Lluís Fàbrega Pere Vilà
Davide Careglio Dimitri Papadimitriou (Eds.)

LNCS 7586

Measurement Methodology and Tools

First European Workshop, FP7 FIRE/EULER Project
Aalborg, Denmark, May 2012
Revised and Extended Papers

Springer

Lluís Fàbrega   Pere Vilà
Davide Careglio   Dimitri Papadimitriou (Eds.)

Measurement Methodology and Tools

First European Workshop, FP7 FIRE/EULER Project
Aalborg, Denmark, May 9, 2012
Revised and Extended Papers

Springer

Volume Editors

Lluís Fàbrega
Pere Vilà
Universitat de Girona (UdG)
Campus Montilivi, EPS-P4, 17071 Girona, Catalunya, Spain
E-mail: {lluis.fabrega, pere.vila}@udg.edu

Davide Careglio
Universitat Politècnica de Catalunya (UPC)
Jordi Girona 1-3, D6-103, 08034 Barcelona, Catalunya, Spain
E-mail: careglio@ac.upc.edu

Dimitri Papadimitriou
Alcatel-Lucent Bell Labs (ALBL)
Building F – Floor 6, Copernicuslaan 50, 2018 Antwerp, Belgium
E-mail: dimitri.papadimitriou@alcatel-lucent.com

ISSN 0302-9743 e-ISSN 1611-3349
ISBN 978-3-642-41295-0 e-ISBN 978-3-642-41296-7
DOI 10.1007/978-3-642-41296-7
Springer Heidelberg New York Dordrecht London

Library of Congress Control Number: 2013949167

CR Subject Classification (1998): C.2, D.4.4, D.2, H.3.5, H.4, K.6.5, I.2, H.5

LNCS Sublibrary: SL 3 – Information Systems and Application, incl. Internet/Web and HCI

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India

Printed on acid-free paper

Springer is part of Springer Science+Business Media (www.springer.com)

Preface

The Future Internet Research Experimentation (FIRE) initiative of the EU 7th Framework Programme (FP7) aims at promoting multidisciplinary research, design and large-scale experimentation of new network and service architectures for the Future Internet. FIRE brings together European research projects addressing two related dimensions: building of large-scale experimental facilities based on the principle of open coordinated federation of testbeds, and instigating disruptive experimentally driven research validated through large-scale experimentation in these testbeds.

Considering that measurements and measurement tools play a very important role in experimentally driven research, it was deemed worthwhile to dedicate a workshop to this challenging topic. The EULER (Experimental UpdateLess Evolutive Routing) research project, a 3-year project part of the FIRE initiative, organized the Workshop on Measurement and Measurement Tools as part of the FIRE activities during the Future Internet Assembly (FIA) conference in Aalborg, Denmark, in May 2012. The initial goals of the workshop were to present the current developments on measurements and tools by the FIRE projects in well-established research areas, including wireless and sensor networks, routing, etc., and to anticipate the needs in new research areas including information-centric networking, programmable components/ networks, etc. For this purpose, researchers from most FIRE projects working in different research areas were invited to present their measurement-based experimental research activities and to share their experience in designing, developing, and using measurement tools. The expected outcomes of the workshop were to identify both current needs with respect to well-established research areas and foreseeable needs and their commonality with respect to new areas of research not currently addressed by existing measurement techniques and tools, to determine the lessons learned and best practices in tools development for measurement-based experimental research, and to determine which initiative(s) could be launched by means of cooperation between projects from a directory of measurement tools accessible to the FIRE community at large, up to the joint development of such tools.

The publication of this volume of the *Lecture Notes in Computer Science* series is directly related to the successful realization of the workshop. Given the quality and interest of the presentations, it was considered worth publishing a book compiling extended versions of them. Speakers were invited to write a chapter focused on the use of measurements and associated tools in experimental research from their experience and the project's perspective, namely, experimental methodology, testbeds, tools developed by the project or external tools, and experiments where such tools are used.

The EULER research project thanks the support of the European Commission through contract number FP7-258307.

June 2013

Lluís Fàbrega
Pere Vilà
Davide Careglio
Dimitri Papadimitriou

Table of Contents

Introduction

Lluís Fàbrega[1], Pere Vilà[1], Davide Careglio[2], and Dimitri Papadimitriou[3]

[1] Institut d'Informàtica i Aplicacions, Universitat de Girona, Spain
{lluis.fabrega,pere.vila}@udg.edu
[2] Departament d'Arquitectura de Computadors, Universitat Politècnica de Catalunya, Spain
careglio@ac.upc.edu
[3] Alcatel-Lucent Bell Labs, Belgium
dimitri.papadimitriou@alcatel-lucent.com

Research on new network and service architectures for the Internet includes many different aspects, from new applications to new network protocols, control models and procedures but also faster and more reliable network technologies. In order to validate scientific results, these new paradigms should be evaluated through experiments carried out at the Internet scale.

The foundational objectives of the Future Internet Research and Experimentation (FIRE) initiative [1] of the EU 7th Framework Programme (FP7) have lead to the inception of experimentally-driven research as a visionary multidisciplinary investigation activity, oriented towards the design and large-scale experimentation of new and innovative paradigms for the Internet (modeled as a large-scale complex distributed system), defining the challenges for it and taking advantage of experimental facilities. Such investigation activity would be realized by means of iterative cycles of research, where the data and observations obtained from experiments performed in each iteration strongly influence research direction in the next iteration. From this research process, the method referred to as "measurement-based research" results logically. In turn, it requires the specification of the relevant criteria and metrics as well as their corresponding measurement techniques and tools.

In this volume of the Lecture Notes in Computer Science series, we have assembled ten articles addressing the design, development and use of measurements and associated tools in experimental research dedicated to the Internet. Each article constitutes a chapter of this book. In each of them, the authors, researchers of eight different projects part of the FIRE initiative, present their view and experience on the subject from their project's perspective, in the research areas of wireless/sensor networks (HOBNET [2], CREW [3] and CONECT [4] projects), routing (CONVERGENCE [5] and EULER [6] projects), and large-scale experimental facilities (OFELIA [7], OpenLab [8] and NOVI [9] projects). Measurement-based research in the wireless/sensor networking research area has to consider devices that are highly heterogeneous (sensors of many types, actuators and different radio technologies), as well as being dynamic and mobile, and it has also to face the fact that experiments are always exposed to interference from unwanted signals, which makes difficult its repeatability. Experimentation of new routing schemes for the Internet is confronted to the situation that there is not any facility that recreates the actual running conditions at the scale of the Internet, where they can be evaluated before being deployed, and therefore experiments have to rely on models that intend to reproduce reality. Large-scale experimental facilities used to be oriented to specific areas and use their own set of tools (to configure and setup experiments, and to generate traffic and perform measurements during the experiments), so research in this area is focusing on providing experimenters with a common interface that facilitates the reproducibility of experiments in different testbeds and that allows the combination of resources from different testbeds as if they were from a single one.

In the next chapter, "Measurement-based experimental research methodology", D.Papadimitriou et al. (EULER project [6]) present the basic principles of experimental methodology and the role of measurements in it. An experimental methodology is an iterative process that includes the following steps: specification of the performance objectives, criteria and metrics; description of the experiment modus operandi (applied on the experimental corpus); observation through measurements and its measurement data analysis, and the feedback at each iteration, which leads to partial conclusions and/or a new iteration. The author also exposes the challenges in performing measurements in a distributed experimental setting (as it is the case in research on network and service architectures for the Internet). Finally, this chapter details the properties and criteria that the experimental measurements shall verify in order to declare scientific validity of the results that measurement-based experimental research produces.

Once the foundations of the measurement-based experimental methodology have been introduced, the reader will learn from the next three chapters several views on the subject from the wireless/sensor networking research area. One is placed in the area of energy efficiency in smart buildings and the other two in the area of wireless testbeds.

In the third chapter dedicated to the "Experimental performance evaluation of sensor-based networking for energy efficiency in smart buildings" by C.M.Angelopoulos et al. (HOBNET project [2]), the authors discuss on measurement-based experimental research in automation and energy-efficiency of smart/green buildings through sensor networks. The way the sensors are deployed, the mobility profile, the sensor network protocol properties and the performance metrics are detected as critical components in the experiments. As an example, a smart watering application developed in the project is described.

In the fourth chapter, "Various detection techniques and platforms for monitoring interference condition in a wireless testbed", W.Liu et al. (CREW project [3]) present the challenges in measuring wireless signals and propose a framework and associated methodologies for wireless testbeds. The framework considers heterogeneous and distributed devices and combines information from both physical layer and higher layer measurements, such as the device performance, reliability of experiments or repeatability.

In the fifth chapter, "Methodology and tools for measurements on wireless testbeds: the NITOS approach" by D.Giatsios et al. (CONECT project [4]), the authors describe the measurement methodology and tools used in the NITOS wireless testbed, and provide several examples of experiments conducted on it. This testbed allows researchers to implement and evaluate new schemes from the MAC to the application layer and it includes a large number of heterogeneous wireless devices. It handles measurements using the cOntrol and Management Framework (OMF), which enables an efficient management of the heterogeneous resources of this testbed, and provides a clear and easy way to define an experiment, execute it and collect the results.

The next two chapters present two facets of the use of measurement and associated tools in the research area of routing, one in the context of the novel paradigm of Information-Centric Networking (ICN) and the other on the wide-scale Internet routing.

In ICN the network layer provides users with content, instead of providing communication channels between hosts. The basic functions are twofold: the addressing of contents through schemes based either on names or identifiers (that do not include references to their location); and routing user requests toward the closest copy of the content with such a name. In the sixth chapter, “Scalability measurements in an Information-Centric Network”, N.Blefari et al. (CONVERGENCE project [5]) present their approach to ICN focusing on the routing scheme, and report performance measurements obtained in simulation and in a testbed.

The seventh chapter, “Iterative research method applied to the design and evaluation of a dynamic multicast routing scheme” by D.Papadimitriou et al. (EULER project [6]), is positioned in the research area of new paradigms for distributed and dynamic routing schemes suitable for the future Internet and its evolution. According to an iterative research cycle process, this chapter present the design and performance evaluation through simulation of a dynamic multicast routing scheme called Greedy Compact Multicast Routing. The authors also present a study of how spatially and temporally concentrated traffic exchanges are in the Internet in order to state the relevance and benefits of deploying multicast routing schemes.

Large-scale experimental facilities are a crucial requirement for validating research on new paradigms and models, new components and systems, new procedures and protocols, and new network and service architectures for the Internet. In the next four chapters the reader will learn about several current large-scale facilities, how the federation of different testbeds can be realized, and the tools they provide to carry out measurement-based experimental research.

OpenFlow [10] is the base technology of the pan-European research testbed OFELIA [7]. OpenFlow defines the interaction between a network element (switch, router) and an externalized control element (the controller), through the separation of control, forwarding and processing of data. In the eighth chapter, “Metrics and measurement tools in OpenFlow and the OFELIA testbed in particular”, M.Körner et al. (OFELIA project [7]) describe the OFELIA testbed, how experimenters can use it, the set of tools to experiment various aspects of OpenFlow switches and controllers and the set of traffic generation and measurement tools to perform the experiments.

Federating testbeds, i.e., enabling transparent access to combinations of resources from different testbeds, each addressing different applications or technologies, offers a richer and more powerful experimental facility to enable heterogeneous and large-scale experimental-based research. However most deployed testbeds have been built with a certain application scope and usually uses its own control framework. The challenges to realize the federation of testbeds are explained in the ninth chapter, “Heterogeneous testbeds, tools and experiments - Measurement requirements perspective” by A.Gavras et al. (OpenLab project [8]). After that the authors describe several examples of experiments being conducted in OpenLab, an experimental platform that brings together a number of different and diverse testbeds (such as fixed, wireless, multimedia, high precision measurement, or emulation testbeds).

One of the problems in the federation of multiple testbeds, is the presence of different and diverse measurement tools. Differences may appear in naming, data representations, units, metadata and data merge; therefore, their integration becomes necessary. In the tenth chapter “Measurements and measurement tools in OpenLab: use cases with measurement data ontologies”, J.E.López de Vergara et al. (OpenLab project [8]) present a possible solution to this problem based on the semantics of the information, unambiguously specifying the set of concepts that comprise a measurement. The proposed solution is composed of three steps: agree on a common ontology for network measurements, define mappings between each particular scheme and the common ontology, and define a semantic interface able of receiving a query from a user and distributing it among all particular measurement repositories.

In the eleventh chapter, “A monitoring framework for federated virtualized infrastructures”, J.Stéger et al. (NOVI project [9]) present a control, management and monitoring framework to allow the federation of various virtualized testbeds, consisting of heterogeneous resources, enabling experimenters to request, reserve, use and update a great deal of virtualized resources in the federation, tailored to their needs. The necessary abstraction of the managed entities is provided by information models, which should support virtualization concepts, vendor independence (of the physical resources), monitoring and measurement concepts and management policies. The information model facilitates the control and management of the individual platforms, and the communication between them.

Finally, in the twelfth chapter “Summary and conclusions”, we summarize the main outcomes of the different chapters, and provide lessons learned, best practices and key recommendations on experimental-based measurement and associated tools.

References

1. EU FIRE initiative, `http://www.ict-fire.eu/`
2. EU FP7 HOBNET project, `http://www.hobnet-project.eu/`
3. EU FP7 CREW project, `http://www.crew-project.eu/`
4. EU FP7 CONECT project, `http://www.conect-ict.eu/`
5. EU FP7 CONVERGENCE project, `http://www.ict-convergence.eu/`
6. EU FP7 EULER project, `http://www.euler-fire-project.eu/`
7. EU FP7 OFELIA project, `http://www.fp7-ofelia.eu/`
8. EU FP7 OpenLab project, `http://www.ict-openlab.eu/`
9. EU FP7 NOVI project, `http://www.fp7-novi.eu/`
10. Open Networking Foundation, `https://www.opennetworking.org/`

Measurement-Based Experimental Research Methodology

Dimitri Papadimitriou[1], Lluís Fàbrega[2], Pere Vilà[2], Davide Careglio[3], and Piet Demeester[4]

[1] Alcatel-Lucent Bell Labs, Antwerp, Belgium
dimitri.papadimitriou@alcatel-lucent.com
[2] Universitat de Girona, Girona, Spain
{lluis.fabrega,pere.vila}@udg.edu
[3] Universitat Politècnica de Catalunya, Barcelona, Spain
careglio@ac.upc.edu
[4] Ghent University, Gent, Belgium
piet.demeester@intec.ugent.be

Abstract. Aiming at creating a dynamic between elaboration, realization, and validation by means of iterative cycles of experimentation, Future Internet Research and Experimentation (FIRE) projects have been rapidly confronted to the lack of systematic experimental research methodology. Moreover, the "validation by experimentation" objective involves a broad spectrum of experimentation tools ranging from simulation to field trial prototypes together with their associated measurement tools. As experimental measurement results and corresponding tools play a fundamental role in experimental research, devising a systematic experimentation and measurement methodology becomes thus crucial for experimental research projects to achieve this objective for their various realizations, including protocols, systems and components. In turn, and in order to meet scientific validity criteria, the measurement results obtained when performing experimental research implies the availability of reliable and verifiable measurement tools, including on-line measurement data analysis and mining.

Keywords: measurement, experimental research, methodology, criteria.

1 Introduction

The Future Internet Research and Experimentation (FIRE) initiative aims to realize a "research environment for investigating and experimentally validating highly innovative and revolutionary ideas" towards new paradigms for the Internet by bridging multi-disciplinary long-term research and experimentally-driven large-scale validation. FIRE foundational objectives were:

- Creation of a multi-disciplinary, long term research environment for investigating and experimentally validating highly innovative and revolutionary ideas for new networking architectures and service paradigms;

- Promotion of experimentally-driven yet long-term research, joining the two ends of academy-driven visionary research and industry-driven testing and experimentation, in a truly multi-disciplinary and innovative approach;
- Realization of a large scale European experimental facility, by gradually inter-connecting and federating existing and new "resource clusters" for emerging or future internet architectures and technologies.

These objectives further evolved toward the inception of experimentally-driven research as a visionary multidisciplinary investigation activity, defining the challenges for and taking advantage of experimental facilities. Such investigation activity would be realized by means of iterative cycles of research, oriented towards the design and large-scale experimentation of new and innovative paradigms for the Internet - modeled as a complex distributed system. The refinement of the research directions should be strongly influenced by the data and observations obtained from experiments performed at previous iterations thus, being "measurement-based" which in turn requires the specification of the relevant criteria and metrics as well as their corresponding measurement tools.

The rationale was thus clear: create a dynamic between elaboration, realization, and validation by means of iterative cycles of experimentation. The realization however was already less obvious and the validation objective rapidly confronted to the lack of systematic experimental research methodology and underlying measurement methodology applicable to computer communication and networked systems. Moreover, the "validation by experimentation" objective opens a broad spectrum of experimentation tools (ranging from simulation to prototype of real systems) and measurement tools. The selection of the experimentation tools depends itself on 1) the object of experimentation (corpus), 2) the nature and properties of the results, and 3) the cost function that itself depends on complexity, experimental and running conditions but also on the level of abstraction (referred to as "realism").

Our main argument are that the systematic experimental validation of the targeted "elaboration and realization" can be described by a continuum that requires a broader set of tools: starting from more abstract tools (not only because their resulting cost is lesser but also because such tools produces results verifying all conditions explained in this chapter) followed by the progressive addition of realism as part of the experimented system to ultimately reach the so-called field trials with real systems. The following sections detail the dependencies with respect to this experimentation chain and the set of criteria that experimental results shall satisfy in order to ensure the scientific validity of the results this chain produces.

Moreover, as measurement results and tools play a fundamental role in experimental research, devising a systematic measurement methodology becomes critical for the various experimental research projects to achieve the objective of validation by experimentation of their numerous realizations, including protocols, systems and components. In turn, the measurement-based experimental validation of the targeted "elaboration and realization" involves a very broad set of measurement tools to obtain measurement results. In order to meet scientific validity criteria, these measurement results obtained when performing experimental research require the availability of reliable and verifiable measurement tools, including on-line measurement data analysis and mining.

2 Experimental Research Methodology

Computer communication networks are characterized by two fundamental dimensions: i) *distribution*: of a large number of dynamically interacting (non-atomic often complex) components, and ii) *variability*: the spatio-temporal variation of their inner properties that in turn influence these interactions. A couple of examples would better describe the landscape: autonomic networking is the transposition of the autonomic computing concept in the communication space, and network "virtualization" is the transposition of the abstraction concept of object-oriented programming in the networking space. More, the dynamic nature of these interactions results in modifying its scaling properties of the individual components besides modifying the properties of the global system. Many other examples can be cited, the fundamental observation is that no experimental model actually exists – or more precisely – the complexity of the resulting system makes its modeling a research discipline on its own.

However, this does not mean or imply that an experimental research methodology could not be defined based on i) a broader set of tools ranging from simulation to experimentation of prototypes/real systems and ii) experience gathered from practicing various and large experiments in the computer communication/networking disciplines. Such methodology would include the following steps (part of each iteration):

1. Specification of the functional and performance objectives, (technical and non-technical) constraints, and description of expected results
2. Definition of the relevant functional and performance criteria and metrics
3. Description of the modus operandi including configuration, initialization, running conditions, and (iterative) procedure(s) to be executed
4. The reporting on observations and the resulting analysis as well as the feedback after performing each iteration before reaching (partial) conclusion.

The following sections describe this methodology in more details.

2.1 Functional and Performance Evaluation

Functional and performance evaluation typically involves the specification of 1) Functional and performance model, 2) Functional and performance measurements and metrics, and 3) Functional and performance results analysis. We describe in the following the experimentation methodology for performance evaluation and refer to [1] for what concerns for the functional evaluation methodology.

- **Performance model:** the specification of a performance model defines the significant aspects of the way by which a proposed or actual system operates in terms of resources consumed, accessed, scheduled, etc. together with the various delays resulting from processing and/or physical/hardware limitations (such as bandwidth, access latency, etc.). A performance model provides useful information on how the proposed model vs. actual system will or does actually work. Based on the information contained in the performance model, the interpretation of the execution of the model (by means of simulation or emulation) provide further insight into the system's behavior, and can be used to identify where the model design is inadequate.

- **Performance measurement:** many performance metrics may be used for this purpose, including computational complexity, memory-space complexity, time complexity (convergence), communication complexity, etc. but also combination of metrics including adaptation cost (which combines communication and computational complexity).
- **Performance analysis:** includes i) the examination of the performance measurement results obtained for the proposed model against the criteria associated to each performance metric, ii) the comparison of the performance measurement results obtained for the proposed model against those obtained for the reference model, and iii) the conformance of the performance measurement results obtained for the proposed model against the results as determined and/or estimated by the associated performance model.

2.2 Experimental Evaluation Methodology

Experimental performance evaluation requires to specify a theoretical model (of the system under study) from which a performance analysis can be performed. Using the feedback from this performance analysis a behavioral/conceptual model is then built that enables the development of an experimental model being either a simulation or an emulation model. A simulation model aims at characterizing the working of the modeled process or system over time by examining a range of behaviors that are similar or analogous to a real world system. On the other hand, an emulation model imitates the externally observable behavior to match an existing real system functionally close enough so that it can be substituted to the real system while internal state of the emulation mechanism does not have to accurately reflect the internal state of the real system which it is emulating. Simulation and emulation are further detailed in Section 4.

The experimental model can then be converted into a computational model. Measuring the metrics on the execution of this model provides the needed information to compare the obtained results with those of the theoretical model. Fig.1 depicts the flow chart used for the purpose of systematic performance evaluation (and analysis).

The (typically iterative) process to build and develop such experimental model is the following:

- Determine the goals and objectives of the experiment
- Build a conceptual model including state variables, which variables are static and dynamic, for the latter are variations continuous and discrete, etc.
- Convert the conceptual model into a specification model of the experiment being a simulation or emulation experiment; the specification typically describes the experimented procedures (pseudo-code) and data structures; the experimented procedures can be either simulation and/or emulation procedures or even prototype(s).

- Convert the specification model into a computational model, i.e., executable computer program(s). The selection of the programming language is part of this step and consists in determining whether a general-purpose programming language or a special-purpose language would be used to develop this/these program(s). Note that in case of simulation, the term program is frequently used to refer to the computational model itself.
- Verify (verification process): Ask the question: Did we build the model right?
  - Determine whether the computational model executes as intended by the specification model.
  - Determine whether the computational model implements the assumptions made about the behavior of the real systems (as transposed in the behavioral model)
  - Verification techniques include: tracing/walk-through, continuity tests (sensitivity tests, i.e., slight change in input should yield slight change in output, otherwise error), degeneracy tests (perform execution with extremes values, e.g., lowest and highest), and consistency tests (similar inputs produce similar outputs).
- Validate (validation process): Ask the question: Did we build the right model?
  - Determine whether the conceptual model is representative of the actual system being analyzed or not. Can the conceptual model be substituted, at least approximately for the real system?
  - The validation process also involves determining whether the computational model is consistent with the actual system being analyzed.

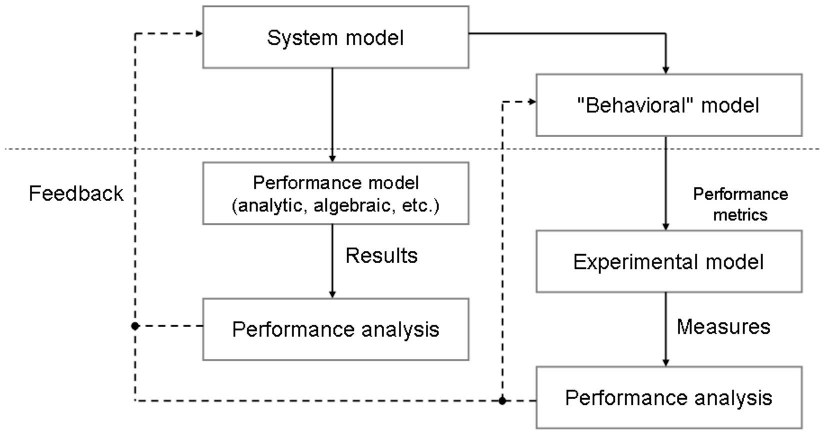


Fig. 1. Performance evaluation and analysis methodology

Once the executable computer program(s) are verified and validated, experimental studies can be performed. Performance evaluation studies include the following steps:

- Design the experiments:
  - Determine the input parameters that should be varied and their interval as well as the initial conditions for proper initialization (note that some characteristics

of the environment may need to be included in the experiment if not accounted as part of the simulation model); this step is critical as it provides means for decreasing the run time of the simulation but still may not provide confidence for stable conditions.

  - Determine the variables to measure (at which frequency, upon which class of event (event-driven), etc. taking into account the tradeoff between too much data (that would in turn require the use of techniques for reducing the amount of collected to a usable form) and too little data (that would in turn introduce the need for representing data by statistical distributions);
  - Determine the execution time taking into account the tradeoff between the resource consumption (by very long runs) and amortization of effects of resulting from transient state will be amortized.

- Execute the program(s) and record tuples of the form {<initial_condition; input>; <running_condition; output>} referred to as observations or data.
- Analyze the output of the program execution towards production of results. Such analysis comprises the following:
  - Results verification (correctness): test whether the results obtained are in accordance to the assumptions made about the behavior of the real systems (as transposed in the behavioral model);
  - Input validation (representativity): validate assumptions about input parameter values and distributions. This step is often associated to the output validation phase;
  - Results/output validation (representativity): test whether the results obtained are representative of those obtained either by real systems or theoretical model of the system.

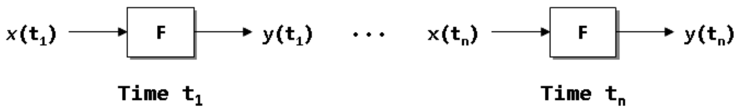
3 Experimental Results Criteria

Let's now proceed with the definitions of the criteria that experimental results shall satisfy when performing experimental evaluation in order to ensure that scientific validity of the results these experiments produce. If we model the corpus of a given experiment (or experimental corpus[1]) by a function F, with input vector of variables $x_1,...,x_n$ and parameters $e_1...,e_m$ such that its output $F(x_1,...,x_n|e_1,...,e_m) = y$, then the following properties shall hold:

3.1 Reliability

Reliability is defined as the probability that system or component will perform its intended function during a specified period of time under stated conditions. Referring to Fig.2, it means that during the pre-defined time interval $[t_1,t_n]$ the output of the experimental corpus model $F(x_1,...,x_n|e_1,...,e_m)[t_k] = y[t_k]$ exists $\forall\ k \in [1,n]$ within a pre-defined range $[y_1,y_p]$ of valid output.

[1] The term experimental corpus refers to the main object of an experiment.



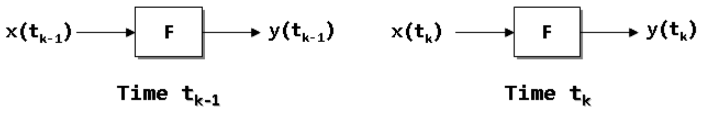
$\exists$ $[t_1,t_n]$ such that $\forall$ $k \in N$, $1 \le k \le n$

$F(x(t_k)) = y(t_k)$ exists and $y(t_k) \in [y_1,y_p]$

Fig. 2. Experimental criteria: reliability

3.2 Repeatability

Referring to Fig.3, repeatability means that $\forall$ $k \in N_0$, if $(x_1,...,x_n|e_1,...,e_m)[t_{k-1}] = (x_1,...,x_n|e_1,...,e_m)[t_k]$ together with the condition that $F(x_1,...,x_n|e_1,...,e_m)[t_{k-1}] = y[t_{k-1}]$ and $F(x_1,...,x_n|e_1,...,e_m)[t_k] = y[t_k]$ exist, then $y[t_k] = y[t_{k-1}]$. The term reliability is thus characterized by persistence of the output in time.



$\forall$ $k \in N_0$, if $x(t_k) = x(t_{k-1})$

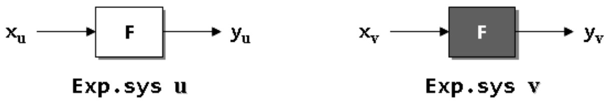
and $F(x(t_{k-1})) = y(t_{k-1})$ exists $\wedge$ $F(x(t_k)) = y(t_k)$ exists

then $y(t_k) = y(t_{k-1})$

Fig. 3. Experimental criteria: repeatability

3.3 Reproducibility

Reproducibility means that the experimental model $F(x_1,...,x_n|e_1,...,e_m)$ can be executed at the same time (simultaneously) on different experimental systems $u, v \in S$ ($\equiv$ experimental system set) and produces the same output if the model input at both systems is identical. Reproducibility is thus characterized by persistence of the output in space. We refer to Fig.4 for a formal definition of reproducibility.



$\exists$ $u, v \in S$ such that if $x_u = x_v$

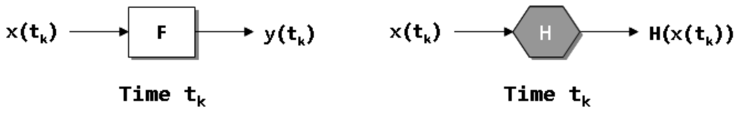
then $y_u = y_v$

Fig. 4. Experimental criteria: reproducibility

3.4 Verifiability

Referring to Fig.5, verifiability means that we can find independently a formal model H: $x(t) \rightarrow H(x(t))$ corresponding to the experimental model F: $x(t) \rightarrow y(t) = F(x(t))$ such that at time t_k, the output of the function F can be confirmed against the output of the formal model H, i.e., referring to Fig.5, $H(x_1,...,x_n|e_1,...,e_m)[t_k] \in [y(t_k)-\varepsilon;y(t_k)+\varepsilon]$ with $\varepsilon << 0$, where $y(t_k) = F(x_1,...,x_n|e_1,...,e_m)[t_k]$.

Note that in practice, one aims at finding a formal model H such that the output of the function F complies with the output of the model H at any time t_k, $\forall$ $k \in [1,n]$, time interval defining the duration of the experiment.



$$\exists\ H: \Re^n \rightarrow \Re: x(t) \rightarrow H(x(t)) \text{ and } \varepsilon << 0$$
$$\text{such that } H(x(t_k)) \in [y(t_k)-\varepsilon\ ;y(t_k)+\varepsilon] \text{ where } y(t_k) = F(x(t_k))$$

Fig. 5. Experimental criteria: verifiability

In order to ensure verifiability, reliability, repeatability, and reproducibility of the experimental results produced, one shall characterize the output of experimentation. Meeting these criteria implies in turn to control the parametrization, the input and output, as well as the running conditions of the conducted experiments. On one hand, verifying the repeatability, reproducibility, and reliability criteria enables generalization of the experimental results produced. On the other hand, ensuring verifiability of these results increases their credibility (results can be "explained").

In the present context, it is also important to underline the fundamental distinction between verification (verifiability) and validation (validity). Verification means that the experimental model output should satisfy the formal model output (e.g. computational model). On the other hand, validity is formally defined as follows: a proposition A is valid if H(A) = TRUE for any model H of A; thus, in experimental research, we can only verify satisfiability (proposition A has at least one model H for which H(A) = TRUE). Note that in propositional logic, one usually verifies validity by applying the following theorem: a proposition A is valid if and only if its opposite (negation) can not be satisfied. Henceforth, the best we can hope concerning "verification" is to find at least one model H of A that verifies the same "output" as the realization F of A by experiment: H(A) = F(A). If this is the case, then one does indeed satisfy the initial proposition (it is verified by one model) but not validate it (the proposition is not verified for any model). Also, one constructs (independently) a model to verify that the output of the experiment F satisfies to the output of the model H. Thus, one does not verify the conformance of experimental execution against the specification of the experimented system but the consistency of the experimental output against a computational model of the experiment drawn independently from it.

Verifiability is thus not synonym of conformity test or conformance test against the specification of the underlying experimented system. Finally, it is interesting to observe from its definition that verification is the formal complement to experimentation (instead of positioning experimentation as the complement to the theoretical model).

Note: scalability is often cited as a criterion to be met by the experimental corpus. The scale of a system is measured by the rate x state x size that the system can sustain when running using a given number of resource units (for processing and storage). Networking systems can thus only scale indefinitely if and only if the rate of change of the state set, the number of states and the size of each state are independent of the global properties of the environment into which the system is operating. It is thus fundamental to mention that the scale of an experimental facility (the number of resource units and their distribution) does not determine the scalability properties of the corpus. However, the scalability properties of the experimented corpus determine the number of resource units that are locally required to be executed at a certain scale. Thus, such experiment can be performed to i) verify a pre-estimated level of scaling of the experimented corpus and/or ii) iteratively determine the scale property of the corpus with the risk that the dependency on the global properties could never be found (hidden dependencies, correlations, non-linearities, etc.). Hence, only the former leads to verifiable experiments. In other words, a large-scale facility can only help verifying scaling properties but not determine these properties. Further positioning the role of so-called large-scale testbeds/experimental facilities is outside the scope of this chapter.

4 Experimental Tools

Different experimental tools can be used. Their selection is neither arbitrary nor religious: it depends on the experimental objective and maturity of the experimental corpus. Nevertheless, each of them needs to ensure that the experimental criteria defined Section 3 are satisfied. However, it is clear that fulfilling these criteria does not come at the same cost for the same level of abstraction. We can distinguish three types of abstraction: i) abstraction of the network/shared infrastructure (network resource consumption model, processing model, etc.), ii) abstraction of the system (processing/memory resource consumption model, computation model, etc.), and iii) abstraction of the environment (traffic model, application model, user/behavior model, etc.). To each (non-atomic) element of this partition of the abstraction space, we can associate a level of realism when the abstraction is replaced by a “real” entity. Without entering into the debate of reality or what reality actually represents or means, we simply consider here a real system as an instantiation of the experimented component models at the hardware and/ or software substrate level depending on the expected level of performance.

4.1 Simulation

Following Shannon [2], simulation is the process of designing a model of a real system and conducting experiments with this model for the purpose either of understanding the behavior of the system or of evaluating various strategies (within the limits imposed by a criterion or set of criteria) for the operation of a system. The execution of the model X is said to simulate another system Y when the internal working processes of X can be described by a mathematical and/or procedural model known to best represent the actual working processes of Y. Ingalls [3], further defines simulation as the process of designing a dynamic model of an actual dynamic system for the purpose either of understanding the behavior of the system or of evaluating various strategies (within the limits imposed by a criterion or set of criteria) for the operation of a system.

Simulation can thus be seen as the process of exercising a model to characterize the working of the modeled entity process, or system over time by examining a range of behaviors that are similar or analogous to a real world system. Simulations are never, by definition, complete. Simulation is one of the most widely used techniques for i) understanding, characterizing and analyzing the behavior of complex systems, ii) construct theories or hypotheses that account for the observed behavior, iii) use the model to predict future behavior, that is, the effects that will be produced by changes in the system. Simulations may be deterministic or stochastic, static or dynamic, continuous or discrete.

4.2 Emulation

Emulation is the process of imitating the outwardly observable behavior to match an existing real system functionally close enough so that it can be substituted to the real system. The internal state of the emulation mechanism does not have to accurately reflect the internal state of the real system which it is emulating. A system X is said to emulate another system Y if the behavior of X is exactly the same as that of Y (same output for same input under similar conditions) but the mechanism to arrive at the output (from the input) is different. Emulation is generally used when we don't exactly know the internal mechanism of the original system but are familiar with the input/output pattern. Performing emulation enables thus to imitate the function of the real system, as by modifications to hardware or software that allow the imitating system to accept the same data, execute the same programs, and achieve the same results as the imitated system without exactly reflecting the inner functioning of the real system. The emulation is "complete" if all the interfaces are present, and the resulting observed behavior matches that of the real world device.

An emulator can thus be defined as a model of a system which will accept any valid input that that the emulated system would accept, and produce the same output or result. Hence, across a well defined interface an emulator is indistinguishable from the real world equivalent (except in performance).

4.3 Simulation and Emulation Are Complementary

In general, spatial measures are more easily achieved in simulation environments whereas temporal measures are more easily achieved by means of emulation. Simulations (in particular, simulation by discrete event) is well suited for experiments involving spatial metrics, structure, and dimensions as it enables handling of large-scale topology cases and produces results that are easier to tune, reproduce, and compare. On the other hand, emulation has the advantage of being more representative of the actual execution of procedures thus providing valuable insight in temporal metrics and behavior (although on smaller scale and at a higher cost). Moreover, emulation experiments enable to check the realism of simulation results and simulation experiments to extend the applicability of emulation results. Indeed, results obtained by means of simulation and emulation experiments are complementary when the experimental scenarios are commonly specified and their execution adapted (without introducing any bias) to the simulation or emulation environment.

Emulation experiments can lead to reproducible and repeatable results but only if their "conditions" and their "executions" can be controlled. Realism can thus be improved compared to simulation (in particular for time-controlled executions of protocol components on real operation system). Nevertheless, such experiments are more complex and time consuming to configure and execute; performance evaluation is possible if the experimental platform comprises a “sufficient number” of machines (representative of the experiment to conduct). Emulation still requires synthetic network conditions (models) if executed in controlled environment and either injecting real traffic or replay traffic traces (not that even when available "spatial distribution" of traffic is available remains problematic to emulate because the spatial aggregation of address prefixes is not necessarily as the routing tables are often not provided together with traffic traces).

Stepping into real system experimentation increases cost but increases realism. As such the loss of control of experimental conditions in such systems raises the issue of persistence of the properties observed earlier in the experimentation chain. In particular, these properties shall already be determined by the earlier experimental stages (leaving them intrinsically part of experimental research activities).

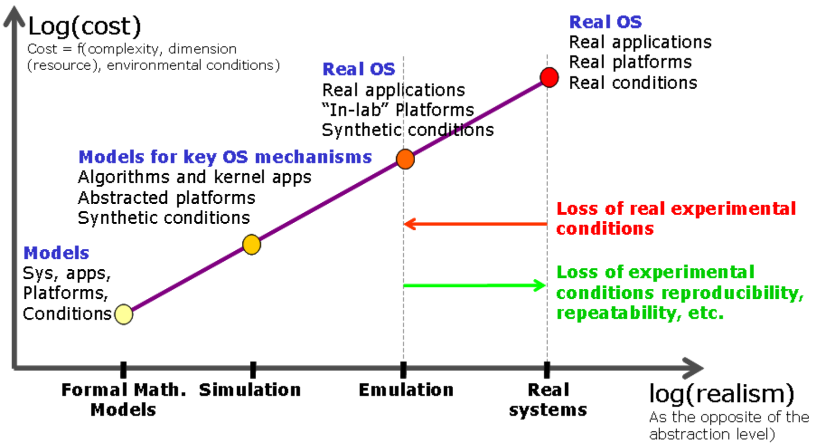

Log(cost)
Cost = f(complexity, dimension (resource), environmental conditions)
Real OS
Real applications
Real platforms
Real conditions
Real OS
Real applications
"In-lab" Platforms
Synthetic conditions
Models for key OS mechanisms
Algorithms and kernel apps
Abstracted platforms
Synthetic conditions
Models
Sys, apps,
Platforms,
Conditions
Loss of real experimental conditions
Loss of experimental conditions reproducibility, repeatability, etc.
Formal Math. Models
Simulation
Emulation
Real systems
log(realism)
As the opposite of the abstraction level)

Fig. 6. Experimental criteria: verifiability

In this context, the validation of a new routing algorithm for instance, would be better conducted on a simulation platform (after formal verification) not only because their resulting cost is lesser but because such tools produces results verifying all conditions explained here above. Afterwards, progressive addition of realism as part of the experimented system would consist in instantiating the execution stratum (remove the system abstraction) in order to perform emulation experiments.

5 Measurement Methodology

Measurement refers to metrology which is defined by the International Bureau of Weights and Measures (BIPM) as "the science of measurement, embracing both experimental and theoretical determinations at any level of uncertainty in any field of science and technology." The term metrology includes all aspects of measurement (theoretical and practical) [4] [5]: starting from the "principles of measurement", which represent the scientific basis for measurement, the "method of measurement" (logical sequence of operations) is instantiated by a measurement (set of operations).

5.1 Measurement Objectives

As documented in [6], measurements aim at determining not only i) the value of a quantity but also at determining, ii) the distributions of quantities in time, in space, and in time and space, iii) the mathematical representations of quantities or their distributions, iv) the relations between quantities, their distributions or representations, and v) the parameters of such relations. The results of measurements of types (i) and (v) are expressed in terms of numbers. The results of measurements of types (ii), (iii) and (iv) may have the form of numbers, series of numbers, functions or series of functions -given in tables-, or analytically. When measurements of type (v) are considered, then the parameters of relations between quantities are often treated as new quantities (e.g., resistance, inductance, capacitance), but the diversity of the investigated relations (e.g., non-linearity, dependence on frequency) breaks the quantitative concept of measurement.

From this perspective, measurement-based experimental research aims at complementing the rigorous performance analysis and simulation-based evaluation. The results are more realistic and can contribute to validate and to fine tune the execution of algorithms. A large variety of realistic topologies, mobility profiles, and traffic patterns is required. Novel network parameters as well as performance monitoring measures (and their trade-offs) arise. Ad hoc approaches are useful but there is a need to converge to widely accepted, common integrated measurement methods, systems and tools.

5.2 Measurement Process

The measurement process is instantiated in a measurement procedure having the *measurand* (quantity that is to be measured) as its inputs, the control variables, and the output representing the *measurement results*. The measurement process comprises

3 distinct steps: 1) design of a measurement method, 2) application of measurement method rules, and 3) analysis of the measurement results. To carry out a measurement task, an experimenter should design and execute a measurement procedure (corresponding to the measurement function μ) which consists of a set of operations, specifically described, for the performance of a particular measurement according to a given measurement method. Note that the results of the measurement can be influenced by external quantity during the measurement process. As experimental testbeds are of different nature (wired, wireless, different hardware, etc.) and offers different level of control of the experimental initial and running conditions the measurement method will have to be specified in order to ensure these external quantities can be identified.

As experimental research in the context of FIRE is by nature distributed, the measurement process shall account for the distributed nature of the experimental environment and the distributed nature of the experimented corpus. This poses additional challenges that should be met by the measurement tools used in the measurement process, including control of the properties of the measurement tools and their calibration, measurement timing and synchronization (timestamps), as well as the measurement granularity, sampling and representativeness.

5.3 Measurement Results Analysis

In order to ensure that measurement results can be systematically analyzed, the control of the experimental execution conditions together with the following elements have to be considered:

- Specify performance analysis methodology together with the necessary mathematical tools to be able to perform data analysis and mining tasks on experimental data coming from various monitoring points (from single or multiple testbeds). This objective also covers specification the necessary mathematical tools to analyze the sensitivity of the performance measures to changes in the *experimental model* parameters. Sensitivity analysis attempts to identify how responsive the results of an experimental model are to changes in its parameters: it is an important tool for achieving confidence in experimentation and making its results credible. Sensitivity analysis quantifies the dependence of system behavior on the parameters that affect the modeled process and in particular its dynamics. It is used to determine how sensitive a model is to changes in the numerical value of the model parameters and changes in the model structure.
- Specify distributed performance monitoring system (while) allowing experimenters to choose the best tool(s) for their experimentation.
- Define a standard experiment description together with a control interface and wrap the measurement tools within this interface. This standard interface will focus on providing a common programming interface to describe every aspect of a networking experiment but will also attempt to provide robust experiment monitoring and management facilities and will integrate with the data analysis and data mining tools developed as described here above (cf. first bullet point). Note that sensitivity analysis of the reliability, the performance, and the performability of the monitoring system is a complementary objective.

6 Measurement Criteria

Measurement results obtained by means of experiments have to verify certain properties and criteria. These properties and criteria mainly include reliability, repeatability, reproducibility, and verifiability. These criteria are indeed the same as those that experimental results shall satisfy (see Section 3) but in the present case they are applied in order to characterize the scientific validity of the measurement results these experiments aim at producing. In turn, they constraint the experimental corpus and research methodology but also determine the fundamental properties and criteria that shall be met by the measurement tools used to perform the measurements implied by these experiments.

6.1 Reliability

Reliability is defined as the probability that the measurement function μ performs its intended measures (output) during a specified period of time under stated conditions. More formally, referring to Fig.7, where the experimental corpus is modeled by a function F with input vector x and output y = F(x), reliability is verified when $\exists [t_1,t_n]$ and $\varepsilon << 0$ such that $\forall k \in N, 1 \leq k \leq n, \mu(y(t_k)) = \mu(F(x(t_k))) \wedge y(t_k) \in [\mu(y(t_k))-\varepsilon,\mu(y(t_k))+\varepsilon]$, where $y(t_k) = F(x(t_k))$.

Reliability implies as a minimum requirement that the components of the experimental corpus remain operational (i.e., do not fail or halt) during this time period. Furthermore, measurement results are reliable if they remain consistent (within a certain well-defined range) during that period.

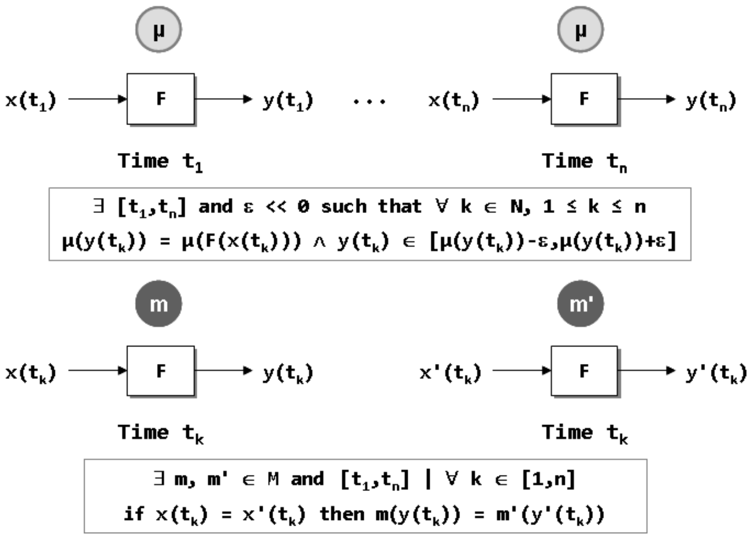

μ
x(t₁)
F
y(t₁)
...
x(tₙ)
y(tₙ)
Time t₁
Time tₙ
∃ [t₁,tₙ] and ε << 0 such that ∀ k ∈ N, 1 ≤ k ≤ n
μ(y(tₖ)) = μ(F(x(tₖ))) ∧ y(tₖ) ∈ [μ(y(tₖ))-ε,μ(y(tₖ))+ε]
m
m'
x(tₖ)
y(tₖ)
x'(tₖ)
y'(tₖ)
Time tₖ
∃ m, m' ∈ M and [t₁,tₙ] | ∀ k ∈ [1,n]
if x(tₖ) = x'(tₖ) then m(y(tₖ)) = m'(y'(tₖ))

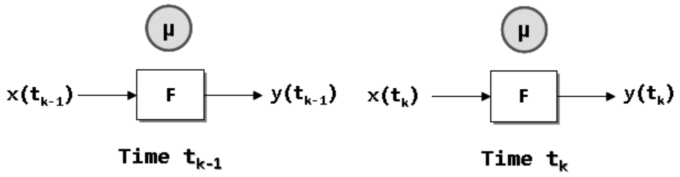
Fig. 7. Measurement criteria: reliability

In practice, in order to assert the reliability of a given measurement tool implementation m ($\in$ M $\equiv$ measurement program set) of a given measurement function μ, it is common to compare the results of measurements produced by m with those obtained for the same time period by means of another implementation m' ($\in$ M) of the same function μ. Referring to Fig.7, $\exists$ m, m' $\in$ M and $[t_1,t_n]$ such that $\forall$ k $\in$ [1,n], if $x(t_k) = x'(t_k)$ then $m(y(t_k)) = m'(y'(t_k))$.

6.2 Repeatability

Repeatability is a temporal criterion associated to measurement results. This term is used when multiple execution of a given experiment (repetition) using the same configuration, running conditions, and input yields the same output. Correct experimental method and usage of models, execution of algorithms, and output processing are required in order to guarantee the repeatability of measurement results.

More formally, referring to Fig.8, repeatability is verified when the following condition is met $\forall$ k $\in$ N, k $\geq$ 1, if $x(t_k) = x(t_{k-1})$ then $\mu(y(t_k)) = \mu(y(t_{k-1}))$, where $y(t_k) = F(x(t_k))$ and $y(t_{k-1}) = F(x(t_{k-1}))$.

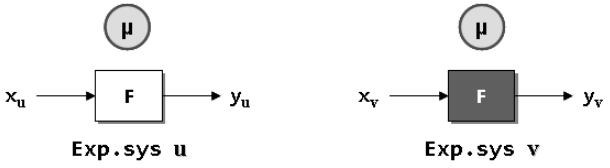


$\forall$ k $\in$ N, k $\geq$ 1, if $x(t_k) = x(t_{k-1})$
then $\mu(y(t_k)) = \mu(y(t_{k-1}))$

Fig. 8. Measurement criteria: repeatability

6.3 Reproducibility

Reproducibility is a spatial criterion associated to measurement results that can be obtained when a given experiment performed on a given experimental system u ($\in$ S $\equiv$ experimental system set) is replicated over a similar but different experimental system v ($\in$ S). This can mean different experimental platform, operating system, etc.



$\exists$ u, v $\in$ S such that if $x_u = x_v$
then $\mu(y_u) = \mu(y_v)$

Fig. 9. Measurement criteria: reproducibility

Typically, reproducibility comes into play when a third party performs the same experiment to determine the scientific validity of the output of an experiment. More formally, referring to Fig.9, reproducibility is achieved when $\exists$ u, v $\in$ S such that if the input vector $x_u = x_v$ then $\mu(y_u) = \mu(y_v)$, where $y_u = F(x_u)$ and $y_v = F(x_v)$.

6.4 Verifiability

The results of experimental measurements are verifiable if the output of the experimental corpus modeled by the function F: $x(t) \rightarrow y(t) = F(x(t))$ can be confirmed against a formal model H: $x(t) \rightarrow H(x(t))$; implying that the measurement results shall comply with the output of the model H (output described as a function of the input vector and the experimental parameters).

Referring to Fig.10, measurement results are verifiable if there exists a formal model H: $\Re^n \rightarrow \Re$: $x(t) \rightarrow H(x(t))$ and $\varepsilon << 0$ such that at time t_k, $H(x(t_k)) \in [\mu(y(t_k))-\varepsilon;\mu(y(t_k))+\varepsilon]$, where $\mu(y(t_k)) = \mu(F(x(t_k)))$. One often considers that verifiability is achieved by comparing the results of an experimental measurement against a reference system RS (assumed as representative of the real system): $\exists$ u $\in$ S and $\varepsilon << 0$ such that if $x_u = x_{rs}$ then $y_{rs} \in [\mu(y_u)-\varepsilon;\mu(y_u)+\varepsilon]$.

Achieving verifiability for a representative sample (to avoid sampling bias) of unbiased measurement results whose size is determined so as to reduce the sampling error (and satisfy a given confidence interval and level given the finite but often large number of available results) enables in turn to generalize the conclusion(s) that can be drawn from experimental measurements.

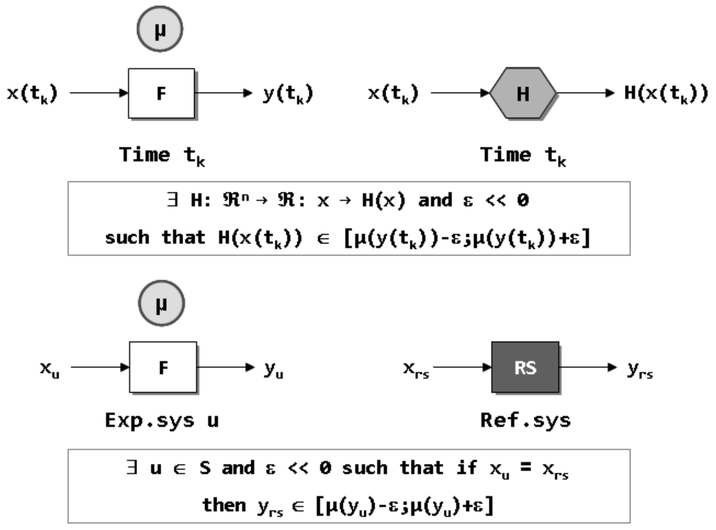

μ
x(t_k)
F
y(t_k)
x(t_k)
H
H(x(t_k))
Time t_k
Time t_k
∃ H: ℜⁿ → ℜ: x → H(x) and ε << 0
such that H(x(t_k)) ∈ [μ(y(t_k))-ε;μ(y(t_k))+ε]
μ
x_u
F
y_u
x_rs
RS
y_rs
Exp.sys u
Ref.sys
∃ u ∈ S and ε << 0 such that if x_u = x_rs
then y_rs ∈ [μ(y_u)-ε;μ(y_u)+ε]

Fig. 10. Measurement criteria: verifiability

7 Conclusion

Starting from the initial objectives of the FIRE initiative and its associated objective of dynamic between elaboration, realization, and validation by means of iterative cycles of experimentation, this chapter positions measurement-based experimental research as a continuum: starting from more abstract tools (not only because their resulting cost is lesser but because such tools produces results verifying all criteria explained in Section 3) followed by progressive addition of realism as part of the experimented system to ultimately reach the so-called field trials with real systems. The addition of realism at increasing cost (resulting from the increasing complexity) is the main purpose of performing experimentation by means of emulation or real systems. However, achieving verifiable, reliable, repeatable, and reproducible emulation results at best cost-complexity (thus experimentation time) can not be achieved if experiments are limited to random trials on emulated platforms. Indeed, emulation experiments can lead to reproducible and repeatable results but only if their "conditions" and their "executions" can be strictly controlled. Realism can thus be improved compared to simulation (in particular for time-controlled executions of protocol components on real operation system). Nevertheless, such experiments are more complex and time consuming to configure and execute.

In order for measurement-based experimental research to reach this objective: 1. This chapter has proposed a systematic experimental research methodology which needs to be commonly shared and applied by projects in order to ensure the scientific validity of the experimental results they produce (otherwise, leaving them as only purpose the proof of executability of implementation instances of experimental corpus); 2. The validation by experimentation of the targeted "elaboration and realization" being actually a continuum, it requires in turn a set of well-defined experimentation tools to systematically implement this methodology starting from more abstract tools such as simulation (not only because their resulting cost is lesser but because such tools produces results verifying all conditions explained in this chapter) followed by progressive addition of realism as part of the experimented system, e.g., by means of emulation, to ultimately reach the so-called field trials with real systems; and 3. As experimental measurements play a fundamental role in experimental research, developing a systematic measurement methodology is crucial to achieve the objective of validation by experimentation of the various project outcomes including protocols, systems and components. Moreover, in order to meet scientific validity criteria, the measurement results obtained when performing experimental research implies the availability of reliable and verifiable measurement tools providing repeatable and reproducible measurement results. Finally, as the amount of collected data increases (due to the scale of experiments), the development of measurement tools shall not be limited to actual measures and their collection but also on-line analysis and mining of measurement data.

Acknowledgements. This research work is conducted by the EULER Project (Grant No.258307) part of the Future Internet Research and Experimentation (FIRE) objective of the Seventh Framework Programme (FP7) funded by the European Commission (EC).

References

1. EULER FP7 Project, Performance objectives, evaluation criteria and metrics. Technical report, `https://www-sop.inria.fr/mascotte/EULER/wiki/pmwiki.php/Main/Deliverables`
2. Shannon, R.E.: Systems Simulation the Art and Science. Prentice-Hall, Englewood Cliffs (1975)
3. Ingalls, R.G.: Introduction to Simulation. In: Proceedings of the 34th Winter Simulation Conference, San Diego, CA, USA (December 2002)
4. International Organization for Standardization (ISO), International vocabulary of metrology - Basic and general concepts and associated terms (VIM), ISO/IEC Guide 99 (2007)
5. Abran, A., Sellami, A.: Initial Modeling of the Measurement Concepts in the ISO Vocabulary of Terms in Metrology, Software Measurement and Estimation. In: Proceedings of 12th International Workshop on Software Measurement (IWSM), October 7-9, Shaker-Verlag, Aachen (2002)
6. Fiok, A.J., Jaworski, J.M., Morawski, R.Z., Oledzki, J.S., Urban, A.C.: Theory of measurement in teaching metrology in engineering faculties. Warsaw University of Technology, PL-00-661 Warsaw

Experimental Performance Evaluation of Sensor-Based Networking for Energy Efficiency in Smart Buildings

Constantinos Marios Angelopoulos and Sotiris Nikoletseas

University of Patras and CTI, Greece
aggeloko@ceid.upatras.gr, nikole@cti.gr
http://students.ceid.upatras.gr/~aggeloko/
http://www.cti.gr/RD1/nikole/

Abstract. This chapter presents measurements based experimental research carried out in the context of the EU/FIRE project HOBNET on on energy efficiency, with application to green, smart buildings of the Future Internet. In particular, we discuss two aspects of experimentation: a) experimental evaluation of sensor network protocols (with a focus on how to efficiently and distributively achieve energy balance in the network towards prolonging its lifetime); critical components of such experimental research (reference deployments, mobility profiles, performance metrics and their trade-offs, protocol properties and families) are also discussed b) specialized application commissioning (which we exemplify via presenting a smart watering application developed in the project). Our experience suggests that measurements based experimental research can nicely complement rigorous performance analysis and simulation based evaluation providing more realistic performance results and contributing to the validation and fine tuning of the high level algorithms. In this respect, we note that current ad hoc approaches are useful but there is a need to further converge to widely accepted, common integrated methodologies, systems and tools.

Keywords: measurements, experimental test-beds, energy efficiency, smart/green buildings, wireless sensor networks.

1 Introduction

Towards a solid development of experimental research, key critical components have to be investigated, defined and developed. This way the community will have common points of reference and a well established experimental framework within which experiments will be designed and corresponding results will be evaluated. We below discuss a few representative components.

Reference Deployments Used. The protocol, algorithm or system under evaluation can be tested against *structured topologies*, such as star, grid or mesh, as well as against *randomized deployments* such as random proximity graphs and nearest neighbour graphs. Another aspect of network deployment addresses the type of devices consisting the network. So, a network may be comprised of a unique type of sensors in a *homogeneous deployment*, or a mix of sensors of high and low capabilities in a *heterogeneous deployment.* A third aspect of network deployments is the spatio-temporal deployment of nodes. In the time domain all nodes of the network can be deployed at the beginning of the experiment, thus having a *flat deployment*, or an *incremental deployment* can be adopted, where network nodes are added throughout the network evolution in order to overcome issues such as bottleneck effects, node failures or network disconnection. In the spatial domain, the placement of the sensors in the network area can follow the uniform distribution, thus leading to a *uniform node density*, or follow a non uniform distribution that leads to *high density diversity.* Finally, according to the nature of the protocol/algorithm under evaluation, a *static* or *mobile* deployment may be preferred as well as a hybrid combination of the two.

The Mobility Factor. When studying mobile sensor networks emphasis needs to be given on *highly dynamic and diverse* motion profiles; this is important because of the corresponding high dynamics of real world scenarios, systems and applications. Such dynamic profiles can be modelled via states that each one describes a motion p (i.e. speed, direction, etc) and by probabilities to shift to a different state given the current one. The set of states and transition probabilities define motion profiles that can be represented via state transition diagrams. Furthermore, particularly for the Control Center (Sink), novel patterns of *accelerated random Sink mobility* need to be investigated. Some examples on this topic include inertia random walk, stretched walk and walk with limited memory; such walks aim to reduce data collection latency by avoiding node overlaps. Finally, new network parameters, like the mobility level capturing both the speed and direction of movement, need to be investigated towards exploiting diverse and dynamic node mobility. An example of the utility of such research is the fact that node mobility can act as a low cost replacement of connectivity and fault-tolerance in sensor networks.

Performance Aspects and Metrics. In the performance evaluation of a network algorithm there exist critical issues that need to be thoroughly investigated. *Scalability* is a crucial aspect related to the performance of the network in relation to its size. While scaling up the size of the network even the correctness of the evaluated protocol/algorithm may be affected. Therefore, systems should be evaluated against very large input sizes (e.g. large number of sensors, long period of time, high event generation rate). Another aspect is how does the network cope with failures, i.e. the *fault tolerance* of the network. Towards that direction, diverse fault models are needed (e.g. temporary/permanent failures, off-line and on-line, random and worst case, etc). *Inherent performance trade-offs* in the network operation emerge (most notably energy vs time) or even competing goals and variations of the same global performance aspect.

Typical examples include minimizing the total energy spent in the network, balancing the energy dissipation or combining energy efficiency and fault-tolerance. Other aspects relate to the *application dependence* of protocols/algorithms, *dynamic changes* and *heterogeneity.*

Protocol Properties and Families. Consolidating a rich, diverse experience of experimentally evaluating protocols and algorithms, the research community may come up with a taxonomy of protocol/algorithm properties and families. A variety of primitive protocols already exists and by combining them hybrid combinations can emerge. Possible nice properties include adaptation with respect to dynamics and diversities, computational simplicity in order to address the constrained nature of sensor motes, randomization in algorithmic decision, distributedness and locality in order to cope with the ad-hoc and distributed nature of WSNs and their applications.

2 The HOBNET Project on Green, Smart Buildings

The HOlistic Platform Design for Smart Buildings of the Future InterNET - HOBNET project is a Specific Targeted Research Project (STREP) under the Future Internet Research and Experimentation - FIRE initiative ([5]). The main objective of HOBNET is to ease and maximize the use of FIRE platforms by multidisciplinary developers of Future Internet applications focused on automation and energy efficiency for smart/green buildings.

The consortium composition represents a carefully chosen mix of the complementary expertise and experience needed to achieve the project objectives, having also the right balance between the theoretical/algorithmic and technological/practical aspects of the planned research. In particular it consists of the Computer Technology Institute (CTI-Patras), the Ericsson Serbia (EYU), the Geneva based public utility foundation Mandat International (MI), Sensinode (Finland), the University College Dublin (UCD-Ireland), the University of Edinburgh (UEDIN) and the University of Geneva (UNIGE).

We note that, the consortium has the appropriate mix of three types of participating organizations (academia, industry, end-users) necessary for the entire project cycle (including end user specifications, experimentally validated research, platform integration, dissemination and exploitation).

HOBNET addresses algorithmic, networking and application development aspects of Future Internet systems of tiny embedded devices. These include:

a) an *all IPv6/6LoWPAN infrastructure of buildings* and how IPv6 can integrate heterogeneous technology (sensors, actuators, mobile devices etc.)
b) *6lowApp standardization* towards a new embedded application protocol for building automation
c) novel *algorithmic models and scalable solutions* for energy efficiency and radiation awareness, data dissemination, localization and mobility
d) rapid development and integration of *building management applications*

e) support for the *deployment and monitoring* of resulting applications on *FIRE test beds.*

The HOBNET research is voluntarily following a user requirements approach, in order to integrate the end user perspective in the research project from the beginning. In particular three end user perspectives were taken into account: a)the building owner (i.e. Mandat Intl.) were the final deployment of the HOBNET architecture will occur b) delegates who are using the building and c)persons working in the building or in charge of the technical maintenance of the building [1]. These include among others safety, comfort and energy saving. The targeted innovations will benefit and exploit two FIRE facilities (CTI-Patras and University of Geneva). The innovations will finally be deployed and demonstrated in a test-bed building with real end users in Geneva, a welcome centre managed by Mandat International to provide support to delegates from developing countries attending UN conferences. If successful and convincing, those innovations will be integrated in larger project of International Cooperation House to support delegates attending UN conferences, with a high visibility.

HOBNET takes a holistic approach addressing critical aspects at different layers (networks, algorithms, applications/tools) in an integrated way, including the following hierarchy:

– At the low level, *network protocols and architectures, mainly based on IPv6*, are studied, with an emphasis on heterogeneity and interoperability.
– At a second layer, we provide *algorithmic models and solutions* for smart buildings, with a special care for scalability.
– An interface layer for the rapid development and the evaluation of *building management applications* is provided at a third level.
– Finally, proposed research solutions and key innovations are organically evaluated in the context of the *platform integration.*

HOBNET intends to use and integrate an end-user perspective in the research cycle, together with all the research partners. Towards this direction the consortium has compiled a list of consolidated scenarios to be implemented. These include local adaptation to presence, emergency management, electric device monitoring, CO2 monitoring, maintenance control, customization, building 3D visualization & monitoring, identification via mobile phones, user awareness, oil tank monitoring, garden watering and resource tracking and monitoring.

3 The Main HOBNET Testbeds

Key research innovations of HOBNET are experimentally validated in three test-beds that act as smart building proof of concept applications. These test-beds are consisted of highly diverse deployed devices (sensors, actuators, etc) and each one has different thematic emphasis. In particular, the Mandat International test-bed focuses on user awareness and technology integration, the CTI test-bed focuses on energy management and data collection protocols and the UNIGE test-bed focuses on target tracking and localisation.

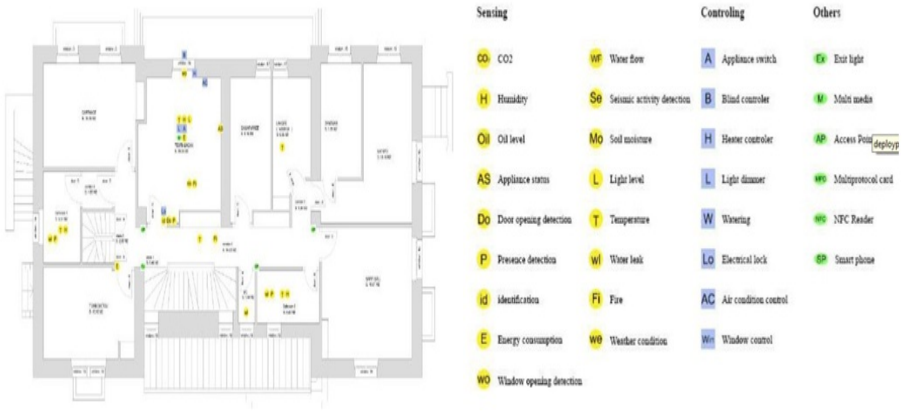
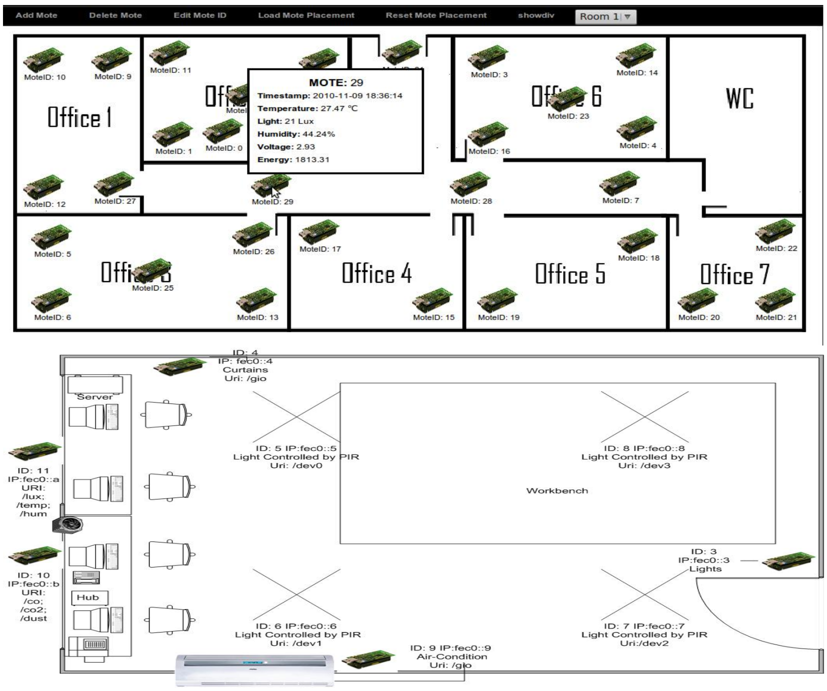


Fig. 1. The Mandat International test-bed

The Mandat International Test-Bed. The MI test-bed consists a highly visible showcase of European green and ICT technologies close to the European headquarters of the UN. It is the test-bed where the final integration of all the components of the HOBNET architecture will occur. These namely include:

– *Core HOBNET components*: HOBNET Resource Directory and Building Web Service Proxy
– *HOBNET interfaces*: IPv6-based Open Building Interface and 6LoWPAN-based Embedded Building Interface
– *Sensors and actuators*
– *Other components and services*: Building Control and Monitoring System, Energy Analysis service, Mobile Applications, 3D Visualisation as well as Remote Applications
– *Sensors* deployed at the MI testbed include: CO2, humidity, oil level, appliance status, door opening detection, presence detection, identification, energy consumption, window opening detection, water flow, seismic activity detection, soil moisture, light level, temperature, water leak, fire detection. *Actuators* to be deployed include: appliance switch, blind controller, heater controller, light dimmer, watering, electrical lock, air condition control. *Other components* include: exit lights, multimedia, access points, multi-protocol card, NFC reader, smart phones.

The CTI Test-Bed at Patras. The HOBNET CTI test-bed at Patras consists of 35 wireless sensor motes (TelosB architecture [11]) spanning across seven offices that form a multi-hop mesh network. Sensor motes deployed in the smart room of the test-bed act either as sensors monitoring ambient environmental conditions (luminance, temperature, air humidity, CO, CO2, dust levels) or as actuators controlling various electromechanical devices (e.g. lights, curtains, air-conditioning



MOTE: 29
Timestamp: 2010-11-09 18:36:14
Temperature: 27.47 °C
Light: 21 Lux
Humidity: 44.24%
Voltage: 2.93
Energy: 1813.31

Fig. 2. The whole CTI test-bed and the smart-room layout

unit, ventilation, etc). Furthermore, these motes are IPv6/6LoWPAN [16] [13] enabled, and each one is assigned a unique IPv6 address. On top of the network stack runs the Constrained Application Protocol (CoAP)[14] allowing representation of the various sensors and actuators as resources by using URIs, thus implementing a RESTful architecture [17]. Several application scenarios (smart watering, emergency evacuation, CO2 monitoring, room adaptation to presence) have already been implemented using the HOBNET architecture.

As part of the CTI test-bed, a demo smart/green room is also developed at the PROKAT building of the CS Department at the University of Patras. In this demo-room we demonstrate selected green/smart building scenarios. By using the sensory data collected the sensors control (via actuators) the various devices (lights, air-condition, heater, appliances, etc.) in order to improve the energy efficiency in the building and improve the comfort levels of the users, e.g. turn off the light/air-conditioning when people walk out of the room, open the curtains when there is enough sun light outdoors etc. The devices are driven by *Control Cube*, a hardware interface between the wireless sensor network and the electromechanical infrastructure of the demo room that the CTI-Patras group has developed. In the near future, this demo room will be (partially) open to FIRE users who will be able to connect to it via a web interface and test it by running their own end-user scenarios.

The UNIGE test-bed at Geneva. A sensor network test-bed, extending an existing FIRE/WISEBED facility [15], is created at the University of Geneva. It includes several motes (both iSense and TelosB) and robotic devices. For hardware, the lab is equipped with about 60 iSense motes with various sensing capabilities (temperature and light sensing, AMR, passive infrared, accelerometers and imaging capabilities), 30 TelosB nodes and 2 Surveyor SRV-1 Robots. At the lab, it is also deployed a wired infrastructure with 5 Dual Core Atom based small servers to take care of massive flashing and reporting without using the wireless medium. Recently, the UNIGEs local lab has been further enriched with additional TelosB motes as well as with 49 Libelium Waspmotes for running algorithmic experiments under real hardware deployment. Several HOBNET scenarios (with a focus on resource tracking) are implemented in that test-bed as well.

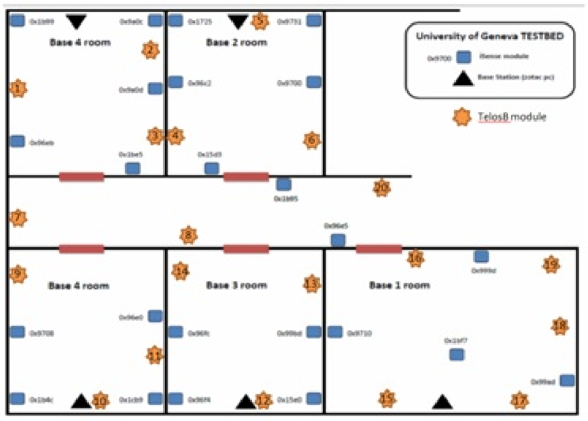


Fig. 3. The UNIGE test-bed

4 Experimental Evaluation of Sensor Network Protocols (Energy Balance)

In this section we present key aspects (hardware, software, topology, networking) of *SenseWall*, a HOBNET experimental sensor network test-bed we have created for the implementation and engineering of distributed sensor network algorithms. This is different to the test-beds in the previous sections since it focuses not on building automations but instead on protocol performance evaluation. We then describe how SenseWall has been in particular used to implement two recent state of the art algorithms for energy balanced sensor data propagation. We elaborate on the issues and challenges created by the restrictions and particularities of the experimental test-bed and how we dealt with them. We also carry out a detailed performance evaluation comparing the energy balance protocols to two baseline protocols that include only either single hop or direct data transmissions.

The Energy Balance Problem. We study the problem of how to achieve energy-balanced data propagation in distributed wireless sensor networks. The energy balance property guarantees that the average energy spent per sensor is the same for all sensors in the network at any time during the network operation. This property is crucial for prolonging the network lifetime by avoiding early energy depletion of sensors and the non-utilization of residual energy on sensors. It is particularly relevant to smart, green buildings of the Future Internet where wireless sensors are used to propagate data related to current ambient conditions (light, temperature, human presence, etc); by balancing the energy dissipation among the sensors in the deployed network its smooth operation is prolonged and the provision of building optimization services is further facilitated.

In this research (see also [6] for details) we have implemented and experimentally evaluated in our test-bed two energy balancing protocols and compared them to two pure data propagation schemes; multi-hop routing and routing with direct transmissions only.

4.1 The Routing Protocols

The Distance-Based Energy Balance Protocol. Lifespan maximization can be achieved by keeping energy balance. The energy balance property guarantees that the average per sensor energy dissipation is the same for all sensors in the network. This property is important since it prolongs the lifetime of the network by avoiding early energy depletion of sensors. To overcome an unbalanced energy consumption scheme, the authors in [2] propose a mixed routing strategy where in each step the algorithm decides probabilistically and locally whether to propagate data one-hop towards the *Sink*, or to send it directly to the *Sink*. This randomized choice balances the one-hop transmissions with the direct transmissions to the *Sink*. Thus, a trade-off emerges between cheap but slow multi-hop propagation and direct but energy consuming transmissions that "bypass" the sensors lying close to the *Sink*, thus propagating data fast. Note that, in most protocols, motes close to the *Sink* tend to be overused and die out early resulting in a disconnected network. In [2], via detailed analysis, the probabilities for each propagation choice are estimated in order to guarantee energy balance.

Authors describe the protocol in such a way that all needed estimations can easily be performed by sensors of current technology, by using simple to obtain information. However, in order to bring the protocol down to real sensors, we need to revise some of the assumptions originally made in [2] for analysis purpose. We also need to cope with the restrictions imposed by the programming language (NesC [4]), the operating system (TinyOS [8]) and the sensor motes themselves (TelosB). These restrictions apart from traditional limitations of Wireless Sensor Networks (i.e. limited energy, low computational power) also include, lack of floating point numbers, limited program memory and no support for dynamic memory allocation.

The Energy-Based Protocol. In [3] the authors revisit the family of mixed strategy routing schemes and propose an on-line distributed algorithm for lifespan maximization. In each step, the algorithm decides to propagate data either one-hop towards the *Sink*, or to send it directly to the *Sink*. This decision is based on explicit information about the energy spent by the current node and the energy spent by the nodes that are one-hop closer to the *Sink* than itself.

Let n be a sensor. V_n is the neighborhood of node n. Node n knows the energy spent by each node in its neighborhood. Let m be the sensor of neighborhood V_n with the lowest energy spent. When n holds data it makes the following decision:

– If node n has spent more energy than m, then n sends the message to m (spending one energy unit).
– Otherwise, n sends the message directly to the *Sink* spending d^2 energy units, where d is the distance from n to the *Sink*.

The Pure Multi-hop Protocol. In this protocol every node propagates data to the sink in a multi-hop way. In particular, let n be a sensor and i the sector it belongs to. The node n can forward its own generated data and relay data from previous sectors $(i+1, i+2, \cdots)$ by forwarding messages to a random node of the next sector $i-1$ towards the sink. This protocol is cheap but its latency can be high since it creates a bottleneck region around the *Sink*.

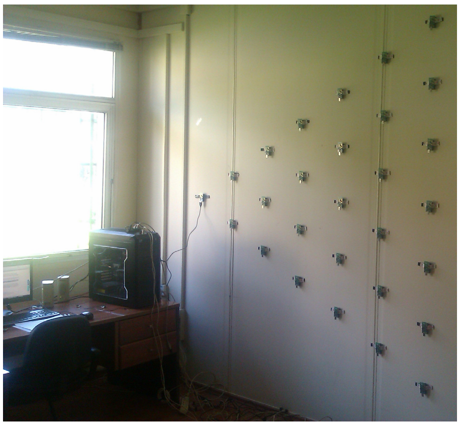
The Pure Direct Transmission Protocol. In contrast to the previous pure multi-hop protocol, in this direct transmission protocol every node of the network sends the data that generates directly to the *Sink* with maximum transmission power. This protocol propagates data very fast but its energy dissipation is very high.

4.2 Hardware Architecture and Networking

The hardware components of our experimental test-bed include a set of 28 TelosB motes connected to a control Base Station PC via a USB tree which consists of USB cables and hubs, for controlling the network and collecting experimental data from the motes without interfering with the wireless communications, thus leaving the wireless medium free for the routing algorithms. We deployed the motes in a sector-shaped topology as shown in Fig. 4 in order to approach the theoretical model of the algorithms described in [2] and [3].

The motes of the test-bed are connected to the Base Station desktop PC via USB cables (see Fig. 4) and each mote is also supplied with two AA rechargeable batteries. Via the USB interface we easily control the whole network (mass flushing/programming of the nodes, reseting the nodes, etc.) and receive packet-statistics through the wired USB backbone.

Desktop PC runs the MySQL server and the MoteProgrammer Java application. The *Sink* sends the received messages to the PC using the serial UART interface and then the Java application stores them in a MySQL database.



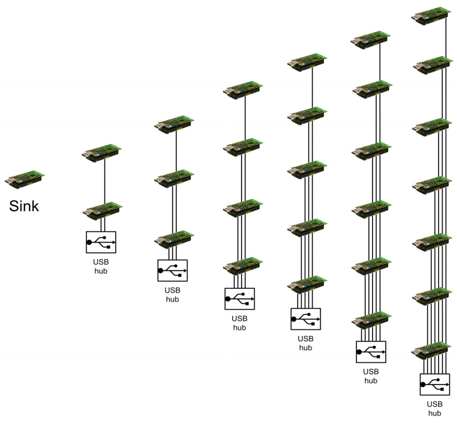


Fig. 4. The USB-wired node topology

The MySQL server is used to log the data from the experiment while the MoteProgrammer application enables the user to control the nodes by flushing, resetting, changing the event generation rate, changing the sampling rate, etc. and to perform off-line analysis of the collected data by making queries for the various performance evaluation metrics.

4.3 Software Architecture and Algorithms

Distance-Based Algorithm Implementation. We implement the energy balancing protocol described in [2] in TinyOS 2.1.0. The module consists of three files. A header file where all necessary structs, message types and macro-variables are defined; a configuration file where the wiring among the components used is defined; and a source file with the source code of the module we implement.

We define *pimsg_t*, which is the message type that we will be using in this implementation. Each message, apart from sensory data, will also contain information about the source of the event (i.e. the ID of the mote that initially generated the event) and the current number of hops the message has made during its propagation so far. Also, we define the array *neighbours*. This array holds the ID's of the motes of the next sector towards the *Sink*, which consists of all the motes that are one hop closer to the *Sink* than the current mote. Finally, every mote maintains locally a counter *msg_counter* that counts the number of events it has generated. This counter is used in the calculation of success rate.

In the configuration file we declare that we will use *DemoSensorC* for generating events, as if we were using real sensors. The reason we made this option is that we are focusing on data propagation and not on monitoring. We also use the *RandomMlcgC* component as random generator. This component is a a fast implementation of the Park-Miller Minimal Standard Generator for pseudo-random numbers and it provides an interface that receives a seed for initialisation. As seed we choose to use the readings from the *VoltageC* component, that is the current voltage that the mote is supplied with. We consider this value to contain adequate randomness, as it depends on many factors (i.e. how long the batteries have been unused). In order to dynamically set the transmitting power of each mote we use the *CC2420PacketC* component.

At the beginning every mote, except for the *Sink*, commences the event generation procedure. Events are generated locally in a periodic manner; therefore the corresponding procedure is controlled by a timer. Every time the timer is fired the mote increases by one the variables indicating the number of events generated so far. Based on which sector the mote belongs to, it decides with probability P_i to transmit the data directly to the *Sink*, or with probability $1 - P_i$ to propagate its data to the next sector. The formula that provides P_i is rigorously computed in [2] to be equal to

$$P_i = 1 - \frac{1}{(i+1)(i-1)}$$

where i the number of hops from the current sector to the *Sink*. Using the component *RandomMlcgC*, the mote generates a random 16-bit number. Then, the mote transmits directly to the *Sink* or propagates to the next sector based on whether the following control is true or not (correspondingly):

```
if ( DiceResult % inverse-Pi  == 0 )
```

where *DiceResult* is the random generated number, % the modulo operator and *inverse-Pi*= $1/P_i * 10000$. The latter is due to the fact that TelosB motes use the MSP430 micro-controller that does not have a hardware floating point unit and therefore fixed point arithmetic is used. If the mote decides to propagate the data to the next sector, then another random number is generated. Then, based on that number, a mote from the next sector is chosen (array *neighbours*) and data are sent to it.

This procedure is also followed when the mote receives data from another sector. It probabilistically decides whether it is going to transmit them directly to the *Sink* or if it is going to propagate them to the next sector.

Energy Based Algorithm Implementation. In order to implement this protocol in TinyOS for TelosB motes, we follow the same architecture as in the previous section. We use the same components and interfaces for starting up and handling the necessary hardware modules (i.e. radio, LEDs, etc), as well as generating random numbers and handling the transmission power. However, in this case we use an extra type of message, $NRGmsg_t$. Each mote uses this type of messages to periodically collect the remaining energy of the nodes of the next sector towards the *Sink* that holds in an array $neighbhours - nrg$. The frequency at which every node updates $neighbhours - nrg$ is much smaller than the event generation rate in order not to induce significant overhead to radio communication. Finally, every time a mote has data to route (either data that generated itself, either data that the mote relays), it compares its own remaining energy to the highest value of the array $neighbhours - nrg$. If the remaining energy of the mote is higher, then it transmits directly to the *Sink*. Otherwise, if there is a mote to the next sector that has more remaining energy (has spent less energy), then the mote propagates the data to that mote.

The MeasurementsLogger Component. We implement a new component in TinyOS 2.1.0 as a binding interface between the desktop software control application and the routing protocol we evaluate on SenseWALL. The role of this component is to setup the parameters of the experiments (i.e. event generation rate, energy sampling rate, experiment duration, etc.) and to monitor the evolution of the routing protocol by enabling the logging of the performance measurements described in 4.4.

The module consists of four files. One containing the signature of the interface provided by MeasurementsLogger component, a header file where all necessary structs, message types and macro-variables are defined; a configuration file where the wiring among the components used is defined; and a source file with the source code of the module we implement. The header file can be automatically created by the Java application and contains all the macrovariables that correspond to the parameters of the experiment.

Software to Control SenseWALL. The Java application that runs in the Base Station automatically detects the motes that are connected to the desktop PC via USB ports and allows the administrator of the network to interact with the motes. Also, the Desktop application provides an interface for computing the following performance evaluation metrics: a) **data delivery latency** b) **average energy consumption** and c) **success ratio** (see definitions in next section). An overview of the system architecture is depicted in Figure 5.

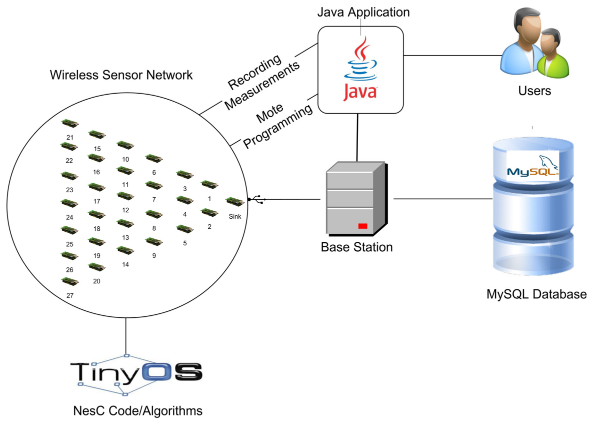


Fig. 5. Software Component Architecture

4.4 Performance Evaluation

Experimental Setup. Each node generates a total of 1.000 events/messages with a rate of 0.2 events/second. Every 50 events (or 250 seconds) a node sends its current energy to the *Sink*. Before conducting any experiment we first fully recharge the batteries in order to have a common reference for all experiments.

Metrics Computation. We compute the above mentioned metrics as following:

a) **data delivery latency** which is the average number of hops needed to reach the *Sink*. Each generated message has a *hop_counter* field. Each time a node relays a message, increases the *hop_counter* field before forwarding the message. In order to calculate the total data delivery latency of the node, we add the *hop_counter* of all received messages.

b) **average energy consumption** is the average energy spent per node during the network operation (we measure the energy consumption per node by subtracting the battery level of the node at the end of the experiment from the initial battery level of the node before the start of the experiment). In order to measure the energy consumption of the motes, we run our experiments by deactivating the USB power supply and the motes use only the battery supply. The motes periodically sample the current battery level using the Msp430InternalVoltage component of TinyOS. At the end of the experiment, we turn on the USB supply and via the Java application motes send their energy measurements to the Base Station that stores all the collected data to a MySQL database.

c) **success ratio** is the ratio of the total number of packets received by the sink to the number of packets generated by the whole network. Each node assigns to a new event/message that generates an incremental packet sequence number *msg_counter*. Upon reception on *Sink* of a node's generated message, we divide the total number of received messages from that node by the maximum packet sequence number (*msg_counter*) of that node.

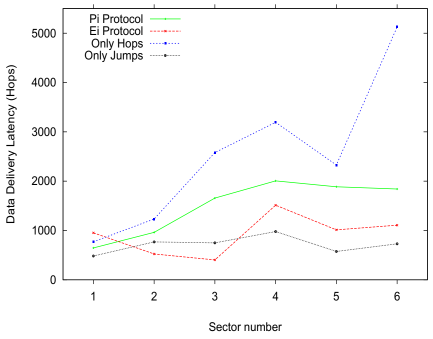


Fig. 6. Average data delivery latency per sector

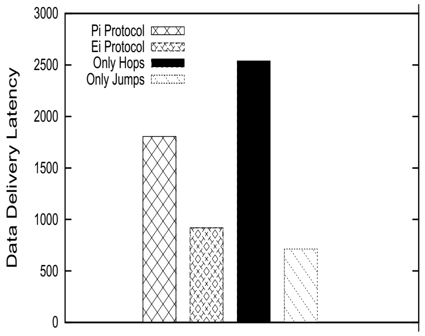


Fig. 7. Total average data delivery latency

Experimental Results. Figures 6 and 7 depict the average data delivery latency per sector and in total for all four protocols. In general, we note that as the distance between the generated event and the *Sink* is increasing, the time the message requires in order to be delivered is growing proportionally. Particularly, the multi-hop propagation scheme is severely affected by the distance, in terms of latency. In contrast, when directly transmitting to the *Sink* distance does not affect the time delivery of messages. This is due to the hybrid nature of these two protocols. Note that due to the different way E_i and P_i decide whether to make a jump or to propagate to the next sector, there is a significant difference in energy consumption. As E_i is more prone to direct transmissions, it also has shorter data delivery times.

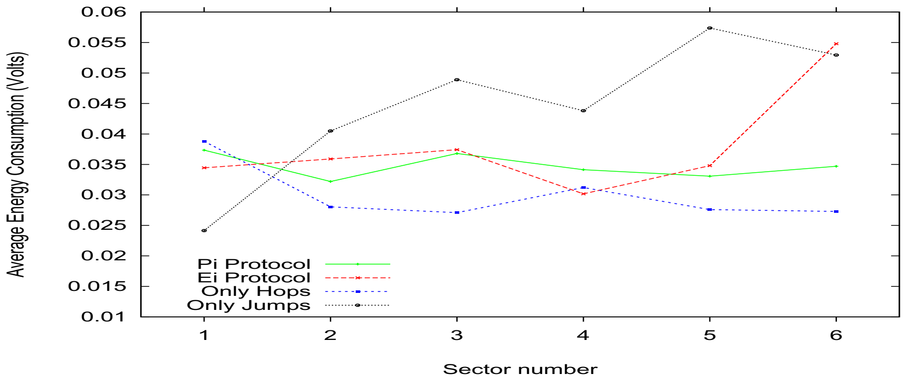


Fig. 8. Average energy consumption per sector

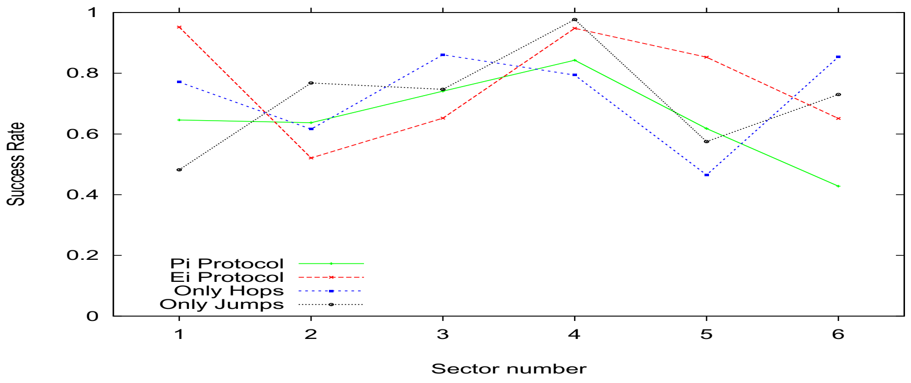


Fig. 9. Success ratio per sector

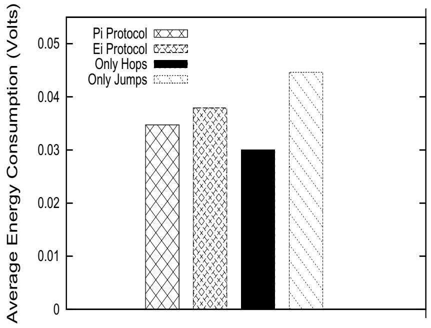


Fig. 10. Total average energy consumption

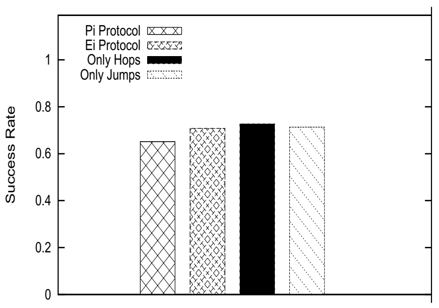


Fig. 11. Total success ratio

Figures 8 and 10 depict the findings about energy consumption. As expected, when using direct transmissions in order to deliver data to the *Sink*, motes lying in distant sectors consume even 240% more energy than motes close to the *Sink*. On the other hand, when using multihop propagation, motes closer to the *Sink* consume nearly double the energy compare to the rest of the network. This is due to the bottleneck effect these motes come up with; they serve as relays to the entire network flow. The P_i protocol actually manages to balance the dissipated energy across the network as all sectors, more or less, dissipate the same amount of energy. In total, again the multihop propagation scheme and direct transmission represent the two extremes with the other two protocols lying between them. Note that the E_i protocol consumes slightly more energy than the P_i protocol. This is due to the fact that E_i is more prone to direct transmissions than the P_i protocol.

Figures 9 and 11 depict the success rate with which each protocol delivers data to the *Sink*. In a per sector basis, we note that all protocols follow the same pattern. The variation of the results are due to the big number of motes deployed in a very confined space (27 motes in a $5 \times 3m^2$ area). However, in total all protocols perform more or less the same in terms of success rate.

5 Specialized Application Commissioning (Garden Watering)

As water supplies become scarce and polluted, there is an urgent need to irrigate more efficiently in order to optimize water use. Particularly, the efficient long-term sustainable management in the context of urban watering is of the out-most importance for maintaining the socio-economical benefits of water resources. In this context, smart buildings play a key role when it comes to efficient water management in cities. In this section, we present a HOBNET/WSN based, smart home-irrigation system that consists of heterogeneous motes, special sensors and actuators. The system is fully adaptive not only to environmental conditions but also to the specific water needs that different plants may have. This way, it manages to perform efficient home irrigation, while it provides an IPv6-capable managing system. More details can be found in [7]

5.1 System Description

Our system architecture includes sensor motes, soil humidity sensors, mote driven electro-valves that control the water flow towards the plants, and a Java application running on a PC, that collects data from the sensor network and stores them in a MySQL database. For the purposes of this study, we abstract a regular home garden with three pots containing different plants with highly diverse watering needs; a geranium that has very limited watering needs (once a week), a lavender that under normal weather conditions has medium watering needs (three times a week) and a mint that requires regular watering (in high temperatures during summertime, even twice a day).

The soil humidity of each pot is monitored by a mote equipped with soil humidity sensor. The watering of each pot is controlled by a corresponding mote-driven electro-valve independently from the other pots. When the soil of the pot is too dry, then the corresponding mote that monitors soil humidity, informs the mote that drives the corresponding electro-valve to start watering the pot. When the soil humidity returns to normal levels, the soil monitoring mote signals the electro-valve mote to cut off water supply for that pot.

Throughout the operation of the system, the levels of the soil humidity of each pot are forwarded to the Sink by the corresponding motes. The Sink is a mote connected to a PC on a USB port that acts as a gateway for the rest of the motes. When it receives a soil humidity measurement, it forwards it to the PC where a Java application receives data and stores them in a MySQL database for post-processing. Apart form soil humidity, measurements concerning temperature are also forwarded to the Sink and logged at the database.

5.2 Hardware Description

Sensor Motes. For our implementation we used two mote platforms, Telosb [11] and IRIS motes [10]. Both of them are ZigBee compliant, small, light weight and when using energy saving protocols can be powered with two AA batteries for several weeks, even months. These characteristics makes them ideal for our smart garden watering system as they can easily be deployed everywhere while being independent of power installations. Furthermore, IRIS was combined with the MDA100CB sensor and data acquisition board which has a precision thermistor, a light sensor/photocell and general prototyping area. This prototyping area was used in order to connect a relay, through which IRIS motes were able to control the electro-valves.

Soil Sensors. The EC-5 soil humidity sensor by Decagon [9] was used for soil monitoring. It consists of a cable, which on one end has two prongs and on the other end has 3 wires. The prongs are pushed inside the potting soil and the three wires of the other end are connected to the 10-pin expansion connector of TelosB motes (Fig 12). The bare wire is connected to the ground pin, the red one is connected to the ADC channel pin (programmed as input) and the white wire is connected to the VCC pin. This sensor had to be used along with TelosB motes as it provides 12-bit data, while the IRIS mote has a 10-bit ADC.

For successful communication between TelosB and the soil humidity sensors for collecting soil humidity data, we used the component Msp430Adc12ClientC and the corresponding interface. In the source code we set the ADC1 (i.e. pin 5 of the 10-pin TelosB expansion connector) in which we had previously shouldered the red wire (analogue out) of the soil humidity sensor. We, also, used the component Sensirion SHT11 of TelosB to monitor the air temperature.

Electro-valves. In order to control the irrigation process we use solenoid valves provided by Irritol (Fig 12). A solenoid valve is an electromechanical valve that is controlled by an electric current. The electric current runs through a solenoid, which is a wired coil wrapped around a metallic core. The solenoid creates a controlled magnetic field when electrical current is passed through it. This magnetic field affects the state of the solenoid valve, causing the valve to open or close. The electric valves operate with a 9V-32V battery.

In order for the IRIS motes to be able to drive the valves we used relays that take as input a 3 VDC current and can control circuits of up to 250 VAC. The relay is connected in series with the electro-valve and an external 9V battery. This way, when the IRIS mote triggers the relay, the circuit is closed, the electro-valve opens and the irrigation process begins.

Finally, in order to have a base of reference, we also installed a common irrigation programmer. This programmer can be set to irrigate at regular time intervals for fixed periods of time; i.e. every second day for half an hour. An overview of the deployment is depicted in Figure 13.

5.3 Software Description

Implementation in TinyOS. The sensor motes were programmed in TinyOS [8]. TinyOS has a component based architecture and forms an event-driven operating system. The motes where assigned unique mote IDs and where programmed so that each TelosB informs about soil humidity the IRIS controlling the corresponding to the pot electro-valve. In order to achieve TelosB - IRIS communication we had to configure them to use the same ZigBee channel. We have chosen to use channel 26 in order to have minimal radio interferences from other wireless networks.

Our system comprises of three TinyOS applications. The *SoilSensorApp* (runs on TelosB motes) is responsible for monitoring the soil humidity and inform correspondingly the IRIS motes that control the irrigation process. The second TinyOS application is the *ValveApp* (runs on IRIS motes). With this application running, when the mote receives a message with soil humidity readings, it reads

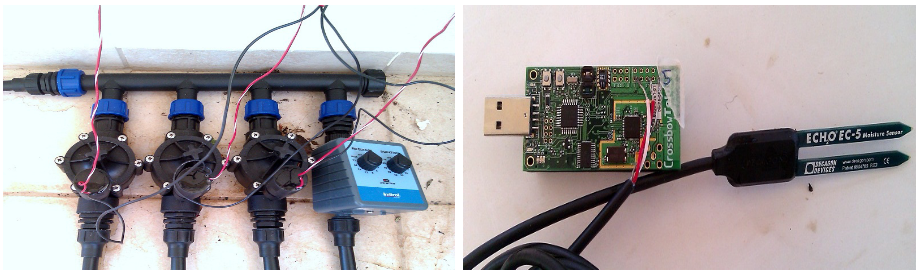


Fig. 12. The installed electro-valves and the common irrigation programmer. The ground and Vcc cables of the solenoids are connected in series with the relay and the external power source (batteries or AC-DC converter). On the right, the EC-5 soil sensor shouldered on a TelosB mote.

its payload and accordingly sets or clears the output ADC pins. This way it can trigger the relay connected to these pins on/off; thus driving the electro-valve.

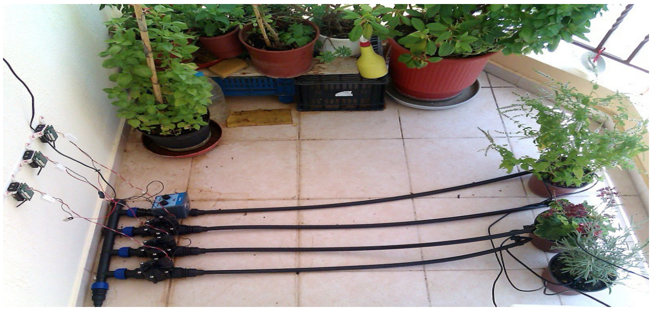


Fig. 13. The entire smart gardening system deployed

The sink node runs a modified version of the BasestationApp that comes along with the 2.1.0 TinyOS distribution. This application receives the radio messages sent to the sink by the portable nodes and forwards them to the PC using UART. From there a Java application receives the messages and stores the corresponding information in a MySQL database.

5.4 Performance Evaluation

We evaluate the performance of the traditional irrigation scheme, using a common irrigation programmer, and the smart irrigation scheme, using a wireless sensor network. We let each system to water for two days three pots; each one containing a different plant. The plants where chosen in order to have diversity in their water needs. Sorted by descending water needs, each pot contains a geranium, a lavender and a mint. The season the experiments were conducted was

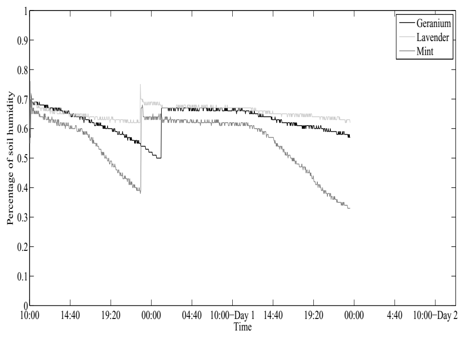


Fig. 14. Soil humidity over time when using common irrigation programmer

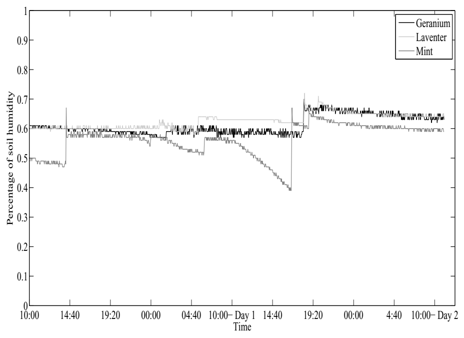


Fig. 15. Soil humidity over time when using WSN smart home irrigation system

summertime and the temperature was around 36^{o}C during the day and 30^{o}C during the night.

Figures 14 and 15 show the soil humidity of each pot over time for the two irrigation schemes respectively. We note that the pot containing the mint, that has the highest water needs from all three plants we used, depletes humidity form its containing soil much faster. On the contrary, the geranium absorbs humidity much slower, therefore the soil dries out at a lower rate.

When using a common irrigation programmer there exist great variations in the concentration of soil humidity. The soil tends to dry out and then is flooded with water causing an almost vertical increase of the soil humidity values. These extreme variations are not to the benefit of the plants, as they require soil humidity to remain at a given level. Furthermore, dried out soil causes great amount of water to pour away as it cannot withhold water to the same degree as even lightly moist soil.

On the other hand, the smart home-irrigation system manages to maintain soil humidity at the same level. It dissipates less water and it provides an irrigation scheme that is adaptable to the specific watering needs of each plant. The most important feature is the fact that by constantly monitoring the humidity levels, it basically adapts to current environmental conditions. Whether there are high temperature or sunlight variations or not, the system will adjust the irrigation process so as to maintain the same level of soil humidity.

6 Conclusions

We have presented selected key aspects of measurements based experimental research. These include experimental evaluation of sensor network protocols (exemplified via the energy balance property), critical components of such experimental research (reference deployments, mobility profiles and performance metrics) and specialized application commissioning (smart watering). Our experience suggests that measurements based experimental research can nicely complement rigorous performance analysis and simulation based evaluation providing more realistic performance results and contributing to the validation and fine tuning of the high level algorithms. A large variety of realistic topologies, mobility profiles and traffic patterns is needed while novel network parameters as well as performance measures (and their trade-offs) arise. In this respect, we note that current ad-hoc approaches are useful but there is a need to further converge to widely accepted, common integrated methodologies, systems and tools.

References

1. HOBNET deliverable - Scenario analysis report presenting the various use case scenarios to be addressed, http://www.hobnet-project.eu/files/D1.1.PDF
2. Efthymiou, C., Nikoletseas, S., Rolim, J.: Energy balanced data propagation in wireless sensor networks. Wireless Networks 12(6), 691–707 (2006)
3. Jarry, A., Leone, P., Powell, O., Rolim, J.: An Optimal Data Propagation Algorithm for Maximizing the Lifespan of Sensor Networks. In: Gibbons, P.B., Abdelzaher, T., Aspnes, J., Rao, R. (eds.) DCOSS 2006. LNCS, vol. 4026, pp. 405–421. Springer, Heidelberg (2006)
4. nesC: A Programming Language for Deeply Networked Systems, http://nescc.sourceforge.net/
5. Holistic Platform Design for Smart Buildings of the Future Internet, http://www.hobnet-project.eu/
6. Angelopoulos, C.M.: D Efstathiou, S. Nikoletseas: Experimental evaluation of energy balance algorithms in the SenseWALL sensor network test-bed. In: DCOSS 2011 (2011)
7. Angelopoulos, C.M., Nikoletseas, S.E., Theofanopoulos, G.C.: A smart system for garden watering using wireless sensor networks. In: MOBIWAC 2011, pp. 167–170 (2011)
8. TinyOS 2.x Documentation, http://www.tinyos.net/tinyos-2.x/doc/
9. Decagon ec-5, http://www.decagon.com/products/sensors/soil-moisture-sensors/ec-5-soil-moisture-small-area-of-influence/
10. Iris datasheet, http://www.memsic.com/support/documentation/wireless-sensor-networks/category/7-datasheets.html?download=135
11. Telosb datasheet, http://www2.ece.ohio-state.edu/bibyk/ee582/telosmote.pdf
12. RFC2616, Hypertext Transfer Protocol – HTTP/1.1
13. http://tools.ietf.org/wg/6lowpan/
14. http://tools.ietf.org/html/draft-ietf-core-coap
15. http://wisebed.eu/site/
16. http://datatracker.ietf.org/wg/ipv6/charter/
17. http://datatracker.ietf.org/wg/core/charter/

Various Detection Techniques and Platforms for Monitoring Interference Condition in a Wireless Testbed

Wei Liu[1,*], Stratos Keranidis[2], Michael Mehari[1], Jono Vanhie-Van Gerwen[1], Stefan Bouckaert[1], Opher Yaron[1], and Ingrid Moerman[1]

[1] Department of Information Technology
Internet Based Communication Networks and Services (IBCN)
Ghent University - iMinds
Gaston Crommenlaan 8 Bus 201, B-9050 Gent, Belgium
{wei.liu,michael.mihari,jono.vanhie,stefan.bouckaert,opher.yaron, ingrid.moerman}@intec.ugent.be

[2] Department of Computer and Communication Engineering,
University of Thessaly, Greece
Centre for Research and Technology Hellas, CERTH, Greece
efkerani@uth.gr

Abstract. Recently the constant growth of the wireless communication technology has caused a huge demand for experimental facilities. Hence many research institutes setup public accessible experimental facilities, known as testbeds. Compared to the facilities developed by individual researchers, a testbed typically offers more resources, more flexibilities. However, due to the fact that equipments are located remotely and experiments involve more complex scenarios, the required complexity for analysis is also higher. A deep insight on the underlying wireless environment of the testbed becomes necessary for comprehensive analysis.

In this chapter, we present a framework and associated techniques for monitoring the wireless environment in a large scale wireless testbed. The framework utilizes most common resources in the testbed, such as WI-FI nodes, as well as some high-end software-defined radio platforms. Information from both physical layer and network layer are taken into account. We observe that feature detection is more sensitive than general energy detection for dedicated technologies, and distributed spectrum sensing can further improve the detection sensitivity. Such observations are applied to achieve better interference detection. The performance is mainly analyzed experimentally.

Keywords: Interference detection, testbed, spectrum sensing.

[*] The research leading to these results has received funding from the European Union's Seventh Framework Programme FP7 under grant agreements number 258301 (CREW project) and number 287581 (OpenLab project).

1 Introduction

Over the past few years, wireless technology has evolved dramatically [1],[2]. The demand for wireless experimentation infrastructure rises accordingly. It is typically time consuming and inefficient to build experimental setups each time for individual researchers. Therefore, many research institutes adopt testbeds — a group of fixed and public accessible infrastructures for experimentation. Such a testbed not only shortens experiment's setup time, it also offers more resources, hence enables more sophisticated experiments.

In a wired-network testbed, the link configuration between network entities is part of the resources that can be reserved by users. Compared to the wired-network testbed, the wireless medium is shared by all the facilities inside the wireless testbed, therefore also shared by different devices reserved by different users. Consequently, simultaneous network experiments performed by independent users on a wireless testbed may interfere each other. Such kind of interference cannot be solved by resource reservation as long as parallel experimentations are allowed. In addition to interference among users, nearby wireless devices, which are not part of the testbed, can also cause interference.

Repeatability and stability are prerequisites for drawing conclusions from any type of experiment. However, in the wireless environment this is usually not the case [3]. Even simple repetition can results in fairly large variations in measurements. The question is what is causing those variations? Sometimes, it is obvious that the variations are due to external interference. But for more complex scenarios, involving tuning of certain parameter sets in the experiment setup, such variations can be extremely confusing and eventually lead to wrong conclusions.

Consequently, we need a good view in the testbed to tell what is really going on in the air. Hence we propose to have a monitor running in the background to detect undesired interference. For this purpose, network layer monitoring tools as well as spectrum sensing techniques can be utilized. In this chapter we focus on monitoring techniques for interference detection in WI-FI experiments. The remaining part of the chapter is organized as follows: Part 2 outlines related work, including existing monitoring techniques and their usage in various testbeds; Part 3 highlights the feature offered by our framework; Part 4 gives a high-level overview of the w-iLab.t testbed and its monitoring platforms. In Part 5 we verify the performance of the tools in w-iLab.t with various interference models. Finally, we conclude this chapter and describe the direction for future work in Part 6.

2 Related Work

Monitoring of the wireless environment can be achieved at different levels. At the network level, many MAC and routing protocols utilize certain types of channel assessment mechanisms. For instance, the IEEE 802.11 standard employs a rate adaptation algorithm ARF (Automatic Rate Fallback) [4] for bit rate adjustment at the physical layer. In ARF, each sender attempts to increase transmission rate after a fixed number of successful transmissions at a given rate, and falls back to a lower bit rate after 1 or 2 consecutive failures. ARF estimates the channel condition via the packet error rate feedback from receivers, more sophisticated protocols involve other statistics to achieve better estimation, [5],[6]. In general, channel assessment at network layer is limited to the links between the relevant transmitters and receivers, thus commonly referred to as link quality estimation.

The view of channel condition at the physical layer is much broader. Physical layer monitoring tools in essence perform spectrum sensing, which can be either technology dependent (feature detection), or technology independent (general detection). A thorough list of sensing algorithms is presented in [7].

Technology dependent detection mostly requires a certain amount of a-priori knowledge of the transmitter. Some detection techniques involve decoding of the received signal. Many popular feature detection techniques exist, such as matched filtering or waveform based detection. These techniques are only able to detect specific types of traffic.

Energy detection is the most common way of technology independent detection. The major advantage is that no a-priori knowledge of the transmitter is required. There are two common approaches to implement energy detection. One is to perform Fast Fourier Transform and calculate the power spectrum density in the frequency domain; another is to derive the received signal strength directly from samples in the time domain.

In the context of a wireless testbed, there are typically many network devices which can perform some level of link estimation, however, resources capable of general energy detection are scarce.

Various channel estimation mechanisms are custom developed for individual testbeds. The solution proposed in [8] utilizes software-defined radio for channel assessment in the NITOS testbed [9]. The estimation is based on energy detection with time domain RSSI (received signal strength indicator) measurement. The framework proposed in [10] is used to inspect link quality between wireless testbed nodes and appropriately map them to user required network topology. The link quality estimation framework proposed in [12] predicts link quality based on packet statistics for pure sensor network environments. We focus on applying sensing and monitoring techniques systematically in a wireless testbed for detecting unwanted interference.

3 The Proposed Framework

In order to obtain optimal observation of the experimental environment, we argue that the relevant channels should be monitored not only during the experiment but also before and after it. More specifically for a given experiment, the relevant channel should be monitored in three phases : before the experiment, during the experiment and also after the experiment.

– Monitoring before the experiment provides an overview of the channel condition. If interference is detected, the system will postpone the experiment

until the channel is clean, or consider to switch to another channel. The purpose of this phase is to avoid invalid experiments. Since any signal present during this phase is interference, a general energy detection is sufficient.
- The monitoring system required during the experiment should be able to distinguish interference from the ongoing experiment. In this case energy detection might be not enough. It is necessary to combine feature detection with network layer information.
- Post-experiment monitoring is similar to pre-experiment monitoring, which requires only simple energy detection. The logic here is, if there is interference detected immediately after the experiment, then most likely the interference was also present during the experiment. The experimenter should be informed that the validity of this experiment needs to be double checked. Additionally the comparison of pre-experiment and post-experiment monitoring provides an insight of the change in the environment.

Our proposed solution is to construct a hybrid system, which employs general energy detection and feature detection, as well as network layer information. In addition to the three-phase observation and hybrid detection technique, we also extend our framework with spatially distributed measurement tools. In comparison to monitoring based on a single device, a distributed system formed by multiple devices provides more insights of the wireless environment. When combining sensing results of multiple sensing devices, two factors must be considered: what weight should be given to each device; when is a particular result valid for combination in terms of time frame. For a non-heterogeneous distributed sensing system, each device share the same weight. The produced sensing information is stored into a database, associated with a timestamp. This helps to compare sensing result from different sensing engines. To obtain more accurate timestamp, the sensing engines are synchronized with ptpd (precision time protocol daemon [11]). The added value of distributed monitoring is further explored in Part 5.

Since pre-experiment and post-experiment monitoring only require simple energy detection, the main challenge is to derive optimal techniques to detect interference during the experiment. The performance of interference detection during the experiment is the main focus of this chapter.

4 The w-ilab.t Testbed

The w-iLab.t testbed is a generic and heterogeneous wireless testbed. It consists of two sub testbeds: the w-iLab.t office and w-iLab.t Zwijnaarde. The w-iLab.t office is deployed in a real office environment while the w-iLab.t Zwijnaarde is located at a utility room. There is little external interference at the Zwijnaarde testbed as no regular human activity is present and most of its walls and ceiling are covered with metal as shown in Figure 1. The majority of devices in both w-iLab.t testbeds are embedded PCs equipped with Wi-Fi interfaces and sensor nodes. Since the Zwijnaarde testbed was deployed more recently, the devices in this testbed are more powerful in terms of processing power, memory

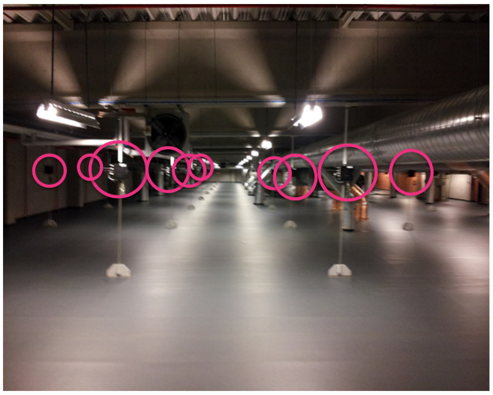


Fig. 1. The w-iLab.t Zwijnaarde testbed

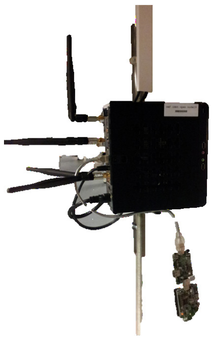


Fig. 2. The w-iLab.t Zwijnaarde node

and storage. In this chapter, our experiments are performed at the Zwijnaarde testbed, therefore we mainly focus on the introduction of this testbed.

There are several types of wireless devices deployed : Zigbee sensors, blue-tooth dongles, Wi-Fi based devices, sensing platforms and some software-defined radio (SDR) platforms. All devices are reachable over a wired interface for management purposes. Each device can be fully configured by the experimenters. When the wireless devices are configured via the same control interface, they are said to be attached to one "node".

A typical node in w-iLab.t Zwijnaarde is shown in Figure 2. It consists an embedded PC with two WI-FI interfaces and one Zigbee sensor. The location of the nodes are indicated with circles in Figure 1. One of the deployed SDR platforms is the USRP N210 [13], abbreviated as USRP throughout this chapter. The USRP's are attached to powerful quad-core servers instead of embedded PC's. There are 60 nodes installed in the Zwijnaarde testbed, among which 6 are USRP's. The topology of the testbed is shown in Figure 3, the locations of the USRP's are marked with hexagons while regular nodes are indicated with circles.

The w-iLab.t Zwijnaarde has adopted OMF (cOntrol Management Framework [14]) as its testbed control and management framework. OMF allows experimenters to describe their experiments systematically. It provides easy data logging services and the ability to configure multiple devices.

There are two main advantages of using OMF framework in the aspect of the interference detection framework — its central control capability and data collection service. Both features are essential for monitoring based on distributed and heterogeneous devices. As an experimenter, the monitoring tools are no more than regular experimental facilities that can be configured via OMF. The data generated by the monitoring system can be logged into the database just like regular measurements as well. For an experienced OMF testbed user, the extra effort of using such a monitoring system is trivial.

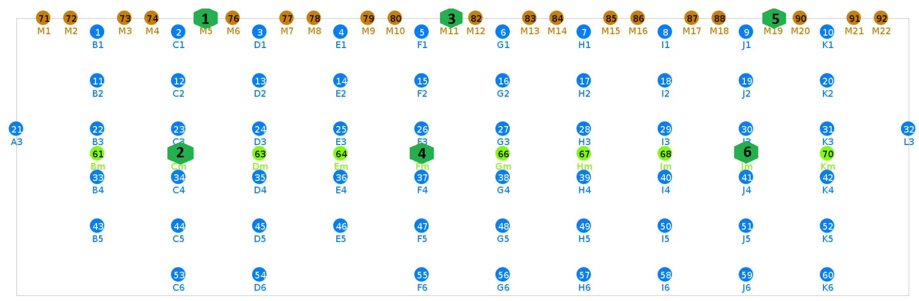


Fig. 3. The Zwijnaarde testbed topology

4.1 The Interference Detection Tools Offered by w-iLab.t

WI-FI Interface in Monitor Mode. As described above, a typical node in the Zwijnaarde testbed has two WI-FI interfaces. Since most experiments do not utilize the second WI-FI interface, it is possible to configure it into monitor mode on selected channel. When configured into this mode, the interface is not associated with any access point (AP). It will capture packets in promiscuous mode. Received WI-FI packets may include a Radiotap header [15], which contains the RSSI of the incoming packet. Therefore, the physical layer information can be extracted directly from WI-FI packets, thus a regular WI-FI card combined with a simple packet sniffer software can serve as a physical layer measurement tool. This is referred as WI-FI monitor throughout this chapter.

The information obtained via the WI-FI monitor contains more details on packet level, and requires less post processing effort. More importantly, there are no special requirements on either hardware or software, hence all nodes with WI-FI interfaces in the w-iLab.t can be configured as WI-FI monitors. However the monitoring functionality is restricted by the capability of the WI-FI card, no information can be provided if the interference can not be decoded. Hence in terms of detection type, WI-FI monitor belongs to the class of feature detection.

To illustrate the capability of the WI-FI monitor, one node in the Zwijnaarde testbed was configured to scan all 13 WI-FI channels in the 2.4 GHz ISM band. The result shown in Figure 4 tells us there are three access points active in the neighborhood, located on channel 1, 6, and 13. The beacon from the access point with essid “robotcontrol” on channel 1 has highest RSSI, which corresponds with the fact that the “robotcontrol” AP is an internal AP which happens to be active during the experiment. The graph also shows that the beacon sent on channel 1 can be measured even up till channel 5. The other two access points’ beacons are received with considerably weaker RSSI due to the fact that they are located outside the testbed. The WI-FI monitor can be used to find out which access point is active on what channel, and how their transmit power is distributed over the neighboring channels. This observation can certainly be applied to the pre-experiment monitoring phase.

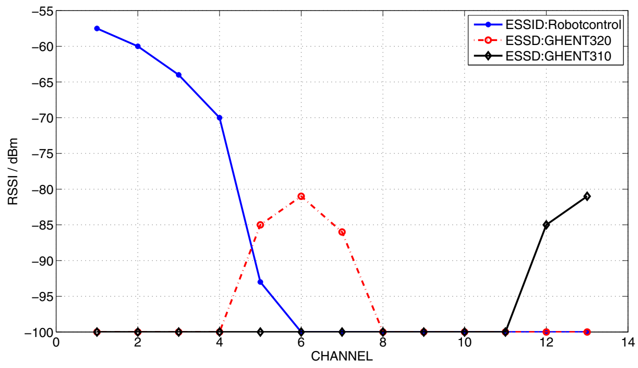


Fig. 4. RSSI Measurement of WI-FI card

USRP Based Spectrum Sensing Engine. The Universal Software Radio Peripheral (USRP) developed by Ettus Research [13], consists of two parts, a fixed mother board and a plug-in daughter board. The daughter board provides basic RF front-end functionality. In the Zwijnaarde testbed, all USRP's are by default equipped with XCVR2450 daughter boards which covers the 2.4 and 5 GHz ISM bands and has a configurable analog front-end filter with maximum bandwidth of 30 MHz.

GNU Radio is by far the most well-known 3rd party application to work with USRP [16]. The platform selected here is Iris — a software platform developed by Trinity College Dublin [17]. It has similar component structure as GNU Radio, but is more suitable for reconfigurability on the fly. Both GNU Radio and Iris utilize UHD [18] driver and firmware to communicate with USRP and C++ to realize the underlying signal processing block. But the glue logic between signal processing blocks is realized differently. Compared to GNU Radio, Iris is more transparent due to its simple structure, and hence easier to get access to low level parameters on the hardware. This high transparency and reconfigurability are more desired in our context, hence we selected this platform.

We have implemented a customized solution within the Iris platform to use USRP for spectrum analysis. The spectrum analysis task is performed by several Iris components. The first component collects complex samples from the USRP device. The received samples are used to calculate power spectrum density (PSD) via the periodogram algorithm. By default, USRP's are configured to sample at 25MHz on a fixed frequency, which is wide enough to cover one WI-FI channel. It is also possible to use the USRP sensing engine in a wide-band mode. In this mode, the front end of USRP is configured to perform fast sweeping across the selected channels. The spectra obtained at multiple channels are assembled into one complete spectrum trace when the sweep is complete. The wide-band mode is suitable for obtaining an overview of multiple channels at the same time, however, the probability of interception on each channel decreases with the increase of the covered bandwidth.

There is an option to add a component for power integration over a certain band based on the PSD. Sometimes, the frequency resolution provided by power spectrum density can overskill if only the received power on the entire channel is interesting. In this case a single RSSI value can be used instead of the PSD trace. More details about the implementation of the USRP sensing engine are provided in [19]. A dedicated OMF wrapper is created for logging the spectrum information into the central database.

The USRP sensing engine belongs to the energy detection category. It is technology independent, however, it can only provide information at the physical layer.

5 Experimental Evaluation

In this section, we evaluate several experimental scenarios that aim at presenting the abilities offered by the different interference detection tools and techniques in w-iLab.t.

More specifically, each experiment is designed to demonstrate a representative interference scenario that can occur during an ongoing experiment. The considered scenarios are listed below:

- Interference may come from devices located in close proximity and operating on the same channel. The performance decrease observed in this case is caused due to the sharing of the medium among devices. This situation is referred as *Channel Contention.*
- When the testing devices are located far away, two transmitters may not be able to detect the existence of each other even when they are on the same channel. This is termed as the “hidden terminal” scenario, where the CSMA MAC protocol would fail and as a result transmitters can start transmitting simultaneously. We refer this type of interference as *Co-channel Interference.*
- IEEE 802.11 set of standards make use of the ISM (Industrial Scientific Medical) bands. The popular 2.4 GHz band, used by 802.11b and 802.11g standards, offers 11 consecutive channels, spaced 5 MHz apart and occupying 22 MHz of bandwidth. As a result, most channels partially overlap with consecutive channels, limiting the number of theoretically non-overlapping channels to three (e.g. 1, 6, 11). As a result, transmissions on a specific channel may interfere with simultaneous transmissions on overlapping channels. This is referred as *Overlapping Channels Interference.*
- Finally within a wireless testbed, interference may also come from none IEEE802.11 compatible devices. We refer to this situation as *Interference from Heterogeneous Technology.*

In the following experiments, we consider a typical scenario of two IEEE 802.11 standard compliant nodes, operating in infrastructure mode with 802.11g standard on channel 11 (2462 MHz) and generating traffic on Uplink.

We refer to these two nodes as the System Under Test (SUT) and also consider the measured application layer UDP throughput of the SUT, as the overall performance metric.

The first three experiments consider channel contention, overlapping channels and co-channel interference respectively. Interference is generated by a collocated pair of IEEE802.11 compliant nodes under various settings. We refer to these two nodes as the interference generating group, abbreviated as INT. In the fourth experiment, we use a narrow band signal generated by a zigbee sensor node in order to examine interference generated by heterogeneous technology.

For all the aforementioned scenarios, we configure the second IEEE802.11 interface on the receiver of the SUT to monitor mode and continuously monitor RSSI of all the packets that are successfully decoded.

In each scenario we configure a different number of USRP devices to perform spectrum sensing on the operating frequency of the SUT. The PSD is recorded with 25 MHz span on the selected channel at the speed of 10 sweeps per second. Alternatively, one single value representing the RSSI of the selected band is recorded instead of PSD trace.

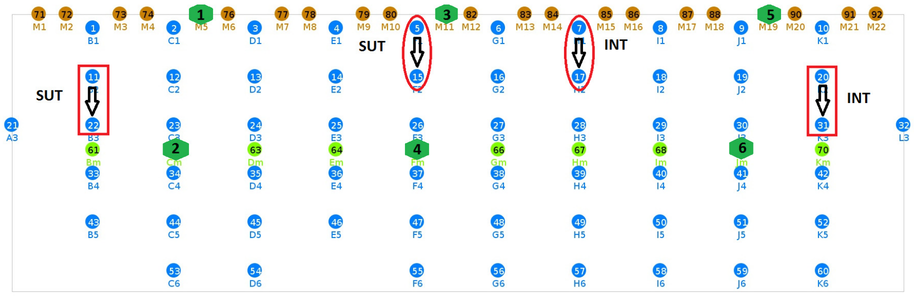


Fig. 5. Experiment topology

5.1 Channel Contention Detection

In the first experiment, we place the INT group close to the SUT and configure it to operate on the same channel as the SUT. These two pairs of nodes are indicated with ovals in Figure 5. We use Iperf to generate traffic on the application layer and set the bandwidth requirement of both the SUT and INT to 30 Mbit/s. The Iperf of the SUT group is active throughout the duration of the experiment, while the Iperf application of the INT group is activated just for a short period, in order to explore how the performance of the SUT is affected. We select 30 Mbit/s as it is slightly above the maximum bandwidth the SUT can achieve without interference, ensuring that contention will happen during the experiment.

The throughput performance of both the SUT and the INT groups are illustrated in Figure 6. Upon the activation of the INT group, the throughput performance of the SUT drops to 15 Mbit/s, which is half of its original throughput and equal to the throughput of the INT. This clearly shows that the available channel capacity is equally divided through the CSMA protocol between the two contending pairs of nodes.

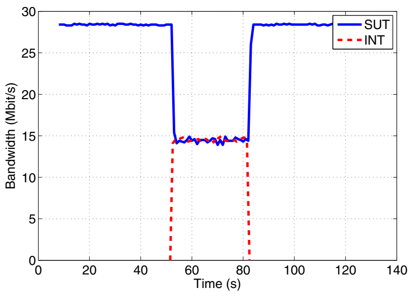


Fig. 6. Throughput performance

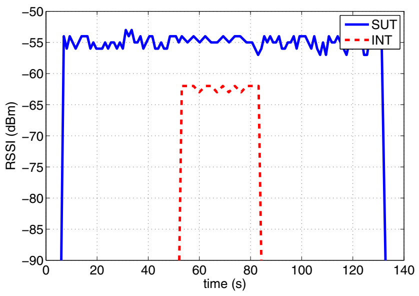


Fig. 7. RSSI trace of WI-FI monitor

Having examined the bandwidth performance of the SUT, the next step is to check the performance of the monitoring tools. The RSSI trace obtained from the WI-FI monitor contains records from both senders. We apply a source MAC address filter on the entire record, and separate the RSSI trace for each group, as illustrated in Figure 7. We conclude that for this scenario, a WI-FI monitor combined with MAC layer information is able to clearly identify which IEEE802.11 compatible device appeared and when exactly the contention happened.

However, the USRP device, not being aware of MAC layer information, is not able to distinguish between different Wi-Fi sources that transmit on the same channel. We use the spectrogram to evaluate the sensing performance of the USRP sensing engine. A spectrogram is a two dimensional graph, with frequency on the horizontal axis and time on the vertical axis. The intensity of the received signal strength is indicated with gray scale. Figure 8 is a spectrogram captured during the experiment by USRP4. The spectrogram does not provide any valuable input that can aid the identification of the different traffic sources. The only valid observation is that the channel under consideration is occupied during the entire experiment and the density of the spectrogram becomes slightly higher, while the INT group is active. However, such observations are not sufficient to detect interference activity and therefore, energy detection is not the preferred detection technique in this scenario.

5.2 Overlapping Channels Interference Detection

The second experiment is designed to evaluate the effect of interference generated by IEEE802.11 compliant devices operating on overlapping channels. For this purpose, we use the same network topology as in the previous experiment. However, instead of configuring the INT group to operate on the same channel as the SUT, we now set it to transmit on the adjacent channels of the SUT group. More specifically the channel index of INT group is varied from 7 to 10 while the SUT is always operating on channel 11.

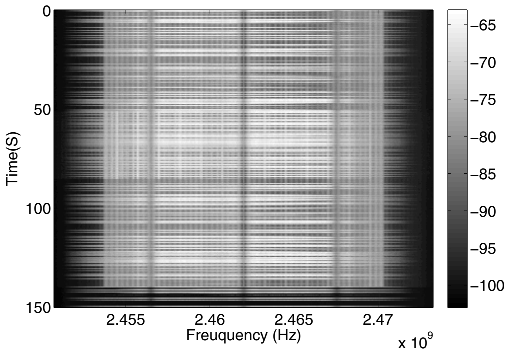


Fig. 8. Spectrogram of USRP4 on channel 11

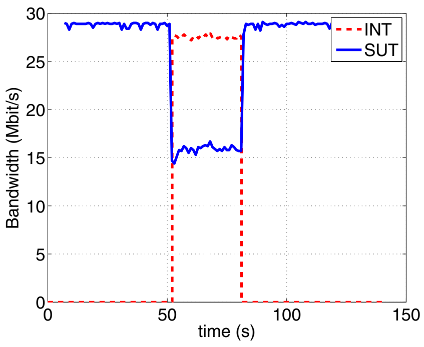


Fig. 9. BW when INT at channel 7

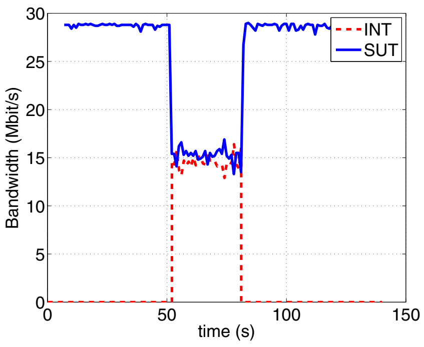


Fig. 10. BW when INT at channel 8

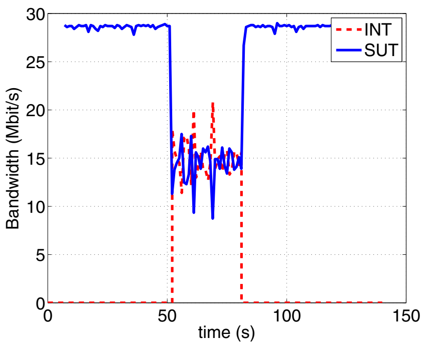


Fig. 11. BW when INT at channel 9

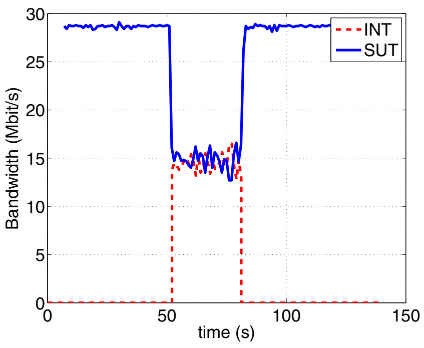


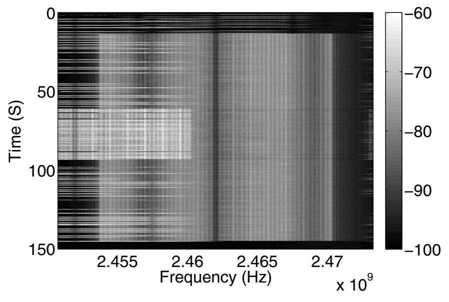
Fig. 12. BW when INT at channel 10

Figures 9, 10, 11, 12 illustrate the throughput performance when the INT group is active on Ch. 7, Ch. 8, Ch. 9, Ch. 10 respectively. We notice that the activation of the INT traffic results in significant reduction of the SUT performance, in all the cases under consideration. Among channels that are closely spaced, such as Ch. 8, Ch. 9, Ch. 10, the bandwidth performance of both groups is around 15 Mbit/s when INT is active, which is close to half of the maximum bandwidth achieved by a single pair of nodes with no interference. This is due to the contention of shared medium as in the previous experiment. Hence the performance of both groups under Overlapping-Channel interference resembles the performance in Channel Contention scenario when selected operating channels are closely spaced.

This observation is clearly violated when the interference is present on Ch. 7, where an overall bandwidth of approximately 45 Mbit/s is achieved and the bandwidth of INT group is significantly higher than SUT. This comes from the fact that when the amount of channel overlapping falls below a certain threshold, the carrier sense mechanism may fail to detect ongoing transmissions and thus results in collisions. In wireless networks, a frame collision does not necessarily result in all the simultaneously transmitted frames being lost. The survival of the collision depends on the relative signal power and the arrival timing of the involved frames. This phenomenon is related to the Capture Effect [20].

According to our experiments, certain topology and channel configurations lead the Capture Effect to either favor the SUT or the INT link. These observations yield interesting insights regarding the impact of the Capture Effect on interference and motivate further investigation.

Unlike the previous experiment and in contradiction to the results shown in Figure 4, the WI-FI monitor does not succeed to decode any packets transmitted on the INT link and as a result, all the recorded RSSI measurements in this case belong to the SUT transmitter. Hence, the Wi-Fi monitor fails to detect any interfering activity in this scenario. Considering the measurements plotted in Figure 4, we notice that WI-FI card is able to decode Beacon frames even when it is on channels that are not adjacent to the Beacon's sender. This can be explained by the fact that Beacon frames are transmitted at the basic rate of



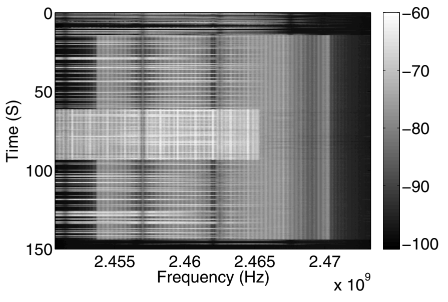


Fig. 13. Spectrogram with INT on Ch. 9 **Fig. 14.** Spectrogram with INT on Ch. 10

1 Mbit/s, while the data from application layer is transmitted at much higher rates, typically 48 Mbit/s or above. Hence it is much easier to successfully decode Beacon frames than regular data frames.

According to the above observations, we conclude that standard compliant devices that operate in monitor mode, are not able to provide valuable monitoring information for the detection of Overlapping-Channel interference.

Fortunately, we can overcome such situations by using another "eye" in the air — the USRP sensing engine. Figure 13 and Figure 14 represent the recorded spectrogram when interference is present on Ch. 9 and Ch. 10 respectively. Based on the spectrogram, it is clear that the activity in adjacent channels is the reason for the SUT throughput reduction. The same conclusions can be drawn when interference is present on Ch.7 or Ch.8.

5.3 Co-channel Interference Detection

This scenario focuses on how distributed sensing can contribute to the detection of co-channel interference. As previously mentioned, co-channel interference may occur when two transmitters are located far away. In this case, the transmitters may fail to detect each other's ongoing transmissions and thus transmit simultaneously, resulting in packet collisions. For this experiment, we setup a distributed spectrum sensing system, using the 6 USRP's that are currently distributed over half of the testbed.

Similar to the setup used in the previous scenario, the experiment here also involves two pairs of IEEE802.11 compliant nodes, however instead of choosing two groups next to each other, the groups are now located at different sides of the testbed. The selected nodes for this experiment are marked with rectangles in Figure 5, while the USRP sensing engines are indicated with hexagons, labeled from 1 to 6.

We configured the SUT group on the left side of the testbed to generate continuous traffic at 20 dBm. The INT group on the right side of the testbed follows an ON-OFF traffic pattern, so that each time the client starts a data stream for 15 seconds, The INT group will wait for another 15 seconds before the client is turned on again. The transmit power of the INT group is reduced from 20 dBm to 0 dBm in the step of 2 dBm. Both the INT and SUT are operating on the same channel.

The ON-OFF traffic pattern of the INT group makes it possible to distinguish between signals that are generated by the SUT and the INT groups, solely based on RSSI measurements.

Unfortunately USRP 1 was not available during the conduction of this experiment, we hence only configured the remaining 5 USRP's to collect spectrum data. Each USRP sensing engine produces an RSSI value every two seconds over the selected WI-FI channel.

The average bandwidth performance when INT is active is plotted against the transmit power of the INT group (Figure 15). The graph shows a clear trend that the impact of INT on the performance of SUT increases with its transmit

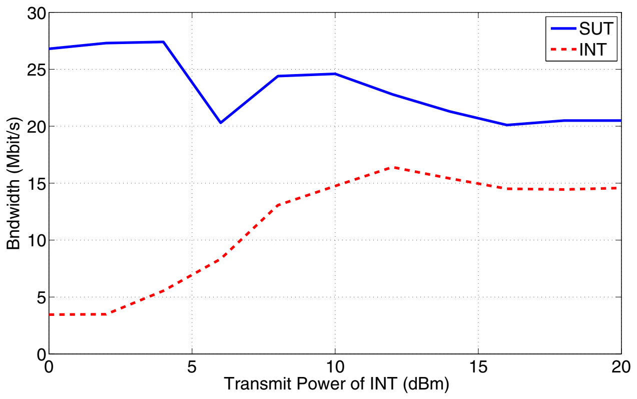


Fig. 15. Bandwidth performance of SUT and INT with various INT transmit power

power. Once the transmit power rises above 6 dBm, the INT group's impact on the SUT becomes visible.

When examining the RSSI records from the WI-FI monitor, the RSSI from the SUT sender is more or less constant, since there is no variation of transmit power at the SUT group. The RSSI recorded from the INT group is also very stable within each experiment. Therefore, the actual value of the RSSI is no longer important, what matters is the length of the packet trace from each sender, since this is the number of packets that can be detected by the WI-FI monitor. Based on this idea, Figure 16 and Figure 17 are generated. The first remark is that the number of detection of SUT is significantly higher than INT. Moreover, when the transmit power of INT is below 14 dBm, there is no detection of INT based on the packet trace at all. When the INT transmit power lies between the interval of 6 dBm and 14 dBm, we do observe the reduction of SUT throughput performance, however, the WI-FI monitor at SUT side does not detect any interference. In this situation, the INT transmitter becomes a "hidden terminal" for the WI-FI monitor and the SUT transmitter.

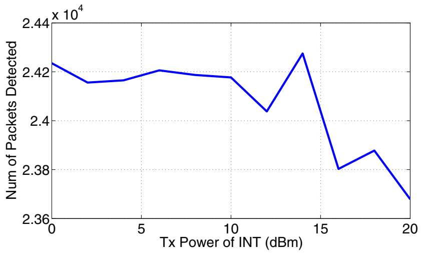


Fig. 16. WI-FI detection of SUT

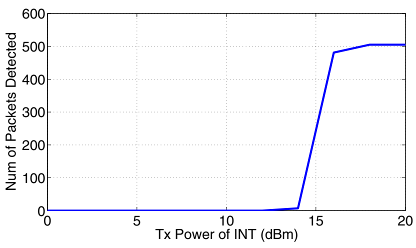


Fig. 17. WI-FI detection of INT

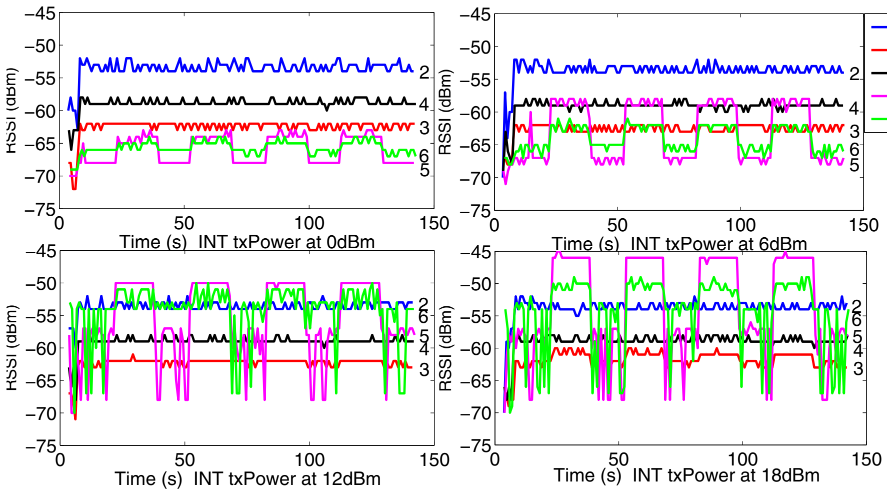


Fig. 18. PSD trace when the transmit power of INT group is at 0 dBm, 6 dBm, 12 dBm, 18 dBm

When the WI-FI detector is incapable of interference detection, how is the performance of USRP sensing engines? We select the RSSI traces of USRP sensing engine when INT transmit power is at 0 dBm, 6 dBm, 14 dBm and 20 dBm to present (Figure 18). At the first glance, USRP 5 and 6 are able to follow the ON-OFF traffic pattern produced by the INT group, while the rest of USRP's are dominated by the SUT group's transmission, since their measurements appear to be stable. This observation is confirmed by Figure 19, where the peak-to-peak value of RSSI in each trace for each USRP is plotted against the INT's transmit power. We clearly see that USRP 5 and 6's peak-to-peak RSSI values increasing with the transmit power of INT group, while the rest USRP's are almost unaffected. Hence, as a single device, USRP located far away from the INT group also fails to detect the interference. We do see that WI-FI monitor can detect the interference when its transmit power is above 14 dBm. Therefore, when using a single device as a monitoring tool, feature detection is more sensitive than energy detection. However, when combining the view of all the USRP's, we are able to identify the interference with energy detection even when the transmit power of the INT group is at 0 dBm. In this case, the advantage of distributed detection over localized detection is evident.

5.4 Heterogeneous Technology Interference Detection

In the last experiment, we focus on the detection of the interference caused by heterogeneous technology. Hence we do not use IEEE802.11 devices to produce

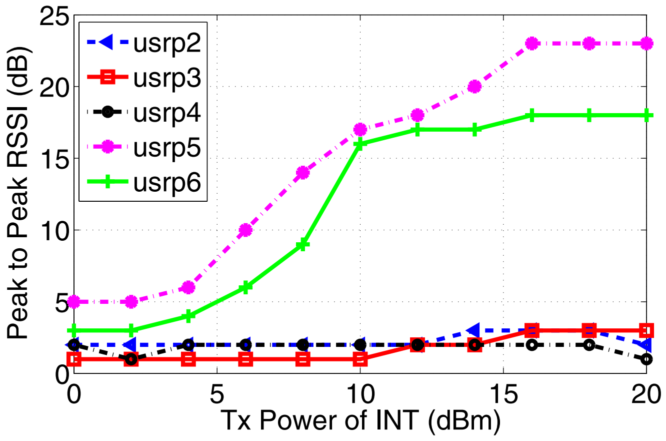


Fig. 19. The peak-to-peak RSSI vs Transmit power of INT group

interference but use a narrow band signal generated by a zigbee sensor node. The interfering signal is centered at 2.465 Ghz. This jamming signal is a simplified representation for interference generated by none IEEE802.11 compliant devices.

Based on bandwidth measurements, we observe that due to the activation of the narrow-band jammer, the performance of the SUT almost drops to zero, as indicated in Figure 20. The RSSI trace recorded by WI-FI monitor, illustrated in Figure 21, does not provide any valuable information except that the RSSI record becomes less dense when interference is present.

In the spectrogram obtained via the USRP sensing engine (Figure 22), the presence of the narrow band jammer becomes evident. Upon the activation of the jammer, the SUT's activity is greatly reduced. Based on this fact, we conclude that the IEEE802.11 transmitter is able to detect ongoing transmissions of the jammer through the Carrier Sense mechanism.

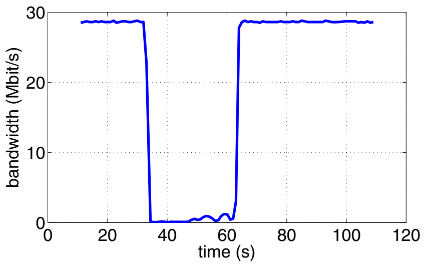


Fig. 20. Throughput performance

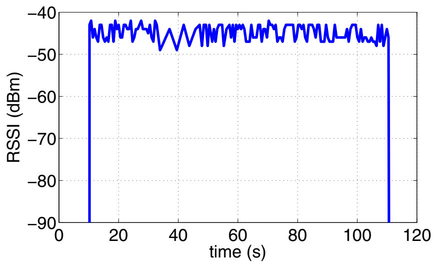


Fig. 21. RSSI trace of WI-FI Monitor

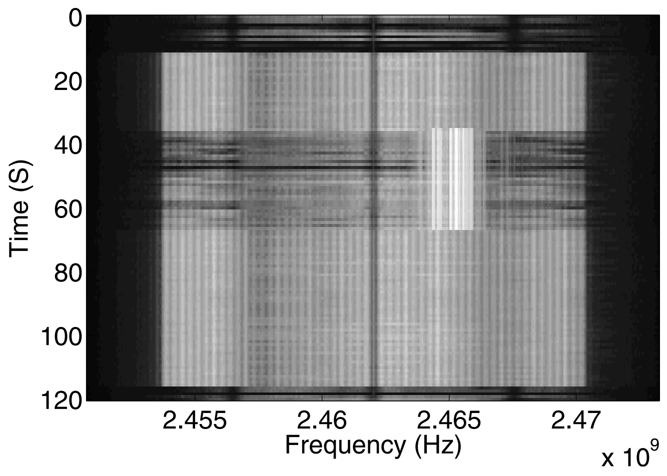


Fig. 22. Spectrogram from USRP

6 Conclusion and Future Work

In this chapter, we prove that wireless experiments are very susceptible to unpredictable interference, a monitoring system is hence necessary for validating experiments. Two types of measurement tools are offered by the w-iLab.t testbed: the standard WI-FI card, and the custom-designed sensing engine based on the software-defined radio platform (USRP). A distributed detection system can be easily formed via the OMF control framework.

We considered several common interference scenarios for WI-FI experiments inside the wireless testbed, including the interference caused by WI-FI device on the same channel, the interference caused by WI-FI device from overlapping channel and finally interference caused by none WI-FI technology. For each scenario, the performance of available measurement tools are examined. Within the co-channel WI-FI interference scenario, we are able to demonstrate the advantage of distributed detection over localized detection when the interfering transmitter is located far away.

In the future, the aforementioned interference detection techniques will be integrated into a benchmarking framework, in which the experiment cycle will be fully automated, offering services such as scheduling an experiment with specific parameter sets, and evaluating the gathered results. The validity of experiments will be evaluated based on the input from the interference detection framework proposed in this chapter.

References

1. IEEE802.11 standards revised due to fast WI-FI growth, `http://www.telecoms.com/43908/802-11-standards-revised-due-to-growth-of-wifi/`

2. IEEE standards, http://standards.ieee.org/findstds/standard/
3. Burchfield, R., et al.: RF in the Jungle: Effect of Environment Assumptions on Wireless Experiment Repeatability. In: IEEE International Conference on Communications, ICC 2009, pp. 1–6 (2009), doi:10.1109/ICC.2009.5199421
4. Kamerman, A., Monteban, L.: WaveLAN-II: A High-performance wireless LAN for the unlicensed band. Bell Lab Technical Journal, 118–133 (Summer 1997)
5. Lacage, M., et al.: IEEE 802.11 rate adaptation: a practical approach. In: MSWiM 2004 Proceedings of the 7th ACM International Symposium on Modeling, Analysis and Simulation of Wireless and Mobile Systems, pp. 126–134 (2004)
6. Baccour, N., Koubâa, A., Youssef, H., Ben Jamâa, M., do Rosário, D., Alves, M., Becker, L.B.: F-LQE: A fuzzy link quality estimator for wireless sensor networks. In: Silva, J.S., Krishnamachari, B., Boavida, F. (eds.) EWSN 2010. LNCS, vol. 5970, pp. 240–255. Springer, Heidelberg (2010)
7. Tevfik, Y., Huseyin, A.: A survey of spectrum sensing algorithms for cognitive radio applications. IEEE Comm. Servey and Tutorial 11(1), 116–130 (2009)
8. Passas, V., Keranidis, S., Korakis, T., Koutsopoulos, I., Tassiulas, L.: An Experimental Framework for Channel Sensing through USRP/GNU Radios. In: Korakis, T., Zink, M., Ott, M. (eds.) TridentCom 2012. LNICST, vol. 44, pp. 383–386. Springer, Heidelberg (2012)
9. NITOS wireless testbed, http://nitlab.inf.uth.gr/NITlab/index.php/testbed
10. 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: The Proceedings of WiNTECH 2009, Beijing, China (September 2009)
11. PTPD protocol, http://ptpd.sourceforge.net/
12. Baccour, N., et al.: A testbed for the evaluation of link quality estimators in wireless sensor networks. In: IEEE/ACS International Conference on Computer Systems and Applications, AICCSA (2010)
13. Ettus Research, http://www.ettus.com/
14. Rakotoarivelo, T., Ott, M., Jourjon, G., Seskar, I.: OMF: A Control and Management Framework for Networking Testbeds. SIGOPS Oper. Syst. Rev. 43, 54–59 (2010)
15. Radiotap, http://www.radiotap.org/
16. GNU Radio wiki, http://gnuradio.org/redmine/projects/gnuradio/wiki
17. Sutton, P., et al.: Iris: an architecture for cognitive radio networking testbeds. IEEE Comm. Mag. 48(9), 114–122 (2010)
18. Universal Hardware Driver(UHD), http://code.ettus.com/redmine/ettus/projects/uhd/wiki
19. Liu, W., et al.: Real-time wide-band spectrum sensing for cognitive radio. In: 2011 18th IEEE Symposium on Communications and Vehicular Technology in the Benelux, SCVT (2011)
20. Lee, J., et al.: An experimental study on the capture effect in 802.11a networks. In: Proceedings of the Second ACM International Workshop on Wireless Network Testbeds, Experimental Evaluation and Characterization, pp. 19–26

Methodology and Tools for Measurements on Wireless Testbeds: The NITOS Approach

Dimitris Giatsios[1,2], Apostolos Apostolaras[1,2], Thanasis Korakis[1,2], and Leandros Tassiulas[1,2]

[1] CERTH, The Centre for Research & Technology Hellas,
78 Filikis Etaireias Str, 38334, Volos, Greece
[2] University of Thessaly,
Department of Computer & Communication Engineering
37 Gklavani - 28th October Str, 38221, Volos Greece
{gidimitr,apaposto,korakis,leandros}@inf.uth.gr
http://nitlab.inf.uth.gr

Abstract. Since its establishment in 2009, the Network Implementation Testbed using Open Source drivers (NITOS) wireless testbed has been extensively used in several research projects for the experimental evaluation of protocols and algorithms. Collection of accurate measurements is of crucial importance for testbed users, allowing them to both select appropriate topologies for their experiments and assess the performance of their implementations. In this chapter, we describe measurement methodologies and tools used in the NITOS testbed in the context of its involvement in some of these projects. We provide examples demonstrating the utility of these tools for specific experiments. We also explain the challenges posed by the complex wireless medium, summarize lessons learned and outline future plans of NITOS towards an enhanced and more integrated measurement framework.

Keywords: Testbed, Experimentation, Measurements, Tools, Management.

1 Introduction

The rapid proliferation of network testbeds around the global research community during recent years is considered as a decisive step for the transition towards the Internet of the future [1]. Experimentation with real network equipment under real conditions addresses the traditional reluctance of the industry to seriously take into account innovative algorithms and architectures tested only in simulation platforms. Programmability of network components and use of open-source technology creates a broad field for hands-on innovation, motivating the interest and involvement of a continuously growing group of researchers. Nowadays, testbeds are not just popular, but thought of as the de facto performance evaluation instruments in network engineering research.

Experience from other research fields, where experimentation is the cornerstone of progress and development, such as Physics or Biology, has shown that the ability to capture accurate measurements is the most crucial component of the experimentation procedure. Indeed, these fields owe large part of their scientific rigor in the use of sensitive measurement equipment and careful methodology.

Measuring experiment variables and collecting observations are an essential part of any scientifically sound evaluation or comparison of technologies, services, or systems being studied. Furthermore, collected measurements provide testbed operators and future experimenters valuable information on the usage of experimental resources, which may influence long-term management decisions or short-term resource selection for a particular experiment.

In this chapter, we describe the measurement methodologies and specific tools used in the NITOS wireless testbed, one of the key testbeds of the FIRE [2] facility. The wireless environment, due to its broadcast nature and inherent uncertainty, poses particular challenges, affecting the experimental procedure. These challenges, as we will see, call for even more advanced and complicated measurements. Thus, many of the tools we describe in this chapter are particularly focused on wireless testbeds.

The remainder of this chapter is structured as follows. In section 2 we provide a brief presentation of the NITOS testbed. In section 3 we focus on measurement tools used for topology assessment, while in section 4 we describe measurement tools used for the evaluation of experiment results. In section 5, we provide some examples emanating from specific experiments conducted in NITOS in the context of research projects, explaining the respective measurement handling procedures. In section 6, we analyze some challenges related to testbed measurements, based on our experience in wireless testbeds, and present the future plans of NITOS in this field. Finally, we conclude the chapter in section 7, summarizing the key points made.

2 The NITOS Testbed

The history of NITOS stretches back to its establishment in 2009, and deals principally with the need to provision and maintain a remotely accessible infrastructure for experimentation and validation in the emerging field of wireless networks. The testbed was built as an outdoor local network with standalone nodes, running on Linux and featuring Wi-Fi cards supporting open-source drivers. Since then it has grown both in size and diversity of features, most notably with the addition of software-defined-radio hardware and an OpenFlow [3] wired experimental network. It has been publicly available through the Internet almost since the beginning and has hosted several experiments, especially in the area of Wi-Fi. Today NITOS is part of the OneLab European testbed facility [4] and the main wireless testbed of the OpenLab project [5], a key project aiming at the federation and further development of network experimental facilities in Europe.

2.1 Description and Architecture

NITOS [*http://nitlab.inf.uth.gr/NITlab/*] is an integrated testbed with heterogeneous features, that focuses on supporting experimentation-based research in the area of wireless networks. It consists of 40 wireless nodes, situated on the two top floors and the rooftop of a campus building (cf. Fig. 1). The nodes are equipped with commercial Wi-Fi cards that are based on Atheros chipsets and support the MadWiFi/Ath5k/Ath9k open source drivers. Through the open source drivers and the Linux operating system running on the nodes, a researcher can modify the MAC (Media Access Control) and the network layer of the nodes and develop new protocols that are directly compatible and comparable with IEEE 802.11 commercial products. 9 of the nodes are connected to software radio Universal Software Radio Peripheral (USRP) boards (GNU-radio). 15 of them are equipped with USB-cameras and 20 of them with temperature and humidity sensors. 10 nodes are equipped with 802.11n cards that allow for experimentation on a Multiple Input Multiple Output (MIMO) setup. NITOS also features three mobile nodes (two mounted on robots and one moving on rails), in order to test mobility-based scenarios.

The testbed's infrastructure is currently being significantly upgraded. A new set of nodes with advanced hardware features, recently assembled in NITOS, has been added to its equipment. These will form an additional indoor wireless testbed, which will be deployed in a campus facility featuring a fairly isolated environment in terms of wireless interference. A small subset of these new nodes has been developed in a more compact design and is aimed for mounting and operation in vehicles. Moreover, NITOS is being extended to support metro-scale wireless communications. In particular, one Worldwide Interoperability for Microwave Access (WiMAX) base station and two Long Term Evolution (LTE) base stations will be soon deployed and installed. Client devices with WiMAX and/or LTE connectivity will also be added to the testbed's infrastructure. Part of them will be deployed as WiMAX/LTE network interface cards in the existing NITOS nodes. Apart from those, handsets with WiMAX/LTE interfaces will be acquired, to be used by real volunteers, in order to experiment with real mobility scenarios.

The architecture of the NITOS testbed is illustrated in Fig. 2. NITOS is comprised of three discrete wired local networks. The first one is the control network, used for logging into the nodes via the NITOS server, sending and receiving OMF/OML traffic, and for some other simple management tasks. The second one is the Chassis Manager (CM) network. Each standalone node in NITOS features an independently powered microcontroller board called CM card, whose main duty is to power the node on or off. These boards feature an Ethernet interface connected to the CM network, through which remote out-of-band power management of the nodes can take place. The third wired local network is an OpenFlow-capable experimental network. This offers the ability to program customized routing schemes inside this network. This network is virtualized by using the FlowVisor tool [6], so that each user can only direct traffic towards resources belonging to his/her slice. In fact, there is a one-to-one correspondence

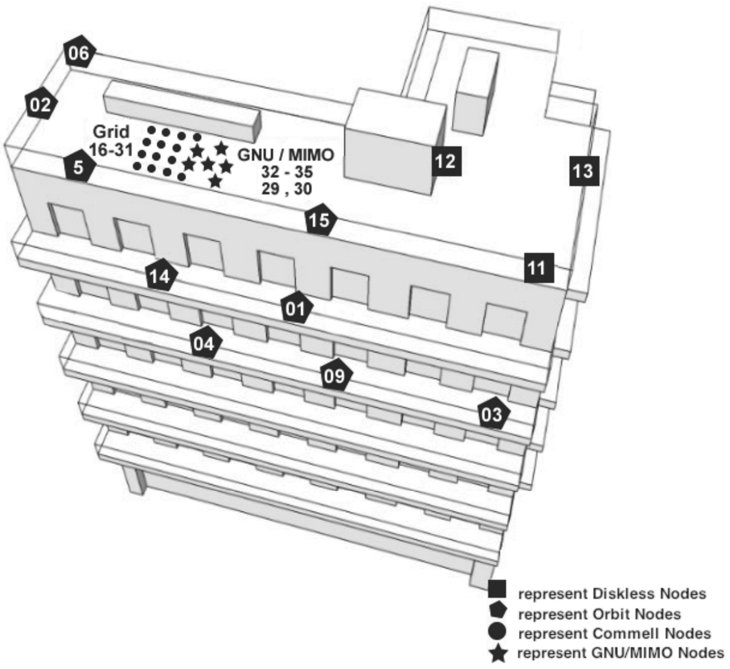

06
02
5
Grid 16-31
GNU / MIMO 32 - 35 29 , 30
12
13
15
11
14
01
04
09
03
■ represent Diskless Nodes
⬟ represent Orbit Nodes
● represent Commell Nodes
★ represent GNU/MIMO Nodes

Fig. 1. Blueprint of the NITOS testbed deployment in the ECE Dept. building of University of Thessaly

between OpenFlow switch physical ports and standalone wireless NITOS nodes, therefore there is no need to reserve OpenFlow resources separately.

Apart from those three wired local networks, there is also the wireless network that is formed by the Wi-Fi-enabled nodes and is primarily used by the users for experimentation purposes.

A central server, called NITOS server, is used to host those networks and also serves as the gate where experimenters enter and gain access to the testbed resources. Moreover, NITOS offers access to a second server, called the Development server, that is internally connected to the NITOS server and can be accessed by registered users. The rationale behind the use of the Development Server is to move high performance demand activities, such as driver compilation and development, out of the main NITOS server.

2.2 Testbed Software

NITOS is remotely accessible through a sophisticated managerial system, developed in University of Thessaly (UTH), called NITOS Scheduler [7]. The front-end of this system is a web interface accessible by registered users through the NITOS website, that allows a researcher to reserve some of the testbed's resources for a specific time duration. At the backend, a set of scripts running on the NITOS main server take care of authorization and access control, based on the reservations.

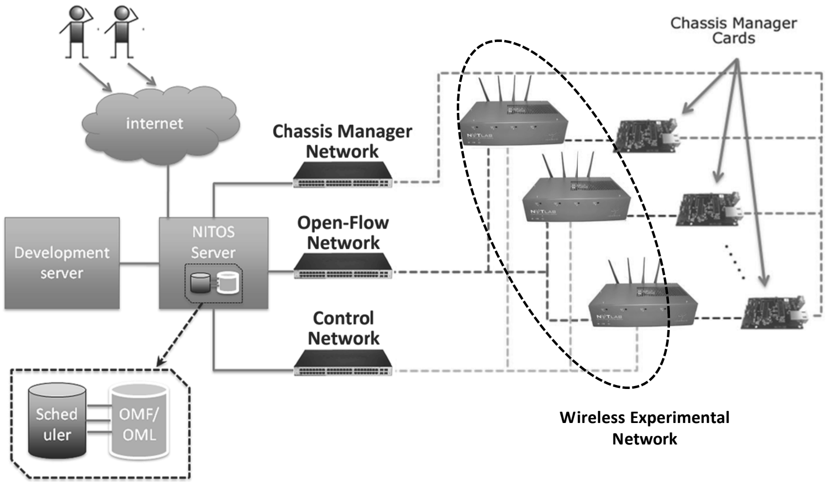


Fig. 2. Blueprint of the NITOS testbed architecture. NITOS testbed's Control, Open-Flow, Chassis Manager, and Wireless Experimental Networks are accessed by remote users via the NITOS server. OMF and NITOS scheduler are software components residing on the NITOS server.

Once granted access to his/her reserved resources, the user has full control over them. However, acknowledging the need to provide to its users an efficient way to handle simple, yet frustrating, management and experiment setup tasks, NITOS testbed has adopted the cOntrol & Management Framework (OMF) [8], already since the beginning of the testbed lifetime. This choice was dictated by the fact that OMF was at the time (and still is) the control and management framework (CMF) with the widest diffusion among testbeds internationally, especially wireless testbeds, so it had been already successively tested by a large user community. OMF, initially developed at the ORBIT testbed [9], but currently maintained and further developed by NICTA [10], allows researchers to describe, instrument and execute their experiments. From a testbed operator's point of view, it provides a set of services to efficiently manage and operate the testbed resources (e.g. installing OS images, resetting nodes).

Apart from these software tools, NITOS features a set of tools related to measurements. The most important is OMF Measurement Library (OML) [11], a framework dedicated to experiment measurements handling, developed by NICTA. Apart from OML, a number of custom applications, developed by UTH, are available for NITOS users. All these tools are described in detail in sections 3 and 4, along with background for their necessity and impact.

Details on how to use the aforementioned software tools in NITOS are provided in the testbed website. A new user is encouraged to start with consulting the *Basic Tutorial* section, available in [12] and then proceed with the rest of the available documentation.

3 Measurements for Topology Assessment

Running a network experiment on a entirely private wired testbed offers the advantage of knowing the capacity of each network link in advance. In a wireless testbed, however, such knowledge can not be safely assumed due to the inherent volatility of the environment. Variance of link capacities is greatly dependent on the specific environment where the wireless testbed is deployed. In the case of Wi-Fi testbeds, we can in general discriminate among the following categories:

- **Indoor deployments in special interference-isolated environments**
  These deployments offer the most stable wireless link capacities. No external sources in the Wi-Fi band operate in the vicinity of the testbed. The link capacities can be considered practically stable over time.
- **Indoor deployments near or colocated with interference sources**
  This is perhaps the most common case, where wireless nodes are located in offices and laboratories in campus or research institute buildings, coexisting with Wi-Fi access points used for Internet access and possibly with other devices operating in the same band (Bluetooth, cordless phones, etc). Interference is a major issue in these deployments, especially during working hours, where external activity in these frequencies reaches its peak.
- **Entirely or partly outdoor deployments in external interference constrained environments**
  These deployments are extremely vulnerable to external interference, but are also affected by weather and atmosphere conditions, due to the effect of varying density of scatterer particles. Changes in the topology of physical obstructions can also affect link capacities, especially in crowded outdoor environments.
- **Entirely or partly outdoor deployments in interference free environments**
  These testbeds are only constrained by fluctuations in the environment, due to weather and atmosphere changing conditions, or due to moving physical obstacles.

The fluctuations in the capacities of the links affect the properties of a given node topology. Typically, a researcher intending to run a wireless experiment selects a subset of resources among the available, based on connectivity between node pairs and the qualities of the different links. Therefore, there is a need for software tools to provide this information to testbed users.

In NITOS there are two software frameworks used in order to assist users in assessing the testbed's wireless environment properties and in selecting an appropriate topology for their experiment. The first one is the *NITOS Topology & Connectivity Tool*, which estimates each link's quality by calculating the Packet Delivery Ratio (PDR) over all combinations of physical transmission rates and frequency channels. The second one is called *NITOS Channel Sensing Framework* [13] and it is based on software-defined radio (SDR) devices that feature highly flexible wireless transceivers and are able to provide highly accurate channel sensing measurements.

3.1 The NITOS Topology and Connectivity Tool

The NITOS Topology & Connectivity Tool [14] collects information for link quality measurements with the aid of *Topology and Link Quality Assessment Protocol* (TLQAP) [15], a framework implemented in the Click modular router [16], taking advantage of the available Click extensions for the MadWiFi driver [17].

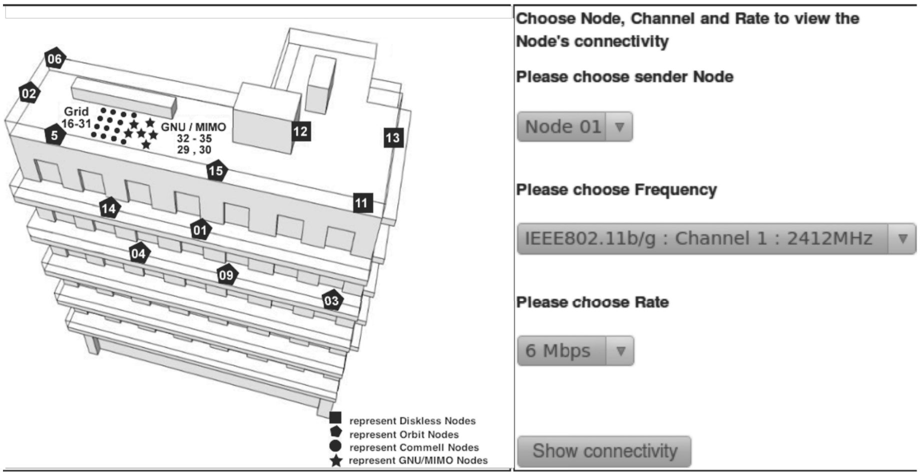
TLQAP is based on actual throughput measurements of a fixed number of consecutive packet transmissions, initiated at each testbed node. Each TLQAP session is comprised of a number of fixed size packets which are transmitted in one burst at a specific channel and physical rate combination. These packets are addressed to an arbitrary neighbor node of the current transmitter and are transmitted without 802.11 support for low level acknowledgements and retransmissions. Otherwise, the captured packet loss ratio would be lower than the underlying, actual loss ratio.

At the same time, each non-transmitting node sniffs (in monitor mode) and logs all the TLQAP packets that it can hear. TLQAP packets feature a special header placed immediately after the Ethernet header. The header fields are the sender IP address along with globally agreed identification numbers for the channel and rate that have been used in the current packet transmission. Such frame header modifications were made possible via the use of the MadWiFi open-source driver.

It must be noted that this mechanism is somewhat equivalent to a typical broadcast transmission. However, in 802.11, broadcast transmissions always use the lowest physical rate. To assess link qualities at higher rates, consecutive unicast transmissions would be required, which would undoubtedly greatly increase the complexity and delay of the framework. Thus, through TLQAP, there is a large gain in efficiency and speed.

The PDR from node X to node Y is calculated by dividing the number of packets received by Y by the number sent by X. Originally, the respective session transmission delay was also recorded, in order to assess congestion at each channel due to transmissions from testbed external Wi-Fi transmitters. Currently, however, congestion assessment measurements are being conducted through another framework, described in the next subsection.

The NITOS Topology & Connectivity tool, developed for NITOS in order to exhibit measurements obtained through TLQAP sessions, is comprised of a web interface, a database and a set of .dot scripts. The experimenter can use the interactive web interface to select particular testbed nodes and it in turn provides him/her with a graphical illustration of the outgoing links in the vicinity of each node and the respective PDR measurements. A snapshot of this interface can be viewed in Fig. 3. The PDR measurement data is stored in a database and collected when TLQAP is enabled by the system administration. TLQAP sessions are scheduled frequently, at periods when none of the nodes is reserved, as a means to keep the link quality information up-to-date. The .dot scripts are then used to generate the graphs showing the PDR metrics by submitting queries on the database. In Fig. 4, the quality of links from node 1



Choose Node, Channel and Rate to view the Node's connectivity

Please choose sender Node

Node 01

Please choose Frequency

IEEE802.11b/g : Channel 1 : 2412MHz

Please choose Rate

6 Mbps

Fig. 3. Web interface for connectivity tool

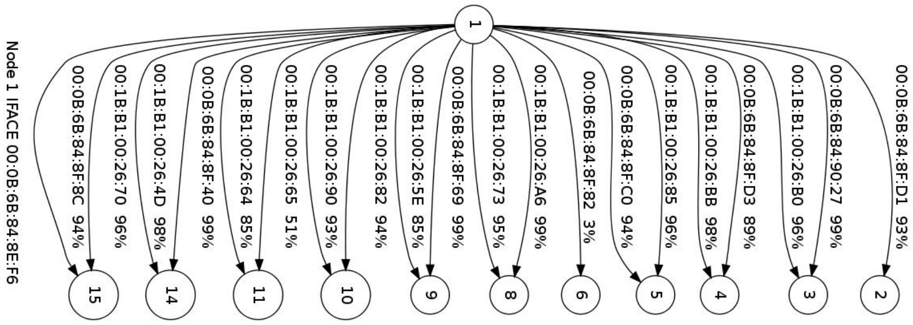


Fig. 4. Link quality graph for node 1

towards its neighbor nodes is illustrated. Each node is indicated by a circle and the PDR of each link is reported on graph edges that indicate link connectivity towards Wi-Fi interfaces of certain nodes.

3.2 The NITOS Channel Sensing Framework

Link quality information is often not sufficient on its own, as congestion and interference from external devices might significantly degrade link throughput. These effects could be caused by external Wi-Fi sources, such as access-points in the vicinity, or non-802.11 devices operating in the unlicensed band, such as Bluetooth devices or microwave ovens. It is important to note that if the signal received by a node is not 802.11-compliant, the receiver will not be able to decode it. Furthermore, if the received signal level is below the channel sensing threshold of a node's wireless card, the node will not back off when it has a frame to transmit. For these reasons, using packet sniffers to detect traffic or relying on the 802.11 backoff mechanism to assess it are not totally reliable methods, and a more sensitive mechanism should be used instead.

Since NITOS is an outdoor-deployed, non-RF-isolated testbed with significant external interference conditions, it is important to provide its users with a powerful spectrum sensing platform, that must be able to exhibit accurate information regarding the 802.11 and non-802.11 activity in each Wi-Fi band channel. For this purpose, the NITOS Channel Sensing Framework [13] is used, which exploits the spectrum sensing capabilities of the NITOS testbed's Software Defined Radio boards. In particular, the used devices are the computer-hosted USRP boards [18] from the Ettus company. Such a board can be viewed mounted on a NITOS node in Fig. 5. These are used in conjunction with GNU Radio [19], an open-source software development toolkit that provides signal processing blocks to implement software radios. A total number of 9 wireless nodes are equipped with USRP1 and USRP N210 devices, spanning the NITOS testbed deployment.

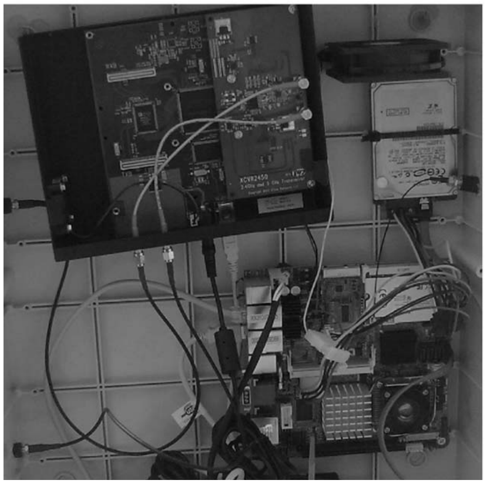


Fig. 5. USRP board mounted on NITOS node

Focusing on the spectrum sensing goal, the Channel Sensing Framework estimates the occupancy ratio per sampled frequency, considering the Received Signal Strength (RSS) measurements that exceed a predefined RSS threshold.

Further details for Channel Occupancy Ratio computation can be found in [20]. The framework allows a user to enable scripts that configure a multitude of sampling parameters, such as:

– the list of frequencies that will be sampled,
– the duration of sampling per individual frequency,
– the number of iterations of the repeated sensing procedure,
– the overall sampling period,
– and the RSS threshold that will be used for measurement filtering.

A flowchart representation of the underlying sensing algorithm is depicted in Fig 6 and shows the execution steps of the spectrum sensing procedure, as triggered by the input configuration parameters provided by the user.

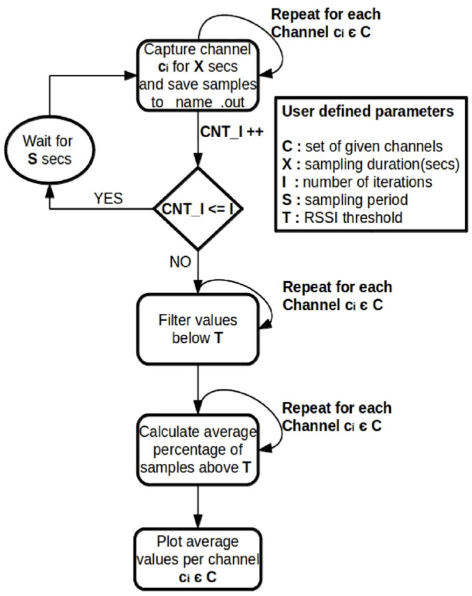

Repeat for each Channel ci ε C
Capture channel ci for X secs and save samples to name .out
CNT_I ++
Wait for S secs
YES
CNT_I <= I
NO
User defined parameters
C : set of given channels
X : sampling duration(secs)
I : number of iterations
S : sampling period
T : RSSI threshold
Repeat for each Channel ci ε C
Filter values below T
Repeat for each Channel ci ε C
Calculate average percentage of samples above T
Plot average values per channel ci ε C

Fig. 6. Flowchart of the Channel Sensing Framework Procedure

The user is able to get a graphical representation of each measurement set that has been already stored in NITOS database, through a web interface. An example is illustrated in Fig. 7, where we can see channel occupancy for the 2.4GHz and 5GHz bands. Various statistical measures can be extracted from the corresponding records, such as average and deviation values per frequency or per individual iteration, and correlation data for measurements captured by different USRP-enabled nodes.

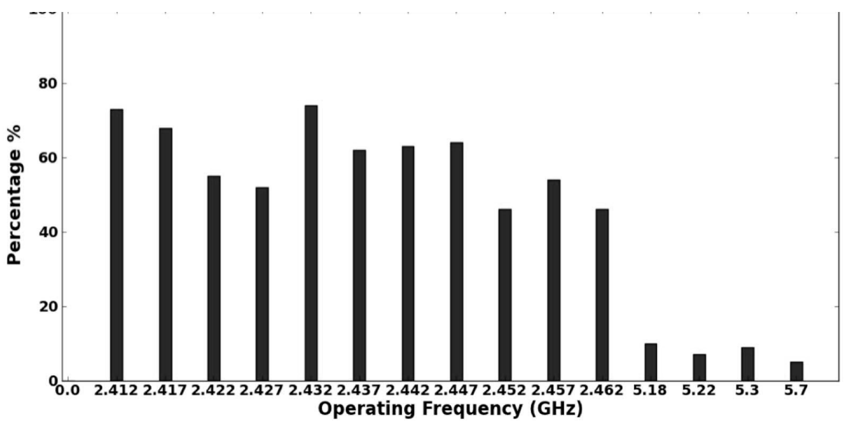


Fig. 7. Plot of the COT measurements per selected frequencies (in 2.4 GHz & 5 GHz bands)

4 Measurements for Analysis of Experimental Results and Experiment Steering

Selection of an appropriate topology for an experiment is only the first step for the experimentation procedure. The most important part of the procedure is, however, collection of measurements at experiment runtime. This constitutes in capturing the values of specific experiment variables emanating from typically multiple sources in a synchronized and organized manner. These variables may be generated in software applications running in some of the experiment's resources, or may be capturable measurements of interest, which assist the user in evaluating the experiment's outcome.

The basic framework used in NITOS for handling experiment measurements is OML [11], which stands for OMF Measurement Library. OML was initially developed as part of the OMF testbed control and management framework, but it evolved in an independent software. Both OMF and OML are being developed by NICTA (Sydney, Australia) as free and open source software. Another tool used in NITOS, but currently still restricted for developers, is spectrum monitoring during an experiment, through the USRP-enabled nodes. Below we describe these components in detail.

4.1 The OML Measurement Framework

OML is a framework designed to support the entire lifecycle of measurements, from their generation and capturing, to their storage and visualization. In Fig. 8 we can see an overview of the framework's architecture. It is based on the client-server paradigm, where an OML server collects the measurements from client applications, running on experiment resources.

Measurements are captured at the resources in so-called measurement points, which are pieces of code which inject given variables to the OML server. OML provides dedicated software libraries which allow insertion of such code inside the source code of the applications or building of wrapper OML-enabled applications around the interface of the original ones. Measurements may pass through a sequence of filters before sent to the server, if the user wishes to perform some processing after their capturing. Examples of available filters are averaging over a given time interval, or selecting the maximum value among a fixed number of measurements.

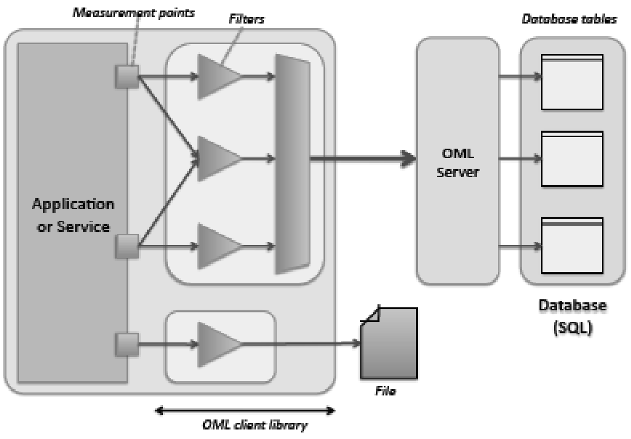


Fig. 8. Overview of OML architecture (source: http://mytestbed.net)

At the server side, measurements are stored in Sqlite databases, organized with timestamps. In an OMF experiment, all the measurements are gathered in a single database corresponding to the experiment, and measurements from different applications are stored in different tables of this database.

Another useful functionality offered by OML is the ability to visualize graphical representations of specific variables at experiment runtime. In this way, the researcher can save a lot of time in the process of evaluating the outcome of an experiment, and perhaps even stop execution if he/she realizes that it does not evolve as expected. In Fig. 9 a snapshot of such a graphical representation can be viewed.

An important featured offered by OML, in conjunction with OMF, is the ability to buffer measurements inside a resource and dump them to the OML server in a batch. This is extremely useful in cases involving mobile resources, where the control network is wireless and connectivity may be temporarily lost between a resource and the server for a particular period. Originally, switching between local buffering of data and sending them to the server was handled manually. UTH contributed an improved version of this feature, where a daemon checking connectivity between the resource and the server is in control of the operating mode.

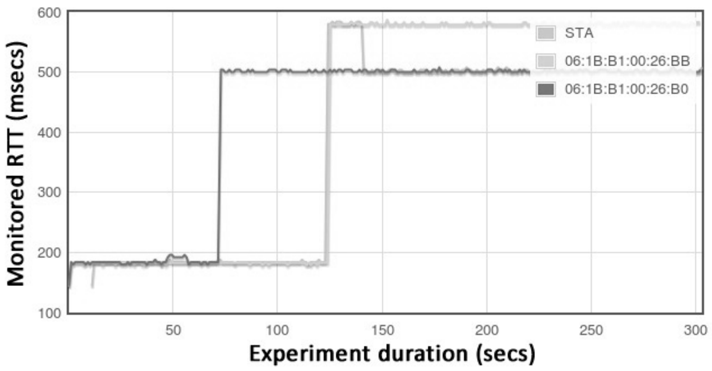


Fig. 9. Runtime visualization of experiment measurements through OML

4.2 Measurement-Based Experiment Steering

As already mentioned, OML was originally a companion framework for OMF. Therefore, it is not surprising that the OMF experiment control framework features various functionalities that exploit the capabilities of OML. Perhaps the most interesting from an experimenter's point of view is the ability to steer an experiment based on the values of specific measurements.

This feature is part of OMF's event-based mechanism. With this mechanism, one can define specific events in an experiment description script, and specific actions to take place when these events are triggered. An example of such a scenario would be for example a dynamic re-association of a wireless station to a new access-point if it experiences congestion over a given level, where the level of congestion is measured and updated dynamically through OML. If all access-points in the area exceed that threshold, then the station could just associate to the access-point experiencing the least congestion (i.e. the trigger condition could be comparison between measurements).

4.3 Spectrum Activity Monitoring

Apart from providing an accurate image of channel occupancy and activity before the execution of an experiment, which serves as a preventive measure against experiencing severe interference, a wireless testbed should ideally offer the ability to experimenters to monitor activity during experiment runtime, particularly activity in the channels utilized in the experiment. This allows to match an unexpected fluctuation in the performance metrics of an experiment with an associated traffic burst emanating from an external source, causing interference or congestion. For instance, a link's throughput could drop to a half of what it was, if during an experiment an external link in the vicinity is using the same channel for data transfer.

NITOS is looking to fill that gap through a more sophisticated framework around its software defined radio enabled nodes. This is currently in development stage, and still not offered to users of the testbed. The idea is that in parallel with an OMF experiment, a background monitoring of the experiment's frequencies takes place from USRPs in the vicinity of the nodes. The user can then compare experiment measurements with spectrum activity data, possibly even by visualizing them in a single graph window.

5 Examples of Experiments Conducted in NITOS

In this section, we describe two experiments conducted in the NITOS testbed, as part of the research activities in European projects which utilize its services. With each description, we also explain the tools used for measurements, so as to provide examples of their usage in real experiments.

5.1 The CONECT Project's Diamond Network Experiment

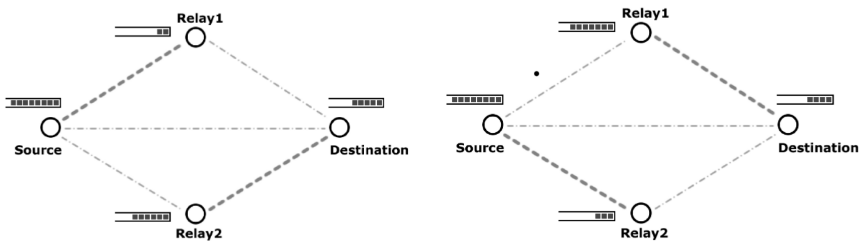
The Cooperative Networking for High Capacity Transport Architectures (CONECT) [21] project proposes a holistic network design approach, by exploiting cooperative forwarding strategies. In order to harvest the benefits of such a research effort and validate the designed cooperative protocols, NITOS is used as a performance evaluation tool that ensures the validity and robustness of the cooperative schemes proposed.

Briefly, the work being done in CONECT involving NITOS includes sending unicast traffic through a proposed cooperative relaying scheme, as well as an extensible scenario over multicast sessions. It also includes insights for applying those cooperative schemes to realistic scenarios, where the benefits of collaboration among nodes can reveal the prospective performance improvements.

The relay-assisted topology in Fig. 10 illustrates the canonical diamond network, containing a source, two relays and a destination (single unicast session), where we consider the joint scheduling and power allocation problem with the objective of stabilizing the network and either maximizing throughput or minimizing total power consumption [21],[22]. There is a network controller that chooses between two feasible scheduling action sets, activating either the set depicted in Fig. 10(a) or the one in Fig. 10(b). The transmissions permitted at each time slot are indicated with bold dotted lines. The information flow is towards the destination, and since the networking conditions (buffer queuing, channel quality) change over time, this will cause the network controller that implements an optimal decision policy to select a particular schedule. The key principle is that the scheduling decision process does not lead in system starvation, since changes in network dynamics are coupled with schedule selection [22].

In Fig. 11 the multicast extension of the problem is depicted. Enhancing the previous scenario of activating single session schedules, to improve networking performance, we consider an extension of the cooperative notion in multicast wireless scenarios, where we assume the existence of a multicast group that desires to experience QoS guarantees.

All these scenarios require careful design in the experimentation methodology and in the evaluation of the results. Instrumentation and repeatability in experimentation are of vital importance for collecting rigorous results and evaluating the performance of the cooperative schemes. An ideal feature would be the ability to exactly match the gathered experimentation results with those being proved by the use of mathematical tools and theory [22]. However, this is not feasible, not only because of the gap between theory and practice caused by modeling simplifications that theory adopts, but also because of a potential lack in a systematic method of experimentation. With the joint use of the OMF framework and the NITOS tools (Connectivity Tool, Scheduler) [7,14,15], reproducibility of the experiment flow is achieved along with monitoring of the varying channel conditions per run. These imply multiple benefits for the experimenter, such as the ability to evaluate the robustness of an algorithm against channel conditions or against different topologies, test the accuracy of the model adopted in theoretical analysis regarding the wireless environment, and derive average and outage performance metrics through repeated experiments.



(a) 1st feasible scheduling set (b) 2nd feasible scheduling set

Fig. 10. Enabling unicast parallel transmissions in the diamond network. Each subfigure indicates a feasible schedule that can be activated each time slot.

In Fig. 12 we can see an example of a diamond network topology in NITOS. It shows some nodes that can be selected to form a diamond network in the NITOS grid. This pattern is not limited to the particular nodes, but it can be extended and repeated among all combinations of nodes forming canonical diamonds, and span over the spatial imprint of the NITOS testbed grid. In order to select topologies that best fit on the relaying scenario objective, a user can exploit the NITOS Connectivity tool that was previously introduced. In this way, an *a priori estimate* of the networking states is given, so as to evaluate the gathered performance measurements.

Collection of measurements takes place with use of the OML framework. The forwarding mechanism is developed inside the IP/MAC layer driver, however the procedure of collecting valuable measurements regarding per packet power consumption, rate configuration and throughput is controlled and configured through OMF/OML. For the sake of evaluating the versatility of the experiments, the experiments were run multiple times, with USRPs sensing the related

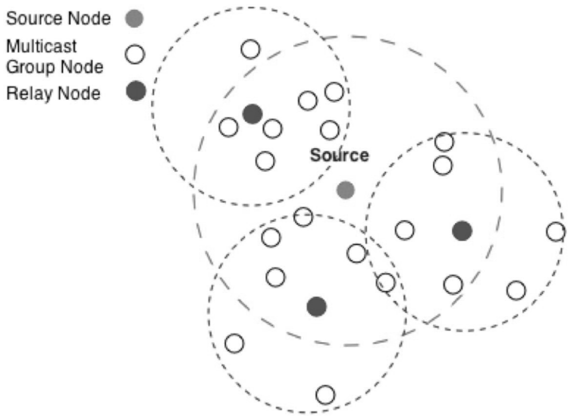


Fig. 11. A source node is aided by some relays that they forward the traffic to the members of the multicast group

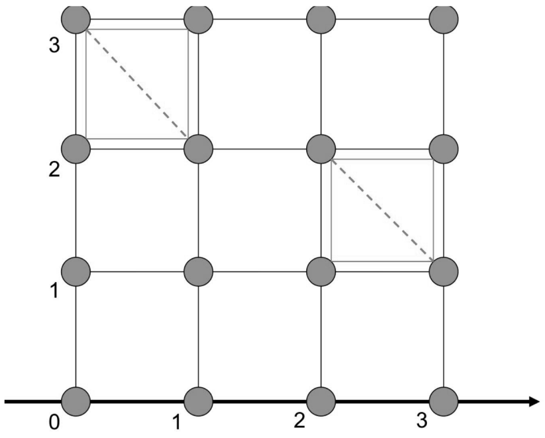


Fig. 12. A diamond network mapping onto the NITOS grid

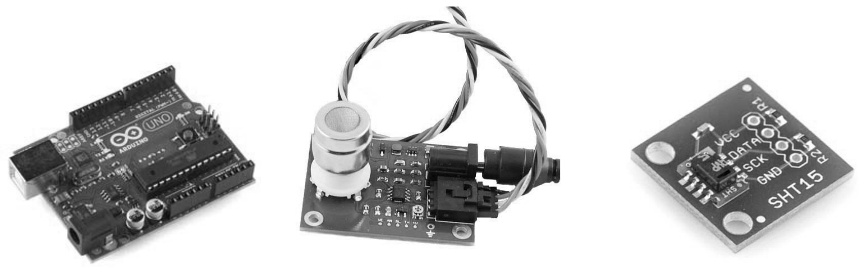
frequencies in the background. Whenever an unexpected pattern occurred in one of the experiment instances, we checked against the spectrum data for unusual activity bursts and, in case of a positive match, this instance was excluded from the results taken into account.

5.2 Vehicle to Infrastructure (V2I) Communication Network

In the context of the European project REDUCTION [23], whose objective is to combine vehicular and Information & Communication Technologies (ICT) for reducing the environmental footprint of vehicles monitored by multi-modal fleet management systems, an experiment involving a Vehicle to Infrastructure (V2I) communication network was conducted. The main idea of the experiment was to demonstrate the scenario where a vehicle equipped with a set of sensors gathers measurements from its environment and communicates opportunistically with Road-Side-Units (RSU), in order to forward the measurements to a centralized framework for storage and analysis.

The connection between the car-mounted node and the RSU was achieved through a Wi-Fi interface. The communication protocol used for this set up was 802.11p, which implies operation in the 5.9 GHz band and use of 10 MHz channel bandwidth (instead of 20MHz used in 802.11a/b/g).

A static NITOS node was used as the RSU, while another NITOS node, mounted on a vehicle (car) was used for gathering measurements and forwarding them to the RSU. A set of sensors was connected to the vehicle-mounted node, sensing temperature, humidity and CO_2. The sensors were connected to an Arduino Uno [24] board attached to the node (cf. Fig. 13). Additionally, a GPS module was connected to the mobile node, enabling measurement of latitude, longitude, altitude, speed, vertical speed and direction.



(a) Arduino board (b) CO_2 sensor (c) Temperature and humidity sensor

Fig. 13. Sensors used on vehicle-mounted node

For the purpose of collecting the measurements, an OML-enabled application was developed for each one of the three types of sensors. Furthermore, the OML disconnected experiment feature was exploited, as the mobile node moved out of the OML server's connectivity range for restricted periods. In order to switch between disconnected and connected mode, the enhanced version with the connectivity monitor daemon developed by the UTH team was used.

In Fig. 14 we can see an overview of the experiment, including a snapshot of the web interface that was created for demonstration purposes, based on the Google Maps API [25]. The data depicted at each point in the map was extracted by the experiment's Sqlite database (in the NITOS server), after conversion into an XML file.

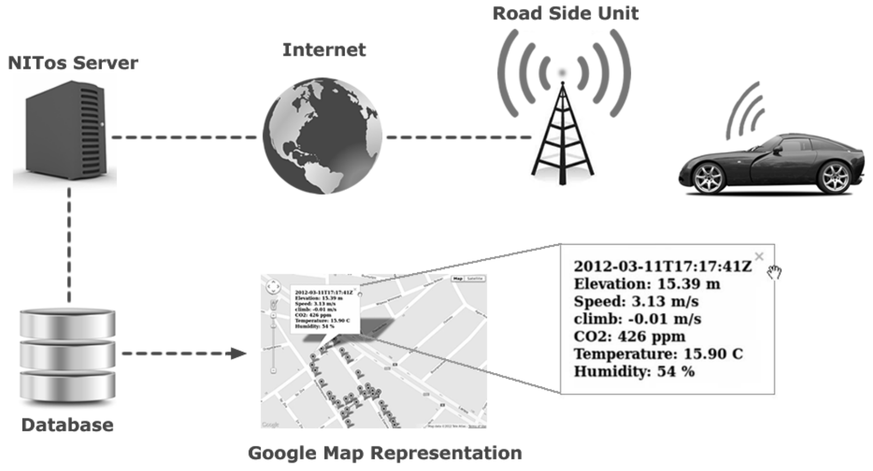


Fig. 14. Overview of V2I demo experiment

6 Challenges - Future Plans

There are a lot of challenges, particularly related to the nature of the wireless environment, which complicate the development of efficient measurement tools for wireless testbeds. We describe some below, along with future plans of NITOS to address them.

As different testbed nodes are affected from different interference sources, depending on their location, channel sensing information should be available on a per-node level. This can be approximated, by spreading the available USRP boards uniformly across the testbed area. Then, for each node, the measurement data from USRPs in its vicinity should be correlated, in order to produce a good estimate of the actual interference perceived by the particular node. Accuracy of the estimates depends of course on the density of USRPs, so there is a tradeoff between the deployment cost of USRPs and the quality of estimation.

Another challenge is related with correlation of experiment results with external interference measurements during an experiment. This is in large part a technical task, involving synchronization between different measurement streams. It is in the plans of NITOS to integrate such a functionality in its measurement toolkit, as already discussed, and preliminary tests are taking place.

Looking at wireless topologies from an experimenter's point of view, further challenges arise. An experimenter might expect the testbed tools to propose him/her an appropriate topology for a specific experiment, instead of using the existing tools to find it out on his/her own. That is, intelligence should be added to the existing topology assessment measurement tools. For instance, an experimenter would just want a "working" diamond topology in the diamond network experiment, not caring about which specific nodes he would be proposed. On the other side, the range of potential experiment setups is so large, that it is not always straightforward to describe the best suited topology efficiently. In case of multiple simultaneous experiments, a combinatorial optimization problem might have to be solved. UTH's group working at NITOS is investigating these issues, with the intention to find the best practical solution possible.

Finally, development of measurement and benchmarking tools for WiMAX and LTE, especially in the form of mobile applications, is an ongoing task. These tools are planned for use in conjunction with the WiMAX and LTE testbed facilities that will be added to NITOS.

7 Conclusions

As measurements constitute the most sensitive part of the experimentation procedure, there is a need for a set of tools and methodologies to efficiently handle related testbed functionalities. In wireless testbeds, where the uncertain environment may affect experiment results, building such a set of tools is particularly challenging. We presented tools and methodologies used by the NITOS testbed in its effort to address these issues, explaining the utility and limitations of each of them. We also provided examples of real experiments, conducted in the context of European research projects, where these instruments were put in practice.

References

1. Feldmann, A.: Internet clean-slate design: what and why? ACM SIGCOMM Computer Communication Review 37, 59–64 (2007)
2. FIRE, Future Internet Research & Experimentation, `http://cordis.europa.eu/fp7/ict/fire/home_en.html`
3. 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)
4. OneLab, Future Internet Testbeds, `https://onelab.eu/`
5. OpenLab, `http://www.ict-openlab.eu/`
6. Sherwood, R., Gibb, G., McKeown, N., Parulkar, G.: Flowvisor: A network virtualization layer. Tech. Rep. OPENFLOW-TR-2009-1, Deutsche Telekom Inc. R&D Lab, Stanford University, Nicira Networks (October 2009)
7. The NITOS Scheduler Management Tool, `http://nitlab.inf.uth.gr/NITlab/index.php/scheduler`
8. Rakotoarivelo, T., Ott, M., Jourjon, G., Seskar, I.: OMF: a control and management framework for networking testbeds. SIGOPS Oper. Syst. Rev. 43, 54–59 (2010)
9. ORBIT Wireless Network Testbed, `http://www.orbit-lab.org/`
10. NICTA, National ICT Australia, `http://www.nicta.com.au/`
11. White, J., Jourjon, G., Rakatoarivelo, T., Ott, M.: Measurement Architectures for Network Experiments with Disconnected Mobile Nodes. In: Magedanz, T., Gavras, A., Thanh, N.H., Chase, J.S. (eds.) TridentCom 2010. LNICST, vol. 46, pp. 315–330. Springer, Heidelberg (2011)
12. Basic Tutorial for using NITOS, `http://nitlab.inf.uth.gr/NITlab/index.php/testbed/instructions/basic-tutorial`

13. Passas, V., Keranidis, S., Korakis, T., Koutsopoulos, I., Tassiulas, L.: An Experimental Framework for Channel Sensing through USRP/GNU Radios. In: Korakis, T., Zink, M., Ott, M. (eds.) TridentCom 2012. LNICST, vol. 44, pp. 383–386. Springer, Heidelberg (2012)
14. Apostolaras, A., Miliotis, V., Giallelis, N., Syrivelis, D., Korakis, T., Tassiulas, L.: A Demonstration of a Management Tool for Assessing Channel Quality Information in Wireless Testbeds. In: Magedanz, T., Gavras, A., Thanh, N.H., Chase, J.S. (eds.) TridentCom 2010. LNICST, vol. 46, pp. 615–618. Springer, Heidelberg (2011)
15. 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, WINTECH 2009, pp. 27–34. ACM, New York (2009)
16. Morris, R., Kohler, E., Jannotti, J., Kaashoek, M.F.: The Click modular router. SIGOPS Oper. Syst. Rev. 33, 217–231 (1999)
17. The MadWiFi project, http://madwifi-project.org/
18. The Ettus Universal Software Radio Peripheral, USRP, https://www.ettus.com/product/
19. The GNU Radio, http://gnuradio.org/
20. Kazdaridis, G., Keranidis, S., Fiamegkos, A., Korakis, T., Koutsopoulos, I., Tassiulas, L.: Novel metrics and experimentation insights for dynamic frequency selection in wireless LANs. In: Proceedings of the 6th ACM International Workshop on Wireless Network Testbeds, Experimental Evaluation and Characterization, WiNTECH 2011, pp. 51–58. ACM, New York (2011)
21. CONECT-Cooperative Networking for High Capacity Transport Architectures, http://www.conect-ict.eu/
22. Apostolaras, A., Cottatellucci, L., Gatzianas, M., Koutsopoulos, I., Li, Q., Navid, N., Wang, L.: D2.2: Advances on packet-level cooperation techniques for unicast traffic transmission. CONECT Project Deliverable (2011), http://www.conect-ict.eu/
23. REDUCTION: Reducing Environmental Footprint based on Multi-Modal Fleet management Systems for Eco-Routing and Driver Behaviour Adaptation, http://www.reduction-project.eu/
24. The Arduino Uno board, http://arduino.cc/en/Main/arduinoBoardUno/
25. The Google Maps API, https://developers.google.com/maps/

Scalability Measurements in an Information-Centric Network

N. Blefari Melazzi, A. Detti, and M. Pomposini

Department of Electronic Engineering, University of Rome Tor Vergata, Roma, Italy
{blefari,andrea.detti,matteo.pomposini}@uniroma2.it

Abstract. Information Centric Networking (ICN) is a new paradigm in which the network layer provides users with content, instead of providing communication channels between hosts, and is aware of the name (identifier) of the contents. In this chapter, first, we briefly describe the FP7 project CONVERGENCE and its approach to ICN. Second, we discuss the needs on measurements required by ICN. ICN is different in several aspects, with respect to the current networking architecture. The measurement needs in an ICN are virtually endless, as designing an ICN is conceptually equivalent to devising a new Internet. Thus, claiming to address this issue in a single chapter would be pretentious. However, the study on ICN is in its initial stage and we want to focus on some of the most pressing and specific aspects of ICN, namely the scalability of its naming and routing functionality. This study is necessary to assess the feasibility of ICN, before addressing other metrics of interest. Thus, the third and main part of the chapter describes our routing-by-name architecture and reports the results of specific measurements on routing issues. Measurements are performed both by means of simulations and by using OneLab, an open, global research network that supports the development of new network services. Our results show that the proposed architecture, designed to improve the scalability of routing tables, is feasible with current technology.

Keywords: Internet Architecture, Future Internet, Information-Centric Networking, Routing, Caching, Scalability, Measurements, Simulation, Test-Bed, Experimental network, OneLab, PlanetLab.

1 Introduction

Information Centric Networking (ICN) is a concept proposed some time ago under different names [1][2], which is attracting more and more interest, recently (see e.g. the papers [3][4][5][6][7][8][9][10] and the projects [11][12][13][14][15][16][17]). ICN proposes a shift from the traditional host-to-host communication to a content-to-user paradigm, which focuses on the delivery of the desired content to the intended users. The basic functions of an ICN infrastructure are to: i) address contents, adopting an addressing scheme based on names (identifiers), which do not include references to their location; ii) route a user request, which includes a "destination" content-name, toward the "closest" copy of the content with such a name; this copy

could be stored in the original server, in a cache contained in a network node, or even in another user's device; iii) deliver the content back to the requesting host.
In our view, the advantages of an ICN are:

1. efficient content-routing. Even though today's Content Delivery Networks (CDNs) offer efficient mechanisms to route contents, they cannot use network resources in an optimal way, because they operate over-the-top, i.e. without knowledge of the underlying network topology. ICN would let ISPs perform native content routing with improved reliability and scalability of content access. This would be a built-in facility of the network, unlike today's CDNs;
2. in-network caching. Caching enabled today by off-the-shelf HTTP transparent proxies requires performing stateful operations. The burden of a stateful processing makes it very expensive to deploy caches in nodes that handle a large number of user sessions. ICN would significantly improve efficiency, reliability and scalability of caching, especially for video [44];
3. simplified support for peer-to-peer like communications, without the need of overlay dedicated systems. Users could obtain desired contents from other users (or from caching nodes) thanks to content-routing and forward-by-name functionality, as it is done today with specialized applications, which, once again, do not have a full knowledge of the network and involve only a subset of possible users;
4. simplified handling of mobile and multicast communications. As regards handovers, when a user changes point of attachment to the network, she will simply ask the next chunk of the content she is interested in, without the need of storing states; the next chunk could be provided by a different node than the one that it would have been used before the handover. Similar considerations apply for multicasting. Several users can request the same content and the network will provide the service, without the need of overlay mechanisms;
5. content-oriented security model. Securing the content itself, instead of securing the communications channels, allows for a stronger, more flexible and customizable protection of content and of user privacy. In today's network contents are protected by securing the channel (connection-based security) or the applications (application-based security). ICN would protect information at the source, in a more flexible and robust way than delegating this function to the channel or the applications [4]. In addition, this is a necessary requirement for an ICN: in-network caching requires to embed security information in the content data-unit, because content may arrive from any network or user node and we cannot trust all nodes; thus, end-users must be able to verify the validity of the received data; caching nodes must make the same check, to avoid caching fake contents;
6. content-oriented quality of service differentiation (and possibly pricing); provision of different performance in terms of both transmission and caching. Network operators (especially mobile ones) are already trying to differentiate quality and priority of content, but they are forced to use deep packet inspection technologies. ICN would let operators differentiate the quality perceived by different services without complex, high-layer procedures [6], and off-load their

networks via caching, a very handy functionality, particularly for mobile operators who can differentiate quality and priority of content transferred over the precious radio real estate;

7. content-oriented access control, providing access to specific information items as a function of time, place (e.g. country), or profile of user requesting the item. This functionality also allows implementing: i) digital forgetting, to ensure that content generated at one period in a user's life does not come back to haunt the user later on, ii) and garbage collection, deleting from the network expired information;
8. possibility to create, deliver and consume contents in a modular and personalized way;
9. network awareness of transferred content, allowing network operators to better control information and related revenues flows, favoring competition between operators in the inter-domain market and better balancing the equilibrium of power towards over the top players;
10. support for time/space-decoupled model of communications, simplifying implementations of publish/subscribe service models and allowing "pieces" of network, or sets of devices to operate even when disconnected from the main Internet (e.g. sensors networks, ad-hoc networks, vehicle networks, social gatherings, mobile networks on board vehicles, trains, planes). This last point is maybe the most important one, especially to stimulate early take up of ICN in selected (and possibly isolated) environments.

On the cons side, ICN has some drawbacks and challenges. A first, obvious, con is that it requires changes in the basic network operation. A second con is that it raises scalability concerns: i) the number of different contents and corresponding names is much bigger than the number of host addresses; this has implications on the size of routing tables and on the complexity of lookup functions; ii) in some proposed ICN architectures [3], delivering contents back to requesting users requires maintaining states in network nodes.

In this chapter, first we briefly describe the approach of the FP7 project CONVERGENCE [16] to ICN. Then, we discuss the needs on measurements required by ICN. ICN is significantly different with respect to the current networking architecture, and poses several new requirements to measurements, which have to be performed both in the current network, to understand some of its aspects useful for the design of ICN, and (experimentally) in the new one. The third and main part of the chapter reports the results of specific measurements performed via simulations, and by using OneLab, an open, global research network that supports the development of new network services [18].

2 The CONVERGENCE Project

The CONVERGENCE project [16] has the aim of designing and evaluating an Information and Communication Technology (ICT) system based on a common and self-contained data unit. The ultimate goal of the CONVERGENCE system is to facilitate, enhance and make more efficient the access to and transaction of resources in networked environments. Resources can be media contents, data about services, or digital representation of real-world objects and people. All information required to attain this objective is embedded within the data unit, including signaling, control, and security information, minimizing the need of using external information or states stored outside the data unit itself (e.g., in network nodes). In the CONVERGENCE system, the basic unit of distribution and transaction is called Versatile Digital Item (VDI).

We can describe the CONVERGENCE system, its features and its expected advantages in terms of four high level components, corresponding also to areas of work and research: the VDI, the applications, the middleware and the network (see Fig. 1).

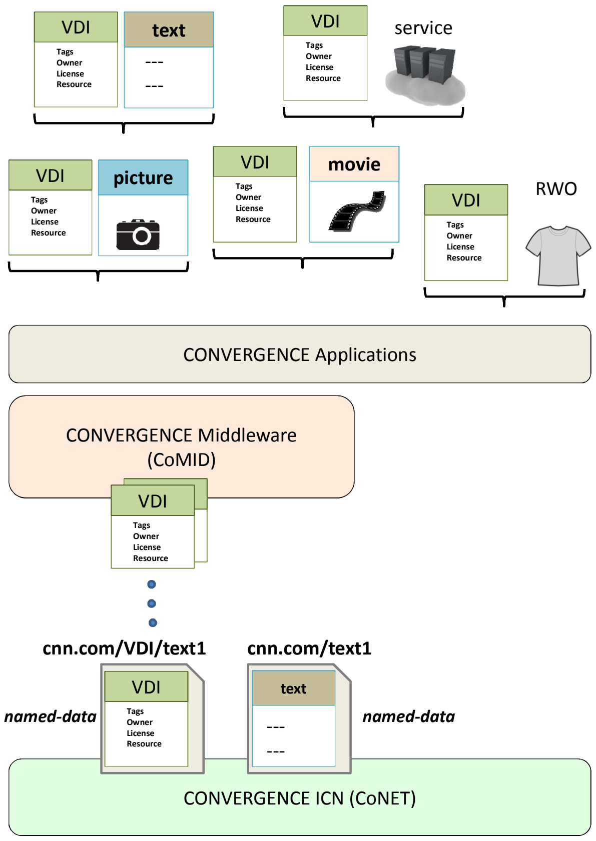


Fig. 1. CONVERGENCE System

2.1 The Unit of Distribution and Transaction

The first area of work is the definition and standardization of a new fundamental unit of distribution and transaction, the VDI. The VDI is a general purpose container which can be used to describe and encapsulate, or make reference to, any kind of resource.

The actual resource can either be physically embedded in the VDI or reside elsewhere and be referenced within the VDI. Resources can be not only classical media files (i.e. texts, pictures and movies), but also data about services, people and Real World Objects (RWOs) (e.g., items of merchandise identified with an RFID). VDIs bind together *meta-information*, which describe the resource, and the *reference to the resource* or the *resource itself* (audio, images, video, text, descriptors of RWOs, descriptors of People, other VDIs, etc.). The meta-data describing the VDI include: i) structural information, describing the content of the VDI; ii) security information (e.g. digital certificates) that enable a recipient to verify integrity and provenance of the resource, and allow legitimate users to decrypt the resource, if necessary; iii) rights information, defining rights to use the resource, and an expiry date for the resource, supporting “digital forgetting”.

VDIs are identified by a unique identifier, which is translated (or which is identical) to the network-level name used to route the VDI. The basis for the definition and standardization of the VDI is the MPEG-21 Digital Item [19].

The advantages of having a unique and standard unit of distribution and transaction are easy to understand and include the possibility of defining common mechanisms for handling structured bundles of different and complex information. The availability of these mechanisms will also provide new possibilities for integrating information about RWOs, services and people. Potential beneficiaries include e-auction sites such as E-Bay, location based services, such as “Friend Finder”, retail, logistic and goods handling companies. In addition, the combination within the same data unit of data and metadata will allow/simplify several important functionality, as it will be described in the following (e.g. searching functions and web engines operation).

2.2 The Applications

The second area of work is the definition and implementation of tools and of applications, relevant to the needs of business and educational organizations participating in the project, and showing the CONVERGENCE potentiality. Tools are re-usable Application elements, which facilitate re-use of code in Applications; an Application can make use of several tools. Our tools and applications exploit the VDI concept and make use of our middleware and network functionality, so offering to end-users the advantages brought about by our system.

The project designed and implemented four main applications to show the usefulness of CONVERGENCE in four real-life scenarios. The four scenarios are: (i) management of audiovisual material; (ii) management of a large photo archive; (iii) customer relationship management and logistics for the retailing sector; (iv) augmented lecture podcast service enabling a collaborative learning environment.

Other two applications have been built later on by integrating the four main applications; the first integrated application merges the first (video) and fourth (podcasts) original applications; the second integrated application merges the second (pictures) and third (retail) original applications. The aim of the integrated applications is to show that our system is flexible enough to combine different applications in one and to exploit common VDIs.

The advantages of our tools and applications include: i) the provision of basic, easy-to-use functionality to applications developers; ii) the solution of specific needs of consortium partners; iii) the possibility of running real world trials to test the system, and the provision of a basis for future commercial exploitation.

2.3 The Middleware

The third area of work is the definition and standardization of a new open source, extensible, middleware. The CONVERGENCE system supports some sophisticated functionalities (publish/subscribe services, searching functions, security functions), which we think are too complex to be implemented at the network layer, inside routers. Thus, we decided to implement them in a subset of nodes and at the middleware layer.

A first important task of the CONVERGENCE Middleware (CoMid) is the support of a publish/subscribe service model: subscribers register their interest in a resource and are asynchronously notified when publishers make available resources that match their interests. Matching between subscription and publications is based on attributes contained in the VDIs of published resources and on conditions specified in the subscriptions. Publish/subscribe differs from the more traditional request-response service model in a number of ways; the interacting parties do not need to "know" each other. Also, they do not need to know how many subscribers will consume the data they have produced. Publishers and subscribers do not need to interact directly: data consumers will receive the desired data when they will be produced by publishers; publishers do not have to care or check or wait that subscribers consume the data they have produced.

Thus, publish/subscribe effectively decouples the application end-points in space and time. This decoupling of publishers and subscribers offers a much enlarged and flexible typology of services. For these characteristics, publish/subscribe is well suited for disseminating data to a wide and dynamic audience.

The data unit of the CoMid is the VDI.

In practical terms, resource providers using CONVERGENCE will publish VDIs to the middleware, making the middleware aware of the characteristics of the resources of such VDIs. Such awareness enables consumers to subscribe to and receive updates (notifications) both for known resources (e.g. the repair manual for a piece of equipment) or for resources satisfying a given search criteria.

For instance, Alice may be interested in receiving offers for a model of camera she wishes to buy. Alice issues a subscription to the CONVERGENCE middleware describing the camera. When a reseller publishes an offer that matches the request of Alice, she will receive such offers by the middleware. Alice will receive the offers asynchronously, i.e. when connected to the pub-sub system, independently of the connection status of the publisher. Offers are carried by VDIs and, in this scenario, the resource in the VDI could be a web-page where Alice can buy the camera.

It is important to observe that not all CONVERGENCE communications must necessarily use a publish/subscribe paradigm. The CONVERGENCE middleware also accepts direct requests to immediately provide specific requested data, with a traditional request-response service model.

A second important task of the CoMid is to support searches, including semantic searches (see [20] for further details on this CONVERGENCE feature).

A third important task of the CoMid is the provision of security mechanisms for: i) assurance of VDI integrity and provenance (i.e., authenticity of the source); ii) governance of VDI access restrictions and confidentiality; iii) issuing and enforcement of licenses; iv) protection of user privacy.

Our CoMid implements the tasks listed above, providing the following overall advantages:

- Dynamicity of VDIs. The information exchanged between providers and consumers is increasingly volatile. Our CoMid allows producers of information to update the information they have released and consumers to check if a digital resource is up to date, to request an update, and to select between several versions of the same item.
- Privacy and security information *built into the VDI*. This feature avoids the need to delegate privacy and security to applications or to transfer protocols, and ensures that VDIs are genuinely trustworthy. Protecting information at the source is more flexible and robust than delegating this function to applications, or securing only the communications channels.
- Support for “digital forgetting”. Our CoMid provides mechanisms allowing users to “unpublish” VDIs and/or to define expiry dates for specific items of information. This ensures that content generated at one period in a user’s life does not come back to haunt the user late. Such mechanisms allow sites and services to perform automatic garbage collection, deleting expired information.
- Incorporation of multimedia standards and Semantic Web technologies in VDIs provides a *homogeneous way of searching and handling structured information*.
- CoMid provides interfaces to manipulate VDIs, together with standard mechanisms for producing, managing and linking VDIs with the corresponding metadata. Characteristic examples of these mechanisms include content protection, rights management and event reporting. This facilitates the production and distribution of content in a uniform, interoperable way, compliant to MPEG-M [21] and MPEG 21 [19] standards.
- CoMid provides users with a global identifier for their work (the VDI identifier).

2.4 The Network: Information-Centric Networking

The fourth area of work is the definition and standardization of a new networking functionality. The middleware, implemented in a subset of all network nodes, needs to transfer data (i.e., VDIs) for its own purposes and at the service of applications. Furthermore, applications need to fetch digital resources described by the VDI.

This functionality could be provided by means of standard TCP/IP means.

Instead, CONVERGENCE has taken an alternative approach, which is more consistent with the use of a common and self-contained data unit at the application and middleware level.

The chosen approach is Information Centric Networking (ICN), briefly described in the Introduction. Our CoNet defines its own data unit at the network layer, called *CONET Information Units (CIUs)*: *interest CIUs* convey requests of named-data (e.g. a VDI); *named-data CIUs* transport chunks of named-data. Named-data is any digital object, uniquely identified by the network with a name (i.e. a string). A named-data can be a VDI or the actual resource referenced to by the VDI. For instance, in Fig. 1 we have both cases: the VDI of text1 of cnn.com, and the actual text1 file of cnn.com. At the network-level, both are named-data; the former is identified by the string "cnn.com/VDI/text1", the latter by the string "cnn.com/ text1".

We identified eight fundamental issues that need to be addressed to design an ICN infrastructure:

1. Primitives & interfaces, which define the relationship of the ICN protocols with the overall architecture.
2. The naming scheme, which specifies the identifiers for the data units (CIUs) addressed by the ICN.
3. The route-by-name mechanism, used by ICN nodes to relay an incoming CIU to an output interface. The output interface is chosen by looking up a "name-based" forwarding table.
4. The routing protocols used to disseminate information about location of CIUs, so as to properly setup the name-based forwarding tables.
5. The data forwarding mechanism that allows CIUS to be sent back to the device that issued a CIU request. Data forwarding cannot use the forward-by-name mechanisms, because, typically, devices/interfaces are not addressed by the content routing plane of an ICN.
6. In-network caching, which concerns the ability of ICN nodes to cache CIUs and to reply to incoming CIUs requests.
7. Segmentation & transport mechanisms (see e.g. [9]) needed to: 1) split a whole content (e.g. a VDI) in different CIUs (or chunks); each CIU is an autonomous data unit with embedded security and addressable by the routing plane; ii) ensure a reliable transfer of CIUs from the origin node (or from a cache node) towards the requesting node; iii) counteract congestion.
8. Security & privacy issues tackling (at least) three specific aspects: 1) how to guarantee content authenticity and protect the network from fake content, which could also pollute network caches; 2) how to guarantee that content be accessed only by intended end users, and 3) how to protect information consumers from profiling or censorship of their requests.

Finally, the network should complement mechanisms provided by the Middleware for the support of the "digital forgetting" and garbage collection functionality. For instance, the network should not forward content whose expiry date is terminated.

The Convergence Network (CoNet) is designed according to these principles.

The advantages of ICN in general and of our CoNet in particular are described in the Introduction.

3 Measurements Needs in an ICN

The measurement needs in an ICN are virtually endless, as designing an ICN is equivalent to devising a new Internet. Thus, claiming to address this issue in a single chapter would be pretentious. However, the study on ICN is in its initial stage and we want to focus on some of the most pressing and specific performance aspects of ICN, namely the scalability of its naming and routing functionality. This study is necessary to assess the feasibility of ICN, before addressing other metrics of interest.

Once the theoretical feasibility of ICN is demonstrated, one could go and study the performance of the other fundamental functionalities, which we listed in the previous section, and to assess the advantages of ICN, as identified in the introduction.

Thus, in this chapter, we focus on measurement issues regarding the scalability of routing-by-name functions, assuming that the ICN is used to fetch current Web contents.

4 Routing-by-Name

In this Section, we briefly recall our reference model [7], and our Lookup-and-Cache solution [10][22], which implements the routing-by-name functionality.

4.1 Reference Model

ICN nodes (see Fig. 2) are interconnected by "sub-systems" [7]. Sub-systems use an underlying technology to connect ICN nodes and can be implemented in several different ways. For instance, a sub-system could be a public or private IP network, an overlay UDP/IP link, a layer-2 network, a PPP link, etc. This is the same concept used in current IP networks, in which different layer 2 technologies connect IP hosts and routers. Nodes can be: ICN end-nodes (or clients) that download contents; ICN serving-nodes (or servers) that provide contents and ICN nodes that relay ICN data-units between sub-systems, which may also cache data.

To provide a content, a server splits the content in blocks of data, named *chunks*, and assigns a unique network identifier to each chunks. A network identifier is a string like "cnn.com/text1.txt/chunk1", which is said to be the "name" of the chunk.

In the CONVERGENCE system, the name could be equal to the VDI identifier or derived from it.

The role of the ICN protocols is to discovery and deliver named chunk. In order to fetch a chunk, a user issues a data unit, named *Interest* message, which contains the name of the chunk. ICN nodes *route-by-name* the Interest message, by using a longest prefix matching forwarding strategy and a name-based routing table. We name the entries of the name-based routing table *ICN routes.* An ICN route has a format like:

<name-prefix, next hop >

A name-prefix should be either the full name of a chunk, e.g. “cnn.com/text1.txt/chunk1”, or a continuous part of it, starting from the first left character e.g. “cnn.com/”.

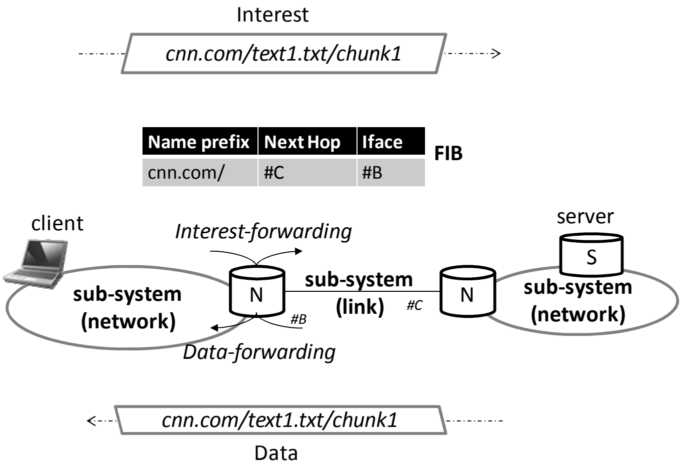


Fig. 2. Network model

The first “en-route” device, be it an intermediate node or the end-server, that has the chunk sends it back within a data unit, named *Data* message, which includes the chunk name. Network nodes forward the Data message towards the requesting client, through the same sequence of ICN nodes previously traversed by the Interest message. These nodes may store the Data in their cache, so as to provide a so-called en-route[1], in-network, caching service. The Data forwarding process exploits reverse-path information either temporary stored in the traversed nodes during the Interest forwarding process (see Pending Interest Table of [3]), or contained in the header of Data message, and previously collected in the Interest message during its forwarding process (see reverse-path source-routing in [7]). Therefore, the routing-by-name process does not involve Data messages, but only Interest messages.

Downloading a whole content is achieved by sending a *flow* of Interest messages to retrieve all the chunks of the content. The sending rate of Interest message is regulated by a receiver-centric congestion control mechanism [23][9], which could be based on the same logic used by TCP. Therefore, in our ICN model, we have endpoints that exchange Interest-Data sequences and the message exchange rate is

[1] We point out that en-route caching does not have an impact on the routing plane. Indeed the routing-plane only routes-by-name requests toward servers. Conversely, in case of off-route caching, the routing-plane should route-by-name requests towards cached contents. The temporal dynamics of these additional “caching routes” is a function of the lifespan of contents in caches, which could be very short. This could cause an excessive routing traffic and processing load. For this reason, an Information Centric Network typically adopts only en-route caching.

regulated by the receiver. Dually, in TCP/IP the endpoints exchange Segment-Ack sequences and the exchange is regulated by the sender.

As regards the naming scheme, several proposals (e.g. [2][3][4][24]) agree in adopting a hierarchical naming. In this chapter, we assume a rather general hierarchical naming scheme where a name is formed by a sequence of *Components*; i.e. a name has the form "Component_1/Component_2/../Component_n". This scheme supports current Web URL, where the Component_1 is the domain name (e.g., "cnn.com") and next Components represent the path of the local resource (e.g., /text1.txt). In addition to these Components, which represent the *content-name*, ICN requires other specific Components, e.g. to represent the chunk number ("/chunk1"), version, etc. The full sequence of Components is referred to as the *chunk-name*.

As said before, in this chapter we focus on a scenario in which the ICN is used to distribute current Web contents and Web servers are replaced by ICN servers. Usually, a Web server provides all contents whose URLs have the same domain-name, e.g. "cnn.com". Therefore, we assume that an ICN sever provides all contents whose names have the same Component_1, which is equal to the domain-name. In this scenario, we argue that the minimal set of routing information needed to route-by-name contents offered by ICN servers depends on the number of domain-names, rather than on the number of content-names or chunk-names. Hence, the name-prefix of an ICN route is a domain-name and, therefore, the number of ICN routes that a node of the default-free-zone should handle is in the order of the current domain-names, i.e. 2 10^8; we assume 10^9 to have some margins [10][22].

We remark that these conclusions are dependent on the assumptions stated above.

Changing the assumption would change the results. For example: i) using a "flat" non-hierarchical naming the number of ICN routes would be higher and likely close to the number of content-names, i.e. 10^{11}; ii) if we allow more than one route per name-prefix, e.g. for routing redundancy or multi-homing purposes, the number of ICN routes would be higher than 10^9; iii) nodes that have a default route, e.g. corresponding to a tier-2 or a tier-3 node of the current Internet, would have a number of ICN routes much lower than 10^9, and so forth.

4.2 Lookup-and-Cache Routing Architecture

The routing-by-name of Interest messages is very similar to the routing of IP packets but, in place of IP-prefixes, the routing-by-name procedure uses name-prefixes, which, in our "fetching web contents" scenario are domain-names. Consequently, it is worth analyzing the feasibility of reusing the architecture of an IP router for an ICN node.

A typical router is composed of three major components: one or two routing engines, line cards that host a forwarding engine and a switch fabric. The routing engine handles the routing protocols and stores the routes in a routing table, named Routing Information Base (RIB). In general, the RIB contains several routes to the same destination and it is implemented by means of cheap and slow memories such as DRAM. The forwarding engine of a line card receives incoming packets and selects the output line card by looking up an on-board routing table, which is named

Forwarding Information Base (FIB) [25][26]. The FIB contains one route per destination, and therefore a smaller number of routes than the RIB. To support packet forwarding at line rate, the forwarding process is carried out by dedicated ASIC chips and the FIB is implemented with fast memories, such as SRAM or TCAM. These memories are expensive, consume a lot of power, and do not follow Moore's Law [27]. After the selection of the output interface, the forwarding engine injects the packet in the switching fabric. The switching fabric is (at least conceptually) an *N*x*N* non-blocking crossbar where *N* is the number of line cards.

If we want to reuse this architecture to route-by-name ICN Interest messages, we should store ICN routes in the FIB and RIB, and properly update the routing and forwarding logics. Hence, a fundamental check is to verify the practical feasibility of storing all required routes in a FIB and in a RIB. As regards the FIB, the maximum size of a SRAM chip is today 32 MByte [28]. Assuming that an ICN routing entry is 45 bytes long [2], the number of routing entries storable in a FIB is in the order of 10^6 (i.e. 32MB/45B). In the previous section we estimated that an ICN node should handle 10^9 routes and thus current FIB technology cannot store the whole set of ICN routes.

Let us now analyze the RIB issue. As in IP, the RIB would contain more than one route per name-prefix; this redundancy mainly depending on the peering relationships among Autonomous Systems. For instance, current BGP data obtained from the AS6447 node [29] show that, on average, its RIB contains 31 routes per destination. As a consequence, we assume that the RIB of an ICN node should handle a number of routes in the order to 10^{10}, i.e. one order of magnitude greater than the number of name-prefixes. In this case, the RIB would require hundreds of Gbytes (i.e., 10^{10} *45B) of DRAM memory and a motherboards with hundreds of memory slots. Current DRAM chips are of 4 GB and motherboards of "expensive" carrier-grade IP routers can host up to 4 memory slot [30][31]. This means that the required increase of capacity is in the order of 10^2. We can conclude that supplying each network node with a motherboard with hundreds of memory slots would dramatically increase the deployment cost of an ICN network, with respect to an IP network.

In order to cope with the capacity issue of the FIB and with the cost issue of the RIB, we propose a *Lookup-and-Cache* routing architecture. In our solution, we use the FIB of a Forwarding Engine as a *route cache* and deploy a *centralized routing engine*, that runs on a server named Name Routing System (NRS), which logically serves all the ICN nodes of a sub-system. Fig. 3 reports an example of Lookup-and-Cache operations. Node *N* receives an Interest message for "ccn.com/text1.txt/chunk1". Since the FIB lacks the related route, the node temporarily queues the Interest message, lookups the route in a remote RIB, gets the routing information and stores it in the FIB, and then it can forward the Interest message. In what follows, we discuss the rationale underlying the Lookup-and-Cache architecture.

4.2.1 FIB as a Route Cache

It is well-known that the relative frequency with which Web contents are requested follows the Zipf's law [32] and that there is a time and space locality of Web content interests. Therefore, a large number of *flows* of Interest messages that an ICN node should *concurrently* route-by-name refer to a small set of contents and, more important, these flows use an even much smaller set of ICN routes, since ICN routes address domain-names rather than single contents. In Section 5, we show that the set of these *active-routes* can be comfortably stored in a SRAM memory. Therefore, we propose to use the FIB as a route cache, which should contain, at least, the entire set of active-routes. When the FIB lacks a route, the node lookups the route in a "remote" RIB and then caches the route in its FIB. When all FIB rows are filled in, new routing entries may substitute old ones, according to a specific *route replacement algorithm.* Furthermore, a routing entry could be removed or updated by a *FIB-RIB consistency mechanism* [22].

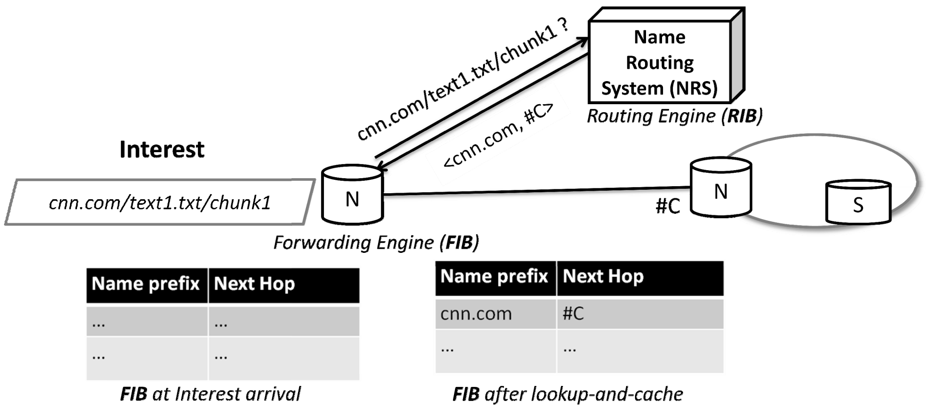


Fig. 3. Lookup-and-Cache concept

4.2.2 Centralized Routing Engine

All ICN routes are contained in the RIB of a Routing Engine, which logically serves all the ICN nodes of a sub-system and runs on a centralized server, named *Name Routing System* (NRS) node. Thus, the cost of an expensive Routing Engine is taken for only one network device, rather than for all network nodes. Of course, the DRAM memory of the NRS node must be able to contain all the ICN routes of all the ICN nodes that it serves. A single NRS node may also serve more than one sub-system; for instance all sub-systems administered by the same company (e.g. a whole autonomous system).

Since many Interest flows use a small set of active-routes, the temporal dynamics of *active*-routes is slower than the flow dynamics. Indeed, a route is used for a period of time that is greater or equal than/to the duration of a single flow. This limits the lookup rate that a centralized Routing Engine should deal with and, in Section 5, we show that this rate is easily supported by current technologies.

So far we have described the "data-plane" of our Lookup-and-Cache architecture, i.e. the procedures carried out to forward ICN messages. In addition to the data-plane, the Lookup-and-Cache architecture (as the IP one) needs "routing-plane" procedures that run on NRS nodes and whose goal is to setup the RIBs. The routing-plane is out of the scope of this chapter; anyhow we point out that our architecture does not impose a specific routing-protocol. For instance, we can support both name-based version of BGP, as suggested in [3], or DONA [2], where the DONA Resolution Handler (RH) has the same function of the NRS node. We conclude by observing that, as it occurs in the current Internet for BGP messages, the ICN nodes should give highest priority to routing signaling messages (e.g., lookup and routing messages), to limit the number of failed communication attempts and the delay.

4.2.3 Route Replacement Algorithm

When a node receives an Interest message for a given content and it is not possible to find a matching route in the FIB, we have a *route-cache-miss* event. In this case: i) if the FIB is not full, the node performs a lookup in the remote RIB and stores the new route in the FIB; ii) the forwarding of the Interest messages is subject to a route-lookup delay. When the FIB is full, the insertion of a new route implies the replacement of an old route. In this case, a *route replacement algorithm* decides whether to lookup the new route or not. In the first case it also decides which old route has to be replaced. In the second case, the Interest message is dropped and subsequently retransmitted by transport level mechanisms.

An inefficient design of the route replacement algorithm would result in an excessive rate of route lookups, with a consequent worsening of delay performance (as more Interest messages will be subject to the route-lookup delay) and an increase of the load of the NRS node. To mitigate these inconveniences, it would be desirable to replace *inactive* routes. Consequently, the design of the route replacement algorithm aims at solving two problems: first, how to identify *inactive* routes and, second, how to behave in case of *FIB overload*, i.e. when there are no inactive routes and a new route needs to be added in the FIB.

In [10][22] we proposed a route replacement algorithm, which assumes that each route contained in the FIB has an inactivity time out (ITO), after which the route is considered inactive; its performance are compared with the Least Recently Used (LRU) policy [43]. Results show that, if the FIB size is over dimensioned and the FIB operates in an unloaded condition, the least recently used route is likely inactive; hence the simple LRU works well, as the more complex ITO. If the FIB size is under dimensioned and the FIB works in overload condition, ITO overcomes LRU as LRU causes an in/out flapping of routes from the FIB.

5 Feasibility Check

In this section we show that our architecture is feasible by using currently available technology. To this end we verify that: i) the capacity provided by current FIB technology is enough to store the expected number of active-routes; ii) the route lookup rate can be supported by current database technology.

On a given node and at a given time, an ICN route is “active” if there is at least one flow of Interest messages using that route. This concept is sketched in Fig. 4, where there are 3 flows of Interest messages toward “cnn.com”. The route toward “cnn.com” becomes active at the start of the first flow and becomes inactive at the end of the last flow. In Fig. 4 there is also a single flow of Interests for “bbc.com”, thus the related route activity has the same duration of the flow.

In the current Internet, a client sends TCP ACK and receives TCP segments from the Web server. In an ICN, a client sends Interest messages and receives Data messages from the ICN server, or from an en-route cache. So, if a client used the ICN to download Web contents, then the traditional flows of TCP ACK messages would be replaced by a flow of Interest messages. Furthermore, on the base of our hierarchical naming assumption, the couple <IP destination address, destination Port> contained in TCP ACK messages would be replaced, in Interest messages, by a chunk-name that contains the domain-name of the destination Web server.

For instance, assume that in the current Internet a host sends an HTTP request towards the domain name "cnn.com". The domain name "cnn.com" will be translated by DNS into an IP address, e.g. 157.166.226.25, a request will be sent to this address and then the data will be directed from 157.166.226.25:80 towards the requesting host, while a flow of TCP ACKs will be directed by the client to 157.166.226.25:80. In the proposed ICN scenario the flow of TCP ACKs would be replaced by a flow of Interest messages for chunks, whose names contain the "cnn.com" name-prefix.

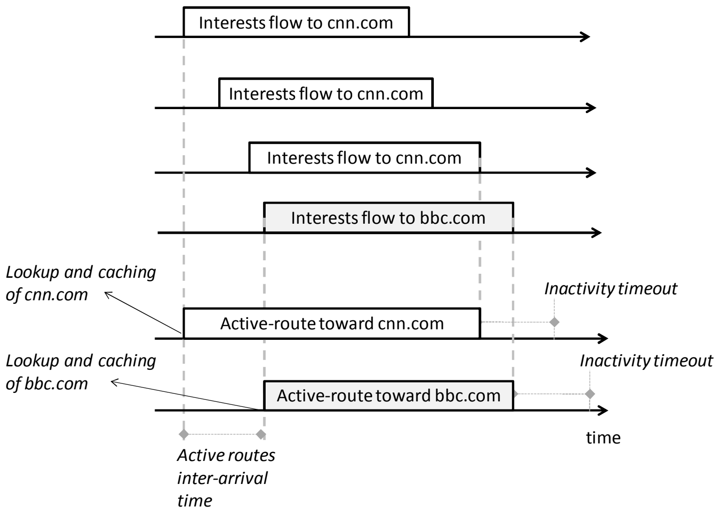

Interests flow to cnn.com
Interests flow to cnn.com
Interests flow to cnn.com
Interests flow to bbc.com
Lookup and caching of cnn.com
Inactivity timeout
Active-route toward cnn.com
Lookup and caching of bbc.com
Inactivity timeout
Active-route toward bbc.com
time
Active routes inter-arrival time

Fig. 4. Flows and active-routes

Using such a mapping between the flow of TCP ACKs and the flow of ICN Interests, we could use current Internet traces to assess the feasibility of ICN. We could replace each ACK of an Internet trace with an Interest message, thus creating a would-be ICN trace. Unfortunately, IP traces usually have anonymized IP addresses, which do not include the domain-names of HTTP GET messages. Hence, we cannot derive the domain-names to be used for the conversion from TCP ACKs to Interest messages, by using such anonymized traces.

To circumvent this problem, we use a simulation approach to associate a domain-name to a flow of TCP ACKs directed towards an anonymized IP address.

The simulation model is depicted in Fig. 5. Briefly, for a given anonymous trace, we randomly associate the web servers' anonymous IP addresses of the trace to a set of public IP addresses, derived (as described below) from the 1 million most used domain-names [33]. Then, we associate each anonymous flow of the trace to a domain-name, randomly extracted among those domain-names that have the public IP address associated with the web server's anonymous IP address of the flow.

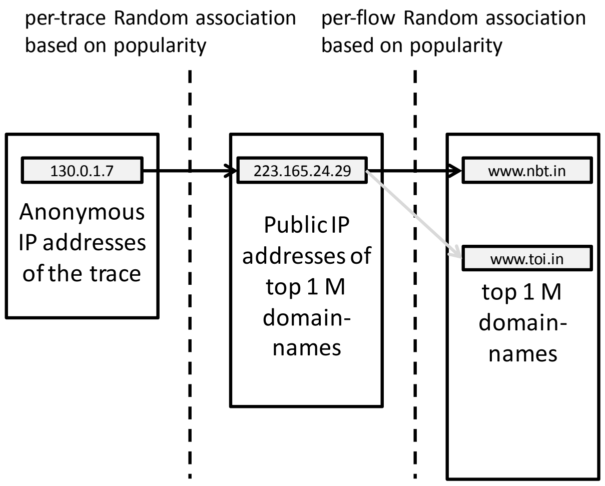

per-trace Random association based on popularity
per-flow Random association based on popularity
130.0.1.7
Anonymous IP addresses of the trace
223.165.24.29
Public IP addresses of top 1 M domain-names
www.nbt.in
www.toi.in
top 1 M domain-names

Fig. 5. Simulation model to associate an anonymous IP address to an actual domain-name

More in details, the simulation model is formed by three phases, as follows:
<u>Phase-1: data structures setup</u>

1. we collect the top 1 million domain-names in a list named $\{DN\}$;
2. for each domain-name DN_i, we model its occurrence probability $opDN_i$ in an Internet trace as a function of its rank position, and according to a Zipf's law. Following the results of [34] we set the value of the Zipf alpha parameter to 1[2];
3. we resolve the list $\{PubIP\}$ of public IP addresses associated to each domain-name[3] (from a machine located in the campus of University of Rome Tor Vergata);
4. for each element $PubIP_i$, we compute its occurrence probability $opPubIP_i$ as the sum of the occurrence probability $opDN_j$ of the domain-names that use the IP address $PubIP_i$;
5. from the anonymized trace, we extract the list $\{AnIP\}$ of unique anonymized IP addresses of web servers;

2 We remind that we are considering the occurrence distribution of domain-names, rather than that of specific contents, and that the parameter alpha of the domain-name Zipf [34] is greater than the one of the content Zipf [32] (e.g, 0.6, 0.8).

3 Since the same IP address may serve several domain-names, the number of unique elements of $\{PubIP\}$ is lower than the length of $\{DN\}$. In our case the ratio between the length of $\{DN\}$ and the number of unique elements of $\{PubIP\}$ is equal to about 1.7.

6. for each element $AnIP_i$, we compute its occurrence probability $opAnIP_i$ as the ratio between the number of HTTP flows that have $AnIP_i$ as destination address and the total number of HTTP flows of the trace.

Phase-2: Random association of anonymous IP addresses to public IP addresses

7. since the number of public IP addresses $\{PubIP\}$ is in the order of 580k while the number of anonymous IP addresses of our trace is lower, we randomly extract a subset of public IP addresses, by using their occurrence probability $\{opPubIP\}$. We refer to this restricted set as $\{rPubIP\}$;
8. we map, one-to-one, elements of $\{AnIP\}$ to elements of $\{rPubIP\}$. We preventively sorted the elements of $\{AnIP\}$ and of $\{rPubIP\}$ on the base of the occurrence probabilities of their elements. Consequently, the element of *AnIPk* with rank *k* in terms of occurrence probability is mapped to the element *rPubIPk* that has the same rank.

Phase-3: Random association of anonymous flows to domain-names

9. for each flow of the trace, we map its destination anonymous IP address $AnIP_i$, to the public address $rPubIP_i$ and we randomly associate to it a domain-name randomly extracted among the ones that use $rPubIP_i$. The extraction is properly weighted by the occurrence probability $opDN_i$.

Since each flow has now an associated domain-name, we can convert the TCP ACKs of a flow in Interest messages, and evaluate the average number of ICN active-routes and the average active-route inter-arrival time by using real Internet trace. Results are reported in Table 1. The Equinix-sanjose-* and Equinix-chicago-* traces [35] are captured on a 10 GigE interfaces of a tier-1 ISP. The Mawi-* traces [36] are captured on a trans-Pacific line operating at 150 Mbit/sec. The Rome-Tor-Vergata trace is captured on the 1 GigE interface of the router gateway of our University [37], which is a tier-3 network. Even in the worst case of the Equinix-sanjose-dirA trace, the average number of active-routes is in the order of 10^3; this value is much lower (by a factor of 10^3) than the capacity provided by an off-the-shelf SRAM based FIB, i.e. 10^6 ICN routing entries, as discussed in Section 4.2.

Table 1. Average number of active routes and inter-arrival times

Trace id	Average value of ICN active-routes (N_{icn})	Average ICN active-routes inter-arrival (I_{icn})
Equinix-sanjose-dirA	4680	0.5 ms
Equinix-sanjose-dirB	1782	1.1 ms
Equinix-chicago-dirB	1576	1.2 ms
Mawi-1	250	4.5 ms
Mawi-2	267	3.3 ms
Rome-Tor-Vergata	185	5.6 ms

Let us now investigate if current database technology can support the required lookup rate. Table 1 reports that the average inter-arrival time between the starts of two consecutive active-routes is in the order of half a millisecond, for the worst trace.

When the FIB memory is dimensioned for containing all active-routes, the inverse of the active-routes inter-arrival time is an upper bound of the lookup rate. Indeed, we need a lookup at the start of the route activity only if that route is not already cached in the FIB. Therefore, an average active-route inter-arrival time in the order of 0.5 ms implies a lookup rate in the order of 2000 lookups per second, in the worst case. This value is easily achievable with current database technology. For instance, we have implemented an NRS node with a Bind9 server, running on an old Linux laptop with an Intel Pentium Processor M at 1.4 Ghz, and we measured a sustainable rate of about 15 000 lookups per second.

We also evaluated the number of active-routes versus time for the Equinix-sanjose-dirA trace (Fig. 6). The number of active-routes has a limited variation around its average value. This simplifies the dimensioning of the FIB size, which can be set close to the observed mean, without requiring a large margin.

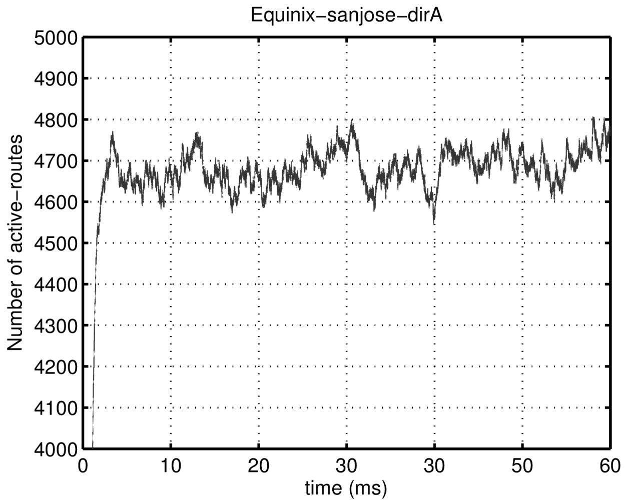


Fig. 6. Number of active-routes for the Equinix-sanjose-dirA trace

Finally, we investigated the effectiveness of FIB *over-provisioning*, to reduce the lookup rate. A FIB is said to be over-provisioned, when it has a capacity significantly greater than then average number of expected active-routes. For this analysis we used an ideal route replacement policy that randomly replaces inactive-routes. Fig. 7 shows the resulting lookup rate vs. the FIB size for the Equinix-sanjose-dirA trace; we observe a significant reduction of the lookup rate as the FIB size increases.

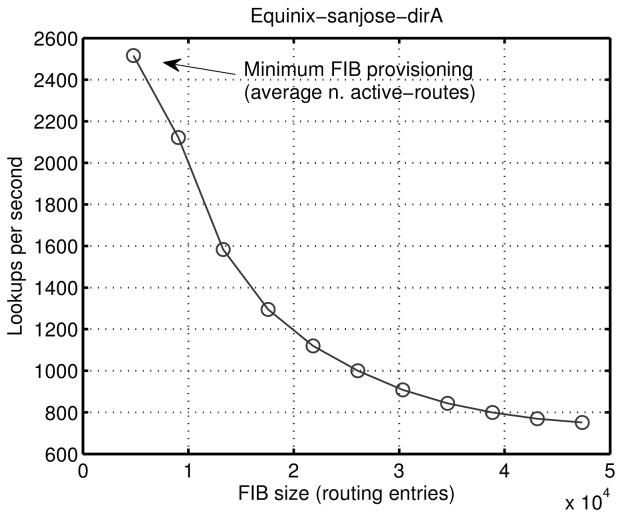


Fig. 7. Lookup rate versus FIB size for the Equinix-sanjose-dirA trace

6 Experiments on OneLab

In this section we show the functionality of Lookup and Cache architecture and evaluate its main performance by using the OneLab test-bed facility [18]. Specifically, we use 20 devices, located in different countries and belonging to the PlanetLab Europe network [18]. We implemented our Lookup-and-Cache architecture with a software package, mainly composed of a modified version of CCNx 0.5.0 [38] and a Java-based implementation of the NRS node. All the software is available in [39]. For the FIB replacement algorithm, we used LRU [43].

We analyzed the case of an ICN network formed by 19 ICN nodes and by a single centralized NRS node. The network topology is shown in Fig. 8, where each ICN node is marked with the country code of the supporting PlanetLab device. The NRS is located in Ireland. The connectivity graph of the network resembles a subset of the Pan-European GEANT research network [40].

As shown in the figure for the IE node, we assume that each ICN node serves a subsystem, containing ICN clients and ICN servers. Furthermore, each ICN node is connected with its neighbors by means of an overlay UDP/IP link. We setup this overlay network by properly configuring the next hop of the ICN routing tables. For instance, the ICN routing table of the IE node has the UK node as next-hop for any content, with the exclusion of contents published by the ICN server handled by the IE device.

This scenario may represent, for instance, the case of a single Autonomous System that uses ICN technology to exchange contents located in internal servers and, to this aim, deploys 19 ICN nodes/sub-systems, whose routing-by-name function is controlled by a centralized NRS device. To simplify the test-bed, we virtualize all the ICN servers and ICN clients contained in an ICN sub-system by using only one client and only one server, both contained in the PlanetLab device that hosts the ICN node of the sub-system. Therefore, each PlanetLab device of Fig. 8 (excluding the NRS) has the role of ICN client, ICN server and ICN node.

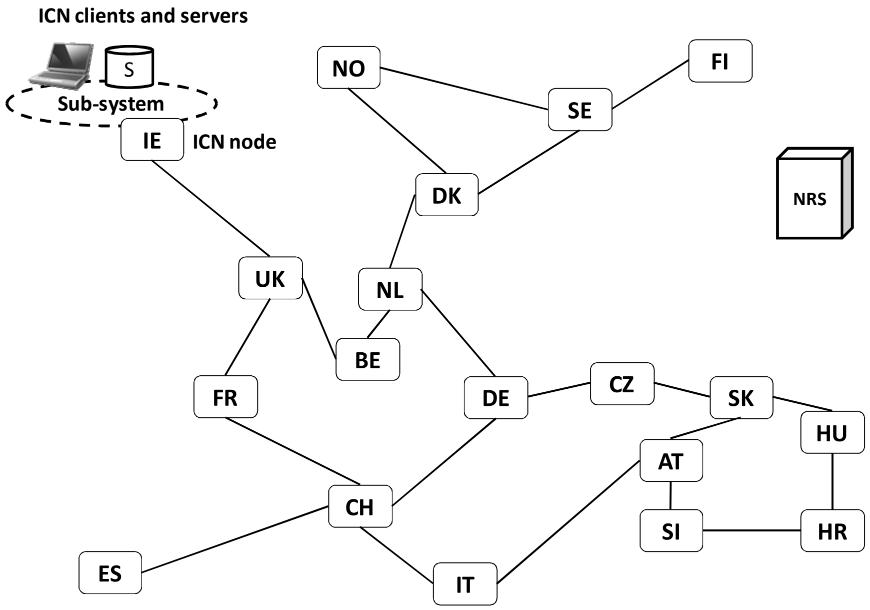


Fig. 8. ICN topology implemented on PlanetLab

Each ICN server handles 20 unique domain-names and, for each domain name, it publishes 5 contents of 500kB. For instance, a server that handles the domain-name www.cnn.com, publishes the contents www.cnn.com/text1.txt, www.cnn.com/text2.txt, ... , www.cnn.com/text5.txt. Therefore, in the whole network we have 380 domain-names, i.e. ICN routes, and 1900 contents (231800 chunks), uniformly distributed among network nodes. Each client generates 300 requests of contents with an inter-arrival time that follows a negative exponential distribution, with average 4s. To select a content, a client first chooses the domain-name according to a Zipf distribution with alpha=1 [34], then it randomly singles out one of the 5 contents associated to the selected domain-name.

The NRS node contains the ICN routes of all 380 domain-names, for all 19 ICN nodes. To compute the ICN routes, the NRS node uses a shortest path routing on the topology depicted in Fig. 8. For instance, a content request issued by the ES node for a content stored in the UK node is routed-by-name on the path: ES-CH-FR-UK.

ICN nodes do not use default routes, even though this could be possible in case of leaf nodes, for instance the IE one. Therefore, each time that a node has to forward an Interest message, if it does not have the related route in its FIB, it has to lookup-and-cache that route, by querying the NRS node.

The queries to the NRS node use a direct UDP/IP connection (not reported in the figure) between the ICN node and the NRS node. Therefore, the signaling traffic between ICN nodes and NRS is not routed-by-name on the ICN topology of Fig. 8, but it is transferred by using underlying, traditional, IP means. As a future work, we will support also NRS queries with ICN means, as proposed in [10].

To conclude the description of the test-bed, we remind that, in addition to the FIB memory that we use as a *cache of routes*, an ICN node has a storage space used as a *cache of content-chunks* (i.e., a cache of network layer data units, CIUs) to implement the in-network caching functionality discussed in the introduction (second advantage, in-network caching). In our ICN nodes, we use the default CCNx content cache replacement algorithm, i.e. FIFO.

Fig. 9 shows the average download time versus the FIB size, comparing the case of nodes without content cache and the case of nodes with a content cache; the content cache size is equal to 10% of the total number of published chunks. The x-axis includes also an out-of-scale point, representative of a full preloaded FIB (labeled "Full-FIB") where, for each node, we use an *unlimited* FIB, pre-loaded with all ICN routes that the node could use. This measurement allows highlighting the worsening of performance deriving from the use of a *limited* FIB as a cache of routes and from the use of a centralized remote RIB.

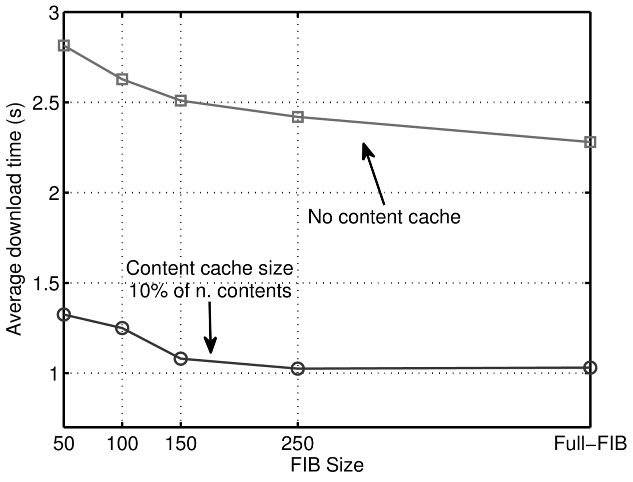


Fig. 9. Average download time versus FIB size

As expected, as the FIB size increase, the performance tends to the full-FIB case, while caching contents leads to a decrease of the download time as some chunks are delivered by the cache of nearby nodes, rather than from far away servers.

If we look at the curve representing the no content cache case, the download time decreases of about 600 ms, when the FIB size increases from 50 to the full-FIB case. We argue that this delay is due to the connectivity/processing delay brought about by the NRS node. This lookup delay (in the worst case equal to about 350 ms) would not occur if the traffic from/to the NRS had priority on the other user traffic and if the NRS were implemented by using a suitable powerful hardware.

Fig. 10 shows the number of lookups per second, measured at the NRS node, versus the FIB size. As measured in real Internet traces (see Fig. 7), also in the OneLab test-bed we obtain a significant reduction of the lookup rate by increasing the FIB size.

However, we have been surprised to see that the in-network caching of contents has a small impact on the lookup rate. We expected that the reduction of the average path length brought about by in-network content would have lowered the number of nodes involved in transferring a content, and the number of NRS lookups.

Thus, we analyzed the ICN network traffic, and found out that a content caching strategy based on chunks, like the one we (and CCNx) are using, may reduce the potential benefit of in-network content caching on the lookup rate.

In fact, even though the probability of finding a single chunk in the cache may be high, the probability of finding all the chunks of a given content (122 in our workload) in the cache is rather small. If only a single chunk of a complete content is not found in a cache of a node, that node will require to forward the Interest message and this may produce an NRS lookup. The same situation may occur if no chunk is stored in the content cache of the node; when the first Interest message is received, the node may perform an NRS lookup and cache the ICN route in the FIB. The FIB will then be used to forward other Interest for other chunks of that content. Thus, in both cases of single-chunk-cache-miss and no-chunk-in-the-cache a single lookup may be executed. This explains the low impact of content caching on the lookup rate.

This result suggests a future work consisting in analyzing in-network content caching mechanisms that cache the whole content, rather than chunks of it.

Fig. 11 shows the amount of total traffic exchanged by ICN nodes during the test, versus the size of the content cache size of the nodes, in case of a FIB size equal to 100. As already stated in [24], we find out that the increase of the content cache size yields a decrease of the network traffic that eventually follows a logarithmic decay. This logarithmic behavior implies that the en-route caching technique has a benefit-to-cost ratio that is good for small caches, whose lookup table can be implemented with off-the-shelf hardware. Conversely, if we want to deploy very big caches, not only are they very expensive, but the performance improvement is relatively small. A way to further reduce traffic is to complement en-route caching with pre-fetching techniques, à la CDN.

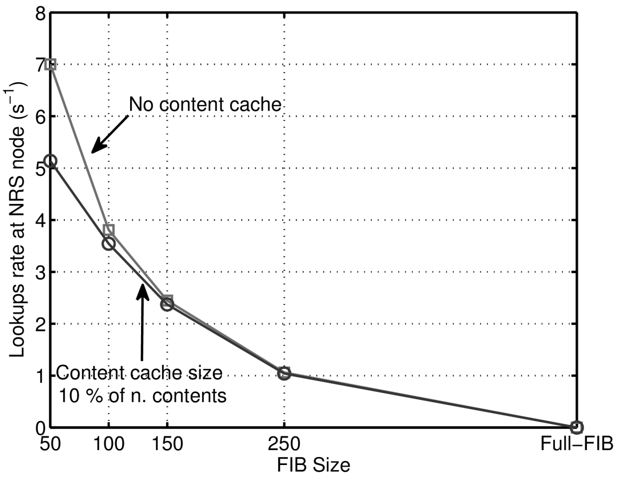


Fig. 10. Lookup rate measured at the NRS node versus FIB size

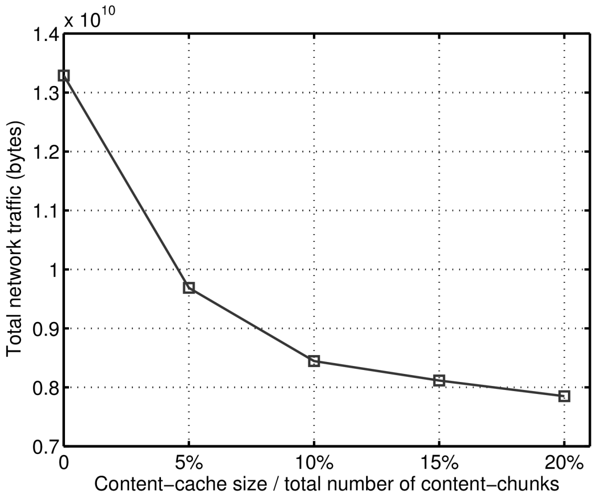


Fig. 11. Total network traffic versus content-cache size, in case of a FIB size equal to 100

7 Conclusions

The Information Centric Networking paradigm poses several technical challenges. Among them, an important one concerns the scalability of its, distinctive, routing-by-name functionality. To evaluate the scalability of routing we must consider two different issues. The first one concerns the size of the routing tables; the second one concerns the rate of routing message updates [45].

In this chapter we presented a proposal for a Lookup-and-cache architecture, which copes with the first scalability issue. The design of an ICN routing protocol that limits the rate of routing messages remains an "orthogonal", open issue, which we leave to further studies. Indeed, our Lookup-and-Cache architecture does not impose the use of a specific ICN routing protocol to exchange routing entries among NRS nodes. For instance name-based versions of BGP [41] or OSPF [42] could be viable candidates.

We used simulations and experiments over OneLab: i) to show that our Lookup and Cache architecture is feasible with current memory technology, and ii) to evaluate its performance. Our findings are that: i) it is necessary to carefully dimension the path from ICN nodes to the NRS, otherwise the lookup delay can become significant. However, this delay has to be compared with the current DNS resolution delay, as far as user perceptions are concerned, so it does not appear that this is a very limiting factor; ii) in-network *content-caching* based on chunks does not seem very useful in reducing the lookup rate; it could be worthwhile to analyze mechanisms that cache the whole content rather than chunks of content; iii) the rate of improvement of en-route *content caching* as a function of the content cache size is not very steep; this suggests to explore also other caching strategies, à la CDN.

Finally, it is worth to note that the Lookup and Cache architecture is very much in agreement with the so-called Software Defined Networking (SDN) paradigm. In

SDN, the network control plane is implemented in a dedicated device, which remotely controls packet switches providing data plane functionality. Indeed, we are implementing the Lookup and Cache architecture in the OpenFlow framework [46][47], which is a popular implementation of the SDN concept.

References

1. Cheriton, D., Gritter, M.: TRIAD: a scalable deployable NAT-based internet architecture. Technical Report (2000)
2. 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. of ACM SIGCOMM 2007, Kyoto, Japan, August 27-31 (2007)
3. Jacobson, V., et al.: Networking named content. In: Proc. of ACM CoNEXT 2009, Rome, Italy, December 1-4 (2009)
4. Smetters, D., Jacobson, V.: Securing Network Content. PARC technical report (October 2009)
5. Trossen, D., Sarela, M., Sollins, K.: Arguments for an information-centric internetworking architecture. SIGCOMM Computer Communication Review 40, 26–33 (2010)
6. Oueslati, S., Roberts, J., Sbihi, N.: Ideas on Traffic Management in CCN, Information-Centric Networking. Dagstuhl Seminar
7. Detti, A., Blefari Melazzi, N., Salsano, S., Pomposini, M.: CONET: A Content Centric Inter-Networking Architecture. In: ACM SIGCOMM Workshop on Information-Centric Networking (ICN 2011), Toronto, Canada, August 19 (2011)
8. Detti, A., Salsano, S., Blefari Melazzi, N.: IPv4 and IPv6 Options to support Information Centric Networking. Internet Draft, draft-detti-conet-ip-option-02, Work in progress (October 2011)
9. Salsano, S., Detti, A., Cancellieri, M., Pomposini, M., Blefari Melazzi, N.: Transport-layer issues in Information Centric Networks. In: ACM SIGCOMM Workshop on Information-Centric Networking (ICN 2012), Helsinki, Finland, August 17 (2012)
10. Blefari Melazzi, N., Detti, A., Pomposini, M., Salsano, S.: Route discovery and caching: a way to improve the scalability of Information-Centric Networking. In: IEEE Global Communications Conference 2012 (Globecom 2012), Anaheim, California, December 3-7 (2012)
11. SAIL project website, `http://www.sail-project.eu/`
12. PURSUIT project website, `http://www.fp7-pursuit.eu`
13. COMET project website, `http://www.comet-project.org/`
14. Named-Data Networking (NDN) project website, `http://named-data.org/`
15. COAST project website, `http://www.coast-fp7.eu/`
16. CONVERGENCE project website, `http://www.ict-convergence.eu`
17. OFELIA project website, `http://www.fp7-ofelia.eu/`
18. OneLab website, `http://www.onelab.eu/`
19. ISO/IEC 21000-2 – Information technology – Multimedia framework (MPEG-21) – Part 2: Digital Item Declaration
20. Chiariglione, L., Difino, A., Blefari Melazzi, N., Salsano, S., Detti, A., Tropea, G., Anadiotis, A.C.G., Mousas, A.S., Venieris, I.S., Patrikakis, C.Z.: Publish/Subscribe over Information Centric Networks: A Standardized Approach in CONVERGENCE. In: Future Network & Mobile Summit 2012, Berlin, Germany, July 4-6 (2012)

21. ISO/IEC 23006 – Information Technology – Multimedia Service Platform Technologies (MPEG-M)
22. Detti, A., Pomposini, M., Blefari Melazzi, N., Salsano, S.: Supporting the Web with an Information Centric Network that Routes by Name. Elsevier Computer Networks 56(17), 3705–3722
23. Kuzmanovic, A., Knightly, E.W.: Receiver-Centric Congestion Control with a Misbehaving Receiver: Vulnerabilities and End-point Solutions. Elsevier Computer Networks 51, 2717–2737 (2007)
24. Ghodsi, A., Koponen, T., Raghavan, B., Shenker, S., Singla, A., Wilcox, J.: Information-Centric Networking: Seeing the Forest for the Trees. In: Proc. of the 10th ACM Workshop on Hot Topics in Networks (HotNets-X), Cambridge, Massachusetts, November 14-15 (2011)
25. Zhao, X., Pacella, D.J., Schiller, J.: Routing Scalability: An Operator's View. IEEE Journal on Selected Areas in Communications 28(8) (October 2010)
26. Trotter, G.: Terminology for Forwarding Information Base (FIB) based Router Performance. IETF RFC 3222
27. Meyer, D., Zhang, L., Fall, K.: Report from the IAB Workshop on Routing and Addressing. IETF RFC 4984
28. Perino, D., Varvello, M.: A Reality Check for Content Centric Networking. In: ACM SIGCOMM Workshop on Information-Centric Networking (ICN 2011), Toronto, Canada, August 19 (2011)
29. BGP Routing Table Analysis Report, `http://bgp.potaroo.net`
30. Cisco Carrier Routing System, `http://www.cisco.com/en/US/prod/collateral/routers/ps5763/prod_brochure0900aecd800f8118.pdf`
31. Juniper T Series Core Routers, `http://www.juniper.net/elqNow/elqRedir.htm?ref=http://www.juniper.net/us/en/local/pdf/datasheets/1000051-en.pdf`
32. Breslau, L., et al.: Web Caching and zipf-like Distribution: Evidence and Implications. In: Proc. IEEE INFOCOM, New York, NY, USA, March 21-25 (1999)
33. Alexa Web Information Company, "Top 1,000,000 Sites", `http://s3.amazonaws.com/alexa-static/top-1m.csv.zip`
34. Jung, J., Sit, E., Balakrishnan, H., Morris, R.: DNS Performance and the Effectiveness of Caching. IEEE/ACM Transactions on Networking 10(5) (October 2002)
35. CAIDA Internet Trace Storage, `https://data.caida.org/datasets/passive-2010/`
36. MAWI Working Group Traffic Archive, `http://mawi.wide.ad.jp/mawi/samplepoint-F/2011/`
37. `http://netgroup.uniroma2.it/Andrea_Detti/Lookup-and-Cache/Feasibility-check/traces/`
38. CCNx project web site, `http://www.ccnx.org`
39. `http://netgroup.uniroma2.it/Andrea_Detti/Lookup-and-Cache-OneLab/lc_onelab.tar.gz`
40. GEANT Pan-European research network, `http://www.geant.net/`
41. Narayanan, A., Previdi, S., Field, B.: BGP advertisements for content URIs, INTERNET-DRAFT draft-narayanan-icnrg-bgp-uri-00 (July 2012)
42. Wang, L., Mahmudul Hoque, A.K.M., Yi, C., Alyyan, A., Zhang, B.: OSPFN: An OSPF Based Routing Protocol for Named Data Networking. NDN Technical Report NDN-0003 (July 2012)

43. Dan, A., Towsley, D.: An approximate analysis of the LRU and FIFO buffer replacement schemes. SIGMETRICS Perform. Eval. Rev. 18, 143–152 (1990)
44. Detti, A., Pomposini, M., Blefari Melazzi, N., Salsano, S., Bragagnini, A.: Offloading cellular networks with Information-Centric Networking: the case of video streaming. In: IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks 2012, WoWMoM 2012 (2012)
45. Elmokashfi, A., Kvalbein, A., Dovrolis, C.: On the scalability of BGP: the roles of topology growth and update rate-limiting. In: Proc. of ACM CoNEXT 2008, Madrid, Spain, December 9-12 (2008)
46. Blefari Melazzi, N., Detti, A., Morabito, G., Salsano, S., Veltri, L.: Supporting Information-Centric Functionality in Software Defined Networks. In: IEEE International Conference on Communications (ICC 2012), Ottawa, Canada, June 10-15 (2012)
47. McKeown, N., Anderson, T., Balakrishnan, H., Parulkar, G., Peterson, L., Rexford, J., Shenker, S., Turner, J.: OpenFlow: Enabling Innovation in Campus Networks. White paper (March 2008), `http://www.openflow.org`

Iterative Research Method Applied to the Design and Evaluation of a Dynamic Multicast Routing Scheme

Dimitri Papadimitriou[1], Florin Coras[2], Alberto Rodriguez[2], Valentin Carela[2], Davide Careglio[2], Lluís Fàbrega[3], Pere Vilà[3], and Piet Demeester[4]

[1] Alcatel-Lucent Bell Labs, Antwerp, Belgium
dimitri.papadimitriou@alcatel-lucent.com
[2] Universitat Politècnica de Catalunya, Barcelona, Spain
{fcoras,arnatal,vcarela,careglio}@ac.upc.edu
[3] Universitat de Girona, Girona, Spain
{lluis.fabrega,pere.vila}@udg.edu
[4] Ghent University and iMinds, Ghent, Belgium
piet.demeester@intec.ugent.be

Abstract. Following the iterative research cycle process, this chapter elaborates a methodology and documents the steps followed for the design of a dynamic multicast routing algorithm, referred to as Greedy Compact Multicast Routing. Starting from the design of the dynamic multicast routing algorithm, we then evaluate by simulation on large-scale topologies its performance and compare them with the Abraham compact multicast routing scheme and two other reference schemes, namely the Shortest Path Tree (SPT) and the Steiner Tree (ST) algorithm. Performance evaluation and comparison include i) the stretch of the multicast routing paths also referred to as multicast distribution tree (MDT), ii) the memory space required to store the resulting routing table entries, and iii) the total communication or messaging cost, i.e., the number of messages exchanged to build the MDT. However, such performance evaluation is a necessary but not a sufficient condition to meet in order to expect deployment of multicast routing. Indeed, if one can determine that traffic exchanges are spatially and temporally concentrated, this would provide elements indicating the relevance for the introduction of such scheme in the Internet. Otherwise (if traffic exchanges are spatially and temporally diverse, i.e., highly distributed), then very few of them would benefit from a (shared) point-to-multipoint routing paths and multicast routing scheme would be less useful. For this purpose, we have conducted a multicast tree inference study. In turn, data and results obtained from these studies provides more realistic scenarios for emulation experiments against the currently deployed approach combining MBGP and PIMdeployed in IPTV or mVPN context.

Keywords: multicast routing, compact, experimental, performance, evaluation.

1 Introduction

The Future Internet Research and Experimentation (FIRE) initiative aims to realize a "research environment for investigating and experimentally validating highly innovative and revolutionary ideas" towards new paradigms for the Internet by bridging multi-disciplinary long-term research and experimentally-driven large-scale validation. FIRE foundational objectives were:

- Creation of a multi-disciplinary, long term research environment for investigating and experimentally validating highly innovative and revolutionary ideas for new networking architectures and service paradigms;
- Promotion of experimentally-driven yet long-term research, joining the two ends of academy-driven visionary research and industry-driven testing and experimentation, in a truly multi-disciplinary and innovative approach;
- Realization of a large scale European experimental facility, by gradually interconnecting and federating existing and new "resource clusters" for emerging or future internet architectures and technologies.

These objectives further evolved toward the inception of experimentally-driven research as a visionary multi-disciplinary investigation activity, defining the challenges for and taking advantage of experimental facilities. Such investigation activity would be realized by means of iterative cycles of research, oriented towards the design and large-scale experimentation of new and innovative paradigms for the Internet - modeled as a complex distributed system. The refinement of the research directions should be strongly influenced by the data and observations obtained from experiments performed at previous iterations thus, being "measurement-based" which in turn requires the specification of the relevant criteria and metrics as well as their corresponding measurement tools. The rationale was thus clear: create a dynamic between elaboration, realization, and validation by means of *iterative cycles* of experimentation.

With the increasing of multimedia streaming/content traffic, multicast distribution process from a source to a set of destination nodes is (re-)gaining interest as a bandwidth saving technique competing with or complementing cached content distribution. Nevertheless, the scaling problems faced in the 90's when multicast routing received main attention from the research community remain mostly unaddressed since so far. Indeed, routing protocol dependent multicast routing schemes such as Distance Vector Multicast Routing Protocol (DVMRP) and Multicast Open Shortest Path First (MOSPF) have been replaced by routing protocol independent routing schemes such as Protocol Independent Multicast Sparse Mode (PIM-SM) [1] and Core Base Trees (CBT) [2]. During last decade, the Single Source Multicast (SSM) variant of PIM, referred to as PIM-SSM [3], has been deployed in the context of IPTV within Internet Service Provider's network (intra-domain multicast). However inter-domain multicast has failed to be widely adopted by most ISPs. The reasons, among others, result from the relative complexity of the protocol architecture. Overlaying multicast routing on top unicast routing topology suffers from the same scaling limits as unicast (shortest-path) routing with the addition of the level of indirection added by the multicast routing, the limits of Multicast Source Discovery Protocol (MSDP) which prevents shared trees between domains (thus, it defeats the objectives of PIM-SM) and its address space structure (multicast addressing also requires firewall upgrades to recognize Class-D addresses). On the other hand, deploying multicast routing requires routing equipment upgrade (both hardware and software) whereas the corresponding cost cannot be compensated by multicast service revenues when the ISP doesn't itself provide access to multicast receivers (or sources).Further analysis on deployment Issues for the IP multicast routing and architecture can be found in [4].

As part of the work conducted in the EULER FP7 project [5], we started by designing a dynamic multicast routing algorithm, referred to as Greedy Compact Multicast Routing (GCMR) [6]. This leaf-initiated routing scheme which runs independently of the unicast routing scheme (and does not share any routing state information) is specialized for the construction of multicast routing paths (or multicast distribution trees) from any source to any set of destination nodes (or leaf nodes). We have then evaluated the performance of the proposed GCMR scheme and compare them for the same topologies with the Abraham compact multicast routing scheme [7] and two other reference schemes, the Shortest Path Tree (SPT) and the Steiner Tree (ST) algorithm. The performance evaluation and comparison by simulation of these multicast routing algorithms include: i) the stretch of the multicast routing paths it produces, ii) the memory space required to store the resulting routing table entries, and iii) the total communication or messaging cost, i.e., the number of messages exchanged to build the entire Multicast Distribution Tree (MDT).

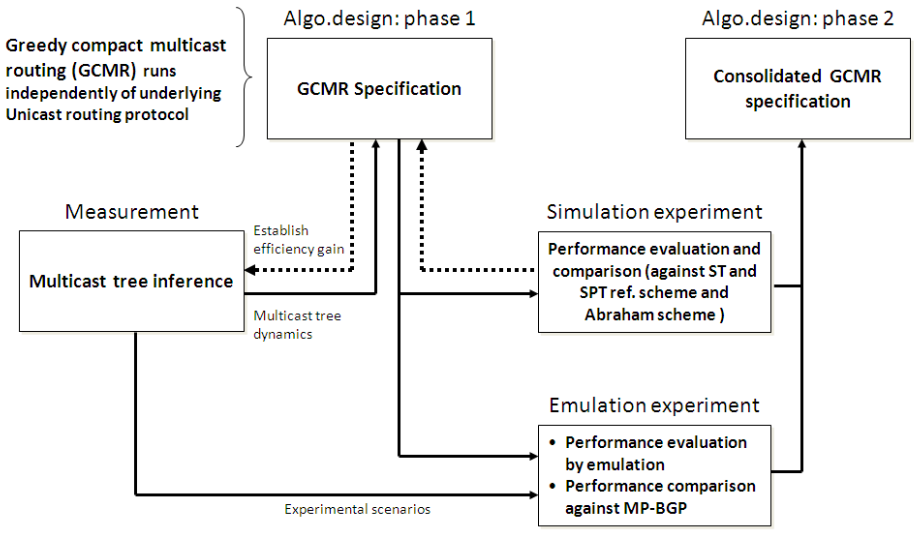

Greedy compact multicast routing (GCMR) runs independently of underlying Unicast routing protocol
Algo.design: phase 1
GCMR Specification
Algo.design: phase 2
Consolidated GCMR specification
Measurement
Multicast tree inference
Establish efficiency gain
Multicast tree dynamics
Simulation experiment
Performance evaluation and comparison (against ST and SPT ref. scheme and Abraham scheme)
Emulation experiment
• Performance evaluation by emulation
• Performance comparison against MP-BGP
Experimental scenarios

Fig. 1. Experimental methodology

However, such performance evaluation is a necessary but not a sufficient condition to meet in order to expect deployment of multicast routing. Indeed, by determining that traffic exchanges are spatially and temporally concentrated, one would increase the relevance for the introduction of such scheme in the Internet. Otherwise, if traffic exchanges are spatially and temporally diverse (highly distributed), then very few of them would benefit from a (shared) multicast routing paths and thus multicast routing scheme would be less useful. For this purpose, we have conducted a multicast tree inference study. In turn, the data and results obtained from these studies enable to design more realistic scenarios for the emulation experiments and the performance comparison against the currently deployed approach combining Multiprotocol BGP specified in RFC 4760 [8] also referred to as MBGP or MP-BGP (for discovery purposes) and PIM (for signaling purposes) deployed in context of IPTV or multicast VPN (mVPN). Figure 1 summarizes our methodology together with the various iterative research cycles until the step where a consolidated version of the GCMR scheme combining all experimental results can be specified.

2 Dynamic Multicast Routing

Dynamic multicast routing algorithms enable the construction of point-to-multipoint routing paths from any source to any set of destination nodes (or leaf nodes). The tree determined by a multicast routing path is commonly referred to as a Multicast Distribution Tree (MDT) as it enables the distribution of multicast traffic from any source to any set of leaf nodes. By means of such dynamic routing scheme, MDTs can dynamically evolve according to the arrival of leaf-initiated join/leave requests. The multicast routing algorithm creates and maintains the set of local routing states at each node part of the MDT. From this state, each nodes part of the MDT can derive the required entries to forward the multicast traffic received from a given source to its leaves.

2.1 Greedy Compact Multicast Routing (GCMR)

In [6], we introduce a dynamic compact multicast routing algorithm that enables the construction of multicast distribution trees (also referred to as multicast routing paths) for the distribution of multicast traffic from any source to any set of leaf nodes. The objective of the proposed GCMR scheme is to minimize the routing table sizes of each node part of the MDT at the expense of i) routing the packets on multicast routing paths with relative small deviation compared to the optimal stretch obtained by the Steiner Tree (ST) algorithm, and ii) higher communication cost compared to the Shortest Path Tree (SPT) algorithm. For this purpose, the GCMR algorithm reduces the local storage of routing information by keeping only direct neighbor-related entries rather than tree structures (as in ST) or network graph entries (as in both SPT and ST). The proposed algorithm relies on the information obtained locally and proportionally to the node degree instead of requiring knowledge of the global topology information (proportional to the network size) while still providing the least cost next hop during the MDT construction. In other terms, the GCMR algorithm requires only the maintenance of local topology information and does not rely on the knowledge of the global topology information or involve the construction of global network structures such as sparse covers. The information needed to reach a given multicast source is acquired by means of a two-stage search process that returns the upstream node along the least cost branching path to the MDT sourced at s. This process is triggered whenever a node decides to join a given multicast source s, root of the MDT. After a node becomes member of an MDT, a multicast routing entry is dynamically created and stored in the local Tree Information Base (TIB). From these routing table entries, multicast forwarding entries are locally instantiated.

More formally, consider a network topology modeled by an undirected graph $G = (V,E,c)$, where the set V, $|V| = n$, represents the finite set of nodes or vertices (all being multicast capable), the set E, $|E| = m$, represents the finite set of links or edges, and c a non-negative cost function $c: E \rightarrow Z+$ that associates a non-negative cost c to each link $(u,v) \in E$. For $u, v \in V$, let $c(u,v)$ denote the cost of the path $p(u,v)$ from u to v in G, where the cost of a path is defined as the sum of the costs along its edges. Let S, $S \subset V$, be the finite set of source nodes, and $s \in S$ denote a source node. Let D, $D \subseteq V \backslash \{S\}$, be the finite set of all possible destination nodes that can join a multicast source s, and $d \in D$ denote a destination (or leaf) node. A *multicast distribution tree* $T_{s,M}$ is defined as an acyclic connected sub-graph of G, i.e., a tree rooted at source $s \in S$ with leaf node set M, $M \subseteq D$. During the MDT construction, the routing information needed to reach a given multicast distribution tree is acquired by means of an incremental two-stage search process. This process, triggered whenever a node decides to join a given multicast source, starts with a local search covering the leaf node's neighborhood. If unsuccessful, the search is performed over the remaining unexplored topology (without requiring global knowledge of the current MDT). The returned information provides the upstream neighbor node along the least cost branching path to the MDT rooted at the selected multicast source node. The challenge consists thus in limiting the communication cost, i.e., the number of messages exchanged during the search phase, while keeping an optimal stretch - memory space tradeoff.

As stated before, the reduction in memory space consumed by the routing table entries results however in higher communication cost compared to the reference algorithms, namely the SPT and the ST. Higher cost may hinder the applicability of our algorithm to large-scale topologies such as the Internet. Hence, to keep the communication cost as low as possible, the algorithm's search process is segmented into two different stages. The rationale is to put tighter limits on the node space by searching locally in the neighborhood (or vicinity) of the joining leaf node before searching globally. Indeed, the likelihood of finding a node of the MDT within a few hops distance from the joining leaf is high in large topologies (whose diameter is logarithmically proportional to its number of nodes) and this likelihood increases with the size of the MDT. Hence, we segment the search process by executing first a local search covering the leaf node's vicinity ball, and, if unsuccessful, by performing a global search over the remaining topology. By limiting the size (or order) of the vicinity ball while taking into account the degree of the nodes it comprises, one ensures an optimal communication cost. For this purpose, a variable path budget π_β is used to limit the distance travelled by leaf initiated requests to prevent costly (in terms of communication) local search or global search. Additionally, as the most costly searches are resulting from the initial set of leaf nodes joining the multicast traffic source, each source constructs a domain (referred to as source ball). When a request reaches the boundary of that domain it is directly routed to the source.

2.2 Comparison with Existing IP Multicast Routing

Independence from the underlying unicast routing algorithm is the fundamental concept underlying multicast routing schemes such as Protocol Independent Multicast (PIM). Its variants for any-source multicast (PIM-SM) [1] and single-source multicast (PIM-SSM) [3] are the most commonly deployed routing protocols even if limited in scope to single carrier networks. It is important though to distinguish between algorithmic independence, i.e., no computational coupling from informational independence, i.e., PIM makes use of the unicast routing table to enable control message exchanges (join, prune, etc.). Indeed, overlaying multicast routing on top of unicast suffers however from the same scaling limitations as current unicast routing with the addition of the level of indirection added by the multicast routing application. Multicast routing protocol enables routers to build a (logical) delivery tree between the sender(s) and receivers of a multicast group. Multicast routing table includes the Multicast Routing Information Base (MRIB) and the Tree Information Base (TIB). The MRIB is the topology table, typically derived from the unicast routing table, which carries multicast-specific topology information. The TIB is the collection of routing state created from the exchange of join/prune messages. This table stores the state of all multicast distribution trees at that node. The implication being that in case of topology change, unicast routing states have to re-converge to a new stable state before multicast routing states can themselves re-converge.

Moreover, we also observe that the scaling problems experienced by these routing protocols and more generally all multicast routing approaches developed by the research community, remain largely unaddressed since so far. Indeed, multicast currently operates as an addressable IP overlay (Class D group addresses) on top of unicast routing topology, leaving up to an order of 100millions of multicast routing table entries. Hence, the need to enable multicast routing paths (for bandwidth saving purposes) while keeping multicast addressing at the edges of the network and building shared but selective trees inside the network. When used in combination to GCMR, multicast forwarding relies on local port information only. Thus, the memory capacity savings comes from i) keeping 1:N relationship between network edge node and the number of multicast groups (N), and ii) local port-based addressing for the local processing of multicast traffic. Further, we argue that the GCMR scheme, by providing the best memory-space vs. stretch tradeoff, can possibly address the memory scaling challenges without requiring the deployment of an underlying unicast routing scheme.

The version 4 of the Border Gateway Protocol (BGPv4) has also been extended to support multicast discovery protocol. This extension relies on the multiprotocol BGP (MBGP) feature defined in RFC 4760 [8]. The multi-protocol capability of BGP enables multicast routing and the connection of multicast topologies within and between BGP autonomous systems. In other words, multiprotocol BGP (MBGP) is an enhanced BGP that carries IP multicast routes. BGP carries two sets of routes, one set for unicast routing and one set for multicast routing. The routes associated with multicast routing are used by the Protocol Independent Multicast (PIM) to build data distribution trees. More recently, this feature has been further extended in RFC 6513 [9] and BGPv4 can now also be used as multicast signaling protocol; hence, avoiding the use of PIM.

2.3 Comparison with Other Compact Multicast Routing

As far as our knowledge goes, prior work on compact multicast routing is, mainly concentrated around the routing schemes developed in the seminal paper of Abraham [7]. One of the reasons we can advocate is that despite the amount of research work dedicated to compact unicast routing, current schemes are not yet able to efficiently cope with the dynamics of large scale networks which is the prime characteristic of dynamic multicast routing schemes.

More formally, the Abraham scheme relies on the off-line construction of a bundle $\mathcal{B}_k = \{TC_{k,2^i}(G) \mid i \in I\}$ of sparse tree covers of the graph G, $TC_{k,2^i}(G)$, where k = log(n). Sparse covers are grown from a set of center nodes $c(T_i(v))$ located at distance at most $k.2^i$ from node $v \in V$. By $T_i(v)$, we denote the tree in the sparse tree cover $TC_{k,2^i}(G)$ that contains the ball $B(v,2^i)$. For each $i \in I$ and $T \in TC_{k,2^i}(G)$, the center node c(T(v)) stores the labels of all nodes[1]contained in the ball $B(v,2^i)$, i.e., the ball centered on node v of radius 2^i. Further, the SPlabel(v) stores the label $\lambda(T,c(T))$ for each $T \in \mathcal{B}(v)$, defined as set of all tree covers T in the bundle $\mathcal{B}_k$ such that $v \in T$. In addition, each node $v \in V$ stores the tree routing information $\mu(T,v)$ for all the trees in its own label SPlabel(v). When a leaf node u joins an MDT, it first determines whether or not one of the MDT nodes is already included in its local tree routing information table. If this is the case, node u sends the join request to the center node $c(T_i(v))$ of the tree $T_i(v)$ that is covered by the MDT at node v. The center node $c(T_i(v))$ then passes the label $\mu(T_i(v),u)$ so that the selected MDT node v can forward the multicast traffic to the newly joining leaf node u. Otherwise, the leaf node u has to obtain via its center node the set of MDT nodes the tree currently includes. Among all index $i \in I$, node u then selects the tree $T_{i^*}(v)$, $v \in$ MDT, whose intersection with its bundle $\mathcal{B}(u)$ is minimum. Once the node, say v, part of this intersection is selected ($T_{i^*}(v) \in \mathcal{B}(u)$), leaf node u directs the join request to the associated center node $c(T_{i^*}(v))$. The latter passes a label $\mu(T_{i^*}(v),u)$ so that the selected MDT node v can forward the incoming multicast traffic to the newly joining leaf node u. In both cases, in order for the source node s to reach node u, node v has to propagate the tuple $[v,c(T_{i^*}(v)),\mu(T_{i^*}(v),u)]$ to source s. Finally, the leaf node u updates all nodes covered by its balls $B(u,2^i)$ to allow them joining the MDT at node u.

Compared to the Abraham compact multicast routing scheme [7], the GCMR name-independent compact multicast routing algorithm is i) leaf-initiated since join requests are initiated by the leaf nodes; however, contrary to the Abraham scheme it operates without requiring prior local dissemination of the node set already part of the MDT or keeping specialized nodes informed about nodes that have joined the MDT, and ii) dynamic since requests are processed on-line as they arrive without re-computing and/or re-building the MDT. Moreover, our proposed algorithm is iii) distributed since transit nodes process homogeneously the incoming requests to derive the least cost branching path to the MDT without requiring any centralized processing by the root of the MDT or any specialized processing by means of pre-determined center nodes, and iv) independent of any underlying sparse cover construction grown from a set of center nodes (which induce node specialization driving the routing functionality): the local

[1] For simplicity, we present here the label-dependent variant of the scheme. In the name-independence version, center nodes store label mappings from names to nodes.

knowledge of the cost to direct neighbor nodes is sufficient for the proposed algorithm to properly operate. It is important to emphasize that the sparse cover underlying the execution of the Abraham scheme is constructed off-line and requires global knowledge of the network topology to properly operate.Moreover, this routing scheme is oblivious, i.e., the path from the source to a given leaf is irrespective of the current set of leaves (even if its iterative construction implies the "local" dissemination of information related to nodes that have already joined the tree) whereas the GCMR scheme is adaptive. The resulting adaptation cost remains to be characterized.

3 Performance Analysis

The performance of the GCMR algorithm, further documented in [6] are evaluated in terms of the stretch of the multicast routing paths it produces, the size and the number of routing table entries, and the communication/messaging cost. Performances are evaluated by simulation on synthetic power-law graphs (generated by means of the Generalized Linear Preference (GLP) model [10]) and the CAIDA map of the Internet topology both comprising 32k nodes. Performance results are compared to the multicast routing algorithms (the Shortest Path Tree and the Steiner Tree algorithm) and the Abraham scheme.

3.1 Reference Routing Schemes: ST and SPT

The execution scenario considers the construction of point-to-multipoint routing paths for leaf set of increasing size from 500 to 4000 nodes (selected randomly) with increment of 500 nodes. Each execution is performed 10 times by considering 10 different multicast sources. We compare the performance of the GCMR algorithm to the Shortest-Path Tree (SPT) which provides the reference for the communication cost and the Steiner Tree (ST) algorithms which provides the reference in terms of stretch. In order to obtain the near optimal solution for the ST, we consider a ST-Integer Linear Programming formulation adapted from [11] for bi-directional graphs.

The communication cost for the ST measures at each step of the MDT construction the number of messages initiated by nodes part of the MDT. Hence, although the ST is computed centrally, the communication cost accounts for the total number of messages exchanged during the MDT building process as a dynamic scenario would perform.

Multiplicative Stretch: defined as the cost ratio between the point-to-multipoint routing paths (underlying the MDT) produced by the proposed scheme and the minimum Steiner Tree. We also compare the cost ratio between the point-to-multipoint routing path produced by the SPT and the minimum Steiner Tree (ST).

- *GLP Topology*: as shown in Figure 2a, the multiplicative stretch for the proposed algorithm is slightly higher than 1 for the GLP topology. As the leaf set increases from 500 to 4000 nodes, its trend curve decreases from 1.09 (maximum value reached for 500 leaf nodes) to 1.05 (minimum value reached for 4000 leaf nodes). Compared to the SPT stretch, our algorithm maintains an average gain of 4% along the different group sizes.

- *CAIDA Map*: as shown in Figure 2b, the multiplicative stretch for the proposed algorithm is slightly higher than 1 for the CAIDA topology. As the leaf set increases from 500 to 4000 nodes, its trend curve decreases from 1.08(maximum value reached for 500 leaf nodes) to 1.03 (minimum value reached for 4000 leaf nodes). Compared to the SPT stretch, our algorithm maintains a maximum deterioration of 4% for sets of 500 leaf nodes; this deterioration becomes negligible as the size of the leaf node sets increases.

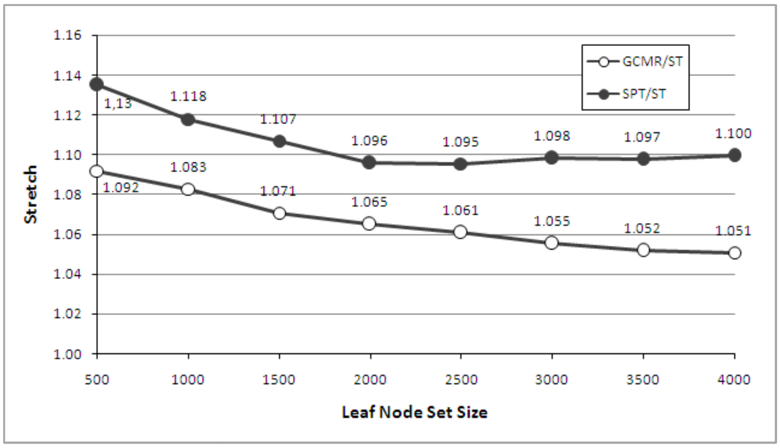


Fig. 2a. Stretch as a function of Leaf Node Set Size

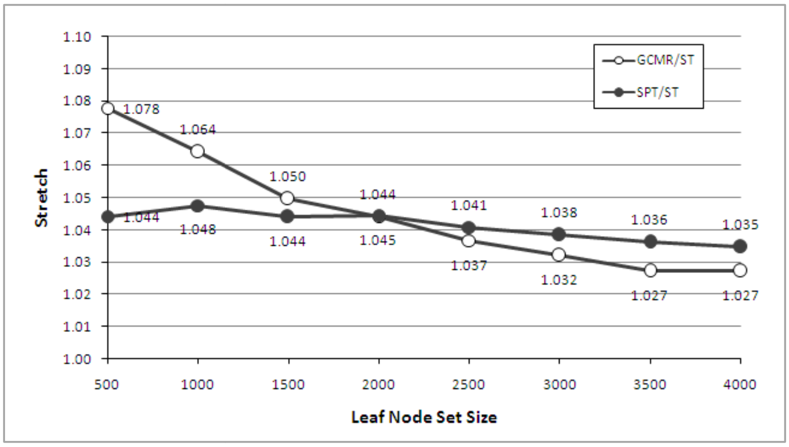


Fig. 2b. Stretch as a function of Leaf Node Set Size

Storage: is concerned with the memory space required to store the routing tables entries (underlying the MDT) and the relative gain we obtain in terms of the ratio between the total number of RT entries produced by our algorithm and the total number of RT produced by the reference algorithms. This ratio provides an indication of the achievable reduction in terms of the memory capacity required to store the routing table entries produced by these algorithms.

- *GLP Topology*: the GCMR scheme produces significantly less RT entries that the ST and SPT reference algorithms. The highest number of RT entries is obtained for a set of 4000 leaf nodes: 10154 RT entries. This value is 4,10 times smaller than the number of RT entries produced by the ST algorithm (41643 RT entries) and 27,92 times smaller than the number of the RT entries produced by the SPT algorithm (283477 RT entries). Figure 3a illustrates the relative gain expressed in terms of the ratio between the total number of RT entries produced by the ST and the SPT references and our algorithm. An increasing gain can be observed as the size of the leaf node set decreases from 4,10 (leaf set of 4000 nodes) to 20,34 (leaf set of 500 nodes) compared to the ST algorithm and from 27,92 (leaf set of 4000 nodes) to 166,08 (leaf set of 500 nodes) compared to the SPT algorithm.

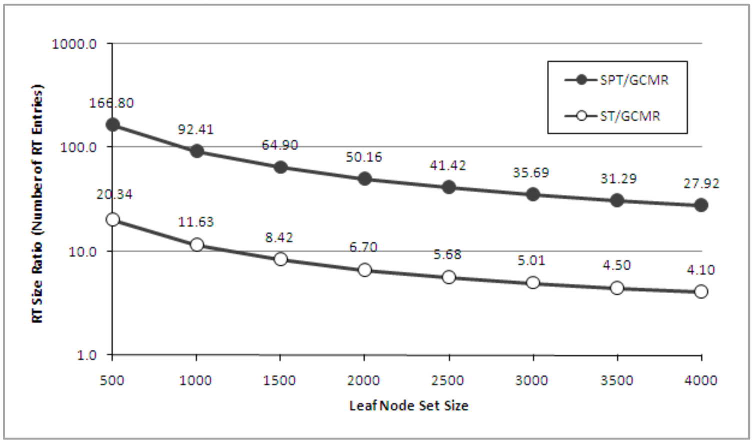


Fig. 3a. RT Size Ratio as a function of Leaf Node Set Size

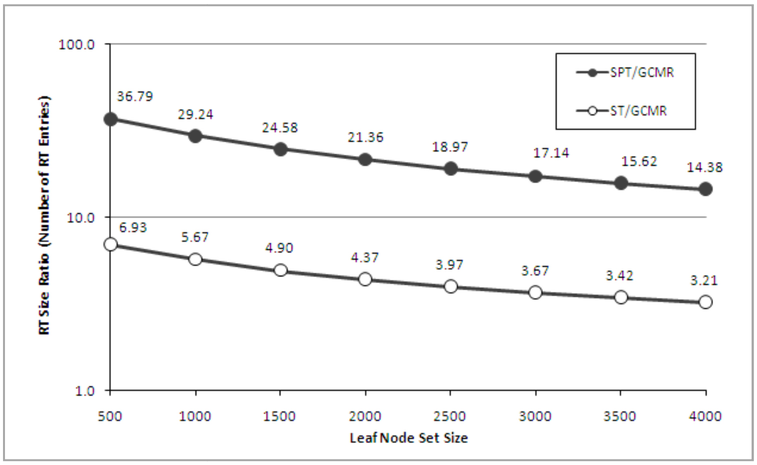


Fig. 3b. RT Size Ratio as a function of Leaf Node Set Size

- *CAIDA Map*: the GCMR scheme produces significantly less routing table entries that the ST and SPT reference algorithms. The highest number is obtained for leaf sets of 4000 nodes: 13169 RT entries. This value is 3,21 times smaller than the number of RT entries produced by the ST algorithm (42277 RT entries) and 14,38 times smaller than the number of RT entries produced by the SPT algorithm (189431 RT entries). Figure 3b illustrates the relative gain in terms of the ratio between the total number of RT entries produced by the ST and SPT references and GCMR. An increasing gain can be observed as the size of thel eaf set decreases from 3,21 (leaf set of 4000 nodes) to 6,93 (leaf set of 500 nodes) compared to the ST algorithm and from 14,38 (leaf set of 4000 nodes) to 36,79 (leaf set of 500 nodes) compared to the SPT algorithm. Interestingly, the obtained gain values for the CAIDA map are smaller than those obtained for the GLP topology. This difference can be explained resulting from the difference in tree-depth: 6 (leaf set of 500 nodes) to 9 (leaf set of 4000 nodes) for the CAIDA map vs. 8 (leaf set of 500 nodes) to 11 (leaf set of 4000 nodes) for the GLP topology.

Communication Cost: defined as number of messages exchanged during the discovery search phase

- *GLP Topology*: as depicted in Figure4a, the communication cost ratio for the proposed algorithm is relatively high compared to the SPT even if much lower than the communication cost implied by the ST (not represented in this figure). Indeed, the communication cost ratio increases from 2,69 (leaf set of 500 nodes) to 8,17 (leaf set of 4000 nodes). This observation can be explained by the presence of high degree nodes (nodes that have a degree of the order to 100 or even higher) in power law graphs. However, as computed this communication cost does not take into account the evolution of the routing topology. This evolution impacts multicast routing algorithms such as the SPT that are strongly dependent on non-local unicast routing information compared to the proposed algorithm. Moreover, as shown in Figure 5, the communication cost of the proposed algorithm compared to the SPT communication cost, decreases as the number of nodes composing the leaf node set increases. This trend leads us to expect that a saturation level can be reached around a communication cost ratio not higher than 10 to 15 as the size of the lead node set continues to grow. It is worth mentioning that the memory and the capacity required to process communication messages are relatively limited.
- *CAIDA Map*: the same trend can be observed for the CAIDA Map (see Figure 4b) where the communication cost ratio between our scheme and the SPT algorithm increases from 7,88 (leaf set of 500 nodes) to 13,77 (leaf set of 4000 nodes). The difference observed between the CAIDA map and the GLP topology can be explained from the following observation the tree-depth differs by a unit (3 vs. 4). This difference induces a relatively higher cost of the SPT when running over the GLP topology.

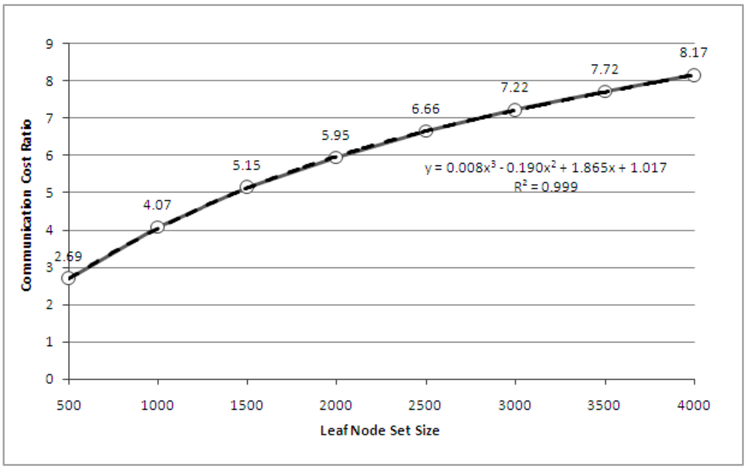


Fig. 4a. Communication Cost Ratio as a function of Leaf Node Set Size

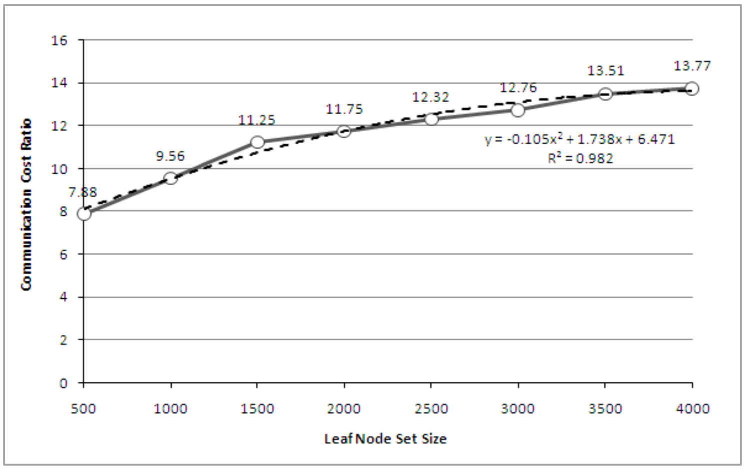


Fig. 4b. Communication Cost Ratio as a function of Leaf Node Set Size

3.2 Abraham Scheme

We further compare the performance in terms of the stretch of the point-to-multipoint routing paths and the memory space required by the GCMR algorithm and by the Abraham routing scheme for dynamic join only events.

Stretch: for the scheme allowing only dynamic join events, the MDT cost is given by Lemma7 of [7]. The authors determine that the proposed dynamic multicast algorithm is O(min{log n, log Δ}.log n) competitive compared to the cost of the optimal algorithm – Steiner Tree. In this formula, the factor Δ is the aspect ratio defined as the ratio between the maximum and the minimum distance d(u,v), for any node pair u,v∈V. Considering an aspect ratio Δ of 6 and a network of 32k nodes the stretch upper bound is about 3.5. Thus the stretch upper bound of the point-to-multipoint routing path produced by the Abraham scheme, even if universal (applicable to any graph), is in the best case more than 3 times higher than the one produced by our scheme. Simulation result show (see Figure 5) that this upper bound is not reached though the GCMR stretch is still twice better than the one obtained with the Abraham scheme. Note also that the comparative gain is weakly influenced by the value k (which determines the sparse cover construction, the higher the value the less the number of trees in the tree cover).

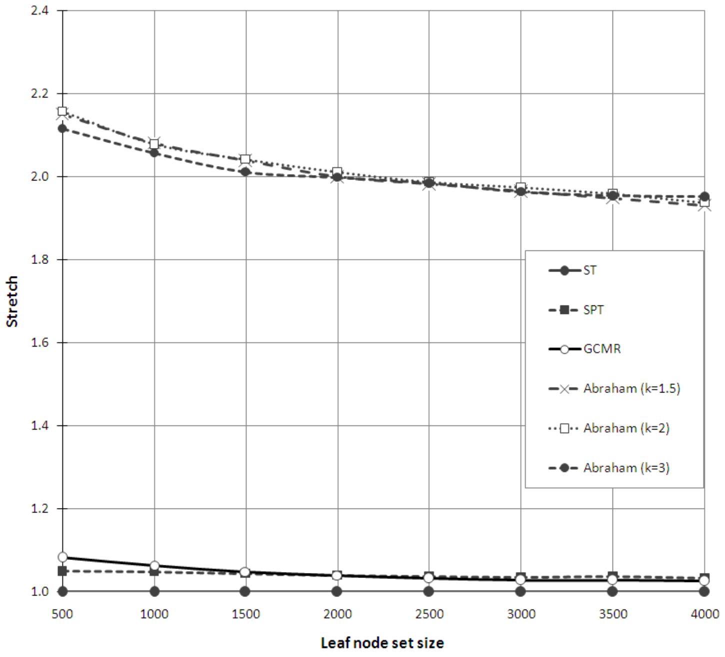


Fig. 5. Stretch as a function of Leaf Node Set Size

Storage: the memory/storage requirement of the Abraham scheme includes i) the tree routing information μ(T,v) stored by each node v, for all trees in its own SPLabel(v) leading to a total storage of $O(\log^3 n.\log\Delta/\log\log n)$ bits, ii) for each $i \in I$ and $T \in TC_{k,2i}(G)$, the center node c(T(v)) of each node $v \in T$ that stores the labels of all nodes contained in the ball $B(v,2^i)$ leading to a total storage over all radii of $O(kn^{1+1/k} \log \Delta)$ bits; in addition, each node v stores $O(\log \Delta)$ labels of size $\tilde{O}(kn^{1/k})$ each leading to a total memory consumption of $\tilde{O}(kn^{1+1/k})$ bits.

To simplify comparison the GCMR memory space complexity is proportional to the number of tree nodes (2τ) and the Abraham scheme proportional to the number of nodes in network times the construction parameter k (k.n). This yields for k = 1 a ratio of 32 (4) for a leaf set of 500 (4000) nodes, for k = 2 a ratio of 64 (8) for a leaf set of 500 (4000) nodes, and for k = 3, a ratio of 96 (16) for a leaf set of 500 (4000). These numbers and trends remain to be further confirmed and validated.

4 Effective Gain Analysis

In this section, we aim at determining the rationale for introducing a multicast routing scheme as part of the Internet-wide routing system. This rationale is based on the following premises if traffic exchanges are spatially and temporally diverse, very few of them would benefit from a (shared) point-to-multipoint routing paths and thus multicast routing scheme would be useless; otherwise (if traffic exchanges are spatially and temporally concentrated), this would indicate the relevance for the introduction of such scheme in the Internet.

In the following we present the method adopted to classify applications from data traffic captures. We further analyze the benefits that a potential multicast routing scheme can provide. Finally, we analyze the traffic statistical characteristics of a streaming video application when transmitting a popular sport event.

4.1 Traffic Classification

Traffic classification is a difficult problem that requires the use of very complex identification techniques due to the variable nature of Internet traffic and applications. Traditionally, the port numbers were used to classify the network traffic (e.g., well-known ports registered by the IANA). Nevertheless, nowadays it is widely accepted that this method is no longer valid due to its inaccuracy and incompleteness of its classification results. The first alternatives to the well-known ports method relied on the inspection of the packet payloads in order to classify the network traffic [12] [13] [14]. These methods consist of looking for characteristic signatures (or patterns) in the packet payloads. Although this solution could potentially be very accurate, its high resource requirements and limitations with encrypted traffic make its use unfeasible in current high-speed networks.

Instead, we developed therefore a traffic classification tool using NetFlow [15] data instead of packet-level traces. We use the well-known C4.5 decision tree technique [16] in order to analyze the impact of traffic sampling on the classification accuracy with Sampled NetFlow [17] [18]. To reduce the impact of traffic sampling on the classification accuracy, the tool implements an automatic Machine Learning (ML) algorithm that does not rely on any human intervention. More details can be found in [12].This tool has been applied to the traffic captured in the connection between the Anella Científica (Catalan NREN [13]) to RedIRIS (Spanish NREN [14]) and to the global Internet. The point of measurement is a 10Gbit Ethernet bi-directional link, which provides Internet connection to 60 different public entities and 50,000 users.

Figure 6 shows the obtained results where the outer ring illustrates the percentage of the outgoing traffic while the inner ring the incoming traffic with respect to the Catalan NREN. In Figure 6a, we can observe that the majority of the outgoing traffic belongs to the Web/HTTP and the P2P application. For the incoming traffic the situation is different: P2P application is the third in terms of traffic percentage while multimedia applications become second. It is worth mentioning that in this figure, the term multimedia application refers to all video/audio applications that are not chat, Web/HTTP, P2P or games. In Figure 6b, we further classify some of the multimedia applications. In this figure, we include all video traffic obtained through web (like YouTube or DailyMotion), P2P(like SOPcast or pplive), or streaming (like Windows Media or Quicktime).

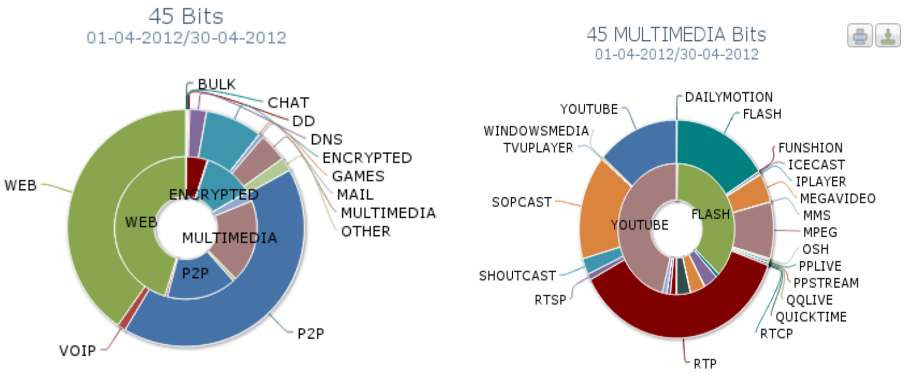


Fig. 6. Outer ring is the outgoing traffic while the inner ring is the incoming traffic a) Classification of the applications (left), and b) Classification of the multimedia traffic (right)

According to these results, the majority of traffic belongs to two types of application, namely Web/HTTP and P2P video streaming. For these very reasons, in the following sections we further analyze these applications to detect the conditions for which a multicast routing scheme would be beneficial.

4.2 Web/HTTP Traffic Analysis

This study aims at determining if Web/HTTP traffic could be served in a one-to-many (multicast) to the requesting clients. For this purpose, we define a time frame *t*. We assume that such *t* is the time a web server can wait before transmitting the packets to the client. If more than one client requests the same content during such *t*, the server will only transmit once while the multicast routing is supposed to take care of replicating the traffic in the network. Given these assumptions, we make use of the same point of measurement described below (i.e., 10Gbit Ethernet link between Catalan NREN and Spanish NREN). We analyze all traffic crossing the point of measurement during periods of 30 minutes and count all identical web content that has been transmitted during the same time frame *t*. If we define as c_i the number of times the same content *i* has been transmitted, we define as traffic saving the percentage $\sum_i(c_i\text{-}1)/\sum_i c_i$. That is the content *i* would be transmitted only once if the multicast routing path would in place instead of being transmitted c_i times. Such saving is then averaged over the total measurement time considered in this study, i.e. 30 minutes.

Figure 7 shows the results obtained for the percentage of traffic savings over time frame *t*. From this figure, we can observe that the percentage of savings provided by a multicast scheme in the case of Web/HTTP traffic application is relatively low even considering high time frame *t* such as 3 seconds (7% of savings). Note that increasing the time frame to more than 3 seconds is expected to yield higher savings. We emphasize though that this study has been realized on an NREN network and not a commercial network.

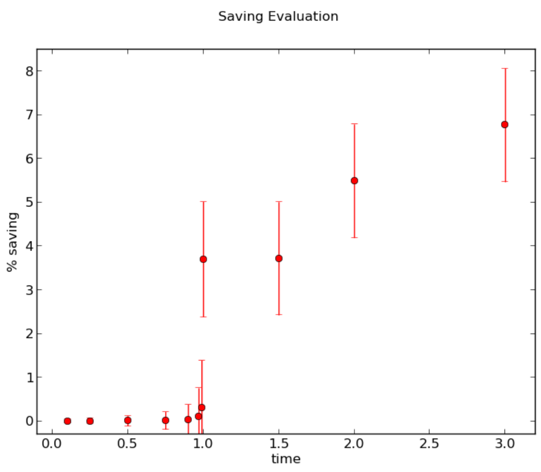


Fig. 7. Percentage of traffic savings as a function of time frame *t*

4.3 Peer-to-Peer Traffic Analysis

In this section, we analyze the P2P traffic to reveal the locality properties of receivers in live P2P streaming systems. The choice of the streaming content was driven by the subjectively perceived importance of the ongoing events at the time the experiment took place. With this in mind, we captured the content streamed by several SOPCast channels during a UEFA Champions League semifinal match. In spite of the fact that interest for such football matches is highest in Europe, the teams involved are both highly appreciated worldwide and amount players spanning many nationalities.

For the capturing process, we used 7 vantage points spanning a total of 6 countries: 2 vantage points in USA (California and Virginia), 3 in Europe (Ireland, Barcelona and Cluj (Romania)) and 2 in Asia (Singapore and Tokyo). The objective is to create an infrastructure capable of performing a world-wide distributed passive capture of large P2P live content streaming overlays. All machines involved ran Ubuntu Linux and have a 100Mbps Ethernet card; four different channels (at different bit-rates) have been analyzed.

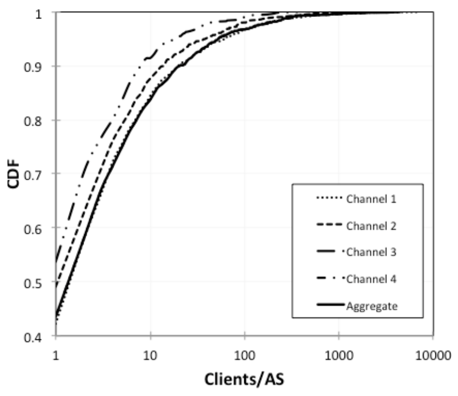
Some of the statistical properties of the traces captured are presented in Table 1. Among them, we count the peering IPs encountered in each trace, which may be used as an estimate for the number of connected end-hosts. However, because we did not identify hosts behind NAT devices, this value should be held as a lower bound estimate. From the peer IPs, the number of Autonomous Systems (AS) exchanging traffic with our nodes is inferred. We define a similarity metric in order to evaluate the breadth of the peer and AS population that we measured. For a vantage point in a channel, the metric was defined as the ratio of IPs/ASNs that overlap with those encountered in traces from other vantage points. The high values measured for IP similarity indicate that in each channel our vantage points exchanged traffic with a large fraction of the peer population, leading to an accurate aggregate view of the whole overlay. Furthermore, AS similarity values suggest that we have a precise estimate of the AS exchanging traffic. Overall, we can infer that streaming channels generally have non-overlapping clients (as expected); however, their clients belong to AS that have a large overlap.

Table 1. Trace properties

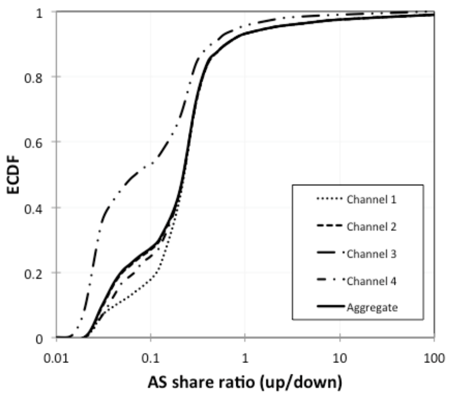
Ch1@850kbps	Download (GBytes)	Number of IPs	Number of AS	Up (%)	Down (%)	Similarity IP (%)	Similarity AS (%)
California	1.45	19250	1469	76	24	93	98
Cluj	1.50	32229	1980	90	10	92	96
Ireland	0.99	13522	1294	66	34	92	98
Barcelona	1.31	34320	1940	91	9	78	94
Singapore	1.39	37164	2039	89	11	89	96
Tokyo	1.18	37822	2028	95	5	91	96
Virginia	1.33	21864	1745	88	12	92	96
Total	9.63	64586	2839	89	11	12	68

The distribution of clients in AS is depicted in Figure 8a. It can be seen that the plots have a similar shape and the differences between them are only due to the inter-channel client variations. As we have seen in Table 1, channels tend to have non-overlapping client populations. Therefore the reasons behind the similarity of the curves have to do with a subtler phenomenon probably related to user behavior and localized user interest.

Within P2P systems, it is the responsibility of the peers to replicate content to other members. In Figure 8b we study the amount of traffic exchanged by our nodes with their AS peers. To evaluate their level of collaboration, we define and compute a sharing ratio for each overlay member. Specifically, for each peer, the sharing ratio is computed by dividing its volume of uploaded traffic by the download one.



a)



b)

Fig. 8. a) Clients per AS, and b) Distribution of clients in AS

Finally, Figure 9 illustrates the degree of collaboration between peers for both download and upload traffic. In absolute terms, the plots show that our nodes obtain between 20 and 50% of the content from peers in the same AS reaching 100% with around 100 AS. On the other hand, our nodes act as seeds for few peers in the same AS (between 10 and 20%) while all peers are in around 100 AS. These results indicate that the majority of the peers are localized outside the AS containing all vantage points. In turn, this means that a multicast scheme could potentially provide an advantage support for inter-domain P2P traffic.

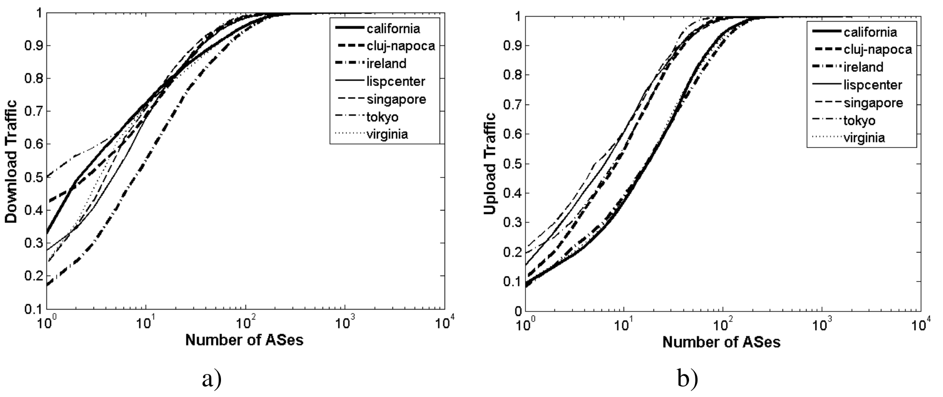


Fig. 9. Amount of a) download and b) upload traffic exchanged by VP nodes with their peers

5 Conclusion

Aiming at applying the iterative research process at the inception of the FIRE initiative, we present in this chapter its application to the development of a dynamic multicast routing scheme that would be able to overcome the known architectural and scaling limits of current protocol independent multicast routing (such as PIM) but also provide a suitable alternative to existing compact multicast routing schemes which exhibit a considerable problem: the sparse cover underlying the execution of the Abraham scheme is constructed off-line and requires global knowledge of the network topology to properly operate.

In terms of performance (Section 3.1), the proposed GCMR scheme shows substantial gain in terms of the number of RT entries compared to the Steiner-Tree (ST) heuristic (minimum factor of 3,21 for sets of 4000 leaf nodes, i.e., 12,5% of the topology size) and the memory space required to store them. The stretch deterioration compared to the ST algorithms ranges between 8% and 3% (for multicast group size of 500 to 4000, respectively); thus, decreasing with increasing group sizes. The proposed two-phase search process -local search first covering the leaf's node vicinity, and if unsuccessful, a global search over the remaining topology - combined with the vicinity ball construction at the source node- enables to keep the communication cost of the GCMR algorithm within reasonable bounds compared to the reference Shortest Path Tree (SPT) scheme and sub-linearly proportional to the size of the leaf node set.

Comparison with the Abraham scheme (Section 3.2), the GCMR scheme provides a better tradeoff between the memory space required to store the RT entries and the stretch factor increase of the produced multicast routing paths.

Having compared performance, the critical question becomes how to take advantage of these properties; indeed, the spatio-temporal distribution of traffic must exhibit locality in order to taking benefit of multicast routing. Initial results (obtained from Catalan NREN access point to the Internet) show a potential gain of less than 10% for Web/HTTP traffic. For "multimedia" traffic, this second level analysis still needs to be conducted. Moreover, the tool will be packaged in order to provide the mean for other NREN to perform similar traffic analysis studies. Ultimately, it would be interesting to initiate traffic captures at different NRENs during same time periods and perform traffic analysis across multiple NRENs.

Acknowledgements. This research work is conducted by the EULER Project (Grant No.258307) part of the Future Internet Research and Experimentation (FIRE) objective of the Seventh Framework Programme (FP7) funded by the European Commission (EC).

References

1. Fenner, B., et al.: Protocol Independent Multicast - Sparse Mode (PIM-SM), Internet Engineering Task Force (IETF), RFC 4601 (August 2006)
2. Ballardie, T., Francis, P., Crowcroft, J.: Core Based Trees (CBT): An Architecture for Scalable Multicast Routing. In: Proceedings of ACM Sigcomm, pp. 85–95 (1995)
3. Holbrook, H., Cain, B.: Source-Specific Multicast for IP, Internet Engineering Task Force (IETF), RFC 4607 (August 2006)
4. Diot, C., et al.: Deployment Issues for the IP Multicast Service and Architecture. IEEE Network 4(1), 78–88 (2000)
5. EULER FP7 Project, Performance objectives, evaluation criteria and metrics. Technical report, `https://www-sop.inria.fr/mascotte/EULER/wiki/pmwiki.php/Main/Deliverables`
6. Pedroso, P., Papadimitriou, D., Careglio, D.: Dynamic compact multicast routing on power law graphs. In: 54th IEEE Globecom, Houston, TX, USA (December 2011)
7. Abraham, I., Malkhi, D., Ratajczak, D.: Compact multicast routing. In: Keidar, I. (ed.) DISC 2009. LNCS, vol. 5805, pp. 364–378. Springer, Heidelberg (2009)
8. Bates, T., et al.: Multiprotocol Extensions for BGP-4, Internet Engineering Task Force (IETF), RFC 4760 (January 2007)
9. Rosen, E., Aggarwal, R. (eds.): Multicast in MPLS/BGP IP VPNs, Internet Engineering Task Force (IETF), RFC 6513 (February 2012)
10. Bu, T., Towsley, D.: On distinguishing between Internet power law topology generators. In: Proc. 21th Annual IEEE International Conference on Computer Communications (INFOCOM 2002), vol. 2, pp. 638–647 (2002)
11. Sage's Graph Library, `http://www.sagemath.org/`

12. Carela-Español, V., Barlet-Ros, P., Cabellos-Aparicio, A., S.-Pareta, J.: Analysis of the impact of sampling on NetFlow traffic classification. Computer Networks 55(5) (April 2011)
13. AnellaCientifica, http://www.cesca.cat/en/communications/anella-cientifica
14. RedIris, http://www.rediris.es
15. Cisco NetFlow, http://www.cisco.com/warp/public/732/Tech/netflow
16. Quinlan, J.: C4.5: programs for machine learning. Morgan Kaufmann (1993)
17. Phaal, P., Lavine, M.: sFlow Version 5. sFlow.org (July 2004)
18. Sampled NetFlow, http://www.cisco.com/en/US/docs/ios/12_0s/feature/guide/12s_sanf.html

Metrics and Measurement Tools in OpenFlow and the OFELIA Testbed

Marc Körner[1], Herbert Almus[1], Hagen Woesner[2], and Tobias Jungel[2]

[1] TU Berlin, Straße des 17 Juni 135, 10623 Berlin
{marc.koerner,herbert.almus}@tu-berlin.de
[2] EICT GmbH, Ernst-Reuter-Platz 7, 10587 Berlin
{hagen.woesner,tobias.jungel}@eict.de

Abstract. OpenFlow is a recent standard that defines the interaction between a network element (switch, router) and an externalized control element (the controller). This article introduces the metrics of interest when developing hardware and software for OpenFlow and shows how these are made available in OFELIA, a Pan-European research testbed for software-defined networks. OFELIA provides software tools like OFLOPS and cbench as well as access to specific test hardware. A section of the article describes how this hardware is integrated into the control framework of OFELIA and how it can be used by external researchers. Some measurement results recently obtained with the use of OFELIA provide an example of experiments that are done to further optimize SDN equipment and controllers.

1 Introduction

To leave traditional paths and reach out for new horizons typically implies greater chances than stepwise improvements. The idea of Software Defined Networks (SDN) is such a step off the established path, and means to re-implement networking hardware and software from scratch, based on early specifications. Testing is essential to prove newly developed solutions. Regarding network protocols early implementations deliver the first impressions of defined functions and their usability in general. With the growing interest and acceptance of a new approach, different implementations of the necessary protocols will show up and raise the question of standard compatibility and interoperability. Because the scientific and technological state of the art is still far away from allowing formal proofs based on mathematical methods, testing is the most important approach not only to prove implementations but also to deliver hints for improvement of protocols and specifications. In addition, over time increasing relevance of new solutions raise the demand to evaluate performance and scalability aspects. Again, testing is a must. At least for this kind of testing, industrial test equipment allowing to generate traffic saturating the transmission capability of the tested systems is required.

1.1 Performance and Conformance Testing of Network Devices

What is typically of interest in the data path of network equipment is the number of frames a device can forward, potentially re-writing parts of the header information (like MAC addresses). The total data rate, delay through the device, jitter (delay variation) is measured by firing a high number of frames into one or more of its ports. IETF RFC 2544 [1] defines a framework of definitions, parameters and standard settings when testing the performance of network nodes.

Conformance testing, in contrast, is typically concerned with either testing the correct format of messages or testing protocols and therefore, the correct behavior of state machines in distributed systems. The former is concerned with expected bits that delineate the header fields and can therefore be tested without keeping state information per packet or flow, the latter is more complicated as it typically involves sequence numbers or timers that inflate the number of states in a state machine.

Further in this chapter, all three dimensions of testing (performance, syntax and protocol machine testing) will be introduced in their specific versions for OpenFlow, and examples of measurement results will be shown that can be achieved with the tools available in OFELIA.

1.2 A Really Short Introduction to OpenFlow

It is probably not too exaggerated to call OpenFlow the latest hype in the telecommunications industry [6]. Similar Application Programmer's Interface (API) had been introduced before by specific Ethernet switch silicon vendors, but after a set of clean-slate research projects starting in the US, Japan and Europe around the middle of the first decade of 2000 it became clear that a renovation of the Internet communication would not only touch the parts above the much cited waist of the IP hourglass, but would have to reach down to the *link* or even to the physical layer of the communication systems. As such, OpenFlow is nothing more than an interface specification between an Ethernet switch and its controller. However, it takes into account that frame headers are checked in hardware in switches and routers today. This hardware check does not stop at the Layer-2 headers, but may look deeper into the header or even the first bytes of the frame's payload. OpenFlow specifies that the first 128 bytes of a header may be evaluated in a single match and forwarding decisions can be met not only on the outermost headers, but on the multilayered information that is contained in a frame header.

Matches, Actions, Stats. OpenFlow allows access to information about frames, and flows consequently, that ranges from the incoming ports over VLAN tags and Ethernet addresses to IP and beyond, ending with transport protocol port numbers. The set of *matches*, i.e. patterns to compare an incoming frame header against, is constantly being extended. The latest standard OpenFlow 1.3 defines 40 field match types, of which 13 are stated to be mandatory for an OpenFlow switch. In consequence of this an arbitrary match function was established, but its support in OpenFlow enabled hardware is not yet available.

Actions are the capabilities of a switch that can be called by a controller. Actions can be as simple as plain output actions as well more complicated like MAC-address-rewriting, which is what happens to a frame when it is forwarded on the IP layer.

Stats tell about the number of bytes and packets that the switch has seen so far since the rule was installed, defining a flow.

OpenFlow defines the interaction between a switch (also called datapath element) and its corresponding controller(s). The switch has a finite number of entries in its Forwarding Information Base (FIB). This switch FIB acts as a cache for the controller's FIB. If an incoming frame does not match any of the FIB rules, this unknown frame is encapsulated into a `PacketIn` message and sent up to the controller. The controller then decides how to treat the frame, it will send a `PacketOut` or `FlowMod` message.

Open Networking Forum Testing and Interoperability Working Group. The Open Networking Forum (ONF) [8] is an industry forum that has been established in March 2011 and since then is pushing forward the standardization of OpenFlow. Little more than a year after its founding, more than 70 member companies are active in several working groups. Working groups are thematically oriented and deal with topics like extensibility, configuration & management, or the northbound controller interface. The Testing and Interoperability Working Group [11] is focused on three areas: Certification, Interoperability and Benchmarking. In the area of certification the group is working on a certification test suite, starting with version 1.0. In the area of interoperability a first plug-fest took place in March 2012 in the bay area. The goal was to test OpenFlow applications for the data center/cloud, Service Provider WAN and Enterprise network. In the final area of benchmarking the working group is working on defining standard benchmarking methodology for performance testing OpenFlow devices.

2 OpenFlow Testing Tools

The currently existing OpenFlow Tools for testing divide into benchmarking and conformance testing tools. Benchmarking tools are available for switches and controllers (see sections 2.1 and 2.2). Conformance testing for OpenFlow switches is supported by OFTest described in section 2.3.

2.1 OFLOPS

OFLOPS (OpenFLow Operations Per Second) is a framework to test an OpenFlow capable switch for Linux[1]. The OpenFlow protocol version used is 1.0. The software architecture is shown in Fig. 1.

[1] Source code at `git clone git://www.openflowswitch.org/oflops.git`

OFLOPS uses the host's network stack for the control channel connection to the OpenFlow switch. The framework considers multiple approaches for the data plane channels. Packet generation is achieved either on the Linux host (user- or kernel space) or on an extended design of the NetFPGA Stanford Packet Generator [2]. Libpcap is used on a Linux host to capture the packets. It is also possible to capture packets with the NetFPGA platform. The latter gives a higher time resolution for measurements.

The framework provides a module interface for benchmarking. In the module a user has to deal with the packets sent to the control and data channels. Furthermore the concurrency control of in- and outgoing packets has to be covered in the module. OFLOPS already provides several modules to cover most of the capabilities of an OpenFlow switch. [12]

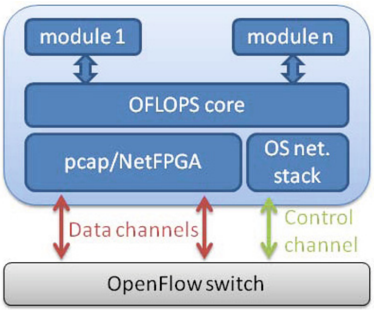


Fig. 1. OFLOPS software architecture

2.2 cbench

An OpenFlow controller benchmarker is provided by cbench, which is part of the OFLOPS framework. Several OpenFlow switches can be emulated by cbench to benchmark a controllers `PacketIn` capability. Currently cbench can benchmark both latency of a controller and throughput of `PacketIn` messages.

2.3 OFTest

OFTest is the successor of the OpenFlow regression test suite of Stanfords OpenFlow reference implementation [9]. It is a framework written in python covering many conformance tests for OpenFlow switches. At the time of writing the framework provides tests for the OpenFlow protocol versions 1.0 and 1.1. OFTest handles like OFLOPS the control and data path connections. Its software architecture is shown in Fig. 2.

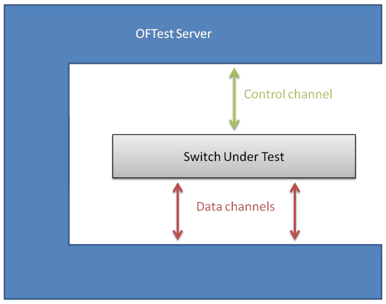


Fig. 2. OFTest software architecture

3 The Pan-European OFELIA Testbed

The OFELIA project (OpenFlow in Europe - Linking Infrastructure and Applications) started in autumn 2010 as an EU-funded research project in the Framework Programme Seven [7]. The aim was to construct (and in the second phase interconnect) a set of campus installations (islands) running a handful of OpenFlow enabled switches each and to make this facility available to all researchers. By mid 2012, 6 islands are up and running, and a total of $\tilde{3}0$ OpenFlow switches form a testbed of decent size for experimental purposes. Figure 3 shows an example of a configuration of an island in OFELIA, i.e. the island at TU Berlin. Apart from a set of physical servers and OpenFlow enabled switches typical for all islands, the Berlin island has two particularities: First, it provides a connection to BOWL, the Berlin Open Wireless Lab, a campus-wide WiFi network with now OpenFlow enabled wireless access points. Second, it gives access to an industry-level testing device, further explained in section 4.

OFELIA uses *FlowVisor*[13] to slice the network. This special purpose controller acts as a network hypervisor. It controls access to individual flows by identifying a network slice with a flow space, a set of values set in the entire set of tuples. As an example, a slice may be defined by a number of MAC addresses, an IP address and a TCP port number or any combination thereof. The underlying picture is that a slice covers an area in an n-dimensional space (for n being the number of tuples in the OpenFlow standard). Experiments are created in slices that consist of virtual machines (VMs) interconnected by OpenFlow capable switches. The experimenter can then install an own controller in one of the VMs and have the OpenFlow switches connect to this one, mediated by the FlowVisor that resides in each island.

Figure 4 indicates two slices A and B that coexist on the network. Both slices share physical servers (where VMs are deployed as Linux VMs on top of a Xen hypervisor) the link and the switch ports, where flows belonging to slice A can only be controlled by controller A and vice versa. The assignment of OpenFlow traffic to individual controllers is performed in the FlowVisor. In the Figure both slices run a NOX controller [3], the first available Open Source controller

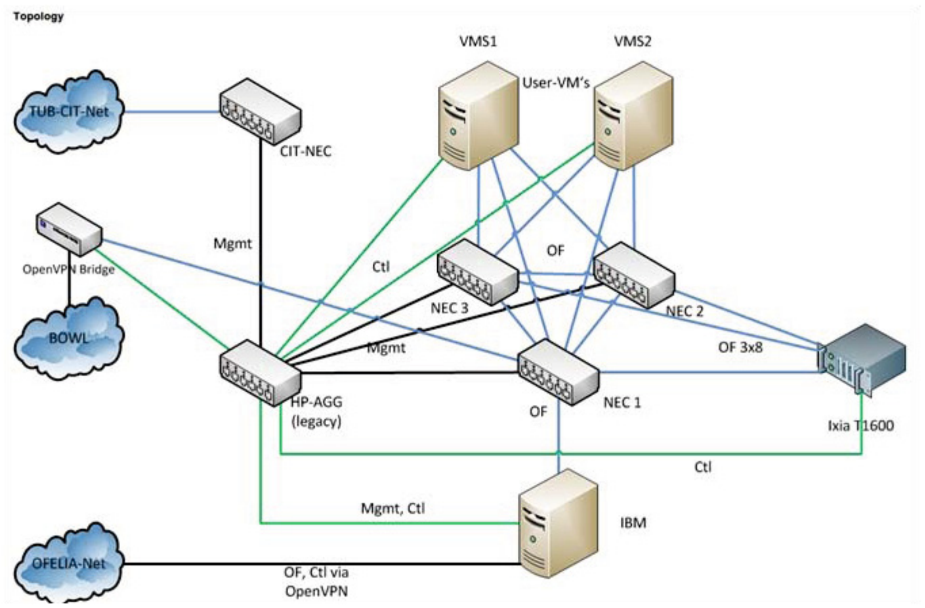


Fig. 3. An example of an OFELIA island is the configuration available at TU Berlin

for OpenFlow. OFELIA thus is more than a simple OpenFlow testbed, as the entire network slice can be generated on demand, combining cloud-like resource allocation with the capability to control the data forwarding between the traffic sources and sinks. The FlowVisor that controls the assignment of OpenFlow protocol traffic between the switches and controllers is configured through the OFELIA control framework, where the researcher has to specify the desired flow space before experimentation. Internally, OFELIA is constructed of two networks: The control network provides access to the control interfaces of the VMs based on Linux and from the FlowVisor to the controller(s). This network is IP-routed and is based on IP ranges pre-assigned to the OFELIA islands. The other network is a flat layer-2 network that connects the data ports of the VMs to the switch ports. It is the job (and the chance) of the experimenter to use any addresses on this layer-2 network.

Experimenters on the OFELIA testbed get access to the facility through an OpenVPN tunnel. Currently this gateway is currently accessing the control network of OFELIA, other gateways for the datapath are under construction and will be deployed soon.

OFELIA is constantly adding hardware to the islands as well as creating new islands with the hope to increase the base of accessible hardware for OpenFlow experimentation. Though all islands are based on a small number of OpenFlow capable NEC switches, the specific research interests and hence the hardware base vary, i.e. an emphasis on optical control framework integration at Univ. of

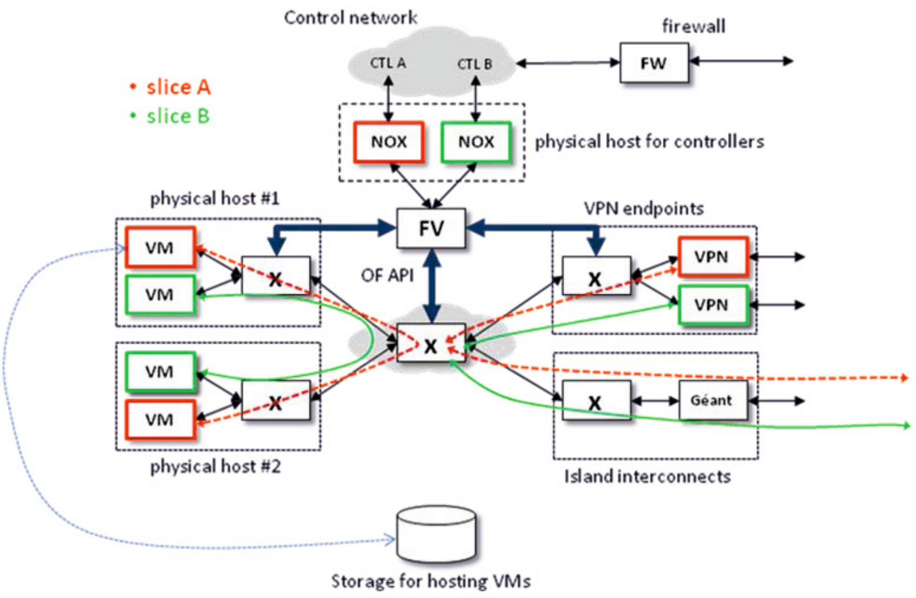


Fig. 4. Slices in Ofelia

Essex, a WiFi campus network at TU Berlin, and a server farm at IBBT of 100 nodes in Ghent.

The virtual machines have a built-in set of testing tools that will be explained in the following section. Further installations are possible, an experimenter may at any point in time install new controllers, traffic sources or sinks, and testing tools. However, the basic set of testing tools is preconfigured in the virtual machines that make up the experimenters' slices.

4 Measurement and Traffic Generation with Ixia Test Systems

In the OFELIA[7] project, testing is used intensively to validate functionality, evaluate performance or just generate load conditions. In many cases, open source based or free available tools (Wireshark[14], etc.) are sufficient to prove functionality or expected results. Nevertheless, those tools typically reach their limits if high load conditions or complex protocols are to be checked or evaluated. In those cases, industrial solutions, based on specific testing equipment in combination with sophisticated testing software are a must.

The OFELIA testbed island at the Technical University Berlin is using an IXIA T1600 tests system with appropriate interfaces together with testing software solutions like IxNetworks and IxLoad[4] to check setup and operation of the installation as well as selected OpenFlow related use cases.

In general, the testing is done by connecting the observed object, the device under test (DUT) to the test system. The test system typically injects well defined bit streams (specific frames, packets, etc.) into one or more ports of the DUT and catches the DUT's output on one ore more ports of the test system itself. Testing typically includes proving that the received bit streams are in line with the expected results (functional testing) and / or the expected performance (throughput, latency, jitter, etc.)

4.1 How to Use Ixia Equipment

Basically there is a three times encapsulated hard- and software architecture. On the line cards every port is driven by its own real-time OS with a dedicated CPU and RAM. Each of the ports communicates over a socket interface with the IxServer which runs on top of a Window system on the Ixia chassis. The applications can be used on a Windows PC or terminal server (TS) in the Ixia control network and communicate with the IxServer over a socket interface. In this model multiple users can share the resources (ports) and access them at the same time.

The OFELIA testing hardware hosted at TU Berlin provides researches the opportunity to evaluate OpenFlow controller implementations and their performance. 24 Gigabit Ethernet ports from the Ixia line cards are directly connected with all three meshed switches (for reference, see Fig. 3). This installation allows all researchers to inject and measure traffic at any point in the network. This includes the option of measuring, e.g. throughput of a single device up to a path through the experimental network.

The control interface for the ports is provided through the Ixia software tools which are installed on a Windows 2008 terminal server for synchronous and shared usage. These tools are able to test traffic on all seven layers of the ISO/OSI model. The main difference in choosing IxNetwork or IxLoad for a test is the network layer the test depends on. IxNetwork is designed for tests of the lower network layers, from one till four. In contrast to that IxLoad is designed for tests of the upper network layers, from layer three to seven.

An OFELIA user needs an account for the facility. The registration is available at the procets website[7]. With this account the user can connect via OpenVPN[10] to the central control network hub in Belgium at the IBBT department in Gent. The control network provides the user access to all available OFELIA resources across the European testbed islands. Every island uses a layer three routed control subnetwork and an interconnected flat layer two experimental network. All services in the testbed are free of charge.

On the project internal wiki pages the user gets the information of which resources are available and how they are connected with each other. With the knowledge of the resources and their connection, the user is able to allocate them by using the Expedient, which is a special web-based resource allocation tool for OFELIA islands in the control network. There, the user can spawn virtual machines and network flow space to test e.g. a new networking protocol or routing algorithm.

With his registered OFELIA account the user is now able to connect to the virtual machines per secure shell (SSH) and use other resources like the terminal server with the Ixia control software. With an remote desktop protocol (RDP) connection over the OpenVPN tunnel to the terminal server where all currently available Ixia programs can be accessed and executed.

Testing with IxNetwork. Basically, the test configuration procedure in IxNetwork consists of the following steps:

1. Add Ixia ports on which the test should be processed on
2. Set protocol configuration for the test or choose a predefined protocol of the configuration wizard
3. Set traffic item for the test
   (a) Add endpoint sets of selected ports and traffic distribution
   (b) Define packet structure, e.g. IPv4, Payload
   (c) Set rate of the packet stream
   (d) Select flow tracking for statistical analysis
4. Start defined test

Alternatively it is also possible to chose one of the predefined tests. The test wizard for this kind of tests guide the user through the whole configuration steps. Predefined tests that can be processed are very versatile. They are structured in different categories like PPPoX, IPTV, IP multicast, RFC2544 and RFC2889. All these categories again consist several tests. One interesting category for example is the RFC2544 which contains the throughput/latency and the frame loss test. An other category is the RFC2889 where the user can measure for example the address cache, address rate or broadcast rate.

Testing with IxLoad. The IxLoad application supports many test scenarios on protocol and application layer. The basic concept of this tool is that a DUT is connected to the Ixia ports where the traffic is originated and terminated from side to side with DUT in the middle. This supports testing of for example load balancer or firewalls. Supported protocols or services amongst other are DDoS, DNS, FTP, HTTP or VoIP. The test configuration procedure is described in the following steps:

1. Define Network and Traffic
   (a) Add and configure originate network and traffic
   (b) Add and configure DUT
   (c) Add terminate network and configure traffic corresponding to originate traffic
   (d) Chose traffic parameters like HTTP page size
2. Set objectives and time line
3. Associate Ixia ports with originate and terminate network
4. Start test

In the statistic section the performance values are shown and can be visualized. There are plenty of statistics available by default and also custom graphs can be displayed.

5 Examples of Experiments

This section gives an draft overview about applications and measurements with two examples. It shows the evaluation of two totally different controllers and their performance in the datapath element of the OpenFlow paradigm. The tests where processed with a Ixia T1600 and a NEC IP8800 switch, configured with OpenFlow real switching instance. The allocation of resources is done through the OFELIA control framework as described in section 4.1. Both examples use an allocated OFELIA slice with ports connected to the Ixia and a VM, which is also configured as the slice controller. The used topology is also depicted by Fig. 5.

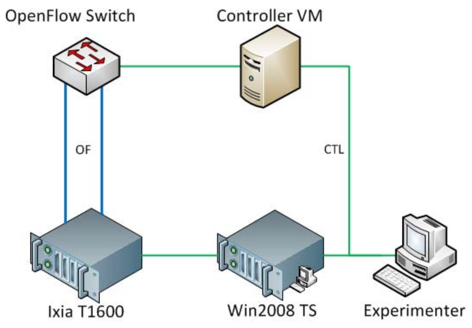


Fig. 5. Allocated test topology at TUB island

NOX Switch Plug-in Test with IxNetwork. The objective of this test is to show the basic function of the Ixia and the OFELIA-TUB testbed by collecting some performance values like throughput. The tool of choice for this test is IxNetwork, which can process tests up to network layer four. Therefore a network slice with two ports attached to the Ixia T1600 is allocated with Expedient. Additionally a VM is spawned for hosting the OpenFlow controller in form of the compiled NOX[3] with switching plug-in from the default OFELIA VM image. The switching plug-in works as a normal MAC learning switch and forwards packets for a specific layer 2 address on the port where they were seen.

After the basic configuration, a secure shell is used to connect to the VM and a RDP connection to connect to the TS server. Then the NOX controller is started on the VM by changing the directory into the OFELIA folder and executing the controller with the above mentioned plug-in by typing:

```
cd /opt/ofelia/software/nox/build/src
./nox_core -v -i ptcp:6633 switch
```

For test configuration IxNetwork is started and configured with with the following parameters:

1. Ixia ports of the project slice are added to the test
2. IP addresses were associated, address resolution protocol (ARP) activated and ports enabled
3. Defined traffic item with the traffic wizard, in this particular case we use a common IPv4 packet with UDP and a up-/down-level data sequence
4. Test is started with the play icon

During the test, IxNetwork shows the transmitted and received packets and other parameters. Basically, all statistic outputs can be customized by the user. Some traffic parameters can be changed on the fly, e.g., traffic injection speed, during the test execution. At the beginning of the test the NOX output shows detection of ARP packets and new flows. After the test is finished and NOX shows the flow expire events, which where sent by the switch to the controller if the flow expires. All these results should be observable and show the error free working of the slice and testbed.

Load Balancer Test with IxLoad. The challenge of this test is to evaluate the function of an implemented plug-in and the performance delivered on real OpenFlow hardware. This example works with a NOX plug-in which was developed as an evaluation for a research paper[5]. This plug-in implements a destination network address translation (DNAT) based load balancer (LB) by writing the flow table entries to the underlying hardware substrate. In detail, the plug-in implements three main blocks. It handles incoming ARP requests and answers them with the LB associated IP's in the particular subnetwork. All incoming service request are distributed to the servers with the instructions pushed from the controller to the switch. The switch again processes these instructions called `FlowMods` and rewrite the source and destination address on layer two and three header and vice versa.

Requirements and basic configuration are the same as mentioned in section 5. Instead of the switching plug-in, the load balancing plug-in is started with NOX.

```
./nox_core -v -i ptcp:6633 loadbalancer
```

On the TS now IxLoad is used and configured with two different subnetworks connected by the DUT where the load balancer option is chosen. In the server network, simulated by the Ixia, two http servers are configured with a content page size of 1024kB. Four clients were configured with http requests in the client network. After the test is started, throughput distribution on the client and server site can be observed in the statistics program panel. In addition the output of the load balancing plug-in shows incoming flow requests, the mapping and distribution to the servers and also the required rewrite operation of the packet header.

6 Conclusion

This article introduced the available testing tools in the OFELIA testbed. OFELIA is generally free to use for any interested researcher, and as such, the article shall also serve as an invitation to come and try out what is available, and to become a part of a growing community that is excited about the possibilities created by letting software define the network.

References

1. Bradner, S., McQuaid, J.: Benchmarking Methodology for Network Interconnect Devices. RFC 2544 (Informational) (March 1999), http://www.ietf.org/rfc/rfc2544.txt
2. Covington, G.A., Gibb, G., Lockwood, J., McKeown, N.: A Packet Generator on the NetFPGA Platform. In: 17th IEEE Symposium on Field Programmable Custom Computing Machines, FCCM 2009, pp. 235–238 (April 2009)
3. Gude, N., Koponen, T., Pettit, J., Pfaff, B., Casado, M., McKeown, N., Shenker, S.: NOX: Towards an Operating System for Networks. ACM SIGCOMM Computer Communication Review (July 2008)
4. Ixia (July 2012), http://www.ixiacom.com/
5. Körner, M., Kao, O.: Multiple service load-balancing with openflow. In: Proceedings of the IEEE 13th Conference on High Performance Switching and Routing. IEEE Publishers (2012)
6. McKeown, N., Anderson, T., Balakrishnan, H., Parulkar, G., Peterson, L., Rexford, J., Shenker, S., Turner, J.: Openflow: Enabling innovation in campus networks. ACM SIGCOMM Computer Communication Review (April 2008)
7. OpenFlow in Europe: Linking Infrastructure and Applications (October 2011), http://www.fp7-ofelia.eu/
8. Open Networking Foundation, http://www.opennetworking.org
9. OpenFlow (September 2012), http://www.openflow.org/
10. OpenVPN (September 2012), http://openvpn.net/
11. Pitt, D., Haugh, M., Sherwood, R.: Testing-Interop Working Group Interoperability Event White Paper, version 1.0. (March 2012), https://www.opennetworking.org/images/stories/downloads/white-papers/onf-testing-interop-march-2012-whitepaper-v1.0.pdf
12. Rotsos, C., Sarrar, N., Uhlig, S., Sherwood, R., Moore, A.W.: OFLOPS: An open framework for openFlow switch evaluation. In: Taft, N., Ricciato, F. (eds.) PAM 2012. LNCS, vol. 7192, pp. 85–95. Springer, Heidelberg (2012), http://dx.doi.org/10.1007/978-3-642-28537-0_9
13. Sherwood, R., Gibby, G., Yapy, K.K., Appenzeller, G., Casado, M., McKeown, N., Parulkar, G.: FlowVisor: A Network Virtualization Layer. Tech. rep., Stanford University (2010)
14. Wireshark (September 2012), http://www.wireshark.org/

Heterogeneous Testbeds, Tools and Experiments – Measurement Requirements Perspective

Anastasius Gavras[1], Andrzej Bak[5], Gergely Biczók[3], Piotr Gajowniczek[5], András Gulyás[2], Halid Hrasnica[1], Pedro Martinez-Julia[6], Felicián Németh[2], Chrysa Papagianni[4], Symeon Papavassiliou[4], Marcin Pilarski[5], and Antonio Skarmeta[6]

[1] Eurescom GmbH
[2] Budapest University of Technology and Economics
[3] Norwegian University of Science and Technology
[4] National Technical University of Athens
[5] Warsaw University of Technology
[6] University of Murcia

Abstract. Federated testbeds for future Internet experimentation have been deployed worldwide to enable large scale and diverse experiments with future Internet technologies ranging from components to complete systems. They serve the purpose of validating new solutions to identified problems and to compare them with current or other evolving solutions. The proliferation of management frameworks and in particular of measurement and monitoring tools poses often a great challenge for the experimenter. OpenLab is addressing the interoperability of testbeds at several levels. In this article we describe how the OpenLab testbeds cope with divergent measurement and monitoring requirements originating from four identified experiments that each of them has its own original research objectives.

Keywords: Future Internet Research and Experimentation, FIRE, monitoring, measurement, experimental research methodology.

1 Introduction

Experimentally-driven research is key to success in exploring the possible futures of the Internet. An open, general-purpose, shared experimental facility, both large-scale and sustainable, is essential for European industry and academia to innovate today and assess the performance of their solutions. OpenLab is a large scale integrating project that brings together the essential ingredients for such a facility [1]. It extend early prototypes of testbeds, middleware, and measurement tools so as to provide more efficient and flexible support for a diverse set of experimental applications and protocols. The prototypes include a set of demonstrably successful testbeds: PlanetLab Europe [2]; the NITOS [3] and w-iLab.t [4] wireless testbeds; two IMS (IP Multimedia Subsystem) testbeds for exploring merged media distribution; the ETOMIC high precision network measurement testbed [5]; and the HEN emulation testbed [6]. Associated with these testbeds are control- and experimental-plane software components. The project advances these prototypes with key enhancements in the areas of mobility, wireless, monitoring, domain interconnections, and the integration of new technologies such as OpenFlow [7]. These enhancements are transparent to existing users of each testbed, while opening up a diversity of new experiments that users can perform, extending from wired and wireless media distribution to distributed and autonomous management of new social interactions and localized services, going far beyond what can be tested on the current Internet. The results advance the goal of a unified Future Internet Research and Experimentation (FIRE) facility.

The chapter is organized as follows: in Sec. 2, systematic experimental methodologies for ICT and related challenges to ensure interoperability among various experimental frameworks are presented. Several experiments conducted over the OpenLab infrastructure, representing a diversity of requirements on measurements and monitoring, are presented in Sec. 3 – 6. A synthesis of the requirements from different experiments is presented in Sec. 7, followed by conclusions and outlook in Sec. 8.

2 Heterogeneous Testbeds and Challenges

Most deployed testbeds today operate control and management frameworks that are tailored to their needs. This applies also to the provided measurement and monitoring functions. They can be compared with the world of telecommunications more than a decade ago, in which each service was tightly integrated in a vertical fashion with the infrastructure that was used to provide the service. Interoperability across administrative and technology domains was difficult or impossible under cost and complexity considerations. Interoperability requirements came as a natural consequence to leverage the best available technology, reducing costs and enhancing customer experience.

2.1 Systematic Experimental Methodology

Experimental research is commonly used in sciences such as sociology and psychology, physics, chemistry, biology and medicine. It is less commonly used in other science disciplines, among others in information and communication related sciences. The Future Internet Research and Experimentation (FIRE) initiative [8] aimed at creating a research environment for investigating and experimentally validating revolutionary ideas towards new paradigms for Future Internet architecture by bridging multi-disciplinary long-term research and experimentally driven large scale validation. FIRE invested significant effort in familiarizing the ICT research community with the methodology of experimental driven research as a necessary research tool in the ICT related science disciplines. This objective evolved further and defined experimentally-driven research in ICT as a visionary multidisciplinary research field in itself, among others, by identifying the challenges for and taking advantage of environments in which experimentation can be conducted.

This was realized by means of iterative cycles of research, oriented towards the design and large-scale experimentation of new and innovative paradigms for the Future Internet – modelled as a complex distributed system. Therefore, research is expected to address all associated aspects in a holistic vision, at all relevant levels and layers.

The refinement of the research directions should be strongly influenced by the data and observations gathered from experimentation in previous iterations thus being "measurement-based" which requires the specification of relevant metrics and measurement tools. So far, a systematic experimental methodology is defined based on the experience from practicing research in ICT and typically includes the following steps:

- specification of the performance objectives, (technical and non-technical) constraints, and description of expected results
- definition of relevant performance criteria and metrics
- description of the modus operandi including configuration, initialization, and running conditions and (iterative) procedure(s) to be executed
- reporting on observations and the resulting analysis and the feedback on each iteration before reaching (partial) conclusion

Important aspects of a sound experimental methodology include:

- formal description of experiments
- degree of control of the relevant input variables of an experiment
- degree of monitoring of the relevant output variables of an experiment
- degree of confidence in the verifiability, reliability, repeatability, and reproducibility of the experimental results

2.2 Interoperability Challenges

In the world of testbeds a similar evolution is underway. The interoperability challenges in testbeds originate mainly from four observations:

1. **Interoperability of tools:** Excellent tools for various tasks exist, but they are usually tailored to the testbeds in which they originate. The requirement that follows is to enable the use of these tools also in other testbeds.
2. **Testbed migration:** Technological progress results in the emergence of new advanced testbeds and the retirement of older testbeds. The requirement that follows is how to migrate existing experimenter communities into these new testbeds. The challenge of migration is that the users often require the same tools for controlling and managing their experiments and collection of measurement results.
3. **Reproducibility of experiments:** Testbeds are not unique and several testbeds that serve the same purpose and community exist. In addition the reproducibility of experiments is part of the experimental methodology. The requirement that follows is how experiments can be reproduced in similar, yet different testbeds.
4. **Experiment expansion:** Finally experiments emerge that try to leverage enlarged capabilities in scale or scope because they address system level problems, rather than component or protocol level problems. The requirement that follows is how cross-testbed experiments can be configured, controlled and managed, whereas "cross-", refers to both vertical and horizontal expansion of an experiment.

The next section analyses four experiments that are conducted over the OpenLab testbeds and which have divergent requirements on measurements and monitoring.

3 Growth of Storage Networks

The aim of the proposed experiment is to create the replicable base for long-running service using OpenLab and PlanetLab environment in order to better observe and track the long-term growth of Storage Networks (Grids, Clouds, Content Delivery Networks, Information-Centric Networks…). The dynamic growth of such systems deployed by content distributors, aggregators and owners, has substantial impact on distribution of network traffic and the scalability of Internet services beyond the "first mile" as well as the "middle mile". As such it is useful to track and map the spread of such Storage Networks throughout the EC and on global scale with at least monthly granularity. This can give researches more insight into the evolution of Internet towards a content-centric, distributed delivery model.

In the recent years we have observed an enormous increase of the popularity of many Internet services, e.g., Facebook, DailyMotion, YouTube etc., first due to the exponential increase in number of broadband users and later due to the increase in bandwidth consumption per user. During the last five years, the backbone traffic has been increasing at a (compound aggregate) rate of approximately 40% to 50% per year with the estimate of cumulated monthly traffic 7,500-12,000 PB for EC countries. At present, a typical user consumes a few GB of video information per day; a typical movie of 90 minutes in standard definition, requiring a bit rate of 1.5 Mb/s, amounts to 1 GB. Spurred by the undeniable trend towards users actively searching for their preferred content and watching it at their most convenient time, the consumption of this video content over the Internet is gradually increasing. This can be confirmed by the success of catch-up services [10, 11], online movie rentals over the Internet [12], and YouTube movies or podcasts on the TV. In order to serve the constantly increasing demand, the Internet Content Service Providers deploy content servers in multiple locations all over the world. To obtain high level of scalability and facilitate optimal distribution of the popular content to geographically diverse population of end users, such content is usually distributed over multiple physical servers, for example by using the CDN (Content Distribution Networks) technology that utilize Storage in the network. However, CDNs today have a number of shortcomings that will become more prominent in the near future when the volume of multimedia content to be delivered will be one or two orders of magnitude higher.

Whenever a client makes a request to get the particular content, the Internet service infrastructure (with the use of DNS servers and HTTP redirections) chooses a suitable server (preferably at a location near to the client) to serve the user request. Content Providers and CDN operators have deployed thousands of servers as part of the Internet infrastructure. However, this infrastructure, operates essentially as an overlay and very little is known about the topologies, geographical spread, expansion and growth of systems that serve the most popular Internet content worldwide.

The knowledge about location of the content servers in the network and the possibility to monitor the long term changes in the infrastructures supporting the most popular Internet services such as Facebook, YouTube or DailyMotion as well as caching solutions deployed by CDN operators, content aggregators and owners, would allow us to better understand the nature and complexity of the Internet of today, and have an insight into its future evolution. This knowledge can also be used for better planning of the Internet underlying transmission resources, which is important due to the proliferation of rich multimedia content. By analysing the trends in the quantity and location (overlay topology) of the deployed servers we would be able to forecast the evolution of the traffic demands. Building such a monitoring tool is a challenging task.

The objective of the SNIFFER experiment is to implement the content server discovery and location mechanism, deployed in a form of the long term service running on the OpenLab infrastructure. Main benefits and impacts of this approach will target in four different groups of users:

- Scientific community,
- European Telco industry and digital media industry,
- European regulators in the scope of digital media industry,
- European SME in the scope of digital media industry,

The experiment will provide the location maps of the content servers related to the particular Internet services, updated periodically to allow observing the long term (in terms of years) trends and changes with at least monthly resolution. This will be achieved by a distributed content server discovery mechanism exploiting the geographical diversity of OpenLab and federated PlanetLab servers and DNS resources.

The discovery of server farms delivering particular Internet services, e.g. YouTube, will allow ISPs to manage the traffic from these services (for example on the network edge) based on operational rules (e.g. ranges of IP addresses) and not on heavily resource consuming and costly DPI approaches. The filtering can be very useful for optimal caching and distribution of the popular, resource-greedy and often demanding (low latency and jitter) video-rich content within the Service Provider's network. It has to be noted that such filtering does not influence the net neutrality principle in any way, as it does restrict neither the user's choice nor his experience of the Internet services – it just allows the operator to better manage the traffic inside its network.

3.1 State-of-the-Art of Storage Network Discovery

During the last years the Internet has grown enormously, in terms of hardware resources such as network links and routers, but also in terms of software and services that it provides to the users. Analysing the internet topology has drawn a considerable attention among the research community, so the literature on measuring and modelling the Internet on the IP interface, router, AS and POP level is very vast. In recent years, however, the enormous success of applications such as P2P systems, social networks, user-created content including video etc. has moved the attention to the overlay level, where such services are provided on top of the underlying Internet resources. Also, the creation of large-scale testbed platforms such as PlanetLab has allowed running measurement experiments on nearly global scale.

Some of the recent activities related to this experiment are as follows. The work presented in [13] is devoted to understanding the mechanisms and policies used to determine the rules of selecting the datacentres for YouTube video downloads. It uses the specific landmark-based geo-location algorithm to find YouTube datacentre locations, however, without providing visualization tools. This study is mainly devoted to analysing the characteristics of video flows and sessions related to YouTube service and to uncover the details of the datacentre selection strategy within the service. In [14] the PlanetLab distributed architecture is used to uncloak the YouTube cloud, providing insights into the cache locations and the topology of logical and physical cache resources using similar methods that are proposed in our experiment. The study is however limited to YouTube service, is not intended to be long-running and therefore lacks tools for analysing the topology changes in various Internet overlays. In [15] the same authors extend the study with charting the uncovered topologies and more insight into the YouTube mechanisms of load balancing and selecting physical servers. The study presented in [16] uses similar methodologies to uncover the internal structures of the CDN network of two largest providers of this service, Akamai and Limelight, providing tools for evaluation of availability, uptime, location and delays related to CDN storage centres.

3.2 Scientific and Technological Methodology for SNIFFER Experiment

The methodology underlying the activities planned in the experiment exploits the functionality of the common Internet protocols, the PlanetLab infrastructure and the capabilities of Telekomunikacja Polska as the largest ISP in Central Europe to obtain a large sample of Web-related customer activities.

The following picture illustrated the proposed experiment setup:

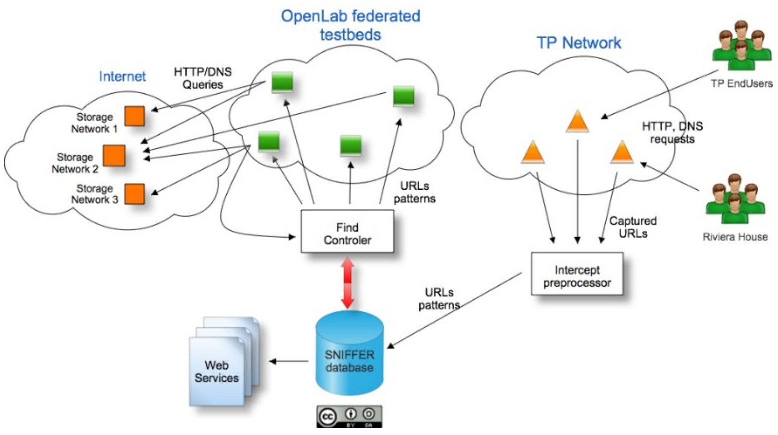


Fig. 1. Experiment components overview

The discovery of the servers hosting the content related to the particular Internet service is a multi-step process. First, we need to obtain the URLs related to popular Internet content. This will be done during the first phase of the experiment by monitoring and logging the Web-related activities of a selected group of trial customers of Orange Polska. This will be based on DPI facilities and logs from systems such as transparent caching appliances. The logs will be then analysed for the presence of specific patterns, related to popular Web services, such as YouTube, Facebook, and DailyMotion, etc., to identify the names of the servers hosting particular content.

The next step consists of resolving the IP addresses of server hostnames using DNS service. As this can be dependent on the user location, we intend to use the geographically diverse OpenLab platform including PlanetLab Europe [2], PlanetLab Central (www.planet-lab.org), testbed(s) to obtain the required geographical spread of the queries, however the experiment will also consider to use external testbed(s) like PlanetLab Japan (www.planet-lab.jp), Measurement Lab (www.measurementlab.net) and Orange Polska PlanetLab private testbed infrastructure to spread the DNS queries into many nodes to uncover the widest possible set of IP addresses related to the specific content. All addresses will be stored in central database for further processing. We will also develop the tools that allow clustering of discovered resources in order to provide consistent view for end-user. This can be done for example using traceroute to discover the routing paths common for a range of servers or using geolocation and well-known clustering algorithms to aggregate data.
The steps will be repeated periodically to allow monitoring of changes in the popularity of services and in the infrastructure that supports their delivery. The discovered infrastructures will be geo-located for the purpose of graphical presentation and charting. Geo-location can be done using different techniques: the publicly accessible geo-location databases, reverse DNS lookups or landmark-based algorithms ("triangulation" based on RTT metric related to the known "beacon" servers). In the latter case, the selected PlanetLab nodes can be used as landmarks for RTT measurements.

The results will be open for publicity, both as XML database and the visualization that will use the Google Earth APIs to provide the graphical view on the discovered infrastructures. We will also develop the tools to visualize historical changes in the infrastructure and topology related to particular Web service.

4 Social-Aware Virtual Network Embedding

Content Delivery Networks (CDN) such as Akamai and Mirror Image are being contracted by Content Providers (e.g. web sites etc.) to host and distribute content, as they effectively overcome the limitations of wide area Internet, with regards to client perceived Quality of Service. For that reason, in a CDN, content is replicated over data centers and edge servers (also known as cache or surrogate servers) scattered over the globe, and client requests are redirected to the most suitable location in terms various performance criteria [17]. However traditional CDNs require significant investments and deployment efforts, and are to a large extent application-specific, rendering a market contestable only for major players. In the emerging Future Internet paradigm the solution lies within the cloud [18, 24], including its wireless extensions. Utilizing existing storage clouds, and more recent Networked Cloud Environments [19], cloud based CDNs can reduce the effort and cost of producing complete, dedicated solutions with ever shorter development and time-to-market cycles. The term wireless cloud refers to an extension of traditional cloud computing to include mainly clients that use wireless links to connect to the network [20].

With the advent and success of social networks, the question that emerged was how this on-going trend and its benefits can influence and inspire the design of algorithms, network topologies and protocols, resulting into socio-aware networked computing systems. Social Network Analysis (SNA) [21] has become a field of its own merit; exploiting social features and structures of interest in the infrastructure through the tools provided by SNA, is a promising solution in order to improve performance and attain the corresponding social growth and user demand, leading to social-aware approaches and implementations. Towards that direction, a cloud based CDN could benefit from taking into account the social aspects of operation.

It is the intersection of all the above trends that has acted as the basic motivation of the SAViNE experiment. The proposed experiment aims at addressing this gap, exploring how social-aware virtual resource mapping can be applied for fostering wireless content delivery within the evolving networked cloud environment. Towards the solution of content distribution over the cloud, where the characteristics of CDNs are inherited to the cloud, the replica placement - that is, selecting the location and number of surrogate servers - becomes the key problem for virtualized resources selection. Thus, the selection of edge servers on the cloud amounts to Virtual Network Embedding (VNE) with location constraints on the virtual nodes - defined by geo-locating of users - and capacity requirements (e.g. storage, CPU, bandwidth) to efficiently facilitate content delivery. VNE problem is central to resource mapping in a network virtualization environment or networked cloud environment as it is essentially the problem of assigning interconnected virtual nodes to a substrate network without violating capacity constraints. The problem can be reduced to the NP-hard multi-way separator problem [19].

4.1 SAViNE Goals and Experimental Procedure

SAViNE aims at establishing, assessing, evolving and prototyping a novel socio-aware replica placement and virtual resource mapping framework for realizing a large-scale Wireless Content Delivery Network scenario. In that sense, SAViNE will provide a proof of concept of the visionary developments carried out through experimentally driven research towards the realization of Future Internet paradigms. Towards this direction two main sets of experiments will be deployed on wireless and wired testbeds of OpenLab: that is NITOS, w-iLab.t and Planetlab Europe.

The first set will involve only nodes from the NITOS and w-iLab.t testbeds supporting 802.11b/g WLAN. The goal is (i) to evaluate the effectiveness of the proposed socio-aware replica placement algorithms on the design of CDNs (ii) to measure - quantitative and qualitative - the impact of the social aspect of the algorithms on the resulting CDN design (iii) to investigate the experiment's repeatability by repeating its execution within a short timeframe (less than a day) on the same wireless testbed and (iv) to investigate the experiment's reproducibility by employing parallel identical experiments on two wireless testbeds.

On the other hand the second set will include also the PlanetLab testbed since the goal is to evaluate the performance of the proposed algorithms for realizing a large-scale Wireless Content Delivery Network. PlanetLab nodes support emulating the wireless medium in terms of bandwidth and delay with appropriately set custom values [2].

The experimental procedure essentially consists of the following phases:

- *Offline Planning Phase* involves (i) devising and adopting efficient methods for solving the socio-VNE problem tailored to the needs of a CDN design and (ii) defining an appropriate set of *embedding scenarios* using these algorithms that will eventually provide the actual CDN topology to be deployed on each testbed. The employed algorithms will be refined within the context of the particular testbed's environment (e.g. topology, resources etc.) on which the CDN will be deployed.
- *Real Time Design Phase*, where based on real time information about the availability of wireless nodes and their resources, various *embedding scenarios* will be executed providing the actual CDN topologies to be deployed on the testbeds.
- *Deployment Phase* involves the actual CDN deployment according to the results of the real time design phase and will help to draw conclusions with respect to the proof-of concept of the proposed solution and study the effect of the problem's varying parameters.

4.2 SAViNE Evaluation and Measurement Requirements

SAViNE experiment essentially consists of two distinct steps; solving the CDN design problem at hand (Offline Planning and Real Time Design Phase) and deploying it (Deployment Phase). Hence evaluation of the overall experiment can be broken down to the corresponding offline evaluation and online evaluation phases.

Offline evaluation refers to the evaluation of the proposed social inspired surrogate placement algorithms related to:

- The efficiency of the mapping solutions on the virtualized environment. The solutions of two different algorithms for mapping virtual to shared physical resources, can be compared on the basis of two commonly used metrics; the *Embedding Cost* which reflects the cost for embedding a request and allocating substrate resources and the *Embedding Revenue* which an indicator of the Infrastructure Provider's gain from accepting mapping requests (Tab. 1).
- The effectiveness of social inspired metrics on the surrogate placement problem for the creation of an efficient content delivery network. Different CDN designs based on different social inspired algorithms can be compared, based on the overall *CDN Deployment Cost* according to the adopted cost model, the *Number of Surrogate Servers* selected along with other relevant QoS/geographical proximity related metrics e.g. the *Path Length* from the end user to the origin server of the CDN (Tab. 1).
- The impact of the adopted social inspired algorithms on the CDN design. In order to compare social inspired and non socio aware replica placement algorithms for a virtualized environment, appropriate social metrics can be adopted like *Shortest Path Betweeness Centrality* [23] for the CDN solution measures the extent to which a node has control over the traffic-carrying capabilities of the infrastructure (Tab. 1).

- The ability of the algorithms to produce equivalent mappings of specific CDN virtual topologies in terms of cost and topological characteristics over different testbeds. Social inspired metrics like *Similarity* [23] as defined in Tab. 1 can reveal the dependency of the mapping solution on the topology of the underlying shared physical infrastructure.

Online evaluation refers to the evaluation of the deployed CDNs, adopting metrics related to:

- Performance measurement of the deployed CDNs. The target is to measure the CDN's ability to serve the customers with the desired content. Typical metrics used by the content providers to evaluate the performance of a CDN are, among others, user perceived response time and surrogate server utilization (Table 1) [17].
- The wireless medium. The correlation between the medium quality and CDN performance leads to the inspection of metrics such as Signal to Noise ratio, since the sustainable rate is directly associated to the *SNR* over the link.
- Experiment repeatability and reproducibility. The aforementioned metrics related to the performance of the CDN and the wireless environment can be utilized to investigate the repeatability of the experiment executed on the same wireless testbed and its reproducibility on different ones with the same operational characteristics.

Table 1. Indicative SAViNE experiment metrics

Embedding Cost [19]	The cost of embedding a request for a virtual topology equals the amount of required bandwidth for all virtual links over the substrate paths as defined by the mapping solution, augmented by the physical resources allocated to each virtual node associated with its type (e.g. disk space for a virtual server).
Embedding Revenue [19]	The embedding revenue of a request for a virtual topology equals the amount of required bandwidth for all virtual links, augmented by the requested resources utilized for each virtual node associated with its type (e.g. disk space etc for a virtual server).
CDN Deployment Cost[22]	The overall cost of deploying a CDN based on the adopted cost models (retrieval, update, storage [24]).
Number of Surrogate Servers [22]	The number of nodes that are selected by the replica placement algorithm to host a surrogate server.
Path Length[22]	The average number hops between end-users and the origin server in a particular CDN solution.
Shortest Path Betweeness Centrality [23]	Shortest Path Betweeness Centrality for a particular CDN solution is defined as the average of individual SPBCs [8] of the selected physical nodes comprising the final CDN solution.

Table 1. (*Continued.*)

Similarity [23]	The Similarity metric is established on the comparison of distances computed over a path stretch distribution for a virtual link. Path stretch for a virtual link mapped to a physical path is defined as the number of hops in the path divided by the virtual link's number of hops. Therefore path stretch distributions for a virtual link are diversified on the basis of different mapping solutions, as a result of either different algorithms on the same testbed or the same algorithm on different testbeds.
Latency [17]	User perceived response time of the replica servers
Server Utilization [17]	It refers to the fraction of time during which the surrogate servers remain busy.
SNR	Signal to noise ratio is a relative measure of the received signal power compared to the noise power in the wireless channel.

5 Architectures for the Internet of Things

Within the general scenario of the Future Internet (FI), the rising of the Internet of Things (IoT) [25] exposes many requirements to the global network that should be met by any architecture for FI. IoT scenarios are mainly composed of devices with limited resources, so the protocols used to interact with them must have low-footprint and other specific capabilities to overcome such limitations.

This experiment – called ANA4IoT – has the main objective of studying the suitability of different architecture proposals for the FI when used in IoT scenarios. It will provide empirical observations and evidences to determine the strengths and weaknesses of such architectures to meet with the requirements imposed by IoT. Moreover, from the experimentation results, we can also provide some recommendations to improve the architectures and make them fit better in IoT scenarios. In addition, we will analyze and determine the benefits that the design principles for the FI [26], as defined by the FIArch group [27], may give to the design of the studied architectures.

Prior to the execution of the experiment and as part of the OpenLab project [1], we will deploy the network architectures used in the experiment on top of some facilities and infrastructures, specifically built for research and experimentation, to form a cross-site testbed. The main architecture that we will deploy is the HIMALIS architecture [29, 30, 31], proposed as part of the AKARI [28] project of the NICT (Japan), which also collaborates with the project. To this respect, we will also investigate the feasibility of the security enhancements of HIMALIS [29] and, as mentioned above, how the general architecture could be used in IoT environments. Finally, we will study how to achieve and enhance the privacy [32, 33] of network entities (persons, machines, etc.) when they communicate.

The HIMALIS architecture proposes the separation of locators and identifiers by the introduction of new elements to resolve identifiers to locators and the distinction between global locators, used in the global transit network (e.g. the Internet), and local locators, used in the access network (e.g a campus or building network). With these mechanisms, the architecture resolves the mobility, multi-homing, and routing scalability problems. However, the introduction of the identifier layer, which is not introduced only in this architecture, may be inadequate for IoT environments, hence the interest to begin the experiment with this architecture.

Below we describe the integrated cross-site testbed that will be used to perform the experiments, the required measurements to obtain the empirical evidences to demonstrate the feasibility of the architectures or the possible aspects to improve, and discuss the problem derived from the requirements imposed to the experimentation infrastructures used to build the testbed.

5.1 Integrated Cross-Site Testbed

Before the proposed experiments can take place we will build a composed testbed from other existing testbeds. The result will give us a cross-site testbed with enough capabilities to perform complete experiments with new architecture proposals for the FI, like the HIMALIS architecture introduced above.

We start the integration work with the GAIA testbed [43], placed at the University of Murcia (UM). On it we will instantiate the basic elements defined in the HIMALIS architecture to have a minimal working environment that permits us perform the necessary tests to know the specific requirements to connect it with the other testbeds that will form the integrated cross-site testbed.

After that, we will integrate the HEN testbed [6], placed at the University College London (UCL). This testbed provides enough resources to instantiate many interconnection components of the HIMALIS architecture. This infrastructure will bring us a comparable environment to real-world network deployments, which is essential when performing research and experimentation on architectures for the FI. Therefore, at this stage we will have various edge network domains at GAIA and the global interconnection elements at HEN, all of them working with the HIMALIS approach.

The next step is to integrate resources from PlanetLab Europe (PLE) [2] to instantiate new network domains and obtain a whole network experimentation environment for the HIMALIS architecture. This step finalizes the construction of the wide and basic infrastructure that will be used to perform the necessary experiments.

Finally, to tackle with some specific issues of IoT we will integrate the NITOS [3] testbed. This provides us the possibility to perform experiments with real low-powered devices (sensors) and see how they integrate with the other infrastructures. The significance of this integration is to obtain results which are as close as possible to the real world.

Once we achieve the integration of the different testbeds and the deployment of the mentioned architectures, we will design and perform the experiment to obtain the results discussed at the beginning of this section. To control the experiment operation we will use OMF (cOntrol and Management Framework) [44] and NEPI (Network Experimentation Programming Interface) [45]. This way, we ensure the easy verifiability, repeatability, and reproducibility of the experiment, as they are the main requirements for experimentally driven research [9].

5.2 Required Measurements

As discussed throughout this section, the objective of the proposed work is to study the suitability and possible improvements of the deployed architectures for IoT environments. In this section we discuss the measurements we need to obtain from the studied architectures to precisely profile them and thus know their general behaviour.

The first problem we find in the proper network interactions of IoT is that the maximum packet size supported by low-power devices (mainly sensors) is much reduced. Thus, we need to measure the packet size of each communication, both in edge networks (access) and in the global transit network (interconnection infrastructure).

An important aspect of any network architecture, not just for IoT, is the latency introduced by the network itself and also by the intermediate elements. In addition, some specific operations of the architecture, such as the resolution from identifiers to locators, may introduce latency at unexpected points of the network. This latency should also be measured, both independently and as part of other network operation.

To know the general performance of the deployed architectures, we also need to measure the effective bandwidth and throughput in comparison with the raw or theoretical capabilities of the experimentation network. Also, these measurements should be obtained for different network sizes to know and demonstrate the scalability of the architectures.

Other important aspect of any architecture for FI is the mobility support. Thus, for the analyzed architectures we need to measure the handover delay of a network node when moves from a network to another network. Also, during this operation, the network may lose packets so we need to measure the packet loss rate and total number of lost packets during the whole operation.

Finally, to study the security and privacy levels we need to observe and measure the packets or messages that can be intercepted or introduced by an attacker. Also, we need to measure the amount of information that a third entity may obtain from the communication taking place between two other entities. In summary, we need to obtain any evidence of active or passive access to non-authorized information by a non-authorized entity.

5.3 Problem Statement

To obtain the measurements discussed above for the analyzed network architectures we need specific tools which should be integrated with the different experimentation infrastructures that form the cross-site testbed we will use in our study.

To measure the packet size we need to intercept the traffic between different nodes, whether they are instantiated in GAIA, HEN, PLE, or NITOS. The resulting measurements should be coordinated from a central point, so they can be matched. Therefore, we need a tool that interacts with those infrastructures, tell them to intercept the packets, and obtain their size. This may be achieved by any general network protocol analyzer but the experimentation infrastructures should offer the support for this type of tools or a compatible mechanism, such as delivering the information through an artificial tunnel to intercept the packets.

Related to the measurement of packet size we have the latency. To measure the latency we only need to detect the time when a packet gets out from a sender node, the time when crossing each intermediate element, and the time when it reaches the destination node. To obtain this type of measurements, apart from the traffic interception, we need to synchronize the clock sources of all elements involved in the experiments, so the timestamp obtained with each measure from the protocol analyzers can be comparable.

To measure the effective bandwidth and throughput we need a special tool connected to the nodes at both ends of each communication of a specific experiment designed to obtain these values. Also, to facilitate the scalability analysis, the tool should support to be connected to many ends at the same time. This tool may be the same or derived from the protocol analyzer mentioned above because it should also be connected to the network interfaces of the involved elements of the experimentation infrastructure. However, it should support precise time measurement together with packet count during specified periods of time.

Apart from the end-to-end or intermediate measurements commented above, to analyze the behavior of the handover operation during a mobility event we need to coordinate the measurements from nodes that are not directly involved in the same communication. In effect, the nodes forming the destination network of a mobile node need to cooperate with the nodes of the original network when obtaining the time and packet loss measurements.

Measuring the security and privacy levels imposes different requirements to the experimentation infrastructures, such as the possibility of differentiating original and modified or illegitimate packets/messages when flowing between any pair of network elements. Moreover, the security experiments are complex to design and perform because of the different aspects to observe and the necessity to introduce artificial elements to the network. This should be provided by a specific tool for experimental security analysis.

In summary, the most important requirement to obtain the necessary measurements is the access to the network interfaces that pertain to the elements of the experimentation infrastructures that form part of each experiment. Also, we need to synchronize the clock sources to ensure that the time measurements that involve different elements are comparable. Finally, specific and complex tools are necessary for complex experiments that involve different measurements and even introduce fake (but controlled) elements to the network.

5.4 Summary

In this section we discussed the specific measurement requirements that should be met by any experimentation infrastructure, either integrated or by a separate tool. This functionality should be adapted to the cross-site testbed we are building, so it should be supported by GAIA, HEN, PLE, and NITOS. This implies that the commented measurement requirements should be applied to those infrastructures, both when working separate and when combined.

6 Geo-location Aware Greedy Routing Architecture

The growth and development of the Internet today have reached a critical point requiring major reformations. There is now a growing body of evidence supporting that the Internet routing system is in the danger of collapsing under its own weight. The most important sign of the problems concerning the Internet routing ecosystem is the alarming rate at which the size of the routing tables, to be looked up by routers to identify the direction in which to forward a packet, is growing [34, 35]. The trend has been super-linear for some time, meaning that the size of the routing tables grows at a higher rate than new hosts and routers are added to the system. This leads to the situation that contemporary core routers nowadays need to manage, and look up, on a packet-by-packet basis, about 350 thousand routing entries in their routing tables [36]. Today, plummeting hardware prices allow the routers to cope with the growth, but it is questionable whether this will be sustainable in the long term.

The ALLEGRA experiment addresses the question whether novel routing schemes can be used and operated efficiently while limiting the size of routing tables. Moreover, our goal is to implement an error tolerant routing solution with fast failure recovery, multicast and multipath enhancement. Furthermore, we plan to achieve this without using high-overhead, communication-intensive routing protocols. To meet these requirements, we deploy and run experiments with the lightweight greedy routing architecture in the OpenLab FP7 world-scale testbed facility. To the best of our knowledge, this will be the first attempt on enabling and conducting hands-on, large-scale experimentation with greedy routing mechanisms in a highly practical networking context. Note that the EULER (http://www.euler-fire-project.eu/) FP7 project concerning greedy routing has similar goals. However the ultimate target of EULER is to investigate whether the routing ecosystem of the Internet can be redesigned to incorporate greedy routing, while still meeting the routing policy requirements. The budget and goal of ALLEGRA is much smaller. We only try to explore how the geo-graphic coordinates of the nodes can be used to design simple yet efficient routing strategies.

6.1 Greedy Routing

In a greedy routing context the elements of the network are embedded into some metric space, which is the two dimensional Euclidean plane in the illustrative example of Fig. 2. The underlying metric space can be used for routing as a guideline to forward the packets to the right direction without explicit knowledge of the topology in a fully distributed manner. In greedy routing the nodes forward packets towards their neighbors, which are the closest to the packets' destination.

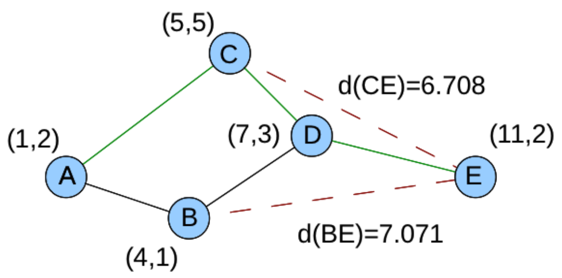


Fig. 2. Greedy routing example

If for example node A wants to send a packet to node E (Fig. 2), it looks up its neighbors, and finds out that C is the closest to E and forwards the packet to C. If C and D repeat the same process then the message gets delivered through the green path, which is called a greedy path. If the embedding of the network is appropriate, this lightweight routing process finds routes between arbitrary pairs of nodes. Note that e.g., the deletion of edge (C)-(D) does not cause routing to stop working, since greedy routing automatically finds the alternate path (through B).

6.2 Experiment Description

The ALLEGRA experiment runs over PlanetLab nodes and starts by selecting a subset of the available nodes on which the experiment will run. After selecting the nodes, we determine their geographical position by using OpenLab's geolocalization services. For this purpose Spotter or the standard PlanetLab localization API are to be used for getting as precise location data as possible. Then a greedy routable overlay topology is designed over the designated PlanetLab nodes. To set up effective overlays in terms of stretch and error-resilience, we combine our hyperbolic-space algorithms described in [37] with the obtained location data. As the next step, we feed this topology as a proper "experiment description" to the experiment controller, e.g. NEPI [45].

From the "experiment description", NEPI (Network Experiment Programming Inerface [39] can configure the PlanetLab nodes and set up the overlay topology by creating the necessary tunnels between the nodes. The greedy routing module will be implemented in OpenFlow software switches, therefore NEPI should ensure an OpenFlow friendly environment on the PlanetLab nodes, and the appropriate libraries should be provided by the time of deployment. The idea behind choosing OpenFlow is to ensure the re-usability of our codes even outside the PlanetLab environment. The tunnels of the overlay topology can be installed in basically two ways at the nodes. First, according to the current philosophy of PlanetLab, virtual network devices can be created in the nodes, which are readable by "PlanetLab compliant" OpenFlow switches (with additional code for supporting standard PlanetLab interfaces). This solution provides the comfort of running our experiments as standard PlanetLab experiments, however, the scenario is not flexible since it requires modifications in the networking interfaces of the OpenFlow switch code. In the timeframe of ALLEGRA this approach seems to be available and reasonable. Another solution is to implement standard virtual devices on the PlanetLab nodes, which can be mapped to the

OpenFlow switches in the standard way. Since PlanetLab does not currently have such solution this can be considered as a future requirement.

For exploiting the inherent fast error recovery feature of greedy routing, the availability of the links between the nodes must be continuously monitored; for multipath routing, the available bandwidth on the links should be measured. For this end, the fine-grain resolution of the ETOMIC (European Traffic Observatory Measurement Infrastructure) (http://www.etomic.org/) could be used. In summary one can derive that the ALLEGRA experiment take a big advantage of the federated nature of OpenLab by applying PlanetLab, ETOMIC and Spotter.

Once the nodes and the topology are set up properly, the measurements can be run focusing mainly on stretch, robustness, error tolerance and overhead. For error tolerance investigation link and node failures will be emulated (node/link disconnected) and the effect on the on-going flows (packet loss, delay, recovery time) will be analysed. For testing the multipath performance, we start multiple flows between randomly selected source-destination pairs and measure the one-to-one throughput with and without multipath functionality.

7 Synthesis of Requirements

From the individual requirements of the above experiments we can identify that the individual requirements on measurements and monitoring that should be fulfilled by the OpenLab project are quite divergent. It is not a surprise that the first set of common requirements that emerged are related to the low level networking parameters such as connection setup time, transit time of packets (latency), packet loss, available bandwidth, handover delay, and other metrics related to the wireless medium. A second set of requirements are related to geo-location of nodes as a performance parameter combined with high precision measurements for monitoring links between nodes.

One major challenge of the OpenLab project is to provide to all experimenters the same access to monitoring and the same measurement functionality regardless of the testbed in which their experiment is deployed and independent of the network type (wireless, fixed…) or other differentiators. Although this will not be possible in all cases, a common framework and integrated tools will also ease operations of the testbeds by their respective owners.

Thus, the OpenLab goal is to provide interoperability of different experiment control frameworks available in the testbeds; e.g. OMF experiment controller, Network Experiment Programming Interface, MySlice, and Federation Computing Interface and Federation Scenario Description Language. The support functions to enable repeatability and comparability of experiments in experimentation scenarios is being provided through the development of algorithms and methodologies for measurement and evaluation of protocols, devices and networks relative to a reference evaluation.

OpenLab establishes a data collection infrastructure to collect and share data among users of different testbeds. Through the data collection infrastructure, the users are empowered to compare their results to other historical traces or they can validate their novel evaluation method with already existing inferred quantities. The Network Measurement Virtual Observatory that already serves as a collection point of raw and evaluated measurement data for several experimental infrastructures is forming the basis for the OpenLab data collection infrastructure. The main objective here is to provide new features enabling the storage of experimental setup and configuration data and to develop new semantic data access, data processing, and visualization methods. The most generic information elements (e.g., basic data structures) of the currently running infrastructures will be collected and provided to the users, and thus owners, of the experiments in a unique semantic representation format.

8 Conclusions and Outlook

As stated in section 2.1 the ICT research community has not treated research methodologies – including monitoring and measurement methodologies – with the same care as other scientific fields. The availability of large shared testbeds makes repeating experiments a simpler task thus enabling peer-reviews and higher precision of collected data. In addition new research questions require large, long-term studies to investigate certain statistical properties. The impact of systematic errors in such studies and the associated cost has led to the requirement to publish these data to the broader community. This in turn requires accepted methodologies and policies to reduce the burden on the data collector while at the same time ensuring better data quality.

The community of testbed operators for Future Internet research and experimentation has started to work on the design and development of an agreed monitoring and measurement framework. This will allow measurement and monitoring resources to be treated as normal infrastructure resources that can be reserved by experimenters. The framework will define related metadata based on existing open standard specifications to allow the unambiguous description of experiments and related data. Certainly existing tools must be adapted to integrate to the framework without disruption of operations and the work of existing experimenter communities.

References

1. OpenLab, `http://www.ict-openlab.eu`
2. PlanetLab Europe, `http://www.planet-lab.eu/`
3. NITOS wireless testbed, `http://nitlab.inf.uth.gr/NITlab/index.php/testbed`
4. w-iLab.t wireless testbed, `http://www.ibbt.be/en/ilabs/ilab-t/wireless-lab`
5. European Traffic Observatory Measurement InfrastruCture (ETOMIC), `http://etomic.org`
6. HEN emulation testbed, `http://mediatools.cs.ucl.ac.uk/nets/hen`
7. OpenFlow, `http://www.openflow.org/`
8. Gavras, A., Karila, A., Fdida, S., May, M., Potts, M.: Future Internet Research and Experimentation. ACM SIGCOMM Computer Communication Review 37(3), 89 (2007), `http://portal.acm.org/citation.cfm?doid=1273445.1273460`
9. Gavras, A. (ed.): Experimentally driven research, white paper, On the existence of experimentally-driven research methodology, Version 1 (April 2010), `http://www.ict-fireworks.eu/fileadmin/documents/Experimentally_driven_research_V1.pdf` (downloaded July 2012)

10. IPLAYER, BBC iPlayer TV Home, http://www.bbc.co.uk/iplayer/
11. HULU, Watch TV. Watch Movies, http://www.hulu.com
12. NETFLIX, Watch TV Shows Online, Watch Movies Online, http://www.netflix.com
13. Torres, R., Finamore, A., Kim, J.R., Mellia, M., Munafò, M.M., Rao, S.: Dissecting Video Server Selection Strategies in the YouTube CDN. In: The 31st International Conference on Distributed Computing Systems (ICDCS 2011), Minneapolis, MN, USA, June 20-24 (2011)
14. Adhikari, V.K., Jain, S., Chen, Y., Zhang, Z.-L.: Reverse Engineering the YouTube Video Delivery Cloud. In: IEEE HotMD 2011 (2011)
15. Adhikari, V.K., Jain, S., Chen, Y., Zhang, Z.-L.: Where Do You "Tube"? Uncovering YouTube Server Selection Strategy. In: IEEE ICCCN 2011 (2011)
16. Huang, C., Wang, A., Li, J., Ross, K.W.: Measuring and Evaluating Large-Scale CDNs. In: IMC 2008, Vouliagmeni, Greece, October 20-22 (2008)
17. Pathan, A.-M.K., Buyya, R., Vakali, A.: Content Delivery Networks: State of the Art, Insights, and Imperatives. In: Content Delivery Networks. Lecture Notes in Electrical Engineering, vol. 9(1), pp. 3–32. Springer, Heidelberg (2008), doi:10.1007/978-3-540-77887-5
18. Broberg, J., Buyya, R., Tari, Z.: MetaCDN: Harnessing 'Storage Clouds' for high performance content delivery. Journal of Network and Computer Applications 32(5), 1012–1022 (2009), doi:10.1016/j.jnca.2009.03.004.
19. Leivadeas, A., Pappagianni, C., Papavassiliou, S.: Efficient Resource Mapping Framework over Networked Clouds via Iterated Local Search based Request Partitioning. To Appear in IEEE Transactions on Parallel and Distributed Systems (2012) (in press)
20. Vassilaras, S., Yovanof, G.: Wireless Going in the Cloud: A Promising Concept or Just Marketing Hype? Special Issue: Distributed and Secure Cloud Clustering, Wireless Personal Communications 58(1), 5–16 (2010), doi:10.1007/s11277-011-0284-9.
21. Katsaros, D., Dimokas, N., Tassiulas, L.: Social network analysis concepts in the design of wireless ad hoc network protocols. IEEE Network 24(6), 23–29 (2010)
22. Pappagianni, C., Leivadeas, A., Papavassiliou, S.: A Cloud-oriented Content Delivery Network Paradigm: Modelling and Assessment. Under Review, IEEE TDSC Special Issue on Cloud Computing Assessment: Metrics, Algorithms, Policies, Models, and Evaluation Techniques
23. Leivadeas, A., Pappagianni, C., Papavassiliou, S.: Socio-aware Virtual Network Embedding. To Appear in IEEE Network Magazine (2012) (in press)
24. Chen, F., Guo, K., Lin, J., La Porta, T.F.: Intra-cloud Lightning: Building CDNs in the Cloud. In: IEEE INFOCOM, pp. 433–441 (March 2012), doi:10.1109/INFCOM.2012.6195782
25. Sarma, A.C., Girao, J.: Identities in the future internet of things. Wireless Personal Communications 49(3), 353–363 (2009)
26. Papadimitriou, D., Zahariadis, T., Martinez-Julia, P., Papafili, I., Morreale, V., Torelli, F., Sales, B., Demeester, P.: Design principles for the future internet architecture. In: Álvarez, F., et al. (eds.) FIA 2012. LNCS, vol. 7281, pp. 55–67. Springer, Heidelberg (2012)
27. Future Internet Architecture Group, http://ec.europa.eu/information_society/activities/foi/events/fiarch-23052011/index_en.htm
28. National Institute of Information and Communications Technology, "AKARI" Architecture Design Project for New Generation Network (2010), http://akari-project.nict.go.jp

29. Martinez-Julia, P., Gomez-Skarmeta, A.F., Kafle, V.P., Inoue, M.: Secure and robust framework for id/locator mapping system. IEICE Transactions on Information and Systems E95-D, 108–116 (2012)
30. Kafle, V.P., Inoue, M.: HIMALIS: Heterogeneity inclusion and mobility adaptation through locator id separation in new generation network. IEICE Transactions on Communications E93-B(3), 478–489 (2010)
31. Kafle, V.P., Otsuki, H., Inoue, M.: An id/locator split architecture for future networks. IEEE Communications Magazine 48(2), 138–144 (2010)
32. Gomez-Skarmeta, A.F., Martinez-Julia, P., Girao, J., Sarma, A.: Identity based architecture for secure communication in future internet. In: Proceedings of the 6th ACM Workshop on Digital Identity Management, pp. 45–48. ACM, New York (2010)
33. Martinez-Julia, P., Gomez-Skarmeta, A.F., Girao, J., Sarma, A.: Protecting digital identities in future networks. In: Proceedings of the Future Network and Mobile Summit 2011. International Information Management Corporation, pp. 1–8 (2011)
34. Huston, G.: BGP reports (July 2007), http://bgp.potaroo.net/
35. Meyer, D., Zhang, L., Fall, K.: Report from the IAB workshop on routing and addressing. RFC 4984 (2007)
36. Csernai, M., Gulyás, A., Rétvári, G., Heszberger, Z., Császár, A.: The skeleton of the Internet. In: Proceedings of GLOBECOM 2010 Conference, Miami, Florida, December 6-10, vol. 2, pp. 1208–1212 (2010)
37. Gulyás, A., Kőrösi, A., Rétvári, G., Bíró, J., Szabó, D.: Network Formation Games Can Give Rise to Realistic Networks. In: Proceedings of ACM PODC, Funchal, Portugal, July 15-18 (2012)
38. Németh, F., Stipkovits, Á., Sonkoly, B., Gulyás, A.: Towards SmartFlow: Case Studies on Enhanced Programmable Forwarding in OpenFlow Switches. In: Proceedings of ACM SIGCOMM Demo, Helsinki, Finland, August 13-17 (2012)
39. Lacage, M., Ferrari, M., Hansen, M., Turletti, T., Dabbous, W.: NEPI: using independent simulators, emulators, and testbeds for easy experimentation. ACM SIGOPS Operating Systems Review 43(4), 60–65 (2010)
40. Triukose, S., Wen, Z., Rabinovich, M.: Measuring a Commercial Content Delivery Network. In: WWW 2011 Proc. of the 20th International Conference on World Wide Web (2011)
41. Leighton, T.: Given the Internet's bottlenecks, how can we build fast, scalable, content-delivery systems. Communications of the ACM 52(2) (2009), http://www.akamai.com/dl/technical_publications/leighton_byline_09.pdf
42. Wong, B., Slivkins, A., Sirer, E.G.: Meridian, A Lightweight Network Location Service without Virtual Coordinates. In: SIGCOMM 2005, Philadelphia, Pennsylvania, USA, August 21-26 (2005)
43. Wong, B., Stoyanov, I., Sirer, E.G.: Octant: A Comprehensive Framework for the Geolocalization of Internet Hosts. In: Symposium on Networked System Design and Implementation, NSDI (2007)
44. Martinez-Julia, P., Jara, A.J., Skarmeta, A.F.: GAIA Extended Research Infrastructure: Sensing, Connecting, and Processing the Real World. In: Korakis, T., Zink, M., Ott, M. (eds.) TridentCom 2012. LNICST, vol. 44, pp. 3–4. Springer, Heidelberg (2012)
45. OMF (cOntrol and Management Framework) (2012), http://omf.mytestbed.net
46. Quereilhac, A., Lacage, M., Freire, C., Turletti, T., Dabbous, W.: NEPI: An Integration Framework for Ntwork Experimentation. In: Softcom (2011)

Measurements and Measurement Tools in OpenLab: Use Cases with Measurement Data Ontologies

J.E. López de Vergara and J. Aracil

High Performance Computing and Networking Research Group
Escuela Politécnica Superior, Universidad Autónoma de Madrid, Spain
{jorge.lopez_vergara,javier.aracil}@uam.es

Abstract. In this chapter we review the network measurement tools and instruments that are in place in the FP7 OpenLab project, which provides a set of different network testbeds. We also focus on the problem of data heterogeneity, which comes from the fact that different measurement tools will provide data in different formats and structured in different ways. Therefore, a common information model becomes necessary and the OpenLab ontology-based solution is presented. Such ontology is being standardized by an Industry Specification Group at ETSI, resulting in the first ever standard on network measurement data ontologies.

Keywords: OpenLab, measurement in testbeds, ontology, network measurement, data integration.

1 Introduction

In the recent past, we have witnessed the deployment of many different networking testbeds, such as Emulab [1], PlanetLab [2], OneLab [3], NITOS [4] and the many testbeds from GENI [5]. Despite that they are targeted towards different scenarios, i.e. mobile, fixed, virtual, etc., all of them have one thing in common: the need for measurement.

Actually, an integral part of any experiment is to measure the results. In networking experiments, the measurement instrumentation can be quite sophisticated, because the different performance metrics require an ever-increasing accuracy. For example, measuring one-way delay in high-speed networks becomes complex due to the need of global synchronization at the microseconds timescale. For example, the ETOMIC testbed [6] provides such synchronization with specialized network interface cards which are directly attached to a GPS device.

Not only the measurement instrumentation becomes sophisticated but it also provides myriads of monitoring information in many different formats. In the one-way delay example the delay units can be milliseconds, microseconds or any other time unit. For more complex measurements, such as jitter, there are many different definitions.

Furthermore, many different measurement instruments can participate in a single experiment. The above one-way delay measurement can be combined with an available bandwidth estimator, which is a different measurement instrument, to correlate link load with latency. Both instruments, the one-way delay and available bandwidth estimator will provide data in a different format from their respective repositories. Combining such data becomes essential in order to have a meaningful and complete view of the experiment outcome. In particular, note that there many different dimensions to measure data (time dimension, capacities, etc) but also many different ways to obtain the measurement (per-segment, end-to-end, etc.). This complicates matters for measurement data integration.

In this chapter we present a state of the art of the different measurement instruments which have been developed under European funding. Then, we focus on the OpenLab measurement instruments. Finally, we dwell on how to integrate measurement data from different instruments. For this, we propose to use the ontology which is being standardized with OpenLab in the ETSI group *Measurement Ontology for IP traffic* (MOI).

2 Measurement Instruments within European Research Programs

In this section we provide a brief overview of the most significant measurement instruments that have been designed under the European funding umbrella. As a first approximation we classify such systems into active measurement, passive measurement and hybrid instruments.

2.1 Active Measurements Instruments

Towards this direction The European Traffic Observatory Measurement Infrastructure (ETOMIC) was created in 2004-05 within the EVERGROW Integrated Project [7] launched by the Future and Emergent Technologies Programme of the European Union VI framework. Universidad Pública de Navarra, Universidad Autónoma de Madrid and Collegium Budapest Association have developed the ETOMIC platform. Its goal is to provide an open access, public testbed for researchers investigating the Internet with active measurement methods, to serve as a Virtual Observatory active measurement data on the European part of the Internet. The developed measurement nodes are fully reconfigurable, extremely accurate (nanoseconds) and GPS-synchronised, which properties make the ETOMIC a measurement infrastructure with unique capabilities. The ETOMIC infrastructure was awarded with the Best Testbed Award in the TridentCom 2005 Conference, and it was opened and introduced to the community in the INFOCOM 2006 Conference. 18 active probing nodes were deployed Europe wide.

Then, the ETOMIC infrastructure was improved and extended in the OneLab2 project. In order to significantly improve the measurement capabilities of PlanetLab Europe, an Advanced Network Monitoring Equipment (ANME) [8], which are the new ETOMIC measurement boxes, are deployed next to the PlanetLab nodes of each Onelab2 consortium member. The main component of ANME, a highly reliable server PC (redundant power supply and interface cards) equipped with an accurate monitoring card (capable of nanoseconds timestamping), offers high precision (GPS-synchronized) active monitoring of the network. An experimental module was also designed for ANME. Another monitoring equipment developed at Onelab2, called APE (Active Probing Equipment) [9], offered a low-cost, low-maintenance monitoring solution in exchange for a less accurate timing precision. Low-cost and low-maintenance come from the fact that the hardware employed is not specialized and the system does not feature an operating system or software that must be updated periodically. The time-synchronization of the different ANME and APE boxes is achieved by new generation GPS receivers.

Another research approach towards active measurements include the DIMES project [10] which is a subproject of the EVERGROW Integrated Project in the EU Information Society Technologies, Future and Emerging Technologies programme. DIMES has been developed by Tel-Aviv University and studies the structure and topology of the Internet in order to obtain a very accurate map and annotate it with delay, loss, available bandwidth, and link capacity. The DIMES project was based on measurement by software agents, which are downloaded by volunteers and installed on their privately owned machines. The DIMES agent performs Internet measurements such as ping and traceroute at a low rate, consuming about 1KB/s. The agent does not send any information about its host's activity or personal data, but only the results of its own measurements. In TridentCom 2009 the DIMES project presented a paper that describes how software DIMES agent can emit packets trains at constant rate which can enhance the ability for making highly distributed and high accuracy measurements together with the Etomic system [11].

PerfSONAR [12] is another infrastructure for network performance monitoring based on active measurements, making it easier to solve end-to-end performance problems on paths crossing several networks. It contains a set of services delivering performance measurements in a federated environment. These services act as an intermediate layer, between the performance measurement tools and the diagnostic or visualization applications. This layer is aimed at making and exchanging performance measurements between networks, using well-defined protocols. The perfSONAR software can also be reused to include different agents from different infrastructures, as long as they comply with the perfSONAR interfaces and data types.

This active measurement instruments primarily provide measurements such as one-way latency, jitter, loss, bandwidth and route discovery.

2.2 Network Monitoring Based on Passive Measurements

Passive measurement systems provide network measurements through the use of packet capturing and NetFlow/IPFIX [13,14] data. Measurements acquired from these systems include traffic classification and network statistics such as packet loss, delay and bandwidth. The LOBSTER project [15] has built an advanced pilot European Internet traffic monitoring infrastructure based on passive network monitoring sensors. LOBSTER has also developed novel performance and security monitoring applications, which have been enabled by the availability of the passive network monitoring infrastructure, and has realized the appropriate data anonymization tools for prohibiting unauthorized access or tampering of the original traffic data.

The passive monitoring applications running on the sensors was developed on top of MAPI (Monitoring Application Programming Interface) [16,17] an expressive API in C for building network monitoring applications, which was developed in the context of the SCAMPI [18] and LOBSTER projects. MAPI enables application programmers to express complex monitoring needs, choose only the amount of information they are interested in, and therefore balance the monitoring overhead with the amount of the received information. Furthermore, MAPI gives the ability for building remote and distributed passive network monitoring applications that can receive monitoring data from multiple remote monitoring sensors through a common access platform. MAPI supports commodity Ethernet network interface, as well as Endace DAG and COMBO cards.

The LOBSTER sensors monitor the network traffic using different measurement applications such as traffic categorization, packet loss measurement, available bandwidth measurement, network traffic statistics and intrusion detection. Depending on the type of measurement or traffic processing, the gathered data or the computed results are stored in files, a database, or Round-Robin Database (RRD) archives, while usually they can be viewed through a Web interface. Basically, the novelty of the project lies in the integration of different measurements from different sensors, in order to correlate them.

2.3 Hybrid (active + passive) Network Monitoring

The FP7 MOMENT project [19] aimed at integrating different platforms for network monitoring and measurement to develop a common and open pan-European infrastructure. The system included both passive and active monitoring and measurement techniques via a common web services interface and an ontology that allows semantic queries. It continuously monitors the macroscopic status of the network and individual domains by leveraging network tomography techniques. One of the projects goals was to offer a pan-European platform for detecting macroscopically relevant events like outages, attacks, world-scale infections, and large anomalies (e.g. BGP storms).

As a general approximation, active methods produce artificial traffic that is injected in the network whereas passive methods only need a traffic copy (splitter or span). This has the advantage of being non-intrusive, but, on the other hand, it does not provide the same granularity, because in active methods the input traffic can be modulated to the specific performance parameter to be measured.

2.4 Outlook

As it turns out, there has been a significant effort within the Seventh Framework Programme in traffic measurement and analysis tools. However, as of today, there are still some pending issues. First, there is no traffic repository at the European level that serves the research and industrial community. This is a clear need for practitioners in the field because the analysis of a given protocol requires packet samples. The traffic analysis process usually involves the implementation of ad-hoc protocol dissectors, which are targeted towards the achievement of a given performance metrics.

For example, one may be interested in the calculation of the message interarrival time for MQ messages, when transmitted on top of TCP. This kind of traffic cannot be found in public repositories easily. However, this is a very common standard in banking datacenters and the traffic analyst demands this type of traffic traces.

Second, we are still missing an integrated view of active measurements within the European Union. Even though many research efforts have been performed (MOMENT, etc) there is no common standard to supply traffic measurements in an integrated way which can be easily accessed by the research community. In that sense, the OpenLab project is making a considerable effort in the integration of heterogeneous measurement data. However, the standardization bodies should push for a wider standardization of the different measurement data formats. As an example, we will report about the Measurement Ontology for IP traffic group at ETSI.

Finally, sustainability of the measurement platforms also becomes an open issue. Many FP7 projects have been funded that deployed a valuable measurement infrastructure. Unfortunately, many of such efforts have ended up as a infrastructure with little or no use for the lack of a continuous funding model. Thus, the measurement infrastructures have to find an adequate business model that guarantees long-term sustainability beyond the first tranche of funding.

As a conclusion, the state of the art in measurement infrastructures shows many interesting efforts in Europe, but there is the need to continue such efforts both in the data heterogeneity aspects and traffic repositories as well as in the related business models for continuous operation.

3 Network Monitoring in OpenLab

As mentioned before, ETOMIC is also a part of the measurement instruments within OpenLab. Integration of ETOMIC with OneLab Europe was performed in the OneLab2 project, with hardware extensions such as the ARGOS card and APE box. The ARGOS card is based on the NetFPGA hardware card and was presented in the second GENI-FIRE workshop in Washington DC, July 20th, 2009. Figure 1 shows the ETOMIC box and APE equipment attached to a port mirror of the switch where the PlanetLab box i connected, so that the traffic can be measured and analyzed.

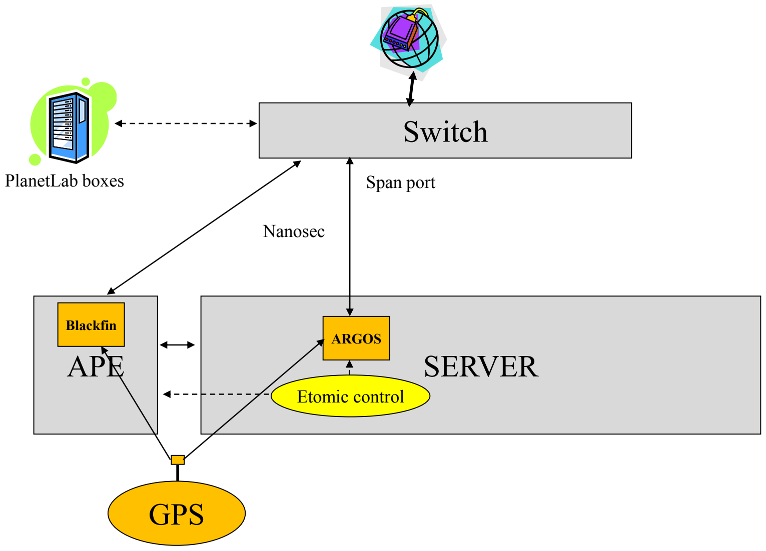


Fig. 1. Integration of ETOMIC/APE in PlanetLab Europe

The Network Measurement Virtual Observatory is a framework to efficiently store and share research data. nmVO (nm.vo.elte.hu) is originally designed by Eötvös Loránd University (ELTE) and developed partially in the OneLab2 project. Beyond the simple data collecting and archiving functions it provides 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.

TopHat [20] is a distributed active measurement system for regular route traces between large numbers of measurement agents. Usually, such agents are "plug-and-play" software that is installed by Internet users. Once the software is installed in the user computer it registers in the control system with no need of human intervention. The innovative aspect of the system consists in its use of techniques to avoid overlap in work between different agents, permitting a higher frequency of probing for constant effort. TopHat draws on measurements from the ETOMIC and DIMES measurement systems. It also is one of the few systems to deploy Paris Traceroute [21], which allows for correct route traces in the face of load-balancing routers (which are ubiquitous in today's Internet).

4 The Problem of Data Heterogeneity

As stated before, there are several active and passive monitoring instruments in every testbed to obtain network measurements. Such measurements can be packet delays, packet losses, link bandwidth usage, available bandwidth, routing, etc. The users of a federated testbed need to have an integrated view of their experiment measurements, where all data is displayed homogeneously, independently of the monitoring instruments used. The integration of such measurements can be valuable to obtain network weathermaps or network tomographies. Actually, a network tomography is an experiments that aims at obtaining performance data about the whole network with only partial information. For example, one may wish to obtain the per-segment delay with the end-to-end delay estimation.

This heterogeneity leads to several problems, such as different names, data representations, units, metadata and data merge: For instance, the values measured in a *traceroute* can include source and destination nodes, intermediate nodes and several samples of the delay between the source node and each other node in the route to the destination. However, this integration in a single view is difficult, because each monitoring instrument provides its own view about these measurements, usually stored in a repository implemented in a relational database [22]. Even this simple network measurement can be represented in different ways depending on the measurement instrument: Names used to represent each field can be different (*naming problem*). For instance, *delay* can be also called *RTT* or *RoundTripTime*. Each node can be represented as a DNS name, or an IPv4 address in dotted decimal or as a 32-bit unsigned integer, or even an IPv6 address (*data representation problem*). Delay can be measured in seconds or milliseconds (*data units problem*). Finally, other information, such as the time the measurement was taken, or the number of times it was repeated can also be found in the monitoring instruments, once again represented in several ways (*metadata problem*). Moreover, the measurements can be of different types, for instance, to obtain a network tomography that provides at the same time delays and bandwidth usage between two nodes (*data merge problem*).

To solve these problems, solutions such as those suggested by the Open Grid Forum Network Measurements Working Group (NMWG) [23] and PerfSONAR [24] have been proposed. To provide a homogeneous data representation, these works have focused on the specification of an XML syntax for network measurement data and a set of web service interfaces. Nevertheless, these technologies have the following drawbacks [22]:

– They are based on XML Schema [25], which just provide a common syntax. In this way, it is not possible to infer any information directly from measurement data. This inference is only possible if a given application analyzes the data.
– Their offered services are restricted to a reduced set of functions, which do not contain every type of network measurements. For instance, the SQL Measurement Archive interface is limited to link utilization, link capacity, input errors, output drops and circuit status measurements.

A more powerful solution is to deal with the information at a semantic level, enabling some degree of inference and automatic reasoning over the retrieved measurement data [26]. At the same time, it is possible to define the information at different abstraction levels, which allows the definition of specific class of measurements that are derived from generic ones. Then, to answer these problems, we propose a solution that deals with the semantics of the information, unambiguously specifying the set of concepts that comprise a measurement (e.g, a ping measurement is a round-trip time delay measurement, and it contains the IP address of the destination and the round trip of each probe), trying to to solve the way in which measurements have been named or the units used in the measurement, as well as including all related information. This solution is composed of three steps:

1. Agree on a common ontology for network measurements. This common ontology will provide the set of concepts and properties of each network measurement, as well as its metadata. This step is explained in subsection 4.1.
2. Define mappings between each database schema and the common ontology. These mappings are the hard part. They are needed to identify which concept is related to each column or table in measurement repository, as well as its data representation and units. This step is shown in subsection 4.2.
3. Define mechanisms to distribute a semantic query, based on the common ontology, among every data sources containing the monitored information. These mechanisms allow to merge information from different monitoring instruments, and let the users to obtain the integrated view of the network measurements. This final step is explained in subsection 4.3.

4.1 Measurement Ontology

The first step to solve the problem of data heterogeneity is to define an ontology for this domain. An ontology provides a vocabulary of classes, instances and relations to describe a domain, stressing knowledge sharing and knowledge representation [27]. Ontologies have some advantages over other information representations:

– The information can be modeled more flexibly than using relational tables. This allows the description of a measurement from a set of previously defined and related concepts and properties, which can also be derived from parent concepts.
– Information described with an ontology can be automatically classified (e.g. a tool that performs passive measurement is a passive tool) and used to infer new knowledge (e.g. if a measurement is higher than a threshold then the network is saturated).
– The information contained in the ontology can also be queried (e.g. retrieve all measurements for a given IP address).

– Finally, semantic web ontologies also provide distributed characteristics and their definition can be downloaded from defined URLs. Moreover, semantic web tools can be leveraged to work with these ontologies (even in other domains such as network measurements).

Currently, the *Measurement Ontology for IP traffic* (MOI) Industry Specification Group at ETSI is doing a standardization effort to do this specification. As it name suggests, this work group is devoted to define the set of concepts and properties for measurements in the scope of packet networks, currently dominated by the IP protocol. The ontology has to describe at the same time very general concepts from network measurements such as delay or network segment, and also the particular measurement values taken from the measurement instruments. MOI's work is based on existing specifications from IETF (SNMP MIBs, IPFIX, PSAMP, IPPM), Open Grid Forum, CAIDA's DatCat or the results from European projects such as MOMENT and PRISM, dealing both with active and passive network measurements [28]. This ontology is intended to be used in OpenLab project to homogenize the network measurements.

Given the complexity of such definition, the MOI ontology is in fact a set of ontologies, each one devoted to a specific subdomain [29,30]. Figure 2 shows the hierarchy of the ontologies:

– *General concepts.* This upper ontology contains the definition of general concepts such as location.

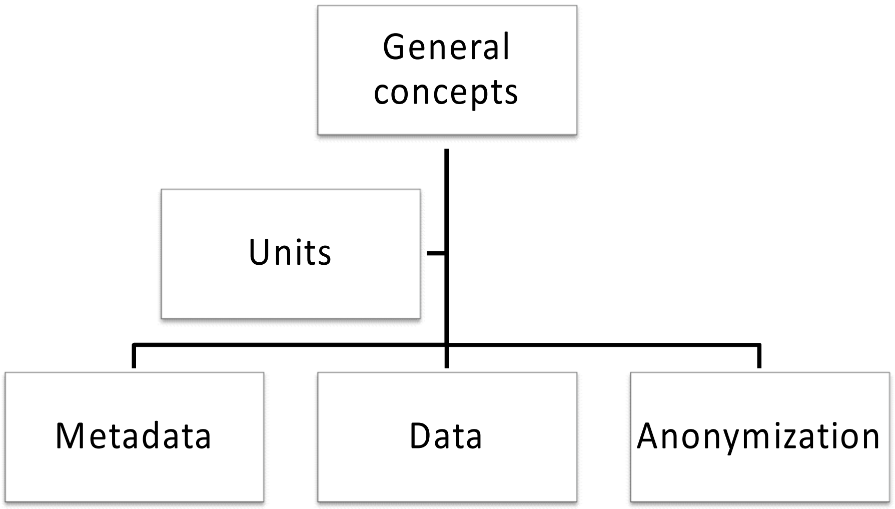


Fig. 2. Hierarchy of MOI ontologies

- *Measurement Units.* It is based on the NASA units ontology [31], but extending it for network measurement units such as data rate.
- *Measurement metadata.* It provides information about the measurements, such as how and when they were measured and where they are stored.
- *Measurement data.* It defines the classes and properties of network measurements, which can be active or passive and their relations to the network nodes where they are measured. There are two main classes, which are *Measurement* (the parent class of all measurements) and *MeasurementData*, which is used to contain the values of a certain *Measurement*. *MeasurementData* also includes a set of facets, such as its default unit or the data type. Each defined measurement (delay, packet loss, etc.) is derived from *Measurement* class.
- *Anonymization.* Last but not least, this ontology several anonymization strategies of the measurement data, given that it may be used by third parties.

It should be noted that this specification does not contain any measurement instances. These will be generated, based on the ontology, by the monitoring instruments.

4.2 Ontology Mappings

Once a network measurement ontology has been agreed (based on the discussions in the ETSI specification group), the second step is to map the schemas of the measurement repositories provided by the monitoring instruments into that ontology. These mappings will allow to represent measurement instances as they have been defined in the ontology. This is not an easy task, because the way in which the information is stored is completely heterogeneous. Mappings can not be obtained automatically. However, there are some heuristics [32] that can be taken into account. If the information is stored in relational databases, which is common, the initial approach would be to look for table to concept and column to property mappings. Nevertheless, this is a naive approach, because sometimes it is necessary to join several tables to obtain instances of a concrete concept, or on the other hand, it is necessary to split a table to obtain instances of several concepts. To look for mappings, it is possible to look for similar words in the definition of tables and rows which are very close to words in the ontology. In this case, given that it is common to find several words merged with a camel-case syntax, these words can be split to find similar substrings (both in the relational database and in the ontology). Finally, it is also possible to look for synonyms of these words. A Levenshtein distance can be used to propose the best candidates for the mapping.

An example of mapping rules is shown below. In this case, we are going to map a database that stores MRTG records about bandwidth usage. These MRTG records are stored in a table named *timeseries*, that has several columns, such as *timeunix*, *net*, *incoming*, and *outgoing*, which provide respectively a timestamp of the measurement, the measured network, ant eh incoming and outgoing traffic. For the sake of clarity, the code provided is restricted only to the mapping of the table and the first column.

```
 1  map:timeseries a d2rq:ClassMap;
 2     d2rq:dataStorage map:database;
 3     d2rq:uriPattern
 4        "timeseries/@@timeseries.net|
 5         urlify@@/@@timeseries.timeunix@@";
 6     d2rq:class MD:MRTGMeasurement;
 7     d2rq:class MD:Measurement;
 8     d2rq:classDefinitionLabel "timeseries";
 9     .
10  map:timeseries_timeunix a d2rq:ClassMap;
11     d2rq:dataStorage map:database;
12     d2rq:uriPattern "timestamp/@@timeseries.timeunix@@";
13     d2rq:class MGC:TimeStamp;
14     .
15  map:timeseries_timeunix_value a d2rq:PropertyBridge;
16     d2rq:belongsToClassMap map:timeseries_timeunix;
17     d2rq:property MGC:timestamp;
18     d2rq:propertyDefinitionLabel "timeseries timeunix";
19     d2rq:column "timeseries.timeunix";
20     d2rq:datatype xsd:long;
21     .
22  map:timeseries_measurementDataTime a d2rq:PropertyBridge;
23     d2rq:belongsToClassMap map:timeseries;
24     d2rq:property MD:hasMeasurementData;
25     d2rq:refersToClassMap map:timeseries_timeunix;
26     .
```

First, lines 1-9 provide the mapping of the *timeseries* table with a class of the ontology named *MRTGMeasurement*, which is a subclass of *MRTGMeasurement*. Next, lines 10-14 provide the mapping of the *timeunix* column with a class of the ontology named *TimeStamp* and the values of this columns are mapped in lines 15-21 to the property *timestamp*. Finally, lines 22-26 link these instances as data values of *TimeStamp*.

4.3 Ontology Data Query

The final step is a semantic interface able of receiving a semantic query from a user and distributing it among all measurement repositories, which obtain the measurement instances that have been mapped to the ontology in the step before. Figure 3 show how this system has to be deployed.

The semantic interface has to know which measurements are stored on each repository. This can be derived from the mapping rules that have been obtained previously. Then, based on this information, the following interaction is done [26]:

1. The query is split in several subqueries that are sent to each repository that contain such information.
2. SPARQL (a recursive acronym for SPARQL Protocol and RDF Query Language) semantic queries are translated to SQL at each repository. For this, an application such as D2R Server [33] can be used.
3. Next, the obtained information is translated back as ontology instances using the mapping rules.

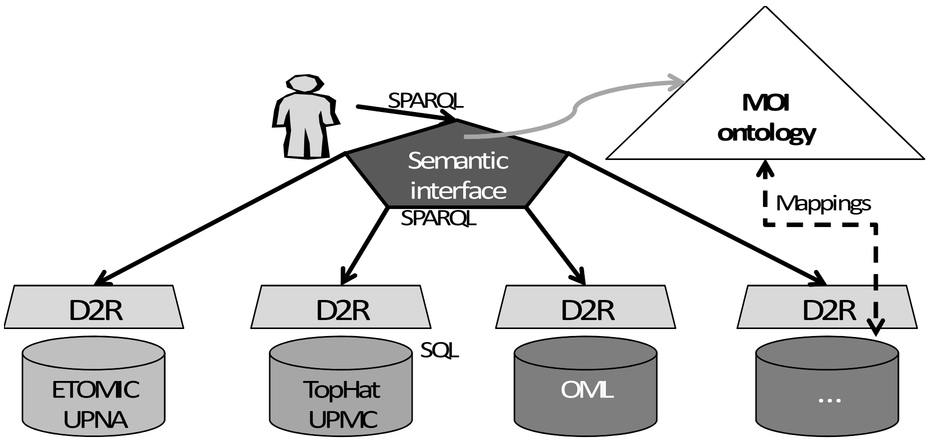


Fig. 3. OpenLab semantic interface for network measurement repositories

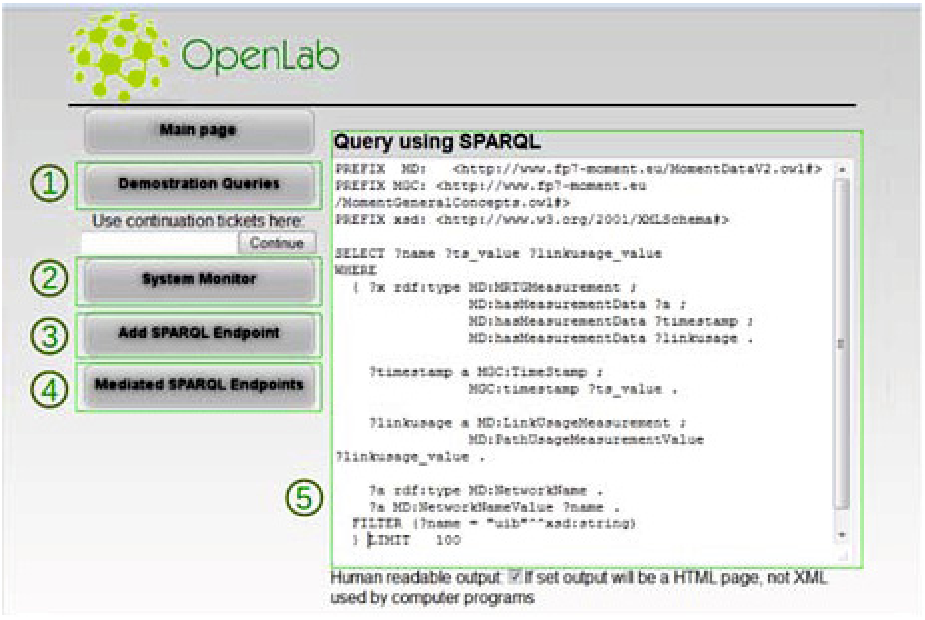
4. Later, the ontology instances that are provided from the different repositories as a result of the query are merged, and queried again to obtain the final response.

This solution allows the retrieval of answers that are composed of data from several monitoring instruments. Obtained data include what is their representation, avoiding the heterogeneity problem explained above. Applications on top of this system will not have to be aware that the input sources have different data representations.

This system has been implemented, providing the graphical user interface shown in Figure 4.

The elements of this interface are the following:

1. In Demonstration Queries, a set of predefined SPARQL queries can be executed. Such queries provide examples that show the capabilities of the system.
2. In System Monitor, currently executing queries can be checked. Queries can take some time until they retrieve all the information from the repositories.
3. In Add SPARQL Endpoint, a new data repository can be added to the system. To register a repository it is necessary to provide the mapping rules between the database and the ontology.
4. In Mediated SPARQL Endpoints, a list of available data repositories can be checked. This list show all registered repositories as well as their mapping rules.
5. This area is the core of the system, allowing a user to write a SPARQL query for the requested network measurements. The result of this query can be shown in a human-readable format, or directly in XML to feed another application.



Query using SPARQL

```
PREFIX  MD:   <http://www.fp7-moment.eu/MomentDataV2.owl#>
PREFIX MGC: <http://www.fp7-moment.eu
/MomentGeneralConcepts.owl#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>

SELECT ?name ?ts_value ?linkusage_value
WHERE
  { ?x rdf:type MD:MRTGMeasurement ;
                MD:hasMeasurementData ?a ;
                MD:hasMeasurementData ?timestamp ;
                MD:hasMeasurementData ?linkusage .

    ?timestamp a MGC:TimeStamp ;
                MGC:timestamp ?ts_value .

    ?linkusage a MD:LinkUsageMeasurement ;
                MD:PathUsageMeasurementValue
?linkusage_value .

    ?a rdf:type MD:NetworkName .
    ?a MD:NetworkNameValue ?name .
  FILTER (?name = "uib"^^xsd:string)
  } LIMIT   100
```

Human readable output ✓ If set output will be a HTML page, not XML used by computer programs

Fig. 4. Graphical user interface [34]

An example of semantic query is shown below:

```
 1  PREFIX MD:   <http://www.etsi.org/ISG/MOI/MoiData.owl#>
 2  PREFIX MGC:
 3     <http://www.etsi.org/ISG/MOI/MoiGeneralConcepts.owl#>
 4  PREFIX xsd:  <http://www.w3.org/2001/XMLSchema#>
 5  SELECT ?name ?ts_value ?linkusage_value WHERE {
 6     ?x a MD:MRTGMeasurement ;
 7         MD:hasMeasurementData ?networkName ;
 8         MD:hasMeasurementData ?timestamp ;
 9         MD:hasMeasurementData ?linkUsage .
10     ?timestamp a MGC:TimeStamp ;
11         MGC:timestamp ?ts_value .
12     ?linkUsage a MD:LinkUsageMeasurement ;
13         MD:PathUsageMeasurementValue ?linkusage_value .
14     ?networkName a MD:NetworkName ;
15         MD:NetworkNameValue ?name .
16    FILTER (?name = "NETWORKNAME"^^xsd:string)
17  } LIMIT   100
```

The first four lines define the prefixes that are going to be used when a concept or a property is used in the query. The fifth line starts the query stating that three variables are going to be retreived (*name*, *ts_value* and *linkusage_value*). Lines 6 to 15 state the conditions of the query. First (lines 6-9), an auxiliary variable x is defined, which class is an *MRTGMeasurement* , and it has different data, such as a *networkName*, a *timestamp* and a a *linkUsage*. Then (lines 10-15), *timestamp*, *linkUsage* and *networkName* are used to obtain their values. Next, line 16 is used to obtain only those results where the network *name* is NETWORKNAME. Finally, line 17 ends the query limiting the number of results up to 100.

5 Conclusions

This chapter has presented several network measurement instruments that have been developed in prior European projects. These instruments can do both active or passive measurements. Some of them are integrated in the OpenLab project, which integrates several network testbeds. The integration of several measurements from different network instruments leads to a data heterogeneity problem. This problem is being solved by using a semantic-based solution, where a common ontology for network measurements is defined and mappings to the data structures have to be specified. The user of this solution only needs to know the structure of the common ontology to query several measurement repositories that are defined independently with different SQL schemas. The system translates the data contained in the repository to the ontology using the mapping rules.

We are currently defining such mappings for static infrastructures, where the data is defined uniformly for every network measurement. This can be for instance the case of ETOMIC or TopHat. We are also working on generating dynamic mappings that should be done on-the-fly, for instance, for OML (OMF Measurement Library) measurements [35], where each measurement structure is defined when the experiment is developed.

References

1. `http://www.emulab.net/`
2. `http://www.planet-lab.org/`
3. `http://www.onelab.eu/`
4. `http://nitlab.inf.uth.gr/NITlab/index.php/testbed`
5. `http://www.geni.net/`
6. ETOMIC Project, `http://www.etomic.org`
7. EVERGROW Project, `http://www.evergrow.org`
8. ANME Box, `http://www.onelab.eu/index.php/services/testbed-components/anme-box.html`
9. APE Box, `http://www.onelab.eu/index.php/services/testbed-components/ape-box.html`
10. DIMES Project, `http://www.netdimes.org`
11. Allalouf, M., Kaplan, E., Shavitt, Y.: On the Feasibility of a Large Scale Distributed Testbed for Measuring Quality of Path Characteristics in the Internet. In: TridentCom 2009, Washington, DC, USA (April 2009)
12. Boote, J.W., Boyd, E.L., Durand, J., Hanemann, A., Kudarimoti, L., Lapacz, R., Simar, N., Trocha, S.: Towards Multi Domain Monitoring for the European Research Networks. Selected Papers from the TERENA Networking Conference, TERENA (2005) ISBN 90-77559-04-3; also published in Computational Methods in Science and Technology, Pozna, Poland, 11(2), 91–100 (2005) ISSN 1505-0602

13. Claise, B.: RFC 3954 Cisco Systems NetFlow Services Export Version 9 (October 2004)
14. Claise, B.: RFC 5101 Specification of the IP Flow Information Export (IPFIX) Protocol for the Exchange of IP Traffic Flow Information (January 2008)
15. Antoniades, D., Polychronakis, M., Papadogiannakis, A., Trimintzios, P., Ubik, S., Smotlacha, V., Oslebo, A., Markatos, E.P.: LOBSTER: A European Platform for Passive Network Traffic Monitoring. In: Proc. 4th International Conference on Testbeds and Research Infrastructures for the Development of Networks & Communities (TRIDENTCOM), Innsbruck, Austria (March 2008)
16. Polychronakis, M., Anagnostakis, K.G., Markatos, E.P.: Arne Oslebo: Design of an Application Programming Interface for IP Network Monitoring. In: Proc. 9th IEEE/IFIP Network Operations and Management Symposium (NOMS), Seoul, Korea, April 19-23, pp. 19–23 (2004)
17. Trimintzios, P., Polychronakis, M., Papadogiannakis, A., Foukarakis, M., Markatos, E.P., Oslebo, A.: DiMAPI: An Application Programming Interface for Distributed Network Monitoring. In: Proc. 10th IEEE/IFIP Network Operations and Management Symposium (NOMS), Vancouver, Canada (April 2006)
18. The SCAMPI Project, `http://www.ist-scampi.org/`
19. `ftp://ftp.cordis.europa.eu/pub/fp7/ict/docs/future-networks/projects-moment-factsheet_en.pdf`
20. TopHat, `http://www.top-hat.info`
21. Augustin, B., Friedman, T., Teixeira, R.: Measuring multipath routing in the internet. IEEE/ACM Transactions on Networking (TON) 19(3), 830–840 (2011)
22. López de Vergara, J.E., Aracil, J., Martínez, J., Salvador, A., Hernández, J.A.: Application of ontologies for the integration of network monitoring platforms. In: Proc. 1st European Workshop on Mechanisms for Mastering Future Internet, Salzburg, Austria, July 10-11 (2008)
23. Zurawski, J., Swany, M., Gunter, D.: A Scalable Framework for Representation and Exchange of Network Measurements. In: Proc. 2nd International IEEE/Create-Net Conference on Testbeds and Research Infrastructures for the Development of Networks and Communities, Tridentcom 2006, Barcelona, Spain (2006)
24. Hanemann, A., Boote, J.W., Boyd, E.L., Durand, J., Kudarimoti, L., Łapacz, 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)
25. Gao, S., Sperberg-McQueen, C.M., Thompson, H.S.: W3C XML Schema Definition Language (XSD) 1.1 Part 1: Structures. W3C Recommendation (April 5, 2012)
26. López de Vergara, J.E., Villagrá, V.A., Guerrero, A., Berrocal, J.: Ontology-based network management: study cases and lessons learned. Journal of Network and Systems Management 17(3), 234–254 (2009) ISSN 1064-7570
27. Gruber, T.R.: A Translation Approach to Portable Ontology Specifications. Knowledge Acquisition 5(2), 199–220 (1993)
28. ETSI ISG MOI: Measurement Ontology for IP traffic (MOI); Report on information models for IP traffic measurement. ETSI GS MOI 010 V1.1.1 (May 2010)
29. Salvador, A., López de Vergara, J.E., Tropea, G., Blefari-Melazzi, N., Ferreiro, Á., Katsu, Á.: A Semantically Distributed Approach to Map IP Traffic Measurements to a Standardized Ontology. International Journal of Computer Networks & Communications (IJCNC) 2(1), 13–31 (2010) ISSN 0975-2293
30. ETSI ISG MOI: Measurement Ontology for IP traffic (MOI); IP traffic measurement ontologies architecture. ETSI draft DGS/MOI-0003 (July 2012)

31. Nasa: Units Ontology, http://sweet.jpl.nasa.gov/ontology/units.owl
32. López de Vergara, J.E., Villagrá, V.A.: Julio Berrocal: An ontology-based method to merge and map management information models. In: Proc. HP Openview University Association Tenth Plenary Workshop, Geneva, Switzerland (July 2003)
33. Bizer, C., Cyganiak, R.: D2R server publishing relational databases on the semantic web. In: Proc. 5th International Semantic Web Conference (November 2006)
34. de Vergara, J.E.L., Acero, V., Poyato, M., Aracil, J.: A semantic interface for OpenLab network measurement infrastructures. In: Korakis, T., Zink, M., Ott, M. (eds.) TridentCom 2012. LNICST, vol. 44, pp. 406–407. Springer, Heidelberg (2012)
35. OMF, http://oml.mytestbed.net/projects/oml/wiki

A Monitoring Framework for Federated Virtualized Infrastructures

József Stéger[1], Sándor Laki[2], and Péter Mátray[2]

[1] Department of Physics of Complex Systems
Eötvös Loránd University, Hungary
[2] Department of Information Systems
Eötvös Loránd University, Hungary
{steger,laki,matray}@complex.elte.hu

Abstract. Monitoring and measurement is a fundamental building block for developing and testing new protocols, routing algorithms and networked applications. In a federated virtualized testbed they allow other service components and testbed-users to follow the current state of the network, and on the other hand they enable intelligent automatic decision-making, e.g. during the embedding of a virtual topology. However, it is not a trivial task to enable federated monitoring functionalities due to the cross-domain nature of the system. The heterogeneity of the federated networks (including network elements and monitoring tools) pose a major challenge. In this chapter we present a framework that tackles some of the most important related problems. We also introduce a specific ontology to describe monitoring and network measurement tasks. This semantic approach enables the flexible integration of a wide range of monitoring tools, metrics and databases. Our Monitoring Framework was created within the NOVI FP7 STREP project which federates two major virtualized testbeds, PlanetLab and Federica.

Keywords: measurement and monitoring, virtualized testbeds, federation, monitoring ontology.

1 Introduction

Experimental facilities have gone through a significant evolution during the past several years: on the one hand, previously independent testbeds are federated under one “supersystem”, and on the other hand virtualization became an important feature. Consequently, monitoring capabilities needs to be reworked in state of the art testbeds to allow other service components and testbed-users to follow the current state of the network, and to enable intelligent, automatized decision-making of the system. For example in virtualized environments, a requested virtual topology could efficiently be mapped to the available physical resources with satisfying certain constraints (e.g. bandwidth limits, free memory or cpu load). However, it is not a trivial task to enable monitoring and measurement in the federated system, since different testbeds consisting of heterogeneous network elements provide basically different possibilities to monitor and access their components. Moreover, cross-domain monitoring problems (e.g. when measuring virtual links between two testbeds) need to be solved as well. In virtualized environments, monitoring and measurement capabilities are also beneficial as they can help testbed users to follow the changes of the network state in their slice while they are running their experiments.

To tackle these challenges, we have built a Monitoring Framework (MF) within the the NOVI FP7 STREP project [1]. NOVI provides control, management and monitoring planes to allow the federation of various virtualized testbeds, consisting of heterogeneous resources. NOVI algorithms and services enable experimenters to request, reserve, use and update a great deal of virtualized resources in the federation, tailored to their needs. To fulfill all the monitoring criteria and facilitate intelligent system functionalities, our framework is designed to provide a generic framework to carry out active or passive measurements on both physical and virtual resources. We have introduced a specific monitoring ontology to describe monitoring and network measurement tasks. With the use of the ontology, a wide range of monitoring tools, metrics and databases can be integrated in the framework. An experimenter only needs to know the metrics to be measured (e.g. bandwidth, rtt, path) independently of the tools installed in the different testbeds, and the monitoring service will ensure the proper mapping between the requested metrics and the available tools automatically.

Related to our work, similar efforts exist within the GENI project [2]. They implement a Scalable Sensing Service (S^3 Monitor) to provide a scalable and extensible monitoring service for the federated experimental facility [3]. A key feature of S^3 Monitor is a flexible design that allows easy “plug in” of new network measurement tools. The perfSonar infrastructure is also related to our work [4]. They provide network performance monitoring, in order to facilitate end-to-end performance problems on paths crossing several networks. PerfSonar contains a set of services delivering performance measurements in a federated environment. These services act as an intermediate layer, between the performance measurement tools and the diagnostic or visualization applications.

1.1 NOVI

The NOVI Project [1], contributes to creating a blueprint of Future Internet federated infrastructures. An important aspect of this blueprint is the development of common information and data models that support the federation of the two main NOVI platforms, Federica and PlanetLab. The intention is to make these models generic and usable by other infrastructures that could at a later stage be included in the NOVI federation. We do this by designing and prototyping a service portfolio based on combined virtualized facilities from the Federica and PlanetLab virtualized infrastructures. In this scope, NOVI tackles the problem of defining an information model and its related data models to capture the concepts and semantics of resources and services offered by several virtualization platforms, focusing primarily on the Federica and the

PlanetLab platforms. Both the information and the data models are intended to be generic so that they can be used by other infrastructures that could participate in the federation.

NOVI identified a number of service components to implement control and management functionalities. These components are put on top of each platform of the federation and they work together to make reservation of virtual resources in the federated environment possible. We mention here only those components, which are related to the monitoring service component. NOVI offers its users freedom to set constraint features of their slice of virtual resources. In order to find the best set of physical resources meeting those constraints Resource Information Service will call the monitoring framework to get the most recent state information. NOVI also offers its users a monitored slice complying with certain customizable requirements. In the case a criteria is not valid any more, the MF sends a signal of failure to the Policy Service component to take actions.

2 Monitoring Tools

A wide range of monitoring tools have emerged in the past decade that enable experimenters to monitor various network characteristics and to follow their changes. Since these tools were developed for different purposes, they only focus on a subset of the available network characteristics and provide their users with basically different capabilities and the way of access also vary (command line, SOAP, XML-RPC, etc.). For example, some tools give free hand to its users to parameterize and perform active measurements on demand, while others do not provide any customization feature, merely return measurement data. In this heterogeneous setting, our MF aims at offering a common and simple access to other system components as well as testbed users to perform their monitoring tasks. To narrow down on the set of tools MF initially supports we had taken into account the following. On the one hand MF needs to handle tools, which are very common and widely used like command line tools (ping, traceroute, iperf, etc.). On the other hand more complex measurement frameworks demonstrate the generality of the monitoring model, detailed in Section 3. Besides command line tools, currently MF supports SONoMA [5], Packet Tracking [6], HADES [7]. We chose these complex monitoring and measurement frameworks because partners of the NOVI project have also contributed to their development. As it will be shown in Section 4, our monitoring framework can easily be extended with new measurement and monitoring tools.

2.1 SONoMA

SONoMA [5] (Service Oriented Network Measurement Architecture) is a Web Service based network measurement platform that is scalable, adaptable and open for scientists and other network developers. It realizes a common and extensible active network measurement framework that aims at decreasing the required time and efforts of network experiment implementation. Its services can be accessed via a standardized SOAP-based web services interface, not constraining the used programming language, so that researchers can use their favorite programming environment to describe measurement scenarios, not wasting time to discover exotic scripting languages. The SONoMA architecture consists of a central management system representing a gateway for SONoMA users to submit their tasks and measurement agents deployed on remote machines from where the measurements can be carried out. The submitted experiments are performed on a proper set of measurement agents in a completely distributed manner. In addition, the results are not only forwarded back to the user, but they are automatically stored in a public data repository, called Network Measurement Virtual Observatory (VO) [8] as well. In contrast to prior approaches, researchers do not have to write separate scripts to check the status of the measurement nodes, spread the probing tasks among the nodes, then collect the results and finally post process the data, since SONoMA can solve all these issues automatically. In NOVI, SONoMA measurement agents are deployed on PlanetLab and certain Federica nodes as well.

2.2 Packet Tracking

The Packet Tracking (PT) tool [6] is a multi-hop packet tracking architecture that provides the experimenter with information on the paths that packets take throughout the network. Besides that, it carries other hop-by-hop metrics like delay and loss as well. The knowledge of the actual network path can be used, for example, for the validation and analysis of different routing algorithms or multicast protocols. In addition, this tool enables to measure the extent of cross-traffic and its influence on the experimenter's traffic. In case of active measurements, it is also crucial to reduce the traffic overhead caused by the probes. To this end, PT tool uses a hash-based packet selection technique that ensures a consistent selection throughout the network while maintaining statistically desired features of the sample. In NOVI, PT tool is currently deployed on PlanetLab nodes only.

2.3 HADES

HADES (Hades Active Delay Evaluation System) [7] is a network monitoring tool providing performance measurements following the IETF approach of RFC3393. It provides its users with different performance metrics such as one-way delay, delay variations and packet loss. HADES was designed for multi-domain delay evaluations with continuous measurements and , in contrast to SONoMA, was not intended for troubleshooting or on-demand measurements. It incorporates active measurements producing probe traffic which delivers the above high resolution network performance diagnostics. HADES measurement agents are generating bursts of UDP probes continuously throughout the day. The packets are sent with gaps of 10 ms to avoid any waiting times at the network card interfaces and each of them contains a unique sequence number and a GPS synchronized timestamp which can be used to calculate the performance metrics at the receiver side. The experimenter can access the measured data via a web service interface, but has no direct influence on the measurement process. In the NOVI federation, HADES tool is deployed on Federica nodes only because of the need of precise timestamping.

2.4 Other Tools

In contrast to the previous examples, the majority of the available monitoring tools do not provide their users with such advanced interfaces and cannot be accessed remotely. They can simply be executed from command line on the given machines. Since these tools are widely used for performing delay, topology or bandwidth measurements, the Monitoring Framework offers an opportunity to execute command line scripts on ordinary remote machines that can be accessed via ssh connection. For example, in our MF it is simple and straightforward to trigger command line tools like `ping`, `traceroute`, `iperf`. In addition, we implemented an extensible driver system to support different interfaces offered by different tools.

3 Information Model

To design a highly flexible and automatized system for federation, it is necessary to introduce a certain level of context-awareness in the components. However, the construction of context-aware algorithms for resource discovery and resource selection is far from being trivial. Resource utilization and constraints limit the set of resources that can be effectively selected. This information can be retrieved from monitoring systems running in the various federated platforms. The support for the monitoring of a variety of resource types is also a challenging task, since we need to comply with both physical and virtual resources that can be used either collectively or individually. There are already a wide spectrum of monitoring tools and techniques in use. Some of them overlap in functionality, some operate only on a restricted subset of the available resources. Thus, we develop a framework for monitoring purposes, which is highly configurable and targets to cover a very versatile set of possible monitoring and measurement tools. To achieve that, a semantic description for the metrics and the underlying tools are constructed along with describing parametrization to remain flexible and practical in use.

In the NOVI project the Information Model (IM) [9] plays key role. Control, management and monitoring functionalities are described by using Web Ontology Language (OWL) representations of the model. In Fig. 1. we sketch the three basic modules of the IM, the resource model, the policy model and the monitoring model. The *Resource Ontology* defines physical and virtual components the control, management and montiring framework can operate on and their relationship. Among them, for example, a computational resource having memory, processor and storage, a link having interfaces bound to computational resources or switching components. Resource model enables us to deal with even higher level elements like a path comprised of an ordered set of links and the topology

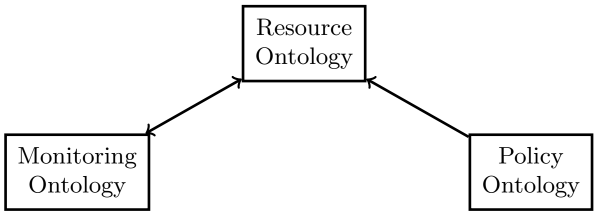


Fig. 1. The high level view of the dependencies between the building blocks of the NOVI information model. The *Resource Ontology* and the *Monitoring Ontology* are closely bound together (see also Fig. 2.) whereas *Policy Ontology* depends only on the *Resource Ontology.*

which is the overall "wired-together" set of virtual resources. The resource model is also describes the binding of virtual components to the physical substrate. The monitoring model and the resource model are highly related to each other. The resource model imports statements about the units while the *Monitoring Ontology* uses the resource concept declared in the *Resource Ontology.* The policy model is out of the scope of this document.

3.1 Monitoring Ontology

The Monitoring Service exploits various methods to retrieve monitoring information. Monitoring data describes the state of the managed resources. It can come from different repositories or it might be generated on the fly using the available monitoring tools. To enable the uniform representation of monitoring functionalities in the federated environment the monitoring model extends the resource model to incorporate the relevant concepts. The monitoring model also follows a modular design, see in Fig. 2. The idea behind defining smaller pieces rather than a huge ontology, comes from two facts. Firstly this model aims to be reusable by new future applications, secondly maintenance and development remains easier for each block. Thus, concepts closer in their meaning belong to the same small document component, and applications may import only pieces relevant to them.

For its operation the Monitoring Service requires formal descriptions of three different groups of concepts, their relationships and properties: *i*) *Resources*: to describe the resources to be monitored. These concepts are defined in the resource model as described in the previous section. *ii*) *Monitoring Tools*: to describe monitoring tools and their parametrization. The main goal of this part of the model is to enable the Monitoring Service to translate incoming monitoring queries to platform-specific commands. *iii*) *Monitored Features and Monitoring Data*: to describe the monitored properties. The concepts of dimension and unit makes it easy to transform between different representation of the same data and also to infer possible relationships between metrics. This part of the model helps the unified, platform-independent and unit-aware handling of monitoring results. In the following we present the main elements of the Monitoring Ontology.

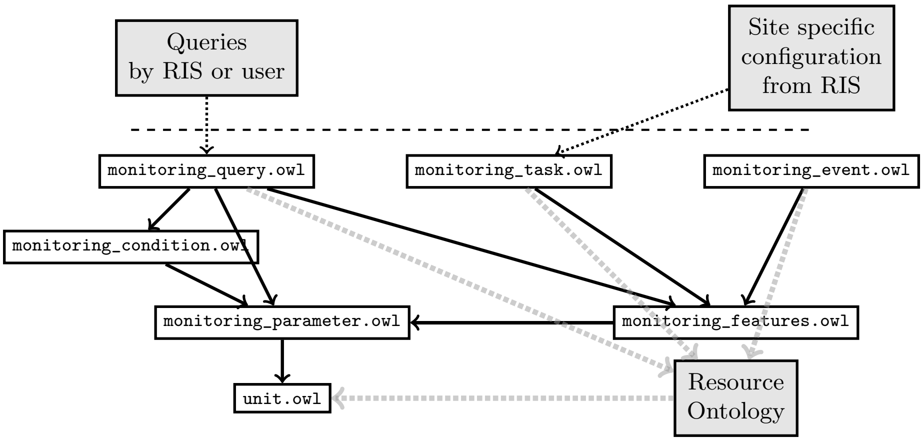


Fig. 2. The monitoring model follows a modular design. The figure shows the dependency between the component modules. Items with a shaded background are components external to the monitoring model.

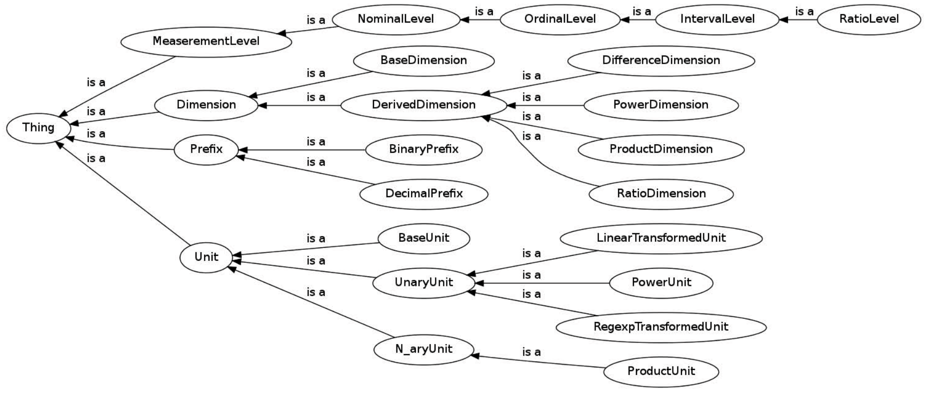


Fig. 3. The hierarchy of classes declared in the unit model

Unit Description. The fundamental concepts of the monitoring model are laid down in the unit ontology document, see Fig. 3. The notion of the MeasurementLevel class comes from the statistics. The "levels of measurement" or scales of measure are expressions that typically refer to the theory of scale types developed by the psychologist Stevens. In [10] all measurement conducted in science are claimed one of the four different types of scales that he called "nominal", "ordinal", "interval" and "ratio". Among all, these scales tell what operations make sense to apply on the samples, like ordering samples, taking their difference or blowing them by a factor. Dimensions are of one well defined level.

For example if somebody is interested in the source port of IP packets captured in an interface card, this particular number is of nominal level, revealing the fact that ordering does not make sense for this type of data, whereas timestamps of reception, which are of interval level, both ordering and differencing makes sense.

In this block we define dimensions, units and unit prefixes. Not only the hierarchy of these classes are defined in the model but their relationships too. In the simplest case these are "is-a" class relationships, but in our semantic model we often describe more complex connections between classes (e.g. describing relationships between monitoring tools and code snippets controlling them). Due to space constraints, in the figures of this chapter we depict only is-a relationships. For a complete view on the ontology, please refer [9]. The possible transformation path between compatible units are stated here using linear transformation, taking a product, raising to a power or applying transformations written in forms of regular expressions. Based on the unit model one can deduce trivial things, for example, the time interval can be derived by taking the difference of to points in time, or similarly, the rate of information flow is the ratio of network volume difference and and the time interval between the two samples.

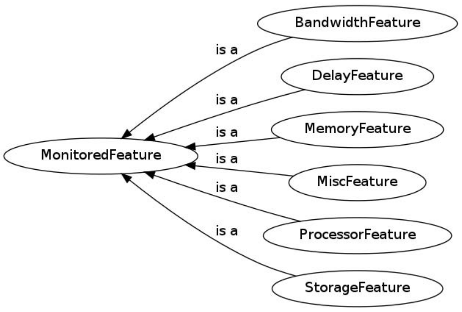


Fig. 4. The hierarchy of classes declared in the feature model

Metrics. Monitored metrics are collected and described in the feature ontology document. The commonly used metrics form classes, as in Fig. 4, the names of the classes are self-explanatory. Individuals of the MemoryFeature class specify the state of a physical or virtual memory resource (total, free, used, swapped). Analogously, individuals of the StorageFeaturee class refer to the state of a storage resource (free, used, quota). Items of the ProcessorFeature refer to the state of a CPU (frequency, cores, load). Descendants of the DelayFeature describe one-way delay or round-trip delay over a network link or path. The rest of the interesting features can be found in the MiscFeature class, like uptime of a resource, or packet loss over a path.

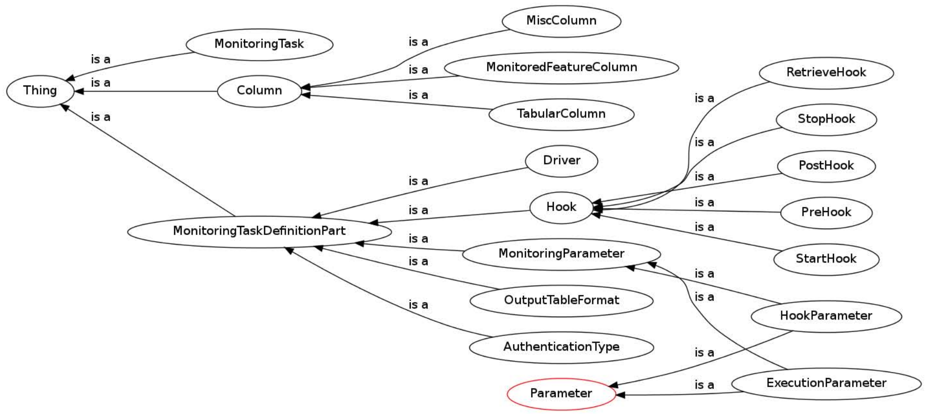


Fig. 5. The hierarchy of classes declared in the task model

Measurement Description. It is essential to bind together the monitoring tools operable in the resources or their results stored in repositories and the metrics a user or a service look for. This middle level binding knowledge is presented in the task module of the monitoring model, which enables the abstraction of monitoring measurements. The key idea is to hide as much details from data consumer as it makes sense. Doing so allows for handling active and passive measurement tools in a common manner. The three main classes in this module are *i*) the *MonitoringTaskDefinitionPart*, holding the atomic information, which are necessary to build the individuals of *ii*) the *MonitoringTask*, whose each individual represents a workflow how to measure and/or retrieve a given metric, and *iii*) the *Column*, which relate to the pure data holding the results of a measurement in a uniform way.

An instance of the Task class represents a measurement session, that is a set of operations including the initialization, execution and termination of measurement processes, and the gathering of measurement results. To generalize the access to the monitoring tools, the task will have a unique driver. The driver represents and in MF it implements the protocol to communicate with a specific measurement tool. Currently, four drivers are in the model and implemented: SshDriver, SOAPDriver, XMLRPCDriver and DBDriver. The former three typically represent connection to active tools generating data in the fly, whereas the last driver represents the linkage to a repository. In order to properly set up the communication channel with the tools that require authentication task has relation to an AuthenticationType class instance. The tools we have access to in NOVI required the definition of UsernamePassword and UsernameRSAKey types.

A monitoring task also bears information of the execution plan for a given network measurement. To this end, it consists of five predefined procedures to be implemented:

Table 1. Monitoring task examples. Hooks implement various procedures of a task. From this information MF dynamically builds and executes the workflow. Depending on which hooks are given, it is possible to handle short term measurements that yield result within seconds (synch) and also long running background tasks (asynch). There are various tools to measure RTT actively. Here we show which hooks need to be present to use either the `ping` command line application or SONoMA services in the MF.

Procedure	CLI-synch	CLI-asynch	SOAP-asynch(SONoMA)
Prehook	Undefined	Undefined	requestsession(. . .)
Starthoop	Undefined	ssh A ping B > file &	longPing(sessionID, A, B, . . .)
Stophook	Undefined	ssh A killall ping	stopProcedure(processID)
Retrievehook	ssh A ping -c N B	ssh A cat file	getData(processID)
Posthook	Undefined	ssh A rm file	closeSession(sessionID, . . .)

1. **prehook** is optional and details what to do before the retrieval of monitoring logs, for example how to initialize a monitoring session.
2. **starthook** is optional and describes how to launch an asynchronous monitoring measurement.
3. **retrievehook** specifies how to get monitoring logs. For example, in the case of a synchronous monitoring measurement it defines how to execute or perform the measurement. On the other hand, for an asynchronous measurement, which has already launched this procedure, it retrieves any available measurement data.
4. **stophook** is optional and describes how to terminate an asynchronous monitoring measurement.
5. **posthook** is optional and details what to do to clean up after data retrieval. For example, how to close an open monitoring session.

One can observe that using these procedures, the MF can easily define both synchronous and asynchronous measurements for short and long term monitoring scenarios. Table 1 presents three different measurement scenarios two for executing command line tools (CLI-synch, CLI-asynch) and one for calling a SONoMA measurement (SONoMA) to get the same information, i.e. the round trip time between two nodes.

4 Monitoring Framework

An ideal monitoring framework enables testbed users, administrators as well as other system components to retrieve valuable information on the current state of the available resources including hosts, slices, links or network paths. In a federated environment such as NOVI, the situation is more complex, since MF also has to support the monitoring of cross-testbed resources such as network links or paths interconnecting two or more testbeds, and must be able to exchange information between MF instances deployed on different testbeds. Besides fulfilling the above requirements, our framework also opens the door to define network state conditions that must be satisfied during the experiment, and if it is constrained, MF will report the event to the user and/or a predefined system component to take care of it.

MF collects information about specific resources and measurement metrics. In a virtualized testbed, monitoring can be performed on either *slivers* (virtualized resources) or *hosts* (physical devices). The corresponding tasks are called *slice monitoring* and *substrate monitoring*, respectively. In addition, it is possible to obtain passive monitoring information from resources directly or from testbed repositories as well, and active monitoring information injecting probe packets into the network. Note that our MF does not distinguish between active and passive monitoring. As we have already mentioned in Section 3, in terms of monitoring, a network measurement can be represented by a monitoring task that represents a general workflow whose execution realizes the measurement on the remote resources and carries the required data. The task concept is independent from whether the measurement is an active or a passive one and enables us to integrate almost every kind of measurement types and tools into MF with ease.

In regular workflows, the MF supports three main tasks: prior to resource allocation substrate monitoring tasks may be performed to ensure that the resource request constraints can be satisfied. Second, after resource reservation MF provides both the NOVI system itself and the NOVI users with means to carry out slice monitoring for diagnostic and watchdog purposes, i.e. to check the current status of given set of resources. Finally, MF provides built-in functions helping the testbed users to perform user-level-experiments.

4.1 NOVI Monitoring Plane Design

As we have already mentioned, our Monitoring Framework has been prototyped in the NOVI project. In NOVI federation, there are NOVI instances on the top of each federated testbed. These instances communicate with each other in a peer to peer fashion. Our MF prototype also takes place in each NOVI instances, offering services for other components and testbed users as well.

The architecture of the Monitoring Plane can be seen in Figure 6. One can observe that MF requires services from two other components: Resource Information Service (RIS) and Policy Service. Resource Information Service serves as a data repository storing all the necessary information about the local testbed that is essential to design and carry out network measurements, while Policy Service is used to authorize and authenticate MF to access resources. MF provides two different interfaces: the Service Interface for submitting monitoring tasks and the Neighbor Monitoring Interface for inter-testbed communication (e.g. to establish cross-testbed measurements that require the cooperation of tools deployed on different testbeds). The Service Interface provides access for other components and NOVI users with different privileges. It enables users to define,control and remove slice monitoring tasks to examine how their experiments affects on different network characteristics in the slices they own. However, other system components can initiate both slice and substrate monitoring tasks on the virtual and physical resources as well. In both cases, MF can carry out

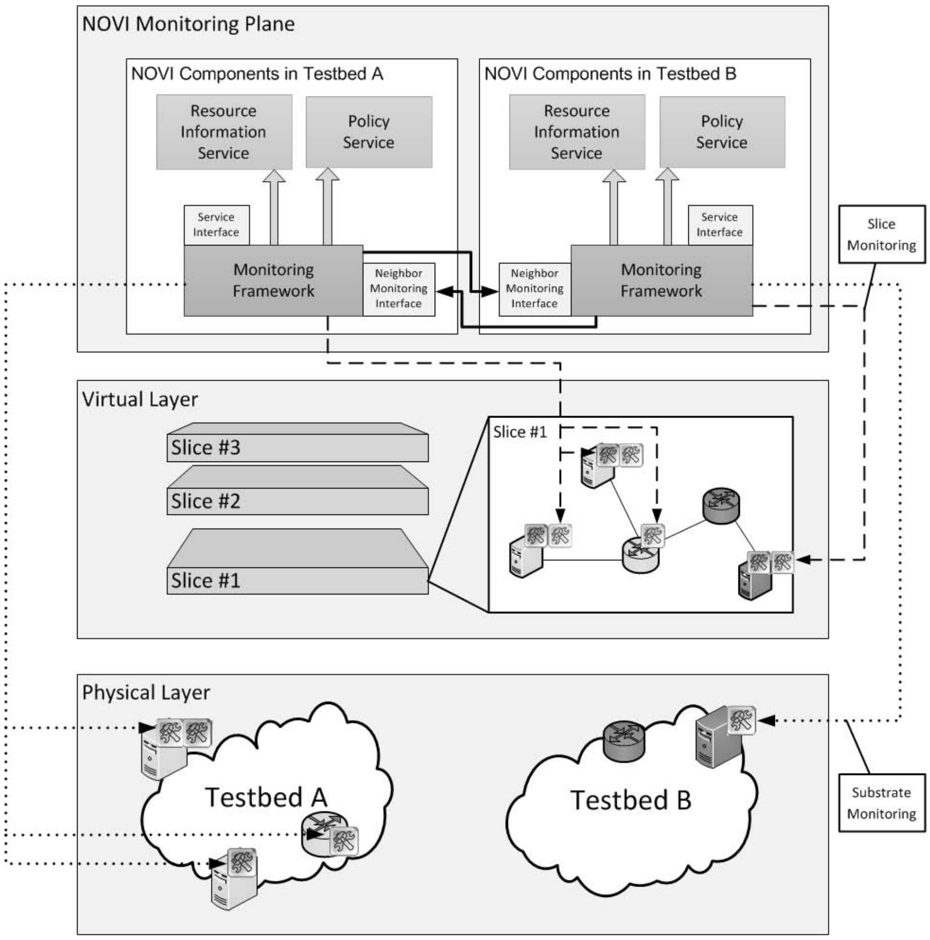


Fig. 6. The architectural overview of NOVI Monitoring Plane. An example federation consisting of Testbed A and Testbed B with the necessary local NOVI instances is illustrated on this figure. Each MF instance is only responsible for its own testbed to instrument monitoring tools and perform network measurements on the available resources. In case of cross-testbed measurements, the local instances communicate with each other via their neighbor monitoring interfaces. The dashed lines illustrate how slice monitoring tools are controlled by the local instance of MF, while the substrate monitoring calls are indicated by dotted ones. Resource Information Service (RIS) serves as a data repository storing all the necessary information on the local testbeds including the available resources and the measurement/monitoring tools installed on them, while Policy Service is responsible for authorization and authentication.

network measurements in a federated environment across different testbeds, like monitoring characteristics (e.g. available bandwidth, round trip delays, etc.) of network paths running from one testbed to another.

One can also observe that each MF instance is responsible for controlling the tools deployed on its own local testbed only. These tools can be installed on the physical and virtual layers as well. Those that are running on the physical resources serve as back-ends for substrate monitoring, providing other system components with useful information (e.g. to map virtual topology to physical resources, etc.). The other set of monitoring tools is used for slice monitoring. These tools are deployed on the virtual resources of a given slice. The installation of these basic monitoring elements can be done by other testbed components or by a user who owns the given slice. In both cases, RIS keeps track which monitoring tool is available on the different resources and provides MF with a well defined interface to get this information.

4.2 GUI

In order to help users to control their monitoring tasks in their virtual testbed we have developed a slice monitoring graphical front end. Resources fall basically to two separate sets: *i*) nodes and *ii*) links or paths. Therefore in the monitoring application user can select a node from his virtual topology or a link between nodes. He also can define paths between nodes that are more than a hop distance from each other.

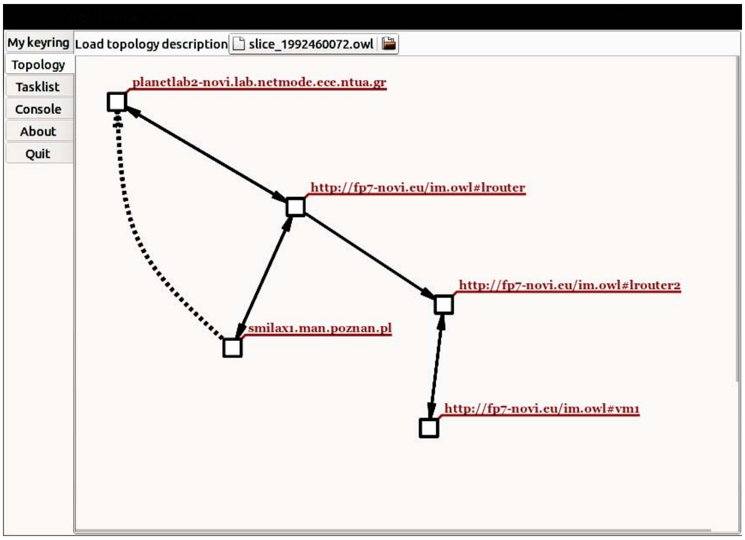
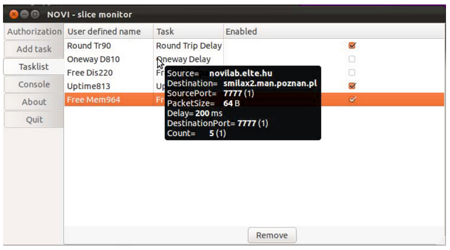

My keyring
Load topology description
slice_1992460072.owl
Topology
Tasklist
Console
About
Quit
planetlab2-novi.lab.netmode.ece.ntua.gr
http://fp7-novi.eu/im.owl#lrouter
http://fp7-novi.eu/im.owl#lrouter2
smilax1.man.poznan.pl
http://fp7-novi.eu/im.owl#vm1

Fig. 7. The slice monitoring GUI makes it easy to select the right resource from the users virtual topology

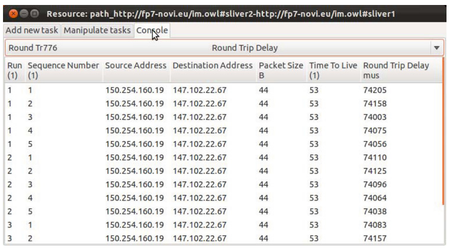


NOVI - slice monitor

User defined name	Task	Enabled
Round Tr90	Round Trip Delay	✓
Oneway D810	Oneway Delay	☐
Free Dis220	Fr	☐
Uptime813	Up	✓
Free Mem964	Fr	✓

Source= novilab.elte.hu
Destination= smilax2.man.poznan.pl
SourcePort= 7777 (1)
PacketSize= 64 B
Delay= 200 ms
DestinationPort= 7777 (1)
Count= 5 (1)

Fig. 8. The application renders the list of monitoring tasks. User can look up parametrization, enable, disable or remove a task by a click.



Resource: path_http://fp7-novi.eu/im.owl#sliver2-http://fp7-novi.eu/im.owl#sliver1

Round Tr776 Round Trip Delay

Run (1)	Sequence Number (1)	Source Address	Destination Address	Packet Size B	Time To Live (1)	Round Trip Delay mus
1	1	150.254.160.19	147.102.22.67	44	53	74205
1	2	150.254.160.19	147.102.22.67	44	53	74158
1	3	150.254.160.19	147.102.22.67	44	53	74003
1	4	150.254.160.19	147.102.22.67	44	53	74075
1	5	150.254.160.19	147.102.22.67	44	53	74056
2	1	150.254.160.19	147.102.22.67	44	53	74110
2	2	150.254.160.19	147.102.22.67	44	53	74125
2	3	150.254.160.19	147.102.22.67	44	53	74096
2	4	150.254.160.19	147.102.22.67	44	53	74064
2	5	150.254.160.19	147.102.22.67	44	53	74038
3	1	150.254.160.19	147.102.22.67	44	53	74083
3	2	150.254.160.19	147.102.22.67	44	53	74157

Fig. 9. Via the console the user can retrieve measurement data in a form of a table. Columns render metrics user is interested indicating their units.

Suppose, a user wants to keep track of the RTT over a path between two PlanetLab nodes routed through a Federica router during experimenting in his virtual topology. In the GUI (see Fig. 7) user can conveniently define a path between `smilax1.man.poznan.pl` and `planetlab2-novi.lab.netmode.ece.ntua.gr`. After the selection of a resource (a node, a link or a path) a new window pops up with further instruction. In this window metrics applicable to the given type of resource are enlisted. In our example a path was defined, therefore only the selection of metrics like bandwidth, one-way delay, packet loss, RTT, etc. are allowed. By selecting RTT metric from the drop down menu, the list of parameters are presented to the user, who in turn can set them unit wise or simply neglect them unsetting the check box besides the named parameter. Our user is now ready to add this new monitoring task. Behind the scenes, MF selects the tool considering what are available in the platform, and based on the parameters the user preset and looking at the credentials of the user. In this example, both SONoMA and the command line ping over SSH could yield the RTTs.

Measurements of selected metrics, associated with monitoring tasks, can be managed individually, independently from other monitoring tasks. Monitoring tasks can be started, stopped or removed from the task list (see Fig. 8). The results of the measurements can be read from the console of the GUI (see Fig. 9), or uploaded to a database within the Resource Information Service, or trigger event-condition-action policies in the Policy Service.

5 Experiments

As we mentioned before, the MF provides other service components as well as testbed users with information on the current state of the physical or virtualized network. In this section, we show both substrate and slice monitoring examples to demonstrate our monitoring framework.

Our first experiment illustrates a substrate monitoring scenario when, for example, a resource allocation algorithm is requesting for additional dynamic information of the underlying physical network to make more optimal decisions. In this example, round trip delay measurements between the physical nodes are performed. The obtained delay data can carry information on the current state of physical links and paths interconnecting the underlying nodes that can be necessary for assembling the required virtual topology.

In the second case, an experimenter has already got a slice with the requested virtual topology depicted in Fig. 10. One can observe that the slice consists of both Federica and PlanetLab resources, and, for example, the network traffic between the two PlanetLab nodes are routed through two virtual routers of Federica. The user can install and execute his/her applications on the virtual hosts, and define the routing in the virtual network. In this case, MF enables the experimenter to follow the effects of his/her application on the network conditions by monitoring different network characteristics (delay, jitter, available bandwidth, etc.).

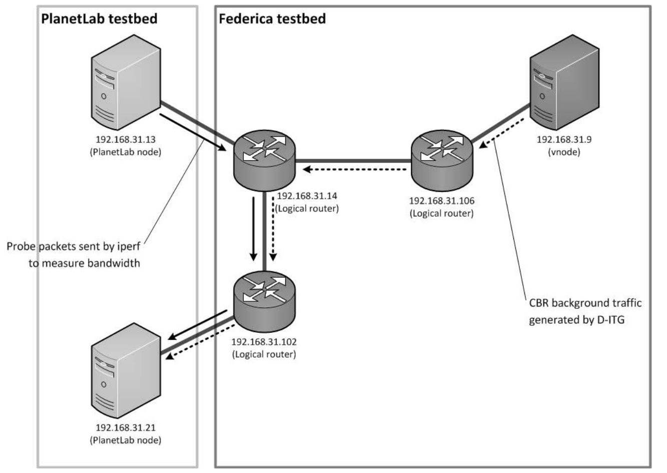


Fig. 10. The virtual topology of our slice. It consists of both PlanetLab and Federica resources as well, indicated by light and dark gray colors, respectively. All the nodes belong to the same logical subnetwork (192.168.31.0/255) with the proper routing settings. The node 192.168.31.13 is sending probe packets to the other PlanetLab node using the iperf tool to continuously monitor the available bandwidth along the virtual path indicated by black thick arrows. At the Federica node 192.168.31.9, D-ITG traffic generator is used to inject artificial background traffic into the virtual network. This background traffic follows the path indicated by dashed arrows.

5.1 Delay Monitoring between Substrate Elements

In this experiment we show how our framework can provide substrate monitoring information for other NOVI components, e.g. to map virtual topologies to physical ones satisfying the predefined requirements/initial conditions of the

Table 2. The list of nodes taking part in the substrate monitoring experiment. Federica core routers (PoPs) are marked A–D and 1–9 stand for PlanetLab computational resources.

Node	Name or IP address (location)
A	*PSNC-PoP (Poznan, PL)*
B	*DFN-PoP (Erlangen, DE)*
C	*CESNET-PoP (Prague, CZ)*
D	*GARR-PoP (Milan, IT)*

Node	Name or IP address (location)
1	188.1.240.194 (Erlangen, DE)
2	157.181.175.243 (Budapest, HU)
3	147.102.22.66 (Athens, GR)
4	147.102.22.67 (Athens, GR)
5	150.254.160.19 (Poznan, PL)
6	150.254.160.20 (Poznan, PL)
7	150.254.160.21 (Poznan, PL)
8	150.254.160.22 (Poznan, PL)
9	150.254.160.23 (Poznan, PL)

experimenter. In this example, MF carries information on the initial conditions only, but it is also possible to define monitoring tasks to follow the running conditions of the network either by other NOVI components or by the user itself.

Let's consider the case when the requested virtual topology must consist of links having delays in a predefined time interval. To fulfill this requirement, the NOVI's resource allocation algorithm needs to call MF to measure RTT metric among the set of the potential nodes (listed in Table. 2). MF measures or retrieves this information in a best effort manner than. In case of federated testbeds, measurements are split between the MF instances of the testbeds taking part in the federation. In our case, PlanetLab sources are running SONoMA shortPing measurements to fetch this information, whereas Federica routers (core PoPs) use command line ping.

The results of the delay measurements are shown in Fig. 11. It can be seen that we will not fetch a full data set, certain measurement data are missing. In this case, Federica core PoPs have forbidden to inject ICMP packets into the public Internet, where our PlanetLab hosts are situated. This is why RTT measurements to PlanetLab nodes have failed to produce results. Nonetheless, this partial information is still useful for other NOVI components to draw decision on the resources to allocate.

5.2 Bandwidth Monitoring in a Slice

In our slice monitoring scenario, the user wants to test his bandwidth consuming application. The user decides to conveniently monitor few characteristics of his virtual testbed topology, namely the RTT, the traffic volume and the available bandwidth over the path, where his application is transmitting data.

The duration of the experiment is one minute. In Fig. 12. the evolution of the network volume and the measured bandwidth are shown. In order to be able to compare data, the traffic volume is transformed to date rate, derived for every .2 seconds. At $t_1 = 10\ s$ one application requesting to send at a bit-rate of 41.2 Mb/s was turned on. Later, at $t_2 = 20\ s$ another data stream is switched on by a new instance of the user application requesting for 58.9 Mb/s. Note, the total data transfer request is in the order of the line speed, as PlanetLab nodes are connected via 100 Mb/s links. At $t_3 = 35\ s$ and $t_4 = 45\ s$ applications are stopped.

Datapoint marked with circles in Fig. 12. represent the fluxes of the data packets inserted in the network by the two applications. It can be seen, during only a single application is on the node can send at the rate requested. However, when both applications are turned on, between t_2 and t_3, the limits of the virtualized resource is hit. The sending rate of the data packets heavily fluctuate.

At the receiver side, network volume is also monitored, and similarly the rate of packet reception is derived. This is the throughput and is marked by green dots in Fig. 12. It is apparent that heavy packet loss occurs when the path is overloaded. The throughput in this period dramatically increases, and the overall packet loss can be as high as $\approx 68\ \%$.

Additionally, bandwidth is also monitored in the slice. The bandwidth metric in this case is measured by the `iperf` tool. This information is also marked in the figure with in blue. One can observe the step-like changes at each interesting points in time: t_i.

In this experiment, besides the metrics related to the speed of data, the user is also interested in the delay statistics. So a slice monitoring task to measure RTT is also launched in the slice. In Fig. 13. the variation of RTT over time is shown. It can be seen, the jitter (related to the variance of RTT) significantly jumps between t_2 and t_3. As to lead our eyes the available bandwidth is also shown in this figure.

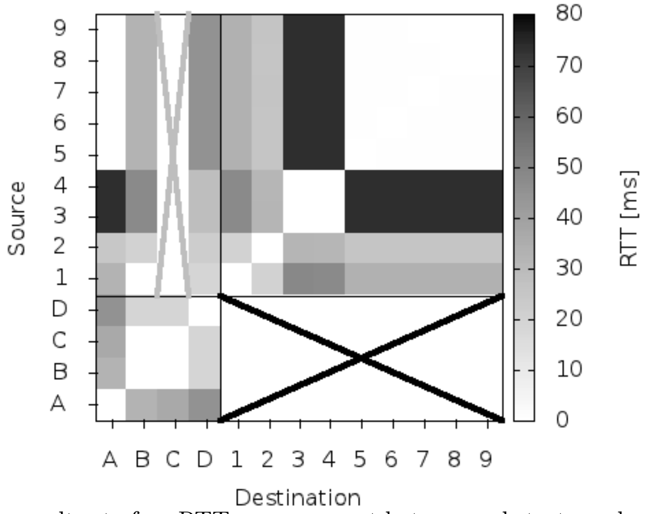


Fig. 11. The result set of an RTT measurement between substrate nodes (see Table. 2. for resource details). The darker shade of gray indicates longer roud trip delays. Data in a diagonal block belong to the same platform domain, while off diagonal elements are across domain information. In our environment Federica nodes were disallowed to probe the public network, missing data are crossed out in black. And the Prague node filters out ICMP traffic, marked in gray.

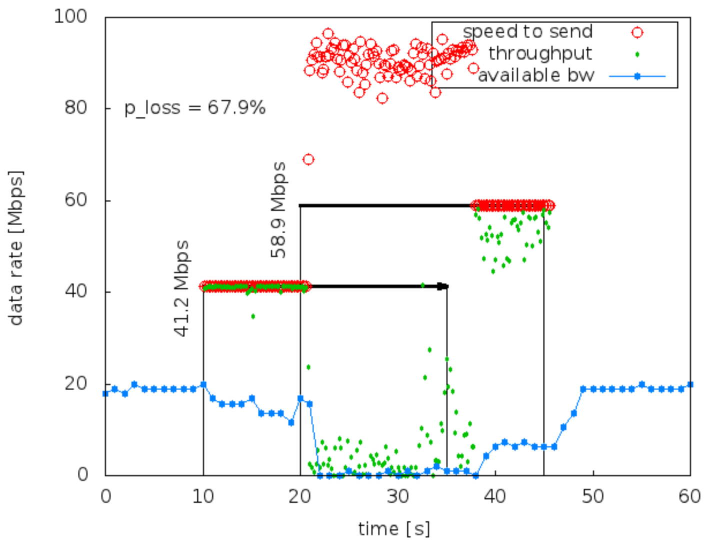


Fig. 12. A hypothetical user is testing his data eager application in a slice of virtual resources. He is keeping track of interesting metrics to see the effect caused by his application to the network. Network volume at both source and destination nodes are logged and also active bandwidth measurement is requested.

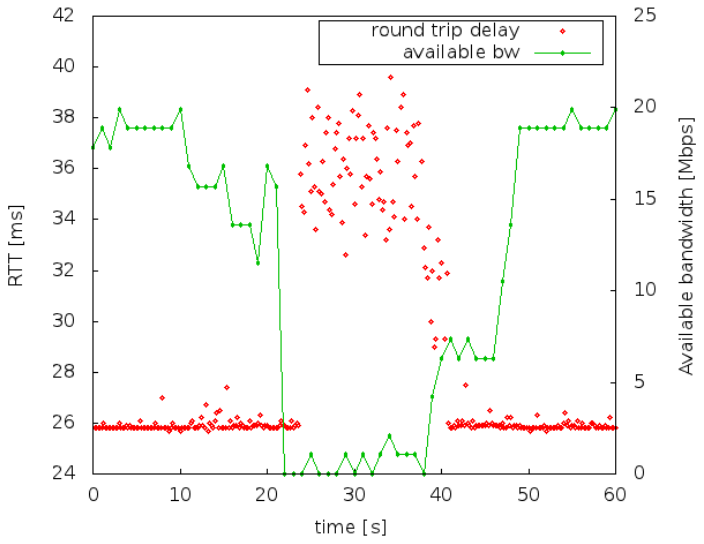


Fig. 13. In the slice the RTT is monitored across the link that carries the bulk of the traffic generated by the users application

6 Summary and Future Work

In this chapter, we have presented an extensible Monitoring Framework for federated virtualized testbeds whose prototype has already been deployed in the NOVI (Networking innovations Over Virtualized Infrastructures) federation. The proposed MF provides the testbed users as well as other system components with a wide range of monitoring capabilities. In contrast to stand-alone testbeds, in a federated and virtualized environment monitoring poses many challenges (e.g. cross-domain measurements, slice and substrate monitoring, heterogeneous tools, resources and way of access, etc.). Our MF aims at answering all these issues using a universal and extensible information model (IM). Since both the measurement workflows, the network characteristics and the resources are described in this common way, new tools and measurement types can be integrated into the system with ease, requiring the modification of the IM only. In NOVI federation, each testbed is of its own MF instance which is only responsible for controlling the local resources including the deployed tools. In case of cross-testbed experiments, MF instances can communicate with each other in a peer-to-peer fashion to instrument the available tools properly.

Our future plans include the integration of NOVI's policy service to make the resource monitoring and probing more controllable by introducing different policies for user groups or other system components. In case of active measurements, especially in a virtualized testbed, it is also a crucial task to reduce the traffic load generated by the probe packets as well as the interference among active probes. To this end, we are going to give a proper solution to synchronize and schedule different slice and substrate monitoring tasks so that the amount of probe traffic could be kept as low as possible. We are also working on the integration of the MF with other NOVI components (e.g. providing information for intelligent resource and topology mapping).

Acknowledgment. The authors thank the partial support of the EU FIRE NOVI project (Grant No. 257867), the EU FP7 OpenLab project (Grant No. 287581), the National Science Foundation OTKA 7779 and 80177, and the National Office for Research and Technology (NAP 2005/KCKHA005).

References

1. Lymberopoulos, L., Grammatikou, M., Potts, M., Grosso, P., Fekete, A., Belter, B., Campanella, M., Maglaris, V.: Novi tools and algorithms for federating virtualized infrastructures. In: Álvarez, F., et al. (eds.) FIA 2012. LNCS, vol. 7281, pp. 213–224. Springer, Heidelberg (2012)
2. Global Environment for Network Innovations (GENI)., `http://www.geni.net/`
3. Gangam, S., Kala, S., Sharma, D., Fahmy, S., Blanton, E., Chatterjee, S., Sharma, P.: Design and evaluation of the s3 monitor network measurement service on geni. In: Proceedings of COMSNETS, COMSNETS 2012 (2012)
4. Boyd, E., Brown, A., Grigoriev, M., Metzger, J., Swany, M., Zekauskas, M., Li, Y.T., Tierney, B., Boote, J., Zurawski, J.: Instantiating a global network measurement framework. In LBNL Technical Report LBNL-1452E (2009)

5. Hullár, B., Laki, S., Stéger, J., Csabai, I., Vattay, G.: SONoMA: A Service Oriented Network Measurement Architecture. In: Korakis, T., Li, H., Tran-Gia, P., Park, H.-S. (eds.) TridentCom 2011. LNICST, vol. 90, pp. 27–42. Springer, Heidelberg (2012)
6. Santos, T., Henke, C., Schmoll, C., Zseby, T.: Multi-hop packet tracking for experimental facilities. In: Proceedings of the ACM SIGCOMM 2010 Conference, SIGCOMM 2010 (2010)
7. Holleczek, P., Karch, R., Kleineisel, R., Kraft, S., Reinwand, J., Venus, V.: Statistical characteristics of active ip one way delay measurements. In: Proceedings of the International Conference on Networking and Services, ICNS 2006 (2006)
8. Mátray, P., Csabai, I., Hága, P., Stéger, J., Dobos, L., Vattay, G.: Building a prototype for network measurement virtual observatory. In: Proceedings of ACM SIGMETRICS (2007), MineNet
9. The NOVI information model (2010-2013), `http://www.fp7-novi.eu/novidissemination/information-model`
10. Stevens, S.S.: On the theory of scales of measurement. Science (1946)

Summary and Conclusions

Lluís Fàbrega[1], Pere Vilà[1], Davide Careglio[2], and Dimitri Papadimitriou[3]

[1] Institut d'Informàtica i Aplicacions, Universitat de Girona, Spain
{lluis.fabrega,pere.vila}@udg.edu
[2] Departament d'Arquitectura de Computadors, Universitat Politècnica de Catalunya, Spain
careglio@ac.upc.edu
[3] Alcatel-Lucent Bell Labs, Belgium
dimitri.papadimitriou@alcatel-lucent.com

Abstract. In this chapter we summarize the main outcomes of this book, and provide lessons learned, best practices and key recommendations on experimental-based measurement and associated tools.

1 Introduction

The Future Internet Research Experimentation (FIRE) initiative [1] of the EU 7th Framework Programme (FP7) promotes research activities on new network and service architectures for the Future Internet, as well as building of large-scale experimental facilities where the scientific results of this research could be validated through experiments at the Internet scale. Such investigation activity would be realized by means of iterative cycles of research, where the data and observations obtained from experiments at each cycle, strongly influence the research direction in the next iteration. This research process, known as measurement-based experimental research, has inspired the main subject of this book. In each chapter, the authors, researchers of projects part of the FIRE initiative, have presented their view and experience on the subject from their project's perspective, namely, experimental methodology, testbeds, tools developed by the project or external tools, and experiments where such tools are used. Eight FIRE projects have participated, belonging to the research areas of wireless/sensor networks (HOBNET [2], CREW [3] and CONECT [4] projects), routing (CONVERGENCE [5] and EULER [6] projects), and large-scale experimental facilities (OFELIA [7], OpenLab [8] and NOVI [9] projects). The aim of this book was the following:

- Collect detailed information on the current developments on measurements in experimental research projects in new and well-established research areas.
- Identify which measurement tools have been developed and applied within the scope of FIRE experimental research facilities.
- Determine what are the foreseeable needs and their commonality by existing measurement techniques and tools.
- Document the lessons learned and best practices in tools development for measurement-based experimental research.

- Identify which initiative(s) can be initiated and realized in the future by means of cooperation between research projects.

This chapter is organized as follows. Section 2 summarizes the measurement-based experiments conducted in several FIRE projects and the measurement tools developed for realizing these experiments. Finally, Section 3 provides the lessons learned, best practices and key recommendations on experimental-based measurement and associated tools.

2 Measurement-Based Experiments and Measurements Tools

This section describes the use of measurements and associated tools in experimental research in FIRE projects, in the research areas of wireless/sensor networks, information networks, routing and large-scale experimental facilities.

2.1 Wireless and Sensor Networks

In general, the experiments conducted in this research area share the following characteristics:

- It is difficult to reproduce and validate the experiments due to the impossibility to isolate them from radio interferences coming from other sources. Also the energy conditions are never exactly the same because of the conditions and life of batteries.
- Experiments make use of highly heterogeneous equipment including different types of sensors (temperature, humidity, cameras, etc), actuators and different radio technologies (Wi-Fi/IEEE 802.11, WiMax/IEEE 802.16, 3G, 4G, ZigBee/IEEE 802.15.4, Bluetooth, etc.).
- Dynamicity and mobility have to be included, e.g., wireless devices can be randomly distributed over an area or placed regularly, be added or removed randomly while the experiment takes place, and be mobile following specific patterns or moving randomly.

The main objective of the HOBNET project [2] (Chapter 3) is to ease the development of applications for automation and energy-efficiency of smart/green buildings through sensor networks. For this purpose, several testbeds are available, with high diversity of devices (sensors actuators, etc.) and different thematic emphasis (energy, tracking, visualization, etc.). Many different applications have been evaluated on these testbeds such local adaptation to presence, CO_2 monitoring, garden watering, maintenance control, or electric device monitoring. Several critical options for the experimental scenario have been detected: structured topologies versus randomized deployments; homogeneous sensor deployments versus heterogeneous deployments; all sensors running at the start of the experiment versus sensors added during network evolution; uniform node density versus high density diversity; static deployments versus mobile deployments (and hybrid combinations). Performance evaluation experiments aim at determining the scalability with respect to the network size, the fault-tolerance properties (this implies the need for diverse fault models), and inherent trade-offs such as energy versus time. The project has developed the REST architecture, being its main features the following:

- Every distinguishable and addressable entity is defined a resource (anything from a physical device, like a sensor/actuator, to a web site, XML file, etc.) uniquely identified by an URI.
- The underlying IPv6/6LoWPAN infrastructure level makes use of IPv6 to integrate heterogeneous technology (sensors, actuators, mobile devices, etc). At a second level of the architecture, several algorithmic models and solutions for smart buildings have been proposed by taking into account scalability. On top of this structure, an interface is available for rapid development of building management applications.
- One of the modules included in this architecture is the Measurements Logger component, whose role is to setup the parameters of the experiments (event generation rate, energy sampling rate, duration, etc) and to monitor the evolution of the evaluated scheme by enabling the logging of the performance measurements (delivery latency, the average energy consumption, and the success ratio).

The problems derived from using measurements from heterogeneous equipment have been investigated by the CREW project [3] (Chapter 4). This project proposes a benchmarking framework for spectrum sensing through heterogeneous devices that comprises the following steps:

- Pre-calibration of the heterogeneous hardware, using a common metrics (the main metric is the Power Spectrum Density or PSD).
- Set up the experiment, using metadata to unambiguously describe the experiment (e.g., transmission power level, device name and location, frequency bins, etc.)
- Measurement of the PSD.
- Comparison using a common data format (a matrix with PSD and timestamp values), post-processing (to compensate for hardware and software heterogeneity) and obtaining of the "device performance score", which abstracts a set of metrics (e.g., device performance, reliability of the experiment, cost, repeatability).

The wireless NITOS testbed developed by the CONECT project [4] (Chapter 5), which allows experimental work both at the packet level and at the MAC layer, includes a large number of heterogeneous wireless devices including Wi-Fi nodes, USRP boards, sensors, WiMax/IEEE 802.16 nodes, 4G and 3G femtocell components. Access to resources takes place through the slice abstraction, i.e., isolated resource containers accessible by one or multiple users. The testbed also provides tools to assist experimenters in assessing the testbed's wireless environment properties and selecting an appropriate topology for their experiment. Examples of experiments conducted in the NITOS testbed include i) the design and evaluation of cooperative networks that exploit different paths through the aid of possible relays that carry out the traffic, and ii) the demonstration of a scenario where a vehicle equipped with sensors gathers measurements from its environment and communicates opportunistically with road-side units to forward the measurements to a centralized framework for their storage and analysis. Measurements in the NITOS testbed are handled using the cOntrol and Management Framework (OMF) adopted from the OpenLab project [8]:

- The OMF framework enables an efficient management of the heterogeneous resources of this testbed, providing a clear and easy way to define experiments, execute them and collect the results.
- The OMF Measurement Library (OML) is based on customizable measurement points inside applications running on the resources and provides a well-structured solution for capturing, processing, filtering, storing and visualizing measurements.
- An OML server is responsible of gathering the measurements and storing them in a database while OML clients are capable of injecting measurements generated at measurement points into streams towards the OML server.
- Extra features include the OMF graph generator (displaying results) and the proxy OML servers to cover disconnected parts of the same experiment thus enabling mobility support.

2.2 Routing

Experimental research on new routing schemes for the Internet is confronted to the difficulties in reproducing the Internet scale and modeling of routing states. The experimental approach requires simplifying the actual experimental corpus through functional abstraction, and reproducing significant phenomena through patterns derived from measurements of the actual environment. This approach involves the following steps i) to perform measurements on the current Internet, ii) derive patterns for modeling the interested quantities (topology, traffic, dynamics) and then iii) use these models for generating experimental scenarios to be executed in large scale routing scheme simulation and emulation. This approach has been followed by the two FIRE research projects we have dealt with, the CONVERGENCE project [5], which works on Information-Centric Networking (ICN), and the EULER project [6], which investigates inter-domain Internet routing algorithmic and related research challenges.

ICN is a novel paradigm where the network layer provides content to applications, instead of providing connectivity between hosts/terminals. The base functionality of ICN relies on i) content addressing through a scheme based on names or identifiers (that do not include references to their location); and ii) routing content requests toward the closest copy of the content with such a name (name-based anycast routing). The architecture proposed by the CONVERGENCE project [5] (Chapter 6) develops the content routing functionality as part of the network layer. Once an end node sends a content request, border routers use a routing table based on content names to forward the request to the next node, till the serving node, which then sends the content back. Besides routing by name, border routers also perform caching, and in the name-based routing table also store entries related to the cached content. Therefore, routing tables in nodes include IP routes, name-based routes and cached content index. Routing tables are in general not complete, and in case of a missing routing entry, a centralized routing engine called Name Routing System (NRS) provides the entry, which is temporary stored in the table.

Several experiments in ICN have been conducted to verify that current technology scales in terms of memory size of the local routing tables and supports the route lookup rate required at the NRS node, while other experiments evaluated the performance of the routing-by-name functionality. Download time, routing table size, number of route lookups at the NRS node, cache size and the amount of protocol messages exchanged by nodes were some of the measured quantities. In all these experiments, the ICN functionality has been used to distribute current web contents; therefore, it was necessary to derive ICN traffic traces between web clients and servers from Internet measurements. A tool was developed to convert Internet web traffic traces into ICN traces: TCP messages were mapped to ICN messages and, since web server's IP addresses in Internet traces are usually anonymous, they were randomly assigned to a set of public IP addresses of the most used domain-names.

Research on new paradigms for distributed and dynamic routing schemes suitable for the Internet short-term dynamics and its long-term evolution is the main goal of the EULER project [6]. The scalability, convergence, and stability properties of the inter-domain routing system currently based on the Border Gateway Protocol (BGP) are some of the major problems faced by the Internet routing architecture. Solving these problems requires to address multiple dimensions altogether: i) the routing table size growth resulting from an increasing number of routing entries, and ii) the routing system dynamics characterized by the routing information exchanges produced by topological or policy changes. Moreover, the re-use of BGP as discovery protocol for multicast routing refrains development of multicast applications. One of the routing schemes under study in this project, called Greedy Compact Multicast Routing (GCMR), has been presented in Chapter 7. This dynamic and multicast routing scheme is leaf-initiated, runs independently of the unicast routing scheme (and does not share any routing state information) and is specialized for the construction of multicast routing paths (or multicast distribution trees) from any source to any set of destination nodes (or leaf nodes). The performance of GCMR has been evaluated and compared with three other multicast routing schemes through simulation using the following metrics: i) the stretch of the multicast routing paths it produces, ii) the memory space required to store the resulting routing table entries, and iii) the total communication or messaging cost, i.e., the number of messages exchanged to build the entire Multicast Distribution Tree (MDT). Moreover the authors also have presented a study of how spatially and temporally concentrated traffic exchanges are in the Internet in order to state the relevance and benefits of deploying multicast routing schemes. In turn, the data and results obtained from these studies will enable to design more realistic scenarios for the emulation experiments and the performance comparison against the currently deployed approach in the Internet (combining Multiprotocol BGP and Protocol Independent Multicast - PIM). The experimental methodology they follow is a clear example of the use of iterative cycles of research, where new research iterations are driven by experimental results obtained in previous iterations.

2.3 Large-Scale Experimental Facilities

The deployment of large-scale experimental facilities is a crucial requirement for validating experimental research activities on new network and service architectures for the Future Internet. Indeed, building such facilities is one of the two dimensions of the FIRE [1] initiative, being the other the innovative research validated through large-scale experimentation in these testbeds.

The pan-European research testbed OFELIA [7] (Chapter 8) makes use of OpenFlow [10]. This technology enables the separation of control and local processing of data by defining the interactions and the operations performed from a (non-)co-located control element (the controller) to a forwarding plane element (e.g., switch,). The header of an incoming packet is compared to a set of defined "matches", i.e., patterns containing specific values of packet fields (Ethernet, IP, transport), and then some "actions" over the packet are performed (forward, rewrite fields, etc.). The set of packets seen by the switch since the rule was installed defines a flow. OFELIA project targets to build (and in the second phase interconnect) a set of campus installations (islands) that consist of GNU/Linux-based virtual machines (VMs) interconnected by OpenFlow switches, and to make these facilities available to all researchers. Internally, the physical network consists of two networks, the control network, which provides access to the control interfaces of VMs and to the switch controllers, and the experimental network, which connects data interfaces of VMs and switch ports. In OFELIA testbeds:

- Experiments run in "network slices", i.e., virtual networks that share the same physical network, built using FlowVisor, a network virtualization layer. FlowVisor, a special purpose OpenFlow controller, acts as a transparent proxy between OpenFlow switches and multiple OpenFlow controllers. It creates rich slices of network resources, delegates control of each slice to a different controller and enforces isolation between each slice, i.e., one slice cannot affect another's traffic.
- The experimenter can get access to a testbed island through an OpenVPN tunnel; then he can generate on demand the entire network slice (using Expedient, a special web-based resource allocation tool), access to the VMs through Secure Shell (SSH), and setup the experiment.
- The experimenter is provided with a built-in set of tools for traffic generation and measurement, and besides he has the capacity to install its own external tools. Provided tools are open source (e.g., Wireshark, iperf, etc.) and also high performance test systems (e.g., IXIA T1600 a1nd associated software), which allow to perform customized and predefined tests but also inject and measure traffic at any point of the network.
- The experimenter is also provided with a set of tools to test various aspects of OpenFlow switches and controllers, such as OFLOPS, cbench and OFTest.

An important aspect in large-scale experimental facilities is the federation of testbeds, i.e., enabling transparent access to combinations of resources from different testbeds, each of them addressing different applications or technologies. The federation of testbeds offers a richer and more powerful experimental facility to enable heterogeneous and large-scale Internet-oriented research.

The OpenLab project [8] (Chapter 9) brings together a number of different and diverse testbeds, such as wireline (PlanetLab Europe), wireless (NITOS, w-iLab.t), multimedia (WIT IMS), high precision measurement (ETOMIC), or emulation testbeds (HEN), which use different control framework and tools (e.g., MySlice, Federation Computing Interface and Federation Scenario Description Language, cOntrol and Management Framework and OMF Experiment Controller, or Network Experimentation Programming Interface). The main goal of the OpenLab project is to enable transparent access to combinations of resources from different testbeds, addressing the interoperability challenges at several levels by i) using tools tailored for a given testbed in other testbeds, ii) migrating experiments performed in old testbeds to new ones, iii) reproducing experiments in similar, yet different testbeds, and iv) extending experiments to enlarged scale or enhancing experiment to a broader scope. One of the problems in the federation of multiple testbeds is the presence of different and diverse measurement tools. Differences may appear in naming, data representations, units, metadata and data merge; therefore, their integration becomes necessary. The OpenLab solution to this problem (Chapter 10) deals with the semantics of the information, unambiguously specifying the set of concepts that compose a measurement. The proposed solution would comprise three steps: i) the agreement on a common ontology for network measurements, ii) the definition of mappings between each particular scheme and the common ontology, and iii) the definition of a semantic interface (based on the ontology) able to receive a query from a user and distribute it among all particular measurement repositories. Such ontology is currently being standardized at the European Telecommunications Standards Institute (ETSI) and several mappings are being defined.

The NOVI project [9] (Chapter 11) aims at providing control, management and monitoring planes to allow the federation of various virtualized testbeds, consisting of heterogeneous resources, enabling experimenters to request, reserve, use and update a great deal of virtualized resources in the federation, tailored to their needs. The necessary abstraction of the managed entities is provided by information models, which should support virtualization concepts, vendor independence (of the physical resources), monitoring and measurement concepts and management policies. The information model facilitates the control and management of the individual platforms, and the communication between them. This project proposes a monitoring and measurement framework to allow the federation of virtualized testbeds. With the use of a specific monitoring ontology, a wide range of monitoring tools, metrics and databases can easily be integrated in this framework. An experimenter only needs to know the metrics to be measured (e.g., throughput, one-way/two-way delay) independently of the tools installed in the different testbeds and the monitoring service will ensure the proper mapping between the requested metrics and the available tools automatically.

3 Lessons Learned, Best Practices and Recommendations

This section aims at documenting the lessons learned and best practices on the use of measurements and associated tools in experimental research in FIRE projects, as well as providing several key recommendations for the future work in this area. The following observations can be derived from the current developments on measurement-based experimental research in FIRE projects:

- The development of dedicated measurement tools is time-consuming (to comply with the verification, reliability properties as documented in Chapter 2) and existing tools when re-used (and thus re-usable) provide relatively limited extensibility potential.
- Testbeds are of different nature (wired, wireless, different hardware, different setups, etc.) and experiments themselves are conducted for different measurement purposes (even when performed on the same testbed); henceforth, measurement tools designed for experiments on a given testbed are likely to be incompatible with each other; thus measurements are often not reproducible. Moreover, the reliability of measurements is more challenging to achieve in open testbeds where all running conditions are not under the control of the experimenter.
- Experiments conducted in wireless and sensor environments are repeatable but only up to a certain probability that the measurement tool performs its intended measure (output) during a specified period of time under stated conditions. Ensuring these conditions are verified is challenging in open testbeds/experimental environments.
- Few (if not none) projects dedicate effort to verify their measurement results. The reasons are multifold: complexity of the experiments and variety of components they involve (thus their modeling), unavailability of comparable real system (since the experimented corpus is by nature unavailable), time required to perform systematic verification, etc. The measurement verification phase is often limited to ad-hoc (or eventually more systematic) comparison with similar experiments acting either as experimental model or reference system. For this purpose, measurement tools should also work / be adapted (from the testbed/prototype experiment) to also run in the context of emulation and simulation experiments. Such practice would facilitate the comparison of results.
- As experiments involve different components the increasing need to specify well-defined interfaces between different measurement tools with implementations adapted to different equipment, to standardize the formats of the collected data and, if possible, also the control of the experiment. Moreover, the factorization of the code by means of well defined and well documented software modules shall be put into practice so that measurement tools developers could use these software modules independently and in turn improve their reusability.
- The advent of new (or rejuvenated) technology research areas including programmable/software-defined networks induce specific needs such as on-line code verification and software robustness testing.

Among the FIRE community consensus raises on the need to setup a dedicated initiative on measurement tools, testbeds and experimental measurement-based research in networking/communication technologies in order to communalize and to factorize the development of measurement tools. Meanwhile measurement and associated techniques and tools play an instrumental role in forefront research on network science. Moreover, this initiative should remain in charge of the research community to avoid introduction of proprietary measurement software. Here below, we summarize the recommendations for the future work in this area:

- Concerning the question "How can the huge number of tools that already exist be shared?", the research community should create joint working groups involving different research projects and identify common developments that can be shared with other projects. At the end of the development and validation phase, tools should be made available as open-source code to the research community at large (by means of openly and easily accessible repository).
- On the other hand, bring measurement tools developers from different research projects is recommended to share their experience and work together in order to progressively specify a common modular baseline when developing measurement tools together with common data formats and generic interfaces (including the control interfaces). Moreover, all measurement tools and associated standards should be open-source.
- The implementation of a measurement tool repository should start with a simple and easily accessible repository with progressive addition of improvements (bottom-up) rather than designing a powerful but empty system (top-down). Even further, building such repository could start by just providing data from experiments (data sets) so that other researchers could re-use these data to further analyze them and obtain additional results. It is also important that all these tools could also work / be adapted from the testbed scenario to the emulation scenario and to the simulation scenario. This practice will also facilitate comparison between results.
- Finally, as the amount of collected data increases (due to the scale of experiments), it is recommended that the development of measurement tools shall not be limited to actual measures and their collection but also to data analytics and related tools.

References

1. EU FIRE initiative, `http://www.ict-fire.eu/`
2. EU FP7 HOBNET project, `http://www.hobnet-project.eu/`
3. EU FP7 CREW project, `http://www.crew-project.eu/`
4. EU FP7 CONECT project, `http://www.conect-ict.eu/`
5. EU FP7 CONVERGENCE project, `http://www.ict-convergence.eu/`
6. EU FP7 EULER project, `http://www.euler-fire-project.eu/`
7. EU FP7 OFELIA project, `http://www.fp7-ofelia.eu/`
8. EU FP7 OpenLab project, `http://www.ict-openlab.eu/`
9. EU FP7 NOVI project, `http://www.fp7-novi.eu/`
10. Open Networking Foundation, `https://www.opennetworking.org/`

Author Index