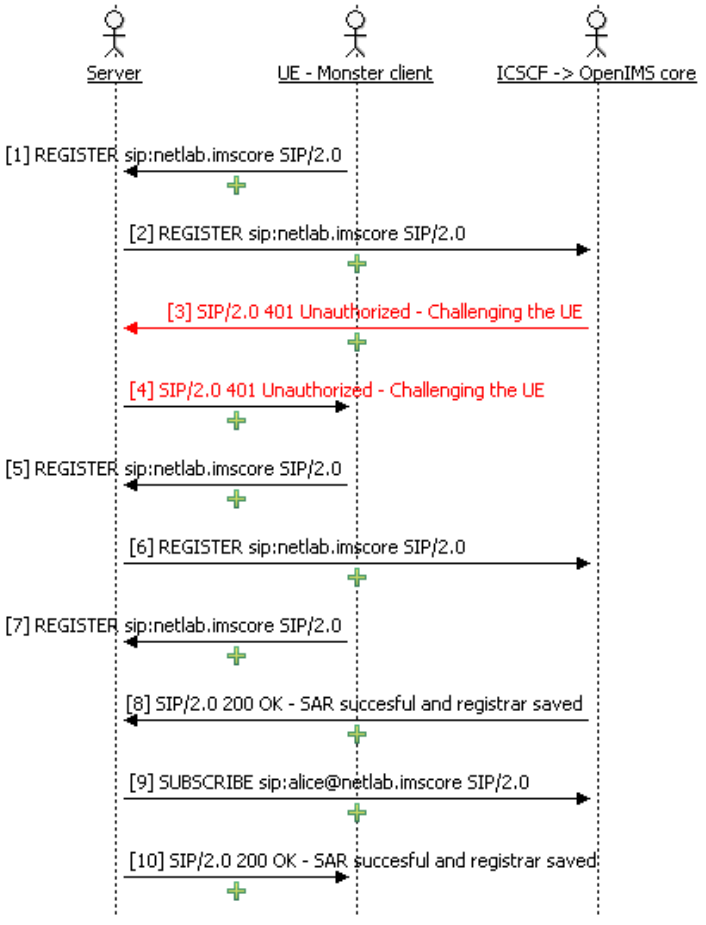


Fig. 5. Roaming flow of SIP messages

4 Conclusions

Within this paper, the Netlab consortium has established a distributed test bed for IMS interconnection tests. During the tests, we have executed several basic test cases and have addressed the difficulties. We were able to register at different IMS cores with several IMS clients, we were also able to establish voice sessions through different IMS cores, send instant messages, test presence through OpenIMS and SDS cores and prove that roaming scenario works in mentioned cores.

We have proved interoperability among different IMS cores although in some occasions specific configuration had to be done. We have achieved positive results however more work will be done before the project ends. The future work consits in more sophisticated scenario and also on sustainability of the NetLab project results.

References

1. Netlab: Use Cases For Interconnected Testbeds and Living Labs, http://www.celtic-iniciative.org/Projects/NETLAB/default.asp
2. RFC 2543 SIP: Session Initiation Protocol (March 1999)
3. ETSI ES 282 001 (TISPAN); NGN Functional Architecture Release 1
4. Camarillo, G., Romero, G., Miguel, A.: The 3G IP Multimedia Subsystem. Wiley, Chichester (2006)
5. Camarillo, G.: SIP demystified. McGraw-Hill, New York
6. OpenIMS web page, http://www.openimscore.org/
7. Ericsson service development studio, http://www.ericsson.com/developer/sub/open/technologies/ims_poc/tools/sds_40
8. Monster IMS client web page, http://www.monster-the-client.org/
9. IMS Mercuro Client, http://www.ericsson.com/developer/sub/open/technologies/ims_poc/tools/sds_40
10. X-lite web page, http://www.counterpath.com/x-lite.html
11. UCTIMSCLIENT web page, http://uctimsclient.berlios.de/
12. ETSI/TISPAN web page, http://www.etsi.org/tispan/

Author Index