Lecture Notes in Computer Science 2839
Edited by G. Goos, J. Hartmanis, and J. van Leeuwen

Springer
Berlin
Heidelberg
New York
Hong Kong
London
Milan
Paris
Tokyo

Alan Marshall   Nazim Agoulmine (Eds.)

Management of Multimedia Networks and Services

6th IFIP/IEEE International Conference, MMNS 2003
Belfast, Northern Ireland, UK, September 7-10, 2003
Proceedings

Springer

Volume Editors

Alan Marshall
Queen's University of Belfast, School of Electrical and Electronic Engineering
Belfast BT9 5AH, UK
E-mail: a.marshall@qub.ac.uk

Nazim Agoulmine
Université d'Evry Val d'Essonne, Laboratoire LSC-CNRS
40, Rue du Pelvoux, 91020 Evry Cedex, France
E-mail: nazim.agoulmine@iup.univ-evry.fr

Cataloging-in-Publication Data applied for

Bibliographic information published by Die Deutsche Bibliothek
Die Deutsche Bibliothek lists this publication in the Deutsche Nationalbibliografie; detailed bibliographic data is available in the Internet at <http://dnb.ddb.de>.

CR Subject Classification (1998): C.2, H.5.1, H.3, H.5, K.3, H.4

ISSN 0302-9743
ISBN 3-540-20050-9 Springer-Verlag Berlin Heidelberg New York

Typesetting: Camera-ready by author, data conversion by PTP-Berlin GmbH
Printed on acid-free paper SPIN: 10956977 06/3142 5 4 3 2 1 0

Preface

Welcome to MMNS 2003.

Multimedia services over IP networks are proliferating at an enormous speed. There is also increasing demand for solutions that provide assured levels of service quality. All of these require novel paradigms, models, and architectures for realizing integrated end-to-end service management rather than managing network elements in isolation. Providing scalable Quality of Service (QoS) while maintaining fairness, along with secure and optimal network resource management, are key challenges for the future Internet. These challenges apply to both fixed and wireless networks.

This book contains all of the papers presented at the 6th IFIP/IEEE International Conference on Management of Multimedia Networks and Services (MMNS 2003) hosted by The Queen's University of Belfast, Northern Ireland, September 7–10, 2003. MMNS 2003 follows the successful conferences held in Santa Barbara (2002), Chicago (2001), Fortaleza, Brazil (2000), Paris (1998), and Montreal (1997). MMNS uses single-track presentations, which provide an intimate setting for discussion and debate. The conference is known for its high quality papers from various research communities. In just six years, MMNS has established itself as one of the premier conferences focusing on the management of multimedia networks and services. The conference objective is to bring together researchers working in all facets of network and service management as applied to broadband networks and multimedia services. MMNS deals with all aspects of designing, developing, and deploying networked multimedia systems, and it serves as a forum for the dissemination of state-of-the-art research and development results. MMNS also includes panel sessions in which experts offer their observations and opinions about current hot topics.

This year MMNS was augmented by a workshop on End-to-End Monitoring Techniques and Services (E2EMON). This one-day workshop offered a unique opportunity for researchers in this area to exchange ideas and experience that foster the development of generic and custom monitoring for next-generation Internet services. Many thanks are due to Professor Ehab Al-Shaer for his sterling efforts in organizing this.

The success of MMNS ultimately depends on the committee members and on the quality of the submissions. As in previous years, we were fortunate to have a large number of enthusiastic members who worked tirelessly to ensure that we had a high quality conference. We thank all the authors who submitted papers, and also the members of the program committee and the additional reviewers who carefully reviewed the submissions. Each paper submitted was reviewed by at least three reviewers, and conflicting reviews were extensively discussed by the program committee. This procedure has once again resulted in a varied and high quality program completed by the excellent tutorials and keynotes presentations, thanks to John Strassner, Petre Dini, and George N. Aggélou for the tutorials and Ian F. Akyildiz and Derek McAuley for the keynotes.

As conference chairs, it was a pleasure to work with such a dedicated and conscientious program committee.

This year once again saw the continued support of IFIP and IEEE. In addition we would like to acknowledge the financial assistance provided by Invest Northern Ireland, which sponsored the conference banquet. We also acknowledge the cooperation of Alfred Hofmann of Springer-Verlag, which has now been the official publisher for the last three conferences. We would like to acknowledge the valuable support and guidance provided by the steering committee: Ehab Al-Shaer, Giovanni Pacifici, Guy Pujolle, and Raouf Boutaba, and to thank the previous conference chairs for the benefit of their experience. Also, thanks to Massum Hassen for his help in organizing the tutorials, and Ahmed Mehaoua, John Vicente, and Tham Chen Kong for helping to advertise the conference and make it a successful event. Finally, we thank all the staff who assisted both at Queen's University Belfast, UK, and at Evry University, France. Particular thanks are due to Qiang Gu, Danielo Gomes, Emi Garcia, Mauro Fonseca, and not least to Paula Matthews. The application of her considerable experience, coupled with her unbounded enthusiasm and professionalism, assured a much more efficient conference than would otherwise have been possible.

For those of you who attended the conference we hope you found this year's MMNS a valuable experience, and were able to savour the technical presentations, renew old friendships, and hopefully make new ones. We hope everyone enjoyed their stay in Belfast.

September 2003 Alan Marshall, Nazim Agoulmine

MMNS 2003 Organizing Committee

Executive Committee

Conference Chairs

Alan Marshall	Queen's University of Belfast, UK
Nazim Agoulmine	University of Evry, France

Steering Committee

Ehab Al-Shaer	DePaul University, USA
Giovanni Pacifici	IBM Research, USA
Guy Pujolle	University Pierre & Marie Curie, France
Raouf Boutaba	University of Waterloo, Canada

Tutorial Chair

Masum Hassen	Cisco System, Inc., USA

Publicity Chairs

Ahmed Mehaoua	University of Versailles, France
John Vicente	Intel Corporation, USA
Tham Chen Khong	National Univ. of Singapore, Singapore

Proceedings Chairs

Emiliano Garcia	Queen's University of Belfast, UK
Mauro Fonseca	University Pierre & Marie Curie, France

Web Chairs

Qiang Gu	Queen's University of Belfast, UK
Danielo Gomes	University of Evry, France

Program Committee

Abdelhakim Hafid	University of Montréal, Canada
Ahmed Helmy	Univ. of Southern California, USA
Ahmed Karmouch	University of Ottawa, Canada
Alaa Youssef	IBM Research, USA
Alan Marshall	Queen's University Belfast, UK
Alexander Clemm	Cisco Systems, Inc., USA
Ana Rosa Cavalli	INT, France
Andrew Campbell	Columbia University, USA
Bert-Jan van Beijnum	University of Twente, The Netherlands
Burkhard Stiller	UniBw Munich / ETH Zurich, Switzerland
Chien-Chung Shen	University of Delaware, USA
David Hutchison	Lancaster University, UK
Dominique Gaiti	Univ. of Technology of Troyes, France
Ed Perry	HP Labs, USA
Gautam Kar	IBM Research, USA
Gerard Parr	University of Ulster, UK
Go Hasegawa	Osaka University, Japan
Greg Brewster	DePaul University, USA
Hanan Lutfiyya	University of Western Ontario, Canada
John Strassner	Intelliden Corporation, USA
Jose M. Nogueira	University Minas Gerais, Brazil
Jose Neuman de Souza	Universidade Federal do Ceará, Brazil
Kevin Almeroth	UC-Santa Barbara, USA
Laurie Cuthbert	Queen Mary, University of London, UK
Lundy Lewis	APRISMA Corporation, USA
Michael Tchicholz	Fokus Fraunhofer, Germany
Mohamed Bettaz	Philadelphia Univ., Jordan
Mohammed Atiquzzaman	University of Oklahoma, USA
Mohammed Erradi	ENSIAS, Morocco
Murat Yuksel	Rensselaer Polytechnic Inst., Germany
Nazim Agoulmine	University of Evry, France
Nikos Anerousis	Voicemate Inc., USA
Petre Dini	Cisco Systems/Concordia Univ., USA/Canada
Puneet Sharma	HP Labs, USA
Ralf Steinmetz	Darmstadt Univ. of Tech., Germany
Russ Clark	Georgia Inst. of Technology, USA
Songwu Lu	UC-Los Angeles, USA
Supratik Bhattacharrya	Sprint ATL, USA
Willie Donnelly	Waterford Inst. of Technology, Ireland

Additional Reviewers

Nadjib Achir	University of Paris VI, France
Ehab Al-Shaer	DePaul University, USA
Marcelo Amorim	University of Paris VI, France
Joaquim Celestino	UECE-LARCES, Brazil
Belkacem Daheb	LIP6, France
Mauro Fonseca	LIP6, France
Yacine Ghamri	LIP6, France
Danielo Gomes	University of Evry, France
Masum Hasan	Cisco Systems, Inc., USA
Joseph Markwei	Telelink ventures, UK
Ahmed Mehaoua	University of Versailles, France
Anelise Munaretto	University of Paris VI, France
Manuel Nunez	University of Madrid, Spain
Clark Russ	Georgia Institute of Tech., USA
Sidi-Mohamed Senouci	University of Paris 6, France
Malgorzata Steinder	IBM Research, USA
Asser Tantawi	IBM T.J. Watson Research Center, USA
Chen-Khong Tham	National University of Singapore, Singapore
John Vicente	Intel Corporation, USA
Fawaz Wissam	University of Paris Nord, France

Table of Contents

CORBA-Based Stream Control and Management for IP-Based Production Studio Networks

Terence Song[1], Dritan Kaleshi[2], and Alistair Munro[2]

[1] Wireless and Networks Research Labs, Centre for Communications Research, University of Bristol, Merchant Venturers Building, Woodland Road, Bristol, BS8 1UB, United Kingdom
Terence.Song@bristol.ac.uk
[2] Department of Electrical and Electronic Engineering, University of Bristol, Merchant Venturers Building, Woodland Road, Bristol, BS8 1UB, United Kingdom
{Dritan.Kaleshi, Alistair.Munro}@bristol.ac.uk

Abstract. Traditionally, device relationships in a production studio are established and managed by physically routing the different cables that carry the relevant streams, together with any required synchronization references. IP-based networks offer a common infrastructure for the distribution of broadcast content as well as the deployment of integrated production and management solutions. This paper describes the design and implementation of components based on the OMG Audio/Video Streams Specification to facilitate the rapid integration of devices during the initial set-up phase, and the control and management of media content flow during normal operations for the purpose of program production within an IP broadcast environment.

1 Introduction

The use of heterogeneous IP-based networks as a basis for the distribution of production studio content presents a number of interesting challenges. The operational requirements of a production studio far exceed those of more common Internet-based streaming applications (e.g. video-conferencing, stock quotes, etc.). The production studio is made up of heterogeneous sources, sinks, and intermediate processing devices; such as microphones, speakers, mixers, recorders, players, and storage devices. These devices form complex relationships with each other, usually under the control of a studio manager or producer, to define the flow of media content during the course of program production (i.e. capture, editing, storage, distribution, and presentation). The two main issues related to the distribution of production content are: the control and management of stream relationships, and the distribution of synchronization information. The work described in this paper relates to the first of these two issues. In particular, we describe the design and implementation of IDL-defined components for the control and management of flows within the audio plane of an IP broadcast studio environment for the purpose of program production.

Traditionally, device relationships in a production studio are established and managed by physically routing the different cables that carry the relevant flows and streams, together with any required synchronization references. IP-based networks, however, offer a common infrastructure for the transport and automatic routing of media data, as well as the deployment of integrated broadcast content production and management solutions. The heterogeneity of IP-based networks allows the stream establishment and management problem to be addressed at the application-level, instead of the physical-level. The control and management of streams within the production studio consists of two main parts—the definition of studio device relationships during the initial set-up phase, and the dynamic re-configuration and modification of those relationships and their properties (e.g. formats, device parameters, etc.) thereafter.

In order to achieve maximum extensibility, maintainability, and interoperability, our objective has been to re-use standard technologies to the greatest extent possible. To this end, the aims have been to evaluate the applicability of the Object Management Group's (OMG) Audio/Video Streams Specification [1] and to develop extensions for the control and management of audio flows within an IP broadcast studio environment. The Common Object Request Broker Architecture (CORBA) [2] distributed processing environment (DPE), which provides a scalable, extensible, and robust environment for distributed objects to co-exist and inter-play, forms the basis for the implementation of the control and management components. The stringent performance requirements for streaming data often preclude the use of distributed object computing middleware as the transport mechanism, as demonstrated in [3]. The focus of the work described in this paper, however, is on middleware support for stream control and management, not stream transport. Instead, the transport of time-based media in our implementation is facilitated by the Real-Time Transport Protocol (RTP) [4] through the Java Media Framework (JMF) [5]. Previous works based on the OMG Audio/Video Streams Specification have focused on media streaming, with the control and management aspect being of lesser importance. Our contribution demonstrates its application to a production studio environment, where the control and management of complex flow and stream topologies is critically important.

The remainder of this paper is organized as follows. Section 2 begins with a definition of what makes up a stream, followed by a brief overview of the OMG Audio/Video Streams Specification. In section 3, we present the production studio's audio plane and describe the definition of flow control and management components to model its layout and contents (i.e. the devices and their relationships). Section 4 provides a description of the production studio's set-up and normal operations using a design patterns-based approach. The relevant issues that influenced our design and implementation of flow control and management objects are highlighted in section 5, followed by a brief description of our prototype implementation in section 6. The conclusions are presented in section 7.

2 Overview of the OMG Audio/Video Streams Specification

The OMG Audio/Video Streams Specification [1] defines a standardized, flexible, and efficient CORBA-based architecture for the control and management of streams. Stream control and management concerns the establishment, control, management, and release of streams (i.e. the specification defines control flow, not data flow). The architecture is independent of media streaming frameworks and is generic enough to support arbitrarily complex streams.

2.1 Stream Model

In the Internet domain, the term *stream* is more commonly used to refer to the continuous transfer of a particular format of media content from a (synchronization) source to one or more sinks. In this paper, however, the continuous transfer of media content from a synchronization source is explicitly qualified by the term *content stream*, and the terms *flow* and *stream* refer to abstractions (i.e. local representations of distributed processes) used to facilitate the control and management of content stream instances.

Figure 1 depicts an abstract model of the basic elements that make up a stream. The origin and destination of a content stream is referred to as the *producer* and *consumer flow endpoint* respectively. Flow endpoints are created by *flow devices* to support flows. A *flow* abstracts one or more content streams of the same type (i.e. format). More precisely, a flow represents the continuous transfer of media content of a particular format (e.g. audio or video) in a clearly identified direction, from one or more producer endpoints to one or more consumer endpoints. A *stream* is an abstraction for aggregating multiple flows in parallel, which may travel in either direction, between *stream endpoints*. Similarly, a stream endpoint, which represents the termination point of a stream, logically contains one or more consumer and/or producer flow endpoints (of which some or all may be involved in the stream). No directionality is attached to a stream endpoint, since it may contain both producer and consumer flow endpoints simultaneously. However, specialised or typed stream endpoints may be used to assist in matching compatible flow endpoints during the stream establishment process. Stream endpoints are created by *multimedia devices* to support streams. A multimedia device abstracts a collection of one or more flow devices. Flow and multimedia devices are responsible for creating and destroying their respective flow and stream endpoints.

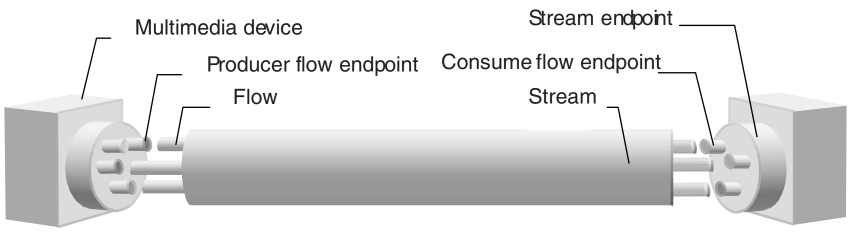


Fig. 1. The basic elements of a stream.

The stream abstraction defines a containment hierarchy, whereby a stream contains one or more flows, and each flow contains one or more content streams. That is, the containment hierarchy defines one-to-many relationships between containers and containees, and by transitivity, a *stream* contains one or more *content streams*. It is important to note that flows can exist independently of streams. The stream-related elements simply provide the ability to aggregate their respective flow-related counterparts, thereby allowing the latter to be controlled and managed as a composite entity.

2.2 Stream Control and Management

The OMG Audio/Video Streams Specification [1] defines the set of component interfaces that abstract the basic elements that make up a stream. There are two conformance levels, allowing implementations to trade-off between flexibility and efficiency. The light-profile defines components that represent a stream and its associated multimedia devices, virtual devices, and stream endpoints. Flow-related functionality is co-located within and accessed through the encapsulating stream-related component interfaces. The full-profile provides for greater granularity of control by also exposing flow-related interfaces (i.e. flows, flow devices, and flow endpoints). Since flows also support the relevant management interfaces, they can exist independently of streams. The component interfaces can be grouped into two distinct sets—interfaces onto flow and stream *control and management objects* (representing the continuous transfer of media content), and interfaces onto flow and stream *interface control objects* (representing the participants involved in the media transfer). More importantly, the specification defines how control messages are transmitted and received in a CORBA-compliant way between these two sets of interfaces to set-up, control, manage, and release flows and streams.

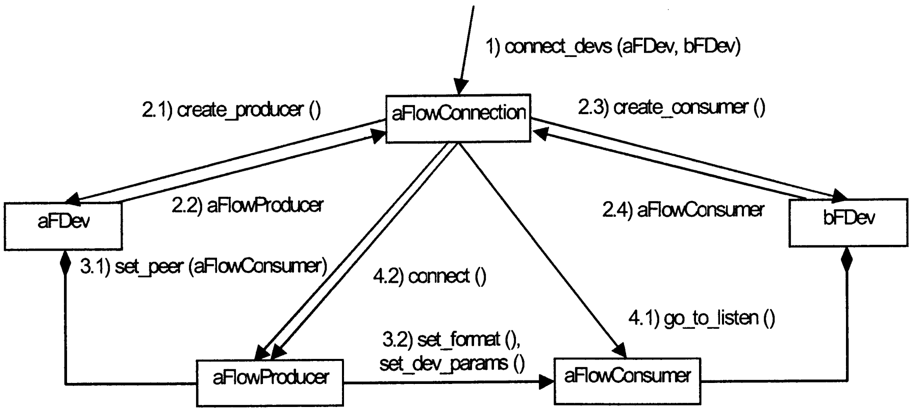


Fig. 2. The point-to-point flow establishment process.

Flow and stream *control and management objects* play a central role in the architecture. A control and management object provides two different but interrelated levels of abstraction. At the highest-level of abstraction, it represents the continuous transfer of media content (i.e. a flow or stream). This abstraction is suitable for modelling potential, as well as established, device relationships. A flow or stream, however, involves multiple distributed processes that facilitate the media transfer (i.e. it is a distributed entity). Therefore, the control and management object is also a local representation of the flow or stream participants. To support these two abstractions, the control and management object supports two sets of operations: operations for binding devices and endpoints, and operations for controlling and managing that binding. Because the binding mechanism is standardized, interoperability between different implementations is possible. The establishment of a point-to-point flow binding is illustrated in Figure 2. The process consists of three main phases: endpoint creation, configuration, and transport set-up. Upon receiving a request to connect two flow devices, the flow control and management object (as a potential binding) requests each device to create an endpoint that will support the flow. The flow control and management object then requests one of the endpoints to ensure that its peer is configured similarly. Once the endpoints have been configured, the flow control and management object completes the binding process by instructing the consumer to begin listening on a particular address, which may be explicitly specified or implicitly determined, and for the producer to connect to that same address. Once established, the flow control and management object (as an established binding) supports operations for starting, stopping, and destroying the flow.

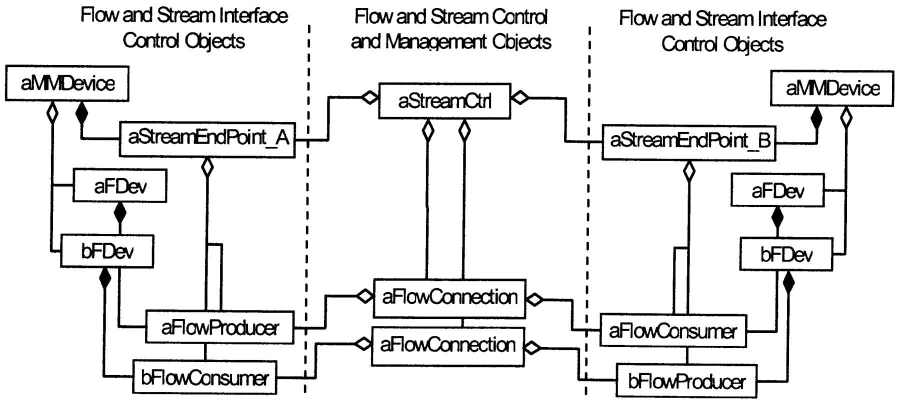


Fig. 3. A stream binding established between full-profile IDL-defined components.

Figure 3 shows the associations between components involved in a full-profile binding between two multimedia devices (c.f. Figure 1). The stream establishment process is similar to that for a flow, except for an additional flow endpoint matching phase; where the compatibility of flow endpoints supported by stream endpoints is determined. In a full-profile implementation, a collection of flows can be established independently and then individually added to a stream via operations supported by the stream interface. The stream-related components simply provide a way of aggregating flow-related components, as illustrated in Figure 3. The component interfaces can be extended to provide custom stream handling appropriate to the application.

Because the Audio/Video Streams Specification provides a programmatic description of the binding process, the fact that an IDL-defined component is a CORBA object is often overlooked. In our modelling efforts, we found that an object service provider/requestor view provides a greater understanding of the mechanics that underlie the architecture. Additionally, in our analysis of the control and management architecture, we discovered that the roles of control and management objects, and indeed the dynamics of the architecture, can be described simply in terms of two design patterns—*Mediator* and *Facade* [6]. It is unclear whether the occurrence of these two design patterns in the architecture is by design, or a consequence of it. In any case, sections 4.1 and 4.2 will further assert the roles of control and management objects within our IP broadcast environment using a design patterns-based approach.

2.3 Property Management

Fundamental to the operation of any control and management application is the ability to represent, access, modify, and exchange management-related information. The Audio/Video Streams Specification makes extensive use of properties to describe devices, streams, flows, and their endpoints. In-line with CORBA concepts, the specification does not build non-typed properties into interfaces. Instead, component interfaces inherit from CORBA's Property Service [7], which is used extensively for managing properties. This inheritance separates the concern of property management from those of stream control and management. Supporting a generic interface for property management allows the management information set to be extended dynamically without requiring changes to stream control and management interfaces. More importantly, however, a standard interface allows for interoperability between the stream components, allowing peers to be queried to establish their status and their compatibility constraints. In addition to a standard interface, a standardised set of parameters is also defined to allow for interoperability between different implementations. These properties are managed internally by stream components and are defined as read-only to clients. Additionally, application-specific parameters may also be associated with stream components; for example device model, serial number, owner, location, etc.

3 Modelling the Production Studio

The production studio is made up of heterogeneous sources, sinks, and intermediate processing devices that form complex relationships with each other dynamically or under the control of a studio manager or producer. The talkback system, which forms an integral part of the production studio, provides participants in the production process with communication and audio monitoring capabilities. An IP-based network forms the underlying infrastructure used for transporting the media streams (live or stored audio and video, and auxiliary data) used as input to produce a program for broadcasting and/or storage. In addition, the infrastructure supports facilities for service discovery and brokering [8]. In this section, we describe the modelling of the production studio's audio plane by full-profile *flow components*. Although stream-related components have not been considered, the concepts discussed extend naturally to include collective control and management of multiple flows through stream-related interfaces (as described in section 2.1 and 2.2).

3.1 The Studio Audio Plane

A schematic of the production studio's audio plane is shown in Figure 4. The schematic shows the flow relationships established between the various devices as well as the direction of those flows. The main source of audio is a microphone. The most common sink of audio is a speaker. Other relevant audio devices include mixers, recorders, playback devices, effects processors, etc. The creation and processing of media content occurs in both digital and analogue form. The point-of-presence (PoP) of a device is the point where the digital content it produces or consumes can be identified and managed. It is realized that not all devices present in the studio can or will be directly controlled at their PoP. However, in our modelling efforts, we have assumed that gateways can be used to provide suitable converting functions to enable the transfer of control and management functions. The synchronization streams shown in the figure will not be modelled as synchronization of content streams will be derived primarily from RTP/RTCP algorithms [4]. This schematic forms the basis for the definition and implementation of audio flow control and management components.

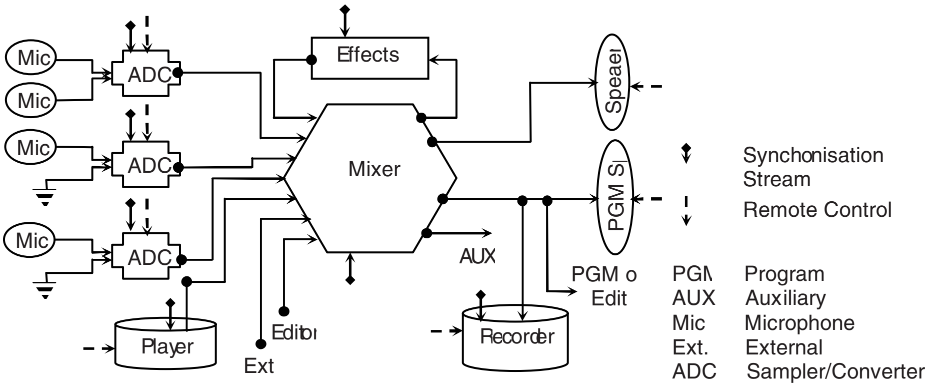


Fig. 4. Studio Schematic: Audio Plane.

3.2 Audio Flow Control and Management Components

Our modelling of the production studio's audio plane is based on full-profile components of the Audio/Video Streams Specification. A full-profile implementation allows for a modular approach to the development of stream components at design-time, and greater flexibility in the composition and distribution of media content at run-time. Indeed, our current production studio environment is modelled and built on audio flow components, and this can subsequently be extended to include video flow components as well as more complex stream configurations as required. At runtime, flows (e.g. audio, video, talkback, etc.) can be structured into layers or hierarchies, and dynamically added to or removed from stream structures, allowing flows to be controlled and managed individually, or collectively, as dictated by the production process.

Multimedia device objects and their associated resources (i.e. virtual devices, endpoints) are anticipated to execute on the devices they represent, or on gateway or proxy devices capable of providing a suitable execution platform (e.g. analogue-to-digital converter devices), such that their execution is closest to the true source or sink of media content. This means that the lifetime of virtual devices and endpoint objects will likely depend on the multimedia device it executes on and represents. Unlike device and endpoint objects however, a flow control and management object does not have a physical equivalent—it is a local representation of the state of distributed processes. Consequently, its existence is not physically or operationally bound to any one particular device. The control and management object need not be created, co-located, nor managed by the same party that invokes it. Depending on application requirements, control and management objects may be transient or persist beyond its service lifetime as well as that of its devices and endpoints. They can be accessed locally or remotely and can have local library or remote service styles of implementations. Furthermore, control and management objects (i.e. their implementations) can be modified without affecting the rest of the components in the system or how they interact. Solutions are anticipated to be application and policy specific.

The fact that flow control and management objects can be run and managed independently of peer devices is of particular importance for production studio environments. In order to create a composite program, the production process relies on a predictable set of flows and streams with a predetermined layout or topology for routing media content. Because control and management objects abstract flows and streams, the production studio's layout can be determined in advance by specifying the set of control and management objects that model the possible connection points into the production studio (e.g. using third-party establishment under the direction of the editor). A predetermined set of flow and stream control and management objects (possibly involving intermediate interconnected components) can be used to specify a system of fixed paths (similar to a pipeline for transporting media content) for routing content streams within the production studio. Once instantiated and activated, the set of flow control and management objects represent the *potential* bindings within the production studio.

Figure 5 shows an object model of the production studio's audio plane. The physical connections established between the audio devices in the production studio have been replaced with appropriate audio flow control and management objects (c.f. Figure 4). These flow control and management objects form the principal components of the production studio's management application, which is responsible for keeping track of all flow control and management objects (i.e. flows). It should be noted that the associations shown in the figure represent object relationships, not flow relationships. The flow relationships are represented by the flow control and management objects themselves. Also, the model illustrates only one of many possible scenarios in which flow control and management objects may be created and used. Furthermore, the flow objects highlighted in the figure are elementary point-to-point and point-to-multipoint flow components (i.e. based on unicast and multicast transports respectively). More complex flow topologies (e.g. multipoint-to-multipoint) can be constructed from compositions of these elementary flows, allowing management clients to perform control and management operations on the composite flow or on its constituent flows. This provides for a more controlled and predictable approach to studio planning, set-up, flow selection and handling, as well as the overall management of the production process during normal operations. Solutions are anticipated to be application and policy specific.

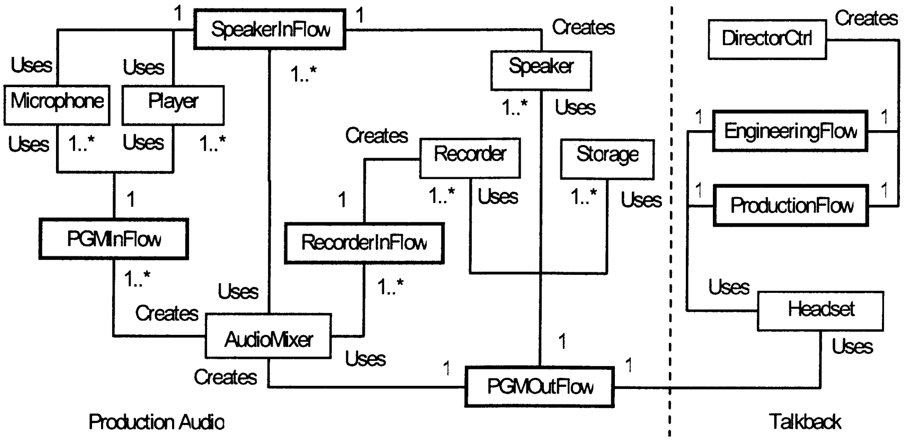


Fig. 5. An integrated object of production studio audio components.

4 Studio Dynamics

There are two main aspects to control and management within the IP broadcast studio environment. The *automatic* object relationship establishment at studio set-up between devices (i.e. studio-on-demand), and the management of the studio functionality during normal operations, expressed in terms of static relationships between the devices in the studio—delivering what streams where (i.e. program production). The flow (and stream) abstraction allows the manager to focus on content management rather than device management. Since the production studio's layout is established by a set of flow control and management objects, and the same set of objects represent both potential as well as established flows (i.e. bindings), the studio set-up and its management thereafter involves the device or studio manager locating the relevant flow control and management object, and invoking one of two sets of services—flow establishment, or flow control and management. One of the fundamental requirements of any distributed processing environment (DPE) is the ability for a component to locate other components with which it can interact. All the necessary information required to invoke a CORBA object is encapsulated in its object reference. Since object references are typed, clients are able to determine the services provided by a particular component, as well as the mechanisms for using those services, via the IDL typing system [9, 10 and 11]. Because the flow components are structured as CORBA objects, there does not need to be a special way of finding control and management objects representing a particular flow. Finding the components is orthogonal to using the services they provide. In other words, clients invoke operations on a CORBA object the same way regardless of how they obtained its object reference. Therefore, flows can be identified and located using generic CORBA facilities or application-specific directory services. For example, an implementation of the *Factory* design pattern [6] can be used to supply a predictable set of control and management objects with varying execution policies (e.g. lifetime, ownership, activation, etc.) and types. Such location and execution transparency, scalability, and robustness are the established traits of CORBA.

4.1 Studio Set-Up

Each flow control and management object supports operations that perform the necessary negotiation between flow devices and endpoints to set-up the flow, which it then represents. The role of the control and management object in the flow establishment process can best be described by the *Mediator* design pattern [6]. The intent of this pattern is to define an object (i.e. the control and management object) that encapsulates how a set of objects (i.e. the devices and endpoints) interact. A mediator promotes loose coupling by keeping peer objects from referring to each other explicitly, and allows their interaction to vary independently.

During the studio set-up phase, components in the studio try to discover and identify suitable peer components and, if required, determine their compatibility and establish the necessary control or consumer relationship with the service (content or otherwise) provider entities. The target "studio-on-demand" provided by the studio DPE requires mechanisms to rapidly integrate sources such as cameras or microphones into the production environment [8]. Devices and endpoints do not interact or negotiate with their peers directly. That is, they do not invoke each other's operations, except for the initial call on a peer to create a binding using first-party flow or stream establishment, and during the configuration phase. Instead, flow control and management objects (acting as mediators) implement and execute the peer negotiation mechanisms and protocols (i.e. determining peer compatibility and negotiating protocols, formats, QoS, security, etc.). The binding process is initiated by invoking the "connect-dev" operation on the control and management object with the target devices or endpoints as parameters. The control and management object conspires with peer devices and endpoints to create the binding. Interoperability between components is facilitated by the set of rules specified in [1] that implementations of the interfaces must follow. Two approaches to flow establishment are possible—first-party and third-party flow establishment. It is important to realize that first-party and third-party flow establishment does not refer to the co-location of the control and management object, but to the initiation of the flow establishment process. First-party flow establishment is initiated by an autonomous peer device or endpoint, and third-party flow establishment is initiated by a client that is not one of the participating devices or endpoints (e.g. a management application).

Since the target flow can be identified as a CORBA object, the integration of sources and sinks simply entail the particular source or sink locating the correct flow control and management object and invoking its "connect-dev" operation to be "automatically" integrated into the existing studio set-up. The term automatic used in the establishment process refers to the fact that no influence external to the control and management architecture (e.g. from the studio manager) is required for peers to establish a flow relationship. The rapid integration of sources and sinks is possible because the flow establishment process is standardised and encapsulated by flow control and management objects. This fact is particularly significant in the case of multicast flows. Potential sinks can simply identify the target flow to connect to and not the device or address to listen on. The necessary information is conveyed to sinks by the multicast flow's control and management object. The IDL typing system ensures that flows can only be established between compatible devices and endpoints.

4.2 Studio Normal Operations

Once the binding is established, the control and management object that previously represented the potential flow now represents the established flow and supports operations for starting, stopping, and destroying that flow. The identifiable flows established during studio set-up result in a topological layout of paths that route content streams from sources to sinks for the purpose of program production. These flows are managed or processed under the control of the editor during normal studio operation. The devices and their associated endpoints participating in a flow can be registered centrally or located via their respective control and management object. The role of the control and management object as a flow can be fully described by yet another design pattern—the *Facade* design pattern [6]. The intent of this pattern is to provide a unified interface to a set of interfaces in a subsystem. It defines a single higher-level, simplified, and localized interface (the flow) to the more general facilities of a distributed subsystem (in this case, the flow participants). This helps to reduce the complexity of the subsystem and makes it easier for clients to use (i.e. control and manage).

During normal operations, management clients simply locate the control and management object that represents the target flow. Once located, the control and management object allows the management client to control the transfer process, modify its QoS profile, or destroy the binding (i.e. close the stream and tear down its transport connections) when it is no longer needed. Since the flow has a distributed state, the control and management object maintains references to all of its participating endpoints. This allows it to exercise control over the transfer of content by issuing requests to participating endpoints whenever changes are requested. As with flow establishment, the management application does not interact with devices and endpoints directly, nor manage them individually. Hidden to all its management clients, a control and management object interacts and conspires with its participants to control and manage the media transfer. The control and management object in its role as a facade is therefore the primary client of peer devices and endpoints.

5 Design Considerations

The flow abstraction is semantically closer to its content than the stream abstraction. The stream abstraction is only relevant in the aggregation of multiple flows. It is important to distinguish between a flow relationship and its content relationships. The use of the general term flow refers to the transport of media content of a single format (e.g. audio, video) between producers and consumers. Although a flow is generally considered as a single entity, it may be composed of multiple content streams. Each producer contributing to a flow defines a content relationship between the producer and its consumers in that flow relationship. In other words, a flow may carry content from multiple sources, raising the issue of source synchronization and flow topology. It is the contractual responsibility of a flow control and management object to define and represent the content relationships between the set of producers and consumers. This section discusses the correlation between a flow abstraction and its content streams, and how this affected the design and implementation of our prototype.

5.1 Handling Multiple Content Streams

Each content stream instance within a flow is identified by its synchronization source. The content stream that is transferred from a producer to its consumers consists of media objects or presentation units (e.g. an audio sample, a video frame) that must be presented according to the temporal relationships that existed during the capturing process of the media objects [12]. Because the presentation units of a content stream are temporally-related, its source—or more precisely its synchronization source—is used to distinguish it from other synchronization sources (including those from the same device or endpoint) so that consumers can identify, synchronize, and present the media objects of the content stream in a timely fashion. Additionally, consumers must identify the type of data being received, detect packet loss, and determine the order in which the arriving packets should be presented. In the Internet, synchronization of information streams is derived primarily from RTP/RTCP algorithms [4].

As a flow may involve more than one producer, the contract between the flow control and management object and its participants are such that, consumers bound to that flow will receive content ‘injected’ into the flow by all producers bound to that flow. Content and flow relationships share all but two characteristics—their type and cardinality[1]. The maximum cardinality of the relationship defined by a content stream stipulate that a content stream always originates from one producer, but more than one content stream may terminate at a consumer. That is, from the producer’s viewpoint, a flow relationship is synonymous with a content relationship. In contrast, from the consumer’s viewpoint a flow relationship may translate to multiple content relationships.

The handling of multiple content streams is particularly important where multicasting is used. At the protocol level, multicasting minimizes the bandwidth and complex-

[1] For each role in a relationship type, the *minimum* and *maximum cardinality* specifies the minimum and maximum number of relationships respectively, in which a role will participate.

ity required to send information to multiple hosts on a network. A multicast group address represents a flat virtual network without any capability to subset group members logically or by physical location/distance in the underlying network. Multicasting in the Internet relies on establishing a well-known shared network-layer address space (so-called Class D addressing in IPv.4 and distinguished by a specific prefix in IPv.6) in which messages, or streams of information, are routed from one source to multiple receivers that recognize that address [8, 13]. The transport or routing of multicast streams is not as important in the discussions here as what happens to the different stream of packets arriving from different producers at the consumer. Content streams transported on a single multicast address are received by every consumer listening on that same address. There are two approaches to handling multiple content streams. Each content stream received by the consumer may either be handled (i.e. processed and presented) separately, or a consumer may support mixer operations directly to create a new compound content stream. The mixing of multiple content streams however is typically performed by a dedicated mixer device. The *audio mixer* plays a central role in the management of multiple audio content streams (as can be seen in Figure 4). As Figure 6 illustrates, it is an intermediate system that receives data packets from one or more (contributing) sources, possibly changes the data format, combines the packets in some manner and then forwards a new data packet. Since the timing among multiple input sources will not generally be synchronized, the mixer will typically make timing adjustments among the incoming content streams and generate its own timing for the combined content stream. Thus, all data packets originating from a mixer will be identified as having the mixer as their synchronization source.

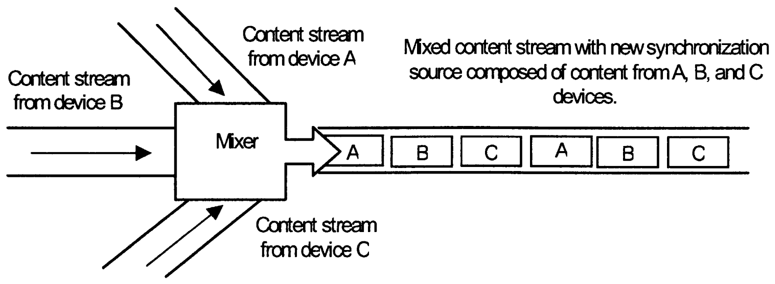

Content stream from device A
Mixed content stream with new synchronization source composed of content from A, B, and C devices.
Content stream from device B
Mixer
A
B
C
A
B
C
Content stream from device C

Fig. 6. The mixing of content streams from multiple synchronization sources.

Although no intermediate mixer device is necessary where mixer operations are supported by consumer devices directly, the use of dedicated mixer devices may be desirable for reasons of compatibility, performance, and controllability. Consumers inherently support at least one content stream. However, not all consumers provide capabilities for handling multiple content streams. Removing this responsibility from consumers ensures that irrespective of whether the flow is composed of one content stream or multiple content streams, the consumers remain compatible. Assigning the responsibility for mixing multiple content streams to a dedicated mixer device means less processing load on consumers. The data packets received can be processed and presented immediately, since all data packets originating from a mixer are identified as having the mixer as their synchronization source. Additionally, the production activities will require selective mixing of multiple content streams. A dedicated mixer device may provide greater flexibility over the mixing and delivery of multiple content streams with respect to the production studio's overall management. The mixer's capability to selectively combine input content streams can be used to provide a more predictable and controllable means of routing multiple content streams.

5.2 Flow Topologies

The flow relationship defines a topological layout that determines what content streams are delivered where. The support for multiple producers by flow control and management objects means that point-to-point, point-to-multipoint, multipoint-to-point, and multipoint-to-multipoint flow configurations are supported directly. The number of producers involved in a flow does not only affect the handling of content streams, but it also influences the flow's resulting topology. The content streams established between producers and consumers describe a maximal bipartite[2] graph where the direction of content flow is from producer nodes to consumer nodes. The graph in effect describes the flow's topology.

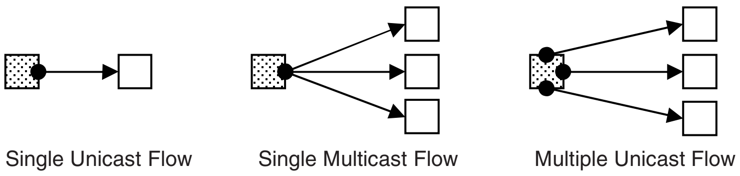


Fig. 7. Point-to-point and point-to-multipoint topologies.

As with the handling of content streams, the implementation of topologies involving only one producer (i.e. point-to-point and point-to-multipoint) is straightforward; since the data received by consumer(s) constitutes only one content stream and as such no mixing of content streams is required. Naturally, point-to-point and point-to-multipoint flows can be based on simple unicast and multicast connections respectively. Notice that a point-to-point topology is a special case of a point-to-multipoint topology where there is only one consumer. As such, the point-to-multipoint flow need not necessarily use a multicast connection. If multicasting is not available, or would be unsuitable (e.g. when the content stream received by each consumer needs to be controlled and managed separately), multiple unicast connections may be used instead. In this case, the sender device creates a producer endpoint for each unicast connection supported, as shown in Figure 7. This implementation detail is hidden from clients.

[2] A bipartite graph is an undirected graph in which the set of nodes can be partitioned into one of two sets, where all edges go between the two sets.

The support of point-to-point and point-to-multipoint topologies by a flow control and management object is implicit. These topologies form the basic components from which more complex topologies can be constructed, which include multipoint-to-point and multipoint-to-multipoint topologies. The support of multipoint-to-point and multipoint-to-multipoint topologies by flow control and management objects is optional. A flow control and management object may refuse to support more than one producer. Figure 8 depicts a multipoint-to-multipoint flow that uses an intermediate mixer device to mix the content streams from multiple producers and multicasts the composite stream to multiple consumers. Producers are connected to the mixer device by unicast connections. Notice also that the multipoint-to-point topology is a specific case of the multipoint-to-multipoint topology where only one consumer receives the mixed stream.

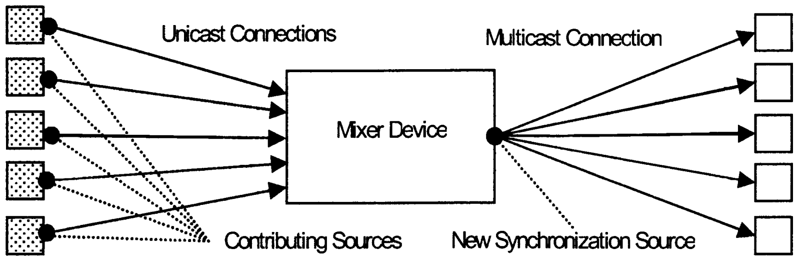


Fig. 8. A mixer device in a multipoint-to-multipoint flow connection.

6 Prototype Implementation

In-line with our objectives in using standard technologies, our prototype implementation of the audio flow control and management components is based on the Java Media Framework (JMF) [5]. JMF provides a unified architecture and messaging protocol for managing the acquisition, processing and delivery of time-based media data. In particular, it provides high-level abstractions of the data sources, sinks, media handlers, processing components, controls, and user interfaces. The framework also defines components for the transmission and reception of audio and video data using RTP [4]. JMF provides facilities for performing the actual processing and transfer of media content, but it does not define interfaces for external (i.e. third-party) control and management of streams. By wrapping JMF components in flow control and management interfaces, the JMF streaming process becomes controllable and manageable. The mixing of multiple content streams is implemented by gathering all the incoming streams under one data source and creating a new outgoing stream with a new synchronization source (SSRC).

Figure 9 illustrates a simple example deployment scenario of flow control and management objects. Device and endpoint objects execute on or closest to the devices. Flow control and management objects exist independently of devices. From the manager's viewpoint, it is irrelevant whether the device is controlled via a gateway or a management agent (See section 3.1). All that matters is that the interfaces defined by the OMG Audio/Video Streams Specification are supported at its point-of-presence. Every other implementation detail is hidden from the manager. The management station is responsible for managing all flow control and management objects within the production studio, which are the objects used to facilitate the integration of devices, and the management of the streaming process thereafter. For example, to establish a flow between the microphone and audio mixer (the hub where flows are mixed), the manager obtains a reference to the target flow control and management object (a potential flow) and calls its "connect-dev" operation passing the microphone and audio mixer references as parameters. The flow control and management object mediates between the device endpoints to establish the binding (Mediator interactions). Once the binding is established, the flow control and management provides the manager with the ability to control the distributed endpoints through a unified interface (Facade interactions).

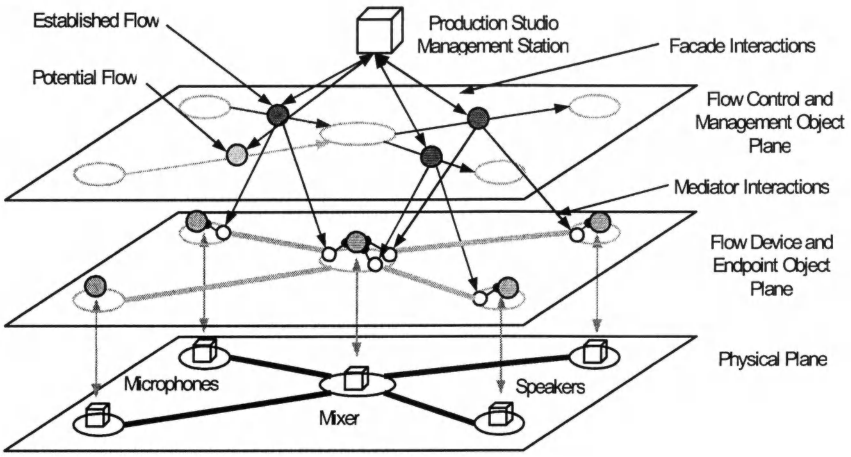

Established Flow
Potential Flow
Production Studio Management Station
Facade Interactions
Flow Control and Management Object Plane
Mediator Interactions
Flow Device and Endpoint Object Plane
Physical Plane
Microphones
Mixer
Speakers

Fig. 9. Conceptual model of a simple example deployment scenario.

7 Conclusions

In this paper, we investigate the applicability of the Audio/Video Streams Specification to a production studio environment. In particular, we describe the design and implementation of components for the control and management of audio flows within an IP broadcast environment for the purpose of program production. The production studio's layout is modelled as a set of flow control and management objects that identify the possible bindings into the studio environment. We emphasize the roles of a control and management object in the OMG Audio/Video Streams Specification's architecture using two design patterns. As a mediator, it provides the necessary services for smoothly integrating peer devices and endpoints into the production studio during the set-up phase. As a facade, it provides services for controlling and managing participating endpoints during the normal program production process. The complexity of binding, controlling, and managing the distributed endpoints of flows is hidden from the studio manager; thus removing the need for the studio manager to interact with devices and endpoints individually. The studio manager can focus on content management instead of device management. The correlation between a flow control and management object and its content, as well as the need for mixer operations and how this affects the flow topology is also described. Because the current model is based on full-profile components, it can be extended in future work to include additional flow types and subsequently incorporated into the existing arrangement. Our prototype implementation is based on the Java Media Framework, which provides support for the delivery of time-based media via the Real-Time Transport Protocol. Being CORBA-based, the architecture allows for considerable flexibility as regards to the implementation, distribution, and management of control and management objects, particularly in the design of ad-hoc solutions for flow and stream establishment. Because of the highly relational nature of the application, the manager needs a management system that heightens the observability and controllability of the network, in real-time. For this reason, we are developing a virtual world supported by a CORBA-based graph drawing system for the visualization and control of the production studio.

References

1. Object Management Group: Audio/Video Streams Specification v1.0. OMG Domain Specifications, Telecommunications, 00-01-03, January (2000)
2. Object Management Group: The Common Object Request Broker: Architecture and Specification v2.6. 01-12-35, December (2001)
3. Mungee, S., Surendran, N., Schmidt, D. C.: The Design and Performance of a CORBA Audio/Video Streaming Service. HICSS-32 International Conference on System Sciences, minitrack on Multimedia DBMS and the WWW, Hawaii, January (1999)
4. Schulzrinne, H., Casner, S., Frederick, R., Jacobson, V., RTP: A Transport Protocol for Real-Time Applications. Internet Engineering Task Force, RFC1889, January (1996)
5. Java Media Framework API Specification, Version 2.0, FCS. 10 March (2001)
6. Erich, G., et al.: Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley (1995)
7. Object Management Group: Property Service Specification v1.0. 00-06-22, April (2000)
8. Munro, A. (Editor) et al.: Studio Production Networking and the Internet – State of the Art Report. Technical Report D1, PRONET LINK project, February (2000)
9. Orfali, R., Harkey, D.: Client/Server Programming with Java and CORBA, Second Edition. John Wiley & Sons, Inc. (1998)
10. Hoque, R.: CORBA 3: Developing Industrial-Strength Client/Server and Web Applications. IDG Books Worldwide Inc. (1998)
11. Siegel, J. (Ed.): CORBA 3 Fundamentals and Programming, 2nd Edition. Wiley (2000)
12. Kaleshi, D., Munro, A.: Studio Production Networking and the Internet – Synchronization Considerations. Technical Report D7, PRONET LINK project, September (2000)
13. Ooms, D., Sales, B., Livens, W., Acharya, A., Griffoul, F., Ansari, F.: Overview of IP Multicast in a Multi-Protocol Label Switching (MPLS) Environment. Internet Engineering Task Force, RFC 3353, August (2002)

IP Video Streaming With Fine-Grained TCP-Friendly Rate Adaptation

Toufik Ahmed[1,2], Ahmed Mehaoua[2], Raouf Boutaba[1], and Youssef Iraqi[1]

[1]University of Waterloo, Dept. of Computer Science
200 University Avenue West, Waterloo,
Ont. N2L 3G1, Canada
{tad, rboutaba, yiraqi}@bbcr.uwaterloo.ca
[2] University of Versailles, CNRS-PRiSM Lab.
45 av. des Etats-Unis, 78035, Versailles, France
{tad, mea}@prism.uvsq.fr

Abstract. Video streaming over the Internet is a challenging task since the Internet is a shared environment offering only best effort service. That is, it offers no quality of service and no guarantee of resources in term of (1) bandwidth, (2) transfer delay, (3) delay variation (jitter), and (4) packet losses. Then, network stability and traffic fairness become critical issues. To solve these problems some source rate control and adaptation should be introduced for UDP traffic as well, in such a way that this traffic becomes TCP-compatible "TCP-friendly". In this article we propose an adaptive streaming framework for unicast MPEG-4 streams over TCP/IP networks. Based on Audio-Visual Content (AVOs) classification and network congestion feedback, video sources dynamically adds and drops MPEG-4 AVO to the streamed multiplex to conform to the TCP-Friendly Rate Control (TFRC) mechanism. Using a content classification model, TFRC automatically adjusts the number of AVOs to be streamed to adapt to network congestion while given much attention to the quality of the service perceived by the end-user. To achieve such a dynamic output rate and video quality adjustment, MPEG-4 AVOs are classified and multiplexed according to both application-level QoS parameters and AVOs semantic descriptors. AVOs requiring same QoS from the network are automatically classified and mapped to one of the available IP DiffServ PHB (Per Hop Behaviors). Performance evaluation shows that transmitted video gracefully adapts to network bandwidth variations while optimizing user perceived quality.

Keywords. Video Streaming, TCP-Friendly Rate Control, QoS, Object Classi fication Model.

1 Introduction

Streaming audio and video on the Internet is becoming more popular. This rapid expansion underlies a new challenge for efficient handling of Internet traffic. The majority of multimedia applications perform over an RTP stack that is implemented on top of UDP/IP. However, UDP offers no congestion control mechanism and therefore is unaware of network condition and unfair towards other competing traffic.

Today's Internet traffic is dominated by TCP. TCP uses several mechanisms to handle network congestion such as: AIMD (Additive Increase and Multiplicative Decrease), slow start, congestion avoidance, fast retransmit and fast recovery. Thus, it is crucial that UDP traffic performs also TCP-friendly congestion control [1].

In this article, we consider the application scenario of streaming object oriented video over IP with congestion control mechanism. We investigate Quality of Service (QoS) interaction provisioning between an MPEG-4 video application and the IP Diffserv network. To achieve the best possible QoS, all the components involved in the transmission process must collaborate together. In this regards, we propose the following mechanism. First, the streaming server classifies the MPEG-4 Audio Visual Objects (AVOs) based on application-level QoS criteria and/or AVOs semantic descriptors according to MPEG-7 framework [2]. Second, the server performs rate adaptation through the adjustment of the number of streamed AVO based on network state. We use a TCP-friendly to adapt the server rate to network condition. The server tries to deliver the maximum number of objects that can fit in the current allowed transmission bandwidth. It begins streaming the important AVO after what it streams the less important AVO. This mechanism allows the server to deal with network congestion by stopping streaming less important AVO when congestion is detected. Finally, the server affects one of the IP diffserv classes of service that guarantee a good quality of service, using an intelligent marking scheme. The more the AVO is important is the scene, the more the server affects low drop precedence class. When network congestion occurs less important AVOs will be dropped automatically by the diffserv router. Lost packets notify the streaming server to reduce its transmission rate.

The idea of adaptive streaming using TCP-friendly helps to prevent the server entering congestion collapse in which the network link is heavily used and little useful work is being done. So to prevent such situation, all applications must perform TCP-like congestion control mechanisms. Traffic that does not perform in TCP-friendly manner is dropped by the router [3].

The paper is organized as follows: Section 2 reviews some background and related work. The proposed adaptation mechanism is presented in Section 3. Simulation model and performance analysis are presented in Section 4 and 5 respectively. Finally, Section 6 concludes the paper.

2 Background and Related Work

Adaptive video streaming is not a new topic. Several researches have been conducted in this area and various approaches have been proposed. While all the work done until now concerns essentially layered video streaming, our work is different from others by using the new concept of object scalability introduced in the MPEG-4 standard.
Video scalability is the key approach for achieving rate adaptation. Many works exploit the video scalability to implement the rate control via TCP-Friendly congestion management. McCanne et al. [4], propose a method that enables different receivers to adapt to bandwidth fluctuations by adjusting the number of layers to which multicast users subscribe. The video stream is divided into a number of multi-resolution layers. Each layer is transmitted to a separate multicast group.

Another type of server rate adaptation using the video scalability is to adjust the codec quantization parameters. In [5] and [6] the video server continually negotiates the available bandwidth and modifies the codec quantization values accordingly. We recall that adapting codec quantization values is a CPU-intensive task which can affects the performance of the video server. The idea of quantizer scale was also used in the context of MPEG-4 in the work presented in [7]. This later employs Fine Granular Scalability which uses a layered coding algorithm. A TCP-friendly rate control algorithm adjusts the rate of each video layer by regulating the level of quantization.

In order to apply a video rate control, it is necessary to have a scheme that returns the transmission rate. The Loss-Delay Based Adaptation Algorithm (LDP) [8] is an end-to-end rate adaptation scheme for adjusting the transmission rate of multimedia applications to the congestion level of the network. Another class of congestion control schemes applies the additive increase, multiplicative decrease (AIMD) algorithm in some form [9],[10]. The Rate Adaptation Protocol (RAP) proposed by Rejaije et al. [9] is an example of an AIMD-based scheme, where the source performs rate adaptation based on acknowledgments sent by the receivers (feedback information). The acknowledgments are used to detect losses and estimate round trip time (RTT). Rate adaptations are performed once per RTT, with transmission rate being increased linearly in the absence of loss, and transmission rate being decreased multiplicatively when congestion is detected. RAP uses the ratio of short-term to long-term averages of RTT to fine tune the sending rate. The RAP protocol was applied in the context of unicast video delivery [11]. The video is a layered constant-bit rate. All the layers have the same throughput. The rate control algorithm used by the server adapts the video quality to network state by adding and dropping layers to efficiently use the available bandwidth. The algorithm takes into consideration the status of the receiver buffer, making sure that base layer packets are always available for playback.

The TCP-friendly congestion control mechanism that was developed recently is TCP-friendly Rate Control Protocol (TFRC) [12]. TFRC provides sufficient responsiveness by taking into consideration all the parameters that affect the TCP rate such as packet size (***s***), loss rate ***(p)***, Round-Trip Time (RTT), retransmission timeout value ($\boldsymbol{t_{RTO}}$), and number of packets acknowledged by a single packet (***b***). The key advantage of TFRC is that it has a more stable rate during the session lifetime. The calculated rate is obtained by using the TFRC is [12]:

$$R_{TCP} \cong \frac{s}{RTT\sqrt{\frac{2bp}{3}} + t_{RTO}(3\sqrt{\frac{3bp}{8}})p(1+32p^2)} \qquad (1)$$

In contrast to previous adaptive video streaming mechanisms, the proposed approach in this paper uses the concept of object scalability introduced in MPEG-4. It adapts the video quality to network state by adding or dropping objects and their associated layers according to the network state. This solves the problems of heterogeneity of receivers and redundancy of data. Objects are encoded separately which does not prevent one object from being decoded if another one is not received.

3 Object-Based Adaptive Video Streaming Using TCP-Friendly Mechanism

3.1 MPEG-4 Framework

The MPEG-4 standard [13] introduces a new technique of coding multimedia scenes called “object-based compression”. This technique allows the encoding of different audio-visual objects in the scene independently.

An MPEG-4 scene consists of one or more AVOs, each of them is characterized by temporal and spatial information. The hierarchical composition of an MPEG-4 scene is depicted in Fig. 1. Each Video Object (VO) may be encoded in a scalable (multi-layer) or non scalable (single layer) form. A layer is composed of a sequence of a Group of Video-Object-Plane (GOV). A Video Object Plane (VOP) is similar to the MPEG-2 frame. VOP supports intra coded (I-VOP) temporally predicted (P-VOP) and bi directionally predicted (B-VOP)

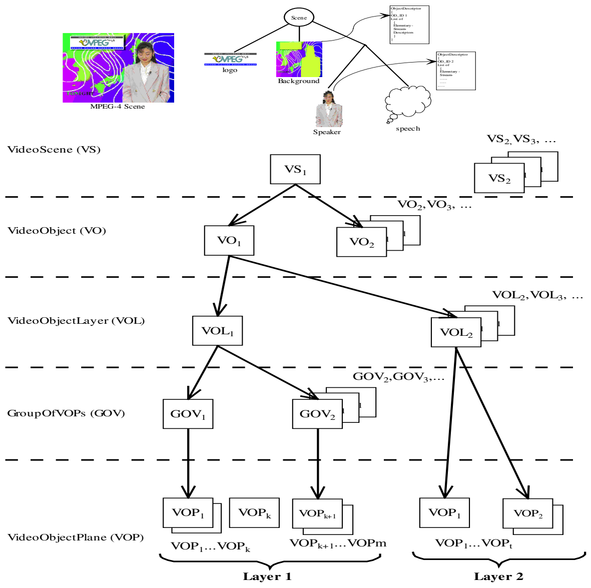


Fig. 1. Hierarchical composition of an MPEG-4 scene

To take benefits from the object-based compression, we propose to use an intelligent adaptation to cope with network congestion and client terminal heterogeneity. We proposed in [14] a mechanism to classify the MPEG-4 AVOs at the video server from most important AVO to least important AVO. Several methods can be used for objects classification. During scene creation, one can affect the adequate priorities to each object in the scene. For scenes with no assigned object priorities, MPEG-4 object descriptors or MPEG-7 [2] can provide the relevant information needed to handle object priority. By classifying these objects, we provide a first level of scalability called object scalability. It gives the server the ability to add and drop video objects dynamically and deal with network congestion intelligently. This technique is presented in next subsections.

3.2 Notations and Parameters

Let $\boldsymbol{S}$ be a set of MPEG-4 AVOs containing $\boldsymbol{n}$ AVOs $\boldsymbol{O}_j$, with $j\in\{\boldsymbol{1}, \boldsymbol{2}\ldots\boldsymbol{n}\}$. Without loss of generality, we assume that these objects are sorted in a decreasing order of priority using our classification process [14]. Each object $\boldsymbol{O}_j$ may consist of $\boldsymbol{m}_j$ layers ($\boldsymbol{m}_j$ •1). Note that lower layers within an object have higher priorities than higher layers.

Let **P** be the function that returns the priority of a particular object or layer. Without loss of generality, we assume that:

$$\begin{aligned}&\forall j, 1 \le j < n : P(O_{j+1}) \le P(O_j)\\&\forall j, 1 \le j < n, \forall l, 1 \le l < m_j : P(L_{j,l+1}) < P(L_{j,l})\end{aligned} \quad \boldsymbol{L}_{j,l} \text{ is the Layer number } \boldsymbol{l} \text{ of the Object } \boldsymbol{O}_j \qquad (2)$$

By using Eq. 2 we can construct an Audio-Visual Entity set called $\boldsymbol{E}$ composed of all object layers ordered by their priorities.
$E=\{L_{1,1}, L_{1,2}\ldots L_{1,m1}, L_{2,1}, L_{2,2}\ldots L_{2,m2}, \ldots, L_{n,1}, L_{n,2}\ldots L_{n,mn}\}$. We will note $\boldsymbol{E}$ as follows:
$E=\{e_1, e_2, \ldots, e_W\}$ with $w=|E|=\sum_{j=1}^{n} m_j$

Note that if two objects have the same priority, then the associated layers of an object have the same priority as the object (in relation to other objects) with the lower layers having higher priorities than higher layers.

At time t_i, the function $\boldsymbol{R}_i$ gives the instantaneous transmission rate of an audio-visual entity. For example, the audio-visual entity $\mathbf{e}_p$ has an instantaneous transmission rate equal to $\boldsymbol{R}_i(e_p)$, and the object $\boldsymbol{O}_j$ has the instantaneous transmission rate equal to $\boldsymbol{R}_i(\boldsymbol{O}_j)$.

The adaptation mechanism operates as follows: The server evaluates the network state from the information gathered (i.e. RTT and loss rate) at time t_i, then computes the allowed sending rate $\boldsymbol{R}_{TCP}$ using Eq. 1. The server tries to send as much as possible of the audio-visual entities without exceeding $\boldsymbol{R}_{TCP}$ taking into consideration entities priorities. Details of the adding and the dropping process will be presented in section 3.4 and 3.5 respectively.

3.3 Example

Assume that we have an MPEG-4 scene composed of four audio-visual objects: $\boldsymbol{O}_1$, $\boldsymbol{O}_2$, $\boldsymbol{O}_3$ and $\boldsymbol{O}_4$. Assume that $\boldsymbol{O}_1$ is composed of a single layer, and that each of $\boldsymbol{O}_2$, $\boldsymbol{O}_3$ and $\boldsymbol{O}_4$ is composed of three layers (one base layer and two enhancement layers). Also assume that the classification layer associates AVO priorities as follows (see Fig. 2):

- $\boldsymbol{O}_1$ is the most important
- $\boldsymbol{O}_2$ and $\boldsymbol{O}_3$ have the same priority
- $\boldsymbol{O}_4$ is the less important

Then, $E=\{L_{1,1}, L_{2,1}, L_{3,1}, L_{2,2}, L_{3,2}, L_{2,3}, L_{3,3}, L_{4,1}, L_{4,2}, L_{4,3}\} = \{e_1, e_2, \ldots, e_{10}\}$. Here w=10.

The video server adds audio-visual entities in the order of their importance (i.e. form left to right in the set E) as shown in see Fig. 2. Entities are dropped in reverse order (i.e. form right to left) until matching the target sending rate.

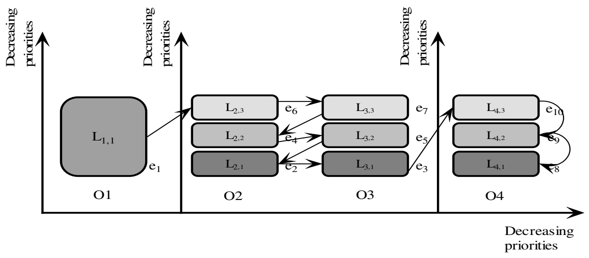


Fig. 2. Transmission Order of different video objects/layers

3.4 ADD of Audio-Visual Objects

The server adds a new audio-visual entity as soon as the target rate exceeds the current sending rate of current entities plus the new entity. Assume that the server is streaming ***k*** entities at time ***t_i***. We assume also that the client has sufficient resources to play all the entities being sent by the server. Therefore, at time ***t_{i+1}*** the server can add a new entity while the following condition is satisfied:

$$\sum_{j=1}^{k+1} R_{i+1}(e_j) \le R_{TCP} \tag{3}$$

At the client side, the new audio-visual entity must be buffered and synchronized to the current playback time. For this fact, the server must start streaming with a synchronization marker of a synchronization frame such as I-VOP. The estimated throughput of the object being streamed is also considered. This allows assuring that the object being streamed has allows sufficient short-time resources. Section 3.6 gives more details about stability management.

3.5 DROP of Audio-Visual Objects

When the estimated throughput of the TCP session indicates that the video server is transmitting more data than it should, then the video server must reduce its sending rate by dropping one or more audio-visual entities. Therefore, the server drops entities while the following condition is satisfied:

$$\sum_{j=1}^{k} R_{i+1}(e_j) > R_{TCP} \tag{4}$$

The server can achieve a minimum throughput by assuring that the most important object or the base layer is always streamed a least. Assuring a minim throughput gives a minimum quality at the receiver side.

3.6 Handling Stability

Since the TFRC compute the new target rate each RTT, adding and dropping audio-visual entities can lead to undesired oscillation and poor video quality at the receiver. To prevent from such behavior, several measures are taken into consideration.

First, the TFRC module copes with oscillation behavior by using EWMA (Exponentially Weighted Moving Average) to detect out-of-control situations quickly. EWMA statistics are used to attempt to respond dynamically to the changing value in the measured RTT and loss and attempt to regulate this value to reflect as much as possible the reality. In TFRC, the loss rate is measured in terms of loss interval which represents the number between two consecutive loss events. The mechanism reacts too strongly to single loss events and ensures that allowed sending rate do not change aggressively.

Second, we propose to adjust the server transmission rate at the beginning of each GOV (Group of video object plane). So the synchronization is assured by starting the transmission with I-VOP. Thus, the new transmission rate obtained from TFRC module is used to adapt video sending rate. Fig. 3 shows four GOVs (each group has twelve VOPs). The average line in the Figure shows the server transmitting rate at the beginning of each GOV of a current Video Object (VO). If this value does not fit in the current available bandwidth then the server does not stream the object. In other words, the server smoothes its sending data to reduce its aggressivity.

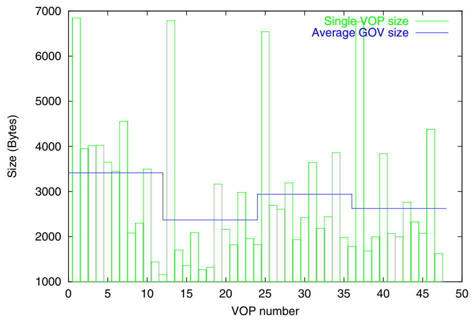


Fig. 3. Stability management

3.7 System Architecture

Fig. 4 depicts the general block diagram of our MPEG-4 Video on Demand system. It is composed of a video server and a video client. The server streams the audio-visual object to the client via an IP network using the RTP protocol [15]. The client decodes and composes the original MPEG-4 scene. As shown in Fig. 1 each AVO is coded separately so the decoding process decodes also each AVO separately and then the composition module composes the original scene. The target transmission rate of the video server is calculated by the TFRC module. This information is sent to the "add/drop module" which adapts the video transmission rate using add/drop algorithms. IP Diffserv Marker module handles the marking of the different RTP packet with Diffserv Code Point before entering the Diffserv network.

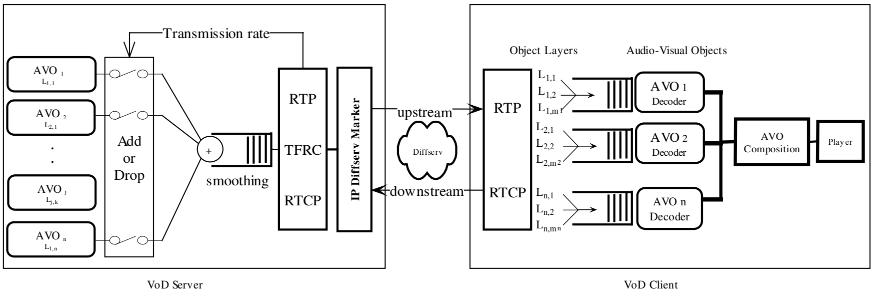


Fig. 4. General Block Diagram of the MPEG-4 VoD System

Diffserv object prioritization aims to privilege the transport of some AVOs compared to others. When network congestion occurs, less important AVOs streams are dropped automatically by the active queue implemented in the Diffserv router. The work detailed in [16] presents a method to handle a layered MPEG-4 stream over IP Diffserv. We extend this approach by handling MPEG-4 AVOs streams prioritization over IP Diffserv network. Recall that the MPEG-4 scene contains many MPEG-4 AVOs sorted according to their importance in the presentation. Therefore, the IP Diffserv Marker tags each video data packet belonging to one AVO with one of the supported Diffserv class of service to reflect object priority. Hence, important objects will be marked with a low drop precedence to guarantee a minimum loss.

It is worth noting that the choice of TCP or TFRC-based congestion control mechanisms is completely orthogonal to whether the traffic is best-effort or not. If the transport protocol is using conformant end-to-end congestion control, then the transport protocol does not have to know whether the traffic is being treated as best-effort or as part of a Diffserv class.

4 Simulation Model

4.1 Network Architecture

Simulations are conduced using the network simulator *ns2*. We used the network architecture shown in Fig. 5 to simulate a unicast service provided by the MPEG-4 server attached to the node “S”. The server sends data to the client attached to the node “C”. We developed an MPEG-4 server in ns2 with TFRC capability. The client is also developed in ns2 and which extends the capabilities of the RTP sink by reporting statistic information to the server. The network is loaded by *n* FTP streams carried over TCP (*n* ranges from 0 to 8). This allows the link between the routers “R1” and “R2” to be on congestion differently. FTP sources always have a packet to send and always send a maximal-sized (1000-bytes) packet as soon as the congestion control window allows them to do so. FTP sink immediately sends an ACK packet when it receives a data packet. The queue in the routers has a size of 50 packets. The core IP Diffserv router examines incoming packets and reacts according to the marking, whereas "R1" is an edge router that implements Marking/Classification policy on incoming packets. R1 uses A Two Rate Three Color Marker (**TR3CM**) [17] to mark the background. Therefore, background traffic is evenly distributed among the different Diffserv classes. We recall that the video traffic is marked at the MPEG-4 server according to AVOs priorities. The bottleneck link between the core router and R2 has a 5 Mbit/s of bandwidth.

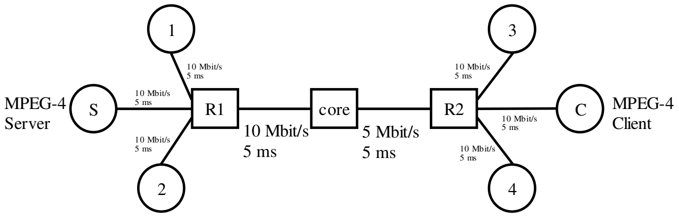


Fig. 5. Network topology

4.2 MPEG-4 Traffic Model

The MPEG-4 traffic is obtained from the MPEG-4 trace file presented in [18]. In our simulation, the MPEG-4 presentation was obtained by using a set of AVOs components. We simulate the weather presentation shown in Fig. 1 by using four multimedia objects: AO (audio speech), VO1 (background), VO1 (speaker) and VO3 (logo). These objects are sorted as follows:

- AO has the priority 1, it is the most important object in this scene. It is marked with Diffserv PHB AF11 (low drop precedence).
- VO1 and VO2 have the priority 2. They are marked with Diffserv PHB AF12 (medium drop precedence). Each Object is composed of 3 layers (one base layer and 2 enhancement layers)
- VO3 has the priority 3, it is the least important object in this scene. It is marked with Diffserv PHB AF13 (high drop precedence).

Fig. 6 shows the bit-rate of the MPEG-4 video objects that can be sent from the MPEG-4 server to the client during a period of 120 seconds. The complete scene is shown in Fig. 6 (a). The Audio Object is a constant bit rate at 64Kbits/s. An Audio packet is sent each 125ms. Video object 1 has an average throughput of 200 Kbit/s and a peak rate of 956 Kbit/s. This object is composed of three Layers: BL (Base Layer), EL1 (Enhancement Layer 1) and EL2 (Enhancement Layer 2). The throughputs of the different layers are shown in Fig. 6 (b). Video object 2 has an average throughput of 650 Kbit/s and a peak rate of 1722 Kbit/s. This object is composed of three Layers: BL, EL1 and EL2. The throughputs of the different layers are shown in Fig. 6 (c). Video object 3 has an average throughput of 124 Kbit/s and a peak rate of 356 Kbit/s. It is composed of one single layer (see Fig. 6 (c)).

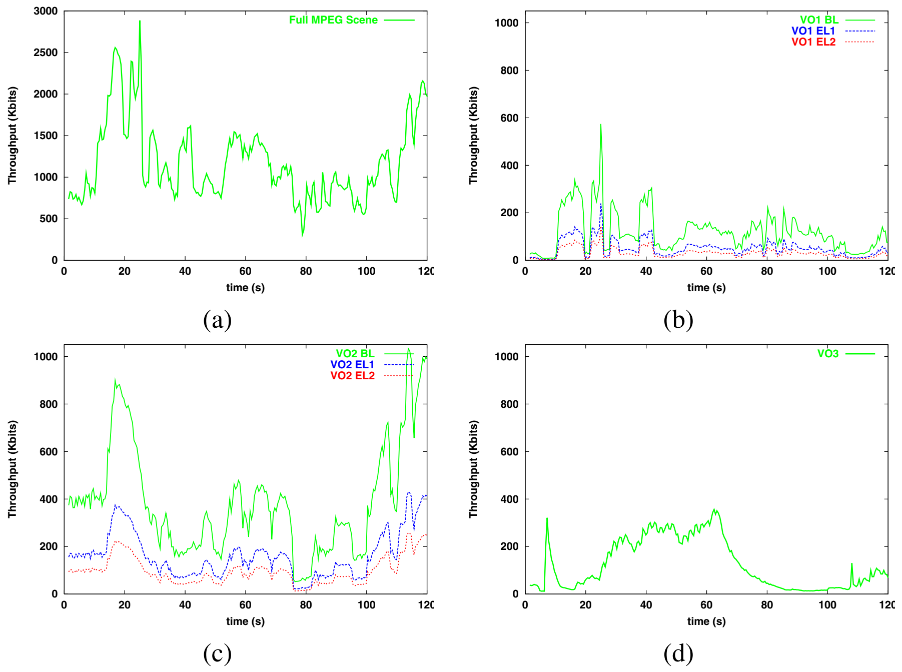


Fig. 6. Instantaneous throughput of the different MPEG-4 Video Object

5 Simulation Analysis

We perform an intensive simulation, each time with different parameters to see the behavior of our video on demand system. We vary the number n of FTP source according to following scenarios: (1) **Scenario A**: one FTP source; (2) **Scenario B**: two FTP sources; (3) **Scenario C**: four FTP sources; (4) **Scenario D**: eight FTP sources. FTP sources send data from time t=30s until time= 90s. We present only results of scenario D due to the lack of spaces.

This section presents some QoS measurement such as, the video server throughput as a function of network state, packet loss and end-to-end packet transmission delay.

5.1 Video Server Throughput

The video server regulates its transmission rate to reflect the allowed rate by adding or dropping audio-visual entities. Results obtained of the different scenarios are shown in Figures below. Also, to simplify the interpretation of the results, Table 1 summarizes the transmission ratio per AVO stream observed during the period of the simulations (120s). Note that the FTP sources begin data transmission at time t=30s, and stop at time t=90s. VO3 has the low ratio since it has the lowest priority is the scene. VO1 and VO2 have the same priority, so the corresponding layers have more or less the same transmission ratio.

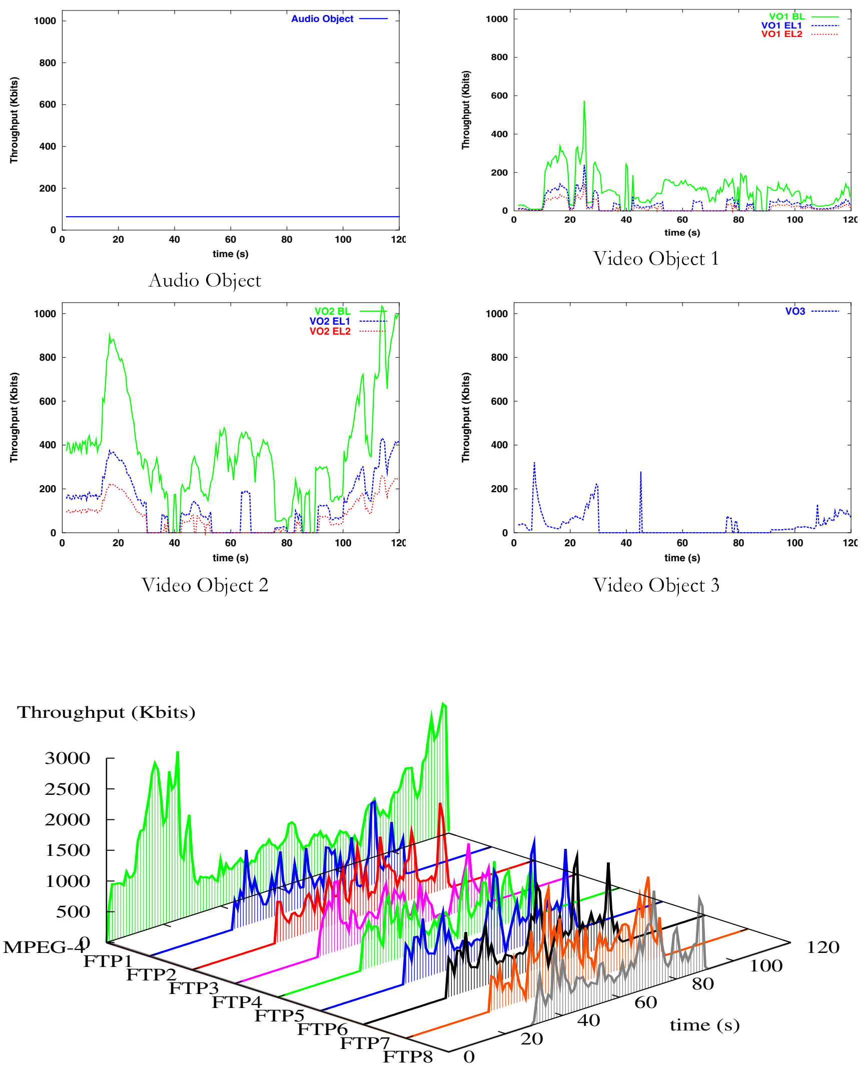


Fig. 7. Scenario D

Scenario D is interesting since we see the effect of our adaptation mechanism. We can see that the audio object is always present and that less important objects (respectively object layers) are not transmitted when the shared bandwidth is not sufficient. Our adaptation mechanism begins transmitting data from important audio-visual entity to less important. We can see that all the streams (FTP and video) fairly share the bandwidth.

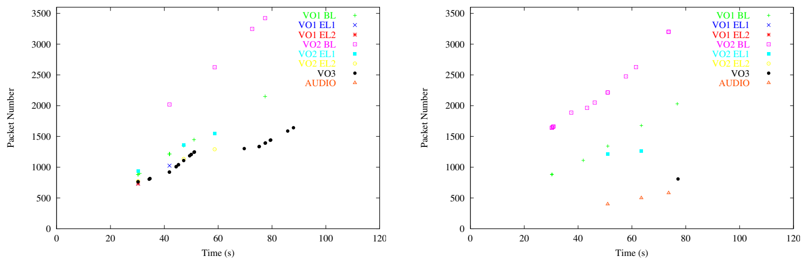
AVO / Scenario	Audio	VO1			VO2			VO3
		BL	EL1	EL2	BL	EL1	EL2	
A	100%	100%	100%	100%	100%	100%	100%	100%
B	100%	100%	100%	100%	100%	100%	100%	100%
C	100%	100%	94%	87%	100%	96%	92%	55%
D	100%	89%	60%	53%	97%	77%	71%	26%

Table 1. Transmission ratio per MPEG-4 objects

Without using our adaptation mechanism, the server transmits the audio-visual entities without any regulation as shown in Fig. 6. The loss may increase and the network may enter in congestion collapse.

5.2 Packet Loss

Fig. 8 shows lost packets for scenarios C and D using our adaptation mechanism. Scenario A does not experience any loss. In scenario B, some lost packets are observed on VO3. This is due to the active queue of the DiffServ router which drops lower priority packets when a predefined threshold is reached to prevent congestion. In scenario C, we observe also some loss on lower priority packets but in scenario D high priority packet are also dropped. This is due to: (1) lower priority packets are not transmitted because our adaptation mechanism regulates the server transmission rate by stopping streaming lower priority packets and (2) AVO1 and AVO2 request more bandwidth in our scene and cause some congestion.



Scenario C Scenario D

Fig. 8. MPEG-4 Packet Loss Using Our Adaptation Mechanism

6 Conclusion

Most of the work done on "TCP-friendly" mechanisms deals with the problems of network stability and traffic fairness. Few attentions have been paid to the applications using them, and in particular, to the quality of service perceived by end users. Therefore, in this article we proposed an adaptive video streaming framework that provides both output rate and visual quality control for MPEG-4 based multimedia applications. The proposed framework relies on two cooperative components, (1) a TCP-Friendly Rate Control mechanism that estimates the targeted output rates for video sources; (2) and media object classification model that automatically adds and drops MPEG-4 Audio-Visual Objects to the multiplex stream to match the allowed source bit rate. These components are combined to form a new "Media Classification Layer" that improve the MPEG-4 system architecture over QoS-capable IP networks. Performance evaluation through intensive simulations and various IP Diffserv configurations, shown a significant reduction of the video packets loss during network congestion, a smooth video source rate adaptation, a fair bandwidth sharing among UDP and TCP streams, and an increase in the Quality of the service with integration of Object descriptors (OD) during the media classification and the multiplexing process.

References

1. Mahdavi, and S.Floyd "TCP-Friendly Unicast Rate-Based Flow Control", Technical note sent to the end2end-interest mailing list, Jan., 1997.
2. ISO/IEC JTC1/SC29/WG11, "MPEG-7 overview," N4980, Jul. 2002.
3. S.Floyd, M. Handley, J. Padhye, and J. Widmer "Equation-based congestion control for unicast applications" Proc. of ACM SIGCOMM, pages 43–56, Aug. 2000.
4. S. McCanne "Scalable compression and transmission over Internet multicast video" Ph.D thesis University of California, Berkeley, Dec. 1996.
5. T.V.Lakshman, P.P.Mishra, and K.K.Ramakrishnan "Transporting compressed video over ATM networks with explicit rate feedback control," in *Proc. IEEE* Infocom 97, pp. 38–47, Apr. 1997.
6. N.G.Duffield, K.K.Ramakrishnan, and A.R. Reibman "SAVE: An Algorithm for Smoothed Adaptive Video Over Explicit Rate Networks" IEEE/ACM transactions on networking, vol. 6, no. 6, Dec. 1998.
7. N.Wakamiya, M.Miyabayashi, M.Murata, and H.Miyahara "MPEG-4 Video Transfer with TCP-friendly Rate Control" in IFIP/IEEE MMNS 2001, pp 29-42, Oct. 2001.
8. D.Sisalem, and H.Schulzrinne "The Loss-Delay Adjustment Algorithm: A TCP-friendly Adaptation Scheme, Network and Operating System Support for Digital Audio and Video" (NOSSDAV), Cambridge, UK, 8–10, Jul. 1998.
9. R.Rejaie, M.Handley, and D.Estrin "RAP: An End-to-end Rate-based Congestion Control Mechanism for Realtime Streams in the Internet" Proc. IEEE Infocom'99, pp. 1337–1345, Mar. 1999.
10. D. Bansal and H. Balakrishnan. Binomial Congestion Control Algorithms. In *Proc. IEEE Infocom '01*, Anchorage, AK, Apr. 2001.
11. R.Rejaie, M.Handley, and D. Estrin "Layered quality adaptation for Internet video streaming" IEEE Journal of selected areas in communications, vol. 18, No. 12, Dec. 2000.
12. M.Handley, S.Floyd, J.Padhye, and J.Widmer "RFC 3448, TCP Friendly Rate Control (TFRC): Protocol Specification", Request for Comments, IETF, Jan. 2003.

13. ISO/IEC 14496-1 “Coding of audio-visual objects, Part 1,2, 3”, final committee draft, May 1998.
14. T. Ahmed, A. Nafaa , A. Mehaoua “An Object-Based MPEG-4 Multimedia Content Classification Model for IP QoS Differentiation”, Proc. of 8th IEEE Symposium on computers and communications ISCC’03, pp. 1091–1096, Jul. 2003
15. H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson “RFC1889 RTP: A Transport Protocol for Real-Time Applications”, Request for Comments, IETF, Jan. 1996.
16. T. Ahmed, A. Mehaoua, and G. Buridant “Implementing MPEG-4 video on demand over IP differentiated services” Proc. IEEE Globecom’01, pp. 2489 - 2493, Nov. 2001.
17. J. Heinanen and, R. Guerin “RFC2698: A Two Rate Three Color Marker TRTCM”, Request for Comments, IETF, Sept. 1999
18. Frank H.P. Fitzek, and Martin Reisslein “MPEG-4 and H.263 Video Traces for Network Performance Evaluation” IEEE Network, vol 5, no 6, pp 40–54, Nov. 2001.

A Proposal for Distributed Conferencing on SIP Using Conference Servers

R. Venkatesha Prasad, Richard Hurni, and H.S. Jamadagni

Centre for Electronics Design and Technology, Indian Institute of Science
Bangalore – 560012, India
{vprasad, hsjam}@cedt.iisc.ernet.in, hurni@ieee.org

Abstract. Session Initiation Protocol (SIP) is the *de facto* standard for Voice over IP implementations. Although this standard does not define conferencing directly, many drafts and papers suggest extensions and solutions to this essential service. In this paper, we first define the issues that are to be considered when building a conference service, and then we motivate our work on some of the limitations of the existing suggested solutions. This framework facilitates the conference control and media handling of a VoIP conference and aims to create a scalable and distributed conferencing system both in terms of load and control. The conference control as well as media is distributed over the network using SIP Servers and Conference Servers, which are described. The floor control is made dynamic by using a metric called Loudness Number [10] that provides a feel of a physical face-to-face conference.

1 Introduction

The growth of Internet encouraged many applications to be ported on the packet-switched network. Transporting the telephony services on the Internet is an exciting assignment and in particular the conferencing facility. We believe that VoIP applications are to be seen from two different aspects: (a) from the technology point of view, and (b) the users' perspective, requirements and interactions.

There are a lot of discussions amongst HCI (Human-Computer Interaction) and CSCW (Computer-Supported Cooperative Work) communities, which mainly deal with the ethnomethodological issues of users' social interactions and about the use of ethnomethodology in designs of applications for CSCW. Unfortunately, the aggressive pace of technical advancements has far outstripped development of metrics and techniques for characterizing and evaluating the novel communication environments. This approach ignores the functional utility of the environment that is used for collaboration [2]. Thus, we take an approach that considers both the aspects, namely, the *technical* and the *functional*.

A conference basically connects many Clients on a single call by mixing their audio streams in the conference into a single stream and playing it at each end-user. For the conference establishment, maintaining and termination the selection of a signalling protocol is necessary. But there are many more issues to be considered when building a conference: packet delay, echo, deciding the number of Clients that can be speaking simultaneously in a conference without degrading conference quality, automatic selection of Clients to participate in the conference (floor control), mixing of audio from selected Clients, handling Clients not capable of mixing audio streams and finally playing of mixed audio at each Client.

The contribution of this paper includes putting together all the bits of design information that are proposed in many Internet drafts and the proposed H.323 recommendations for a Multipoint Processor (MP) to come up with a scalable and distributed architecture for a SIP conferencing system, both on the conference control as well as the audio media. We introduce Conference Servers that work with SIP Servers and Clients and propose SIP messages for their communication needs. A conference solution using Conference Servers has been tested on a test bed.

2 Problem Formulation

We primarily take voice-only conferences by looking from the telephony service provider's view of the VoIP without considering Video and Whiteboard media. The main argument of our approach is that there is no crucial need for complex floor control methods for voice-only conference. We believe that the conference quality depends not only on the individual speech quality but also on the ability to allow impromptu interruptions. We believe that assuming only one [4] participant speaks at any time often results in losing the interactivity in a conference. There is a recent study by Radenkovic et al. [16], which identifies this need for allowing many speakers to speak at the same instant, supporting our observation of requirement of a conference. They use distributed partial mixing of the speech streams but the drawback is speech losing its spatial aspect when many streams are unnecessarily mixed. Thus we allow very few streams to be mixed only at the time of playing out at the user terminals.

To be precise, we set out with the following wish list for the conferencing service:

1. Prospective Clients should first "register" with a control point to use the conferencing software (allowing for billing if required);
2. An ongoing one-to-one call should be upgradeable to a conference by adding at least a third participant;
3. The voice traffic on the network should be as low as possible;
4. Many participants are allowed to speak simultaneously (only a few of them are selected to speak since allowing everyone does not increase the interactivity; fixing the maximum *number* of simultaneous speakers is dealt in [13]);
5. Each Client should get the same set of audio streams for mixing, ensuring consistency;
6. It should be simple to add/delete a Client to/from an ongoing conference;
7. The scalability must ensure that a large number of Clients dispersed over a wide geographical area are easily handled;
8. Many modes of conference building must be supported (e.g. participants joining the conferences by themselves or by invitation);

9. Conference architecture should also allow the use of features such as floor control without disturbing interactivity;
10. No assumption about the availability of Multicasting in the network.

In this paper we take all the above requirements, some of them implicitly and some explicitly. We use *Users*, *Clients, User Agents* (SIP terminology) and *front end* interchangeably depending on the context.

3 The Motivation

Ramanathan and Rangan [15] have studied in detail the architectural configurations comparing many ways of building the conferencing architecture taking into consideration the network delay and computation requirements for mixing. In [19], a discussion on the architecture for a distributed multimedia conference as well as required control protocol is reported. An overview of many issues involved in supporting a large conference is dealt in [6]. The floor control is another major aspect that should be taken into account while designing/building a conferencing tool and is well documented in [3]. An IETF draft [18] discusses conferencing models with SIP in the background. Implementation for centralized SIP conferencing is reported in [21]. There is a new approach called partial mixing by Radenkovic [16] that allows mixed and non-mixed streams to coexist allowing every ones' speech to be heard. In all the above proposals, while there are some very useful suggestions, they also have the following limitations:

- In an audio conference, streams from all the clients need not be mixed. Actually, mixing many arbitrary streams [16] from the clients degrades the quality of the conference due to the reduction in the volume (spatial aspect of speech). If the number of streams mixed varies, this would lead to fluctuations in the volume of every individual participant causing severe degradation in quality. There is a threshold on the number of simultaneous speakers above which increasing the number of speakers does not improve the conference quality.
- There cannot be many intermediate mixers, because it introduces a lot of delay by increasing the number of hops, therefore it is not interactive.
- The floor control for an audio conference with explicit instructions for participants to talk would result in the conference becoming excessively an artificial one-way speech rather than a live, free to interrupt physical conference.
- Partial mixing [16] has the same problem as that of mixing when more streams are mixed but to allow impromptu speech, mixing is not done when the network can afford high bandwidth requirements for sending/receiving the streams.
- For large conferences [18, 8] a centralized conference cannot scale up. With Multicasting, clients will have to parse many streams and traffic on clients' network increases unnecessarily.

A discussion about the limitations of existing SIP conference models (end-system mixing, users joining, large multicast conference, dial-in conference server, ad-hoc centralized conferences, dial-out conference server, centralized signalling and distributed media) is provided in [12].

4 Our Approach

We will now highlight our approach towards building a scalable and distributed conferencing system. The two parts that must be taken care of in a VoIP conferencing system are the following: (i) the *front-end,* consisting of the application program running on the Clients' computer (end users) and (ii) the other application programs that facilitate conferencing and conference control providing the *back-end.* We will not deal with the first part in detail as we use a regular User Agent as described in [17]. The latter part, however, will be of great interest as it encompasses both the controlling and the media handling sides. In this outlook, we propose to have a distributed control through SIP Servers (SIPSs), whereas Conference Servers (CSs) are special entities that will handle only the audio part of the conference, including floor control.

4.1 Description of the Involved Entities

We consider a large conference with hundreds of participants that are situated in several *domains* distributed geographically and interconnected by a WAN, such as the Internet. A domain can be seen as a LAN with high-speed data transfer, etc. This distributed architecture asks for distributed controlling and media handling solutions, as centralized systems would not scale for such very large conferences. Looking into these domains in more detail (Fig. 1), we distinguish the following relevant entities:

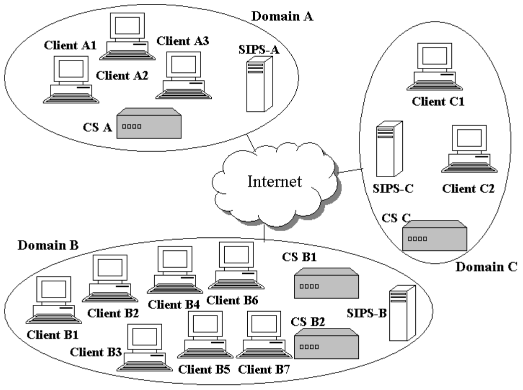


Fig. 1. Description of the different physical components in the proposed architecture.

- An arbitrary number of Clients (User Agents) following the prescriptions in [17].
- A single SIP Server (that will be abbreviated with the acronym SIPS) is present in each domain and is entrusted with all the controlling aspects for the conference(s) under way between one or many Clients in its domain. A SIPS can be viewed as a physical entity that encompasses different logical roles, namely a SIP Proxy Server (which was assumed to be able to generate SIP messages in [12], an assumption that we suppress here because of the presence of the B2BUA), a SIP Registrar Server, a SIP Redirect Server and a SIP B2BUA (Back-to-Back User Agent), which are defined in [17]. This physical implementation enables the handling of in-

coming/outgoing SIP messages by one or another logical entity according to the needs and let the other logical components know about the messages passing through, even if they are not concerned, allowing them to take other actions if necessary. In short, the SIPS can behave like any of the logical roles at any point of time. The B2BUA (and therefore the SIPS) needs to be stateful (in the SIP terminology) and as it is a concatenation of an UAC and UAS, the B2BUA does not need explicit definitions for its behaviour, i.e., it can perform the same tasks and/or handle the same request messages as an UAC or an UAS. The B2BUA, being able to perform more actions than a regular PS, will be the core entity for our conference-related signalling (explained in next subsection).

For the communication needs of the four logical entities in the SIPS, many interfaces are devoted to communication with the domain Clients (User Agents) [12]. Interfaces are also required to communicate with the SIPSs in other domains and with the local Conference Server (see below). Furthermore, objects regarding the calls or conferences currently under way to/from the domain are stored in the SIPS, so that it has complete information about them (fulfilling the stateful condition necessary for the B2BUA as described above). These conference objects (fully described in [12]) contain relevant conference-level information, a list of all the Clients taking part in this conference, irrespective of their local or remote location. A list of the SIPSs from other domains involved in that conference and a list of the CS(s) operating in the local domain are included as well.

The SIPS taking care of all the control aspects has many advantages, viz., (a) it works as a centralized entity that can keep track of the activities of the UAs in a call/conference; (b) it does all the switching for providing PBX features; (c) it locates the UAs and invite them for a conference; (d) billing can be done as well.

- A Conference Server (CS) is a proposed entity that comes into the picture only for the media handling part. Similar to SipConf in [21], a CS has the function of supporting the conference; it is therefore responsible for the audio part, handling audio streams using RTP [20]. A CS only takes commands from its peer SIPS. It can also be used to convert audio stream formats for a given Client if necessary and can work as *Translator/Mixer* of RTP specification behind firewalls. Even if the SIP specification enables direct UA-to-UA media communication in a one-to-one call, it is also possible to use the Conference Server for two-party calls, especially because it is then more functional to create a real conference by adding a third and subsequently more participant(s).

  A CS may be serving many conferences and many Clients at an instant of time in its domain. However, when the number of Clients in a domain is very large, many CSs can coexist in a single domain, each one of them taking care of a fraction of the active Clients. We could implement a CS on a 1.3 GHz Intel PC running Windows NT that supports up to 300 Clients, which shows that even if the number of clients is large, the number of CSs can be kept small.

  In each CS, the information about all other CSs involved in that conference is stored, along with a flag indicating whether its router supports multicast. A list of all the local Clients connected to this particular CS is also available.

We have based the design of our Conference Server on H.323's Multipoint Processor (MP) [7]. In short, the MP receives audio streams from the endpoints, processes these media streams and returns them to the endpoints. An MP that processes audio shall prepare *N* audio outputs from *M* audio input streams by selecting, mixing, or a combination of these. Audio mixing requires decoding the input audio to linear signals, performing a linear combination of the signals and re-encoding the result to the appropriate audio format. The MP may eliminate or attenuate some of the input signals in order to reduce noise and other unwanted signals. The limitation of H.323 is that it does not address the scalability of a conference as the architecture proposes a cascaded or daisy chain topology [8].

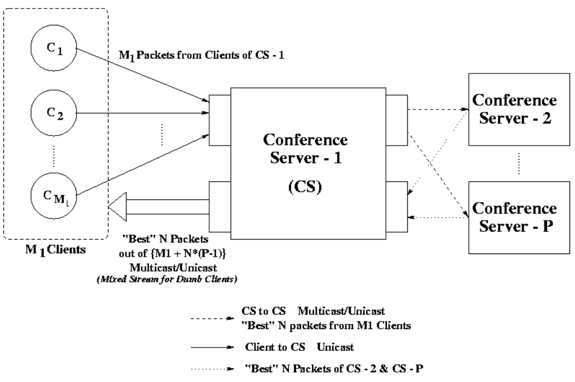

C_1
C_2
C_{M_1}
M_1 Clients
M_1 Packets from Clients of CS - 1
Conference Server - 1 (CS)
Conference Server - 2
Conference Server - P
"Best" N Packets out of {M1 + N*(P-1)} Multicast/Unicast (Mixed Stream for Dumb Clients)
CS to CS Multicast/Unicast "Best" N packets from M1 Clients
Client to CS Unicast
"Best" N Packets of CS - 2 & CS - P

Fig. 2. Schematic diagram of a Conference Server

Continuing with the definition of H.323's MP, the main task of a CS (shown on Fig. 2 and also in [12]) is to pick *N* Clients out of *M* from the local domain that are in the same conference based on some criterion. Audio packets from those *N* Clients, along with their ID are to be sent to the other CSs handling the same conference. Similarly, for each time slot (packet time), a subset *F* of Clients are selected from the pool of packets from all other CSs plus the *N* Clients selected locally and these packets are mixed and played out at the Clients. According to [13], the cardinality of *F*, $|F|$ is *N* and is fixed to three. The flow of audio packets in the system is depicted on Fig. 4 (plain lines).

When comparing our proposal with the ones in the literature, we see that CSs take the commands from a SIPS to which they are attached, similar to Selectors of [9]. However, it makes sense to keep CSs out of all the controlling work from the point of view of a total VoIP service. This is not the case of the Conference Server of [21, 8] which has to do both the work, i.e., conference enabling and floor control.

4.2 Making the Entities Work Together

In order for all the involved entities to work together and provide the desired conferencing service, they need to exchange messages of two different types:

- SIP control messages which are either sent via TCP or UDP, using a timeout in the latter case.
- RTP audio packets, which can be between two UAs in the case of a point-to-point call. Otherwise, first, a Client sends its audio packet to its associated local CS and later CSs exchange their packets (if there are more than one). Then, according to the algorithm defined above, at most three Clients' packets are selected for every time slot and sent on multicast/unicast to other CSs. Then, the CS multicasts the packets back to its associated Clients after selecting the best three streams amongst the streams from other CSs and from its own Clients.

We have already seen that partial or complete information about a conference resides in the User Agents, the SIP Servers and the Conference Servers. To be precise, we draw a list of the conference-level control requirements in the different entities:

- User Agents know the IDs (IP addresses) of their local Register and Proxy Servers (that are fixed and does not have to be dynamically communicated to the UA), which are necessary to start a call or conference. On the other hand, the address of a local CS is not known to the UA in advance, as there can be more than one CS in a domain. There is thus a need to dynamically communicate that address to the UA from the SIPS in its domain.
- A SIP Server knows a lot of information about an ongoing conference: First, it knows the IDs of all registered Clients in a domain and also the IDs of the Clients involved in a given conference. It is also aware of the IP address(es) of the CS(s) handling the media part of the conference(s) in the domain. It also knows the identity of remote SIPS in other domains. Most of the information must be dynamically obtained by the SIPSs when the conference is under way.
- A Conference Server knows the IDs of all the local Clients it must handle the audio packets and the address of remote CSs so that it is able to send a subset of local audio packets in each time slot. As it was decided that CSs would only communicate to their local SIPS for SIP messages, new and dynamic information will have to go through the SIPS before reaching a given CS, and will not go directly between those entities, unlike media packets.

The SIP standard defined in RFC 3261 [17] and in later extensions such as [14] can establish, modify and terminate multimedia sessions. This standard does not offer conference control services, however SIP can be used to initiate a session that uses some other conference control protocol.

For these entities to communicate together and in order to maintain the SIP philosophy in our conferencing system, we assume that the involved entities can both receive and send SIP messages, which is the case of the UAs, the PSs and the B2BUAs that have communication abilities [17]. The only extension needed is to assume that the newly created CSs can send and receive a subset of available SIP requests.

As the different entities need to communicate quite different information between them, we selected a SIP request messages pair that could convey pre-agreed information between these Clients/Servers, namely, SUBSCRIBE and NOTIFY. These two messages are described thoroughly in [14] and can be shortly described as the SIP framework for event notification where entities in the network subscribe to the state of resources and are notified when those states change. The B2BUA will have a central role, as it will be the SIPS logical component entrusted with handling of all the subscriptions/notifications for all the entities within the SIPS.

In Figure 3, we present a diagram showing the messages pair when used. In message 1, the CS subscribes to be notified of events by the SIPS which message body contains a description of the resource it is subscribing to. This first message is acknowledged by message 2. The SIPS immediately sends a NOTIFY message (3) so that the CS knows the present state of the resource. Later, as soon as an event happens to the subscribed resource, messages 5 and 6 (similar to 3 and 4) are exchanged.

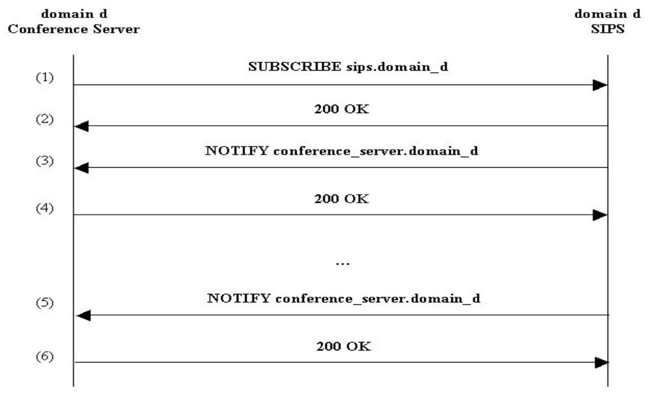


Fig. 3. SUBSCRIBE-NOTIFY message exchange example

In our conferencing environment (and with the help of the defined list of required information every entity is holding), we can draw a precise list of resources a given entity will subscribe to, and also to what other entity it will hold for that particular subscription. The subscription dependencies are drawn on Fig. 4 (dotted lines) and we make the following list as well:

- UAs will subscribe to their local SIPS to be notified to know which CS will handle their audio packets. They may also subscribe to their SIPS to be notified of the joining of every other UAs in order to have a good picture of the conference.
- A SIPS has a subscription with the local registered UAs so that it is informed of changes about their participation in the conference (for e.g. joining, BYE, etc.), making it stateful. They also have to subscribe to the other SIPSs in other domains to pass information to them about the ongoing conference (so it can also provide the local CS(s) with some pertinent information) about UAs joining, and the address of other CSs to pass it on to CSs of its domain. We thus see that by extension all the SIPSs must subscribe with each other (full mesh).
- CSs will subscribe to the local SIPS to be informed about the other CSs present in the conference. It will help in knowing about the UAs, media packets from which it will have to handle in a conference.

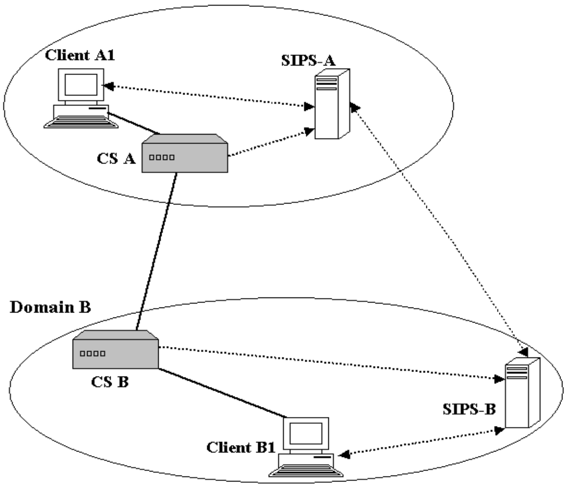


Fig. 4. The message flow between different entities. Signalling messages are shown in dotted lines (with the subscription dependencies), whereas audio packet flow is shown in plain lines.

4.3 Loudness Number (LN)

A basic problem to be solved by the CSs is the following: in a mixing interval, how should it choose *N* packets out of the *M* it might possibly receive? One way to do this would be to rank the *M* packets received according to their energies, and choose the top *N*. This is usually found to be inadequate because random fluctuations in packet energies can lead to poor audio quality. This indicates the need for a metric different from mere individual packet energies that should have the following characteristics:

- A person who is speaking should not be easily cut off by transient spikes in amplitude of the other participants. This implies that a speaker should have some “weight” depending on his past activity (also called “persistence” or “hangover”).
- By the same token, a participant who wants to interrupt the speaker will have to raise his voice and keep trying for a little while in order to break in. In a real-life conference, the body language of a participant often indicates that he wants to interrupt. But in the audio conferencing scenario under discussion, a participant’s intention to interrupt can only be conveyed through the loudness metric on the basis of which the packets to be mixed are selected.

As a result, we use LN for conference control and floor control. A CS selects at most three Clients to speak to the participants of a conference. The LN and its generation are discussed in [10] and as such we have omitted a thorough discussion about it here.

4.4 Hybrid Floor Control

We propose two floor control mechanisms: *(1) Distributed Floor Control* that preserves the interactivity by means of using LN with three floors. The interactivity is preserved due to the design of LN and allowing all the participants to speak impromptu to the participants. *(2) Compensatory Floor Control* is used for avoiding the conference becoming messy, which is a remote possibility. For the purpose, we allow one reserved floor apart from the freely available three. A distributed mutual exclusion algorithm suggested in literature [4] running at PSs may be used for controlling this floor; besides, we propose to grant this floor to the conference moderator (if any, and if required). These two floor controls co-exist since the mixing is done at the clients (or last CSs if there are some dumb clients).

4.5 An Example

We will now introduce a real example of the capabilities of the proposed architecture. We assume the following configuration: two domains, namely `Domain A` and `Domain B` that are interconnected by the Internet. Both those domains have a SIP Server (`SIPS-A` and `SIPS-B`) as well as one Conference Server (`CS A` and `CS B`). Furthermore, there are two potential SIP Clients in `Domain A`, i.e., `Client A1` and `Client A2`, and one in `Domain B`, `Client B1`.

Intra-domain setup phase. We will first introduce the phase that occurs in each individual domain before any conference can take place and is necessary so that all involved entities perform some pre-conference required tasks. The message exchange for domain A is shown on fig. 5, where we can distinguish into 3 different parts:

- Part 1 sees the registration of `CS A` to `SIPS-A`: That way, `CS A` will be informed about UAs joining on one hand and about the addresses of other CSs it will have to send media packets to, on the other hand. The "`NOTIFY CS A`" request contains no information but is compliant with the SIP specification.
- Part 2 and Part 3 consist of each Client registering with its local Registrar Server. Then, the clients subscribe to their local SIPS so that they are kept informed of future calls/conferences and CS handling them. After that, `SIPS-A` registers to the client to be kept informed about its state and leading to a stateful SIPS.

After this initial setup phase, a conference may start at any time inside a domain.

Inter-domain setup phase. After the intra-domain setup phase and before any conference can take place between different domains, an inter-domain setup phase must take place. This phase would actually be contained in a “remote invitation phase” (explained below). When the SIPS at `Domain A` receives the third message of the INVITE/200/ACK three-way handshake used to establish SIP sessions, it is then sure that the remote Client situated in `Domain B` will be included in the conference and thus `SIPS-A` subscribes to `SIPS-B` so that it is kept informed about the events in their respective domains. In part 2, the same occurs in the opposite way. Then, if many more domains are involved in the conference (e.g. `Domain C`), `SIPS-A` will make sure that those other SIPSs know about the joining of the domain:

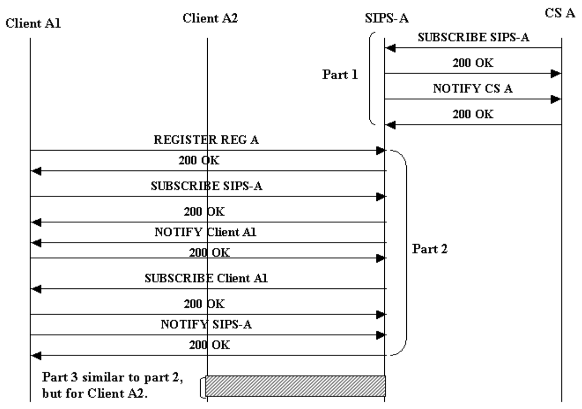


Fig. 5. SIP messaging during intra-domain setup phase

- SIPS-A tells SIPS-B about SIPS-*j* (and all other SIPSs if any). Then SIPS-B would contact each and those would also subscribe to SIPS-B. Whenever SIPS-B subscribes to SIPS-*j*, it would be reciprocated by SIPS-*j* subscribing to SIPS-B.
- In a conference where users join themselves (also called an open conference), then the SDP [5] would have the initiator domain (say SIPS-*j*) to which the new SIPS-*k* would subscribe and get the information about all the other SIPSs.

However, only one inter-domain setup phase can occur between SIPSs in each conference as after this phase, they will not need to start another setup phase as they can already communicate. When all SIPSs in a conference have subscribed to each other, a full mesh of SIPSs is created, which is a guarantee that signalling information is passed according to the needs. In part 3, each SIPS notify their local CS about the ID of the remote CS that they have just been notified of in the form of NOTIFY message.

Local Invitation. Now, we will assume that `Client A1` invites `Client A2`, which are in the same domain. This will start the conference (actually a one-to-one call).
In the first part, `Client A1` invites `Client A2` with a normal SIP invitation messages exchange as defined in [17]. The messages go through the local Proxy Server, which is contained in `SIPS-A`. In the second part, `SIPS-A` sends a NOTIFY message to `CS A` to inform it that two local Clients (with their IDs) have joined the conference. That way, `CS A` will be able to handle their audio packets. In the third part, `SIPS-A` notifies `Client A1` and `Client A2` with the ID of `CS A` so that they will know which is their media handling Conference Server.

Remote Invitation. The next motivating step happens when one of the Clients in `Domain A` wishes to invite a Client from `Domain B`. Let us assume that `Client A1` invites `Client B1` in the conference. The message exchange for remote invitation is depicted in Fig. 6.

In part 1, a normal SIP invitation takes place between `Client A1` and `Client B1`. But as the ACK message flows through `SIPS-A`, and if the inter-domain setup has not already occurred between the two involved domains, this phase must take place right away. In the future, this will not happen again as this is an only-once phase. In part 2, `SIPS-A` notifies `SIPS-B` about relevant conference-level information (such as the handling CS in the domain, the ID of the involved local Clients). Then `SIPS-B` does the same with `SIPS-A`. Then, in part 3, `CS A` is notified of the ID of `CS B` that has just joined the conference (known through the previous messages exchanged). In part 4, `SIPS-B` does the same with its local Conference Server, `CS B`.
Now that all the entities have been provided with the necessary information to make the conference take place between the involved parties, the conference can start. The deletion of clients is done according to the same scheme when necessary, i.e. sending of normal BYE SIP messages between entities and notification of the other parties.

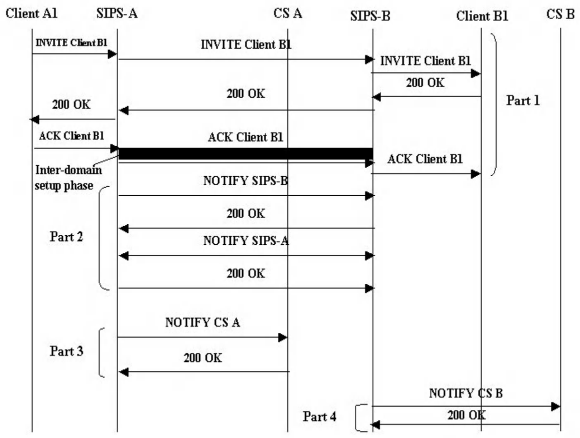


Fig. 6. Flow of SIP messages when Client A1 invites Client B1

5 Conclusion

In this paper, we have presented a new paradigm for VoIP conferencing. The fully distributed nature of SIPS and Conference servers make the scheme highly scalable. Multicast is not a mandatory requirement for our solution. However, it can be used with advantage, if available. The maximum number of hops for audio packets is limited to two (between two CSs), reducing the delay. The audio packets are transmitted using an RTP stack. The traffic is further reduced by VAD techniques [11], as an option. This paradigm, along with LN, supports an interactive conference without explicit floor control and allows impromptu speech while compensatory floor control co-exists. Further, this paradigm supports conferencing requirements listed in [8] and in section 2 of this paper. Moreover, we can support other facilities such as chat, PBX, and Voice Mail Service on the same set up. These facilities are based on a peer-to-peer communication mode. The main contribution of this work is a greatly improved quality of conference due to the impromptu speech permitted without the use of 'gagging' floor controls. Further reduction of traffic is possible by using the characteristics LN. At present we are investigating the effective use of SCCP [1] (and CCCP) in our implementations.

References

[1] C. Bormann, J. Ott, "Simple Conference Control Protocol", Internet Draft, Dec. 1996.

[2] E. Doerry, "An Empirical Comparison of Copresent and Technologically-mediated Interaction based on Communicative Breakdown", Phd Thesis, Graduate School of the University of Oregon, 1995.

[3] H. P. Dommel and J.J. Garcia-Luna-Aceves, "Floor Control for Multimedia Conferencing and Collaboration", *J. Multimedia Systems*, Vol. 5, No. 1, January 1997, pp. 23–38.

[4] A. J. González, "A Distributed Audio Conferencing System" *MS Project Department Of Computer Science Old Dominion University*, Norfolk, VA 23529, July 28, 1997.

[5] M. Handley and V. Jacobson, "SDP: Session Description Protocol", RFC 2327, IETF, April 1998.

[6] M. Handley, J. Crowcroft et al., "Very large conferences on the Internet: the Internet multimedia conferencing architecture", *Journal of Computer Networks*, vol. 31, No. 3, Feb 1999, pp. 191–204.

[7] ITU-T Rec. H.323, "Packet based Multimedia Communications Systems", vol. 2, 1998.

[8] P. Koskelainen, H. Schulzrinne and X. Wu, "A SIP-based Conference Control Framework", *NOSSDAV'02*, May 2002, pp. 53–61.

[9] R Venkatesha Prasad et al., "Control Protocol for VoIP Audio Conferencing Support", *International Conference on Advanced Communication Technology*, Mu-Ju, South Korea, Feb 2001, pp. 419–424.

[10] R Venkatesha Prasad et al., "Automatic Addition and Deletion of Clients in VoIP Conferencing", *6th IEEE Symposium on Computers and Communications*, July 2001, Hammamet, Tunisia, pp. 386–390.

[11] R Venkatesha Prasad, H S Jamadagni, Abhijeet, et al "Comparison of Voice Activity Detection Algorithms" *7th IEEE Symposium on Computers and Communications.* July 2002, Sicily, Italy, pp. 530–535.

[12] R. Venkatesha Prasad, Richard Hurni, H S Jamadagni, "A Scalable Distributed VoIP Conferencing using SIP", *To appear in Proc. of the 8th IEEE Symposium on Computers and Communications*, Antalya, Turkey, June 2003.

[13] R Venkatesha Prasad, H S Jamadagni and H N Shankar, "On Problem of Specifying Number of Floors in a Voice Only Conference", *To appear in Proc. of IEEE ITRE 2003*, Newark, NJ, Aug. 2003.

[14] A. B. Roach, " Session Initiation Protocol (SIP)-Specific Event Notification", RFC 3265, IETF, June 2002.

[15] S. Ramanathan, P. Venkat Rangan, Harrick M. Vin, "Designing Communication Architectures for Interorganizational Multimedia Collaboration", *Journal of Organizational Computing,* Vol. 2, Nos. 3&4, 1992, pp.277–302.

[16] M. Radenkovic et al, "Scaleable and Adaptable Audio Service for Supporting Collaborative Work and Entertainment over the Internet", *SSGRR 2002*, L'Aquila, Jan. 2002.

[17] J. Rosenberg, H. Schulzrinne et al., "SIP", RFC 3261, IETF, June 2002.

[18] J. Rosenberg, H. Schulzrinne, “Models for Multy Party Conferencing in SIP”, Internet Draft, IETF, July 2002.

[19] E. M. Schooler, "A Distributed Architecture for Multimedia Conference Control", Technical Report ISI/RR-91-289,USC/ISI, Marina del Rey, CA, Nov. 1991.

[20] H. Schulzrinne et al., "RTP: a transport protocol for real-time applications", RFC 1889, IETF, Jan 1996.

[21] K. Singh, G. Nair and H. Schulzrinne, "Centralized Conferencing using SIP", *Proceedings of the 2nd IP-Telephony Workshop (IPTel’2001)*, April 2001.

Dynamic Configuration of Multimedia Applications

Slim Ben Atallah, Oussama Layaida, Noel De Palma, and Daniel Hagimont

INRIA, SARDES Project
ZIRST-655, Avenue de l'Europe - 38334 Montbonnot Saint-Ismier Cedex – France
Slim.Benatallah@inria.fr

Abstract. Streaming multimedia applications, such as video on demand or conferencing are increasingly deployed in heterogeneous and mobile environments including Workstations, PDAs, mobile phones, etc. These applications are very resource demanding and in general, they need to be dynamically adapted when executed on low capability terminals. The proxy-based content adaptation approach is well suited to transparently adapt in real time multimedia data on intermediate nodes without modifying the application. In this paper, we report on experiments on dynamic configuration of such proxies by using a configuration language called APSL (Adaptation Proxy Specification Language). We developed a configurable proxy allowing adaptation of existing videoconferencing applications, and evaluated the performance benefits of the proxy approach using a *DirectShow/COM*-based framework.

1 Introduction

Due to the significant growth of network technologies, distributed computing environments are becoming increasingly heterogeneous. This heterogeneity mainly covers two aspects:

- Hardware/Software. Many mobile hand-held devices have appeared such as PDAs or mobile phones. In general, these devices are characterized by low CPU, memory, network or display capabilities. On another hand, the Internet, the operating systems, middleware environments or software libraries which are used by applications may introduce heterogeneity in terms of availability, reliability, latency, network bandwidth, network protocols, data encoding formats, …
- Users' requirements. The potential users of a given application may have very different needs while using the application in a particular context. These requirements may also vary for a same user between different application runs. Moreover, it is important to consider the point of view of many different principals such as a service or content provider, a service client, a network operator, etc.

An adaptation to hardware/software requirements may consist to adapt the encoding format of a streamed video into the unique format supported by an end user terminal. An adaptation to user requirements could consist in selecting in a multi-user videoconferencing session the participants to be displayed on a terminal (e.g. all the participants or one favorite participant). In both cases, the application has to be adapted according to the requirements.

Since it is not realistic to develop, for each application, multiple versions that respond to several constraints, we propose to tackle the various forms of heterogeneity by the development of adaptive distributed multimedia applications. Our objectives are therefore two-fold:

1. Provide distributed multimedia applications with *dynamic adaptation*; that is the ability for applications to adapt their behavior, at launch-time or run-time, to various context constraints.
2. Propose *transparent adaptation* that does not require any modification of the original applications' code.

One important motivation is to be able to adapt applications without any modification to the original code. More importantly, we don't want to reconsider the legacy software (web servers, videoconferencing tool, video players) installed on end-user machines. In order to achieve this goal, we follow a proxy-based architecture where, in a distributed client/server multimedia application, a proxy site between the client and the server is responsible for intercepting interactions (streams) between the two parties and performing the adaptation. An adaptation consists in modifying the content of the stream which traverses the proxy.

The second important motivation is to respond to the various constraints previously enumerated. Since these constraints may be very different and are only known at runtime, we need to dynamically configure the adaptation software on the proxy nodes according to these constraints. Therefore, our approach is to provide a framework for *dynamic configuration of a multimedia proxy*.

We conducted several experiments on a *DirectShow* [5] platform. With its widespread distribution and its efficient implementation, this platform is now considered as a reference platform for building multimedia applications. These experiments aim at validating this approach by implementing several applications and adaptation scenarios. We also evaluated the obtained performance benefits. The lessons learned from this experiment are the following:

– Performing multimedia application adaptations on proxy machines is a means to take into account a very broad range of requirements coming of the hardware, software and user environments, without reconsidering the applications installed on end-user machines.
– Dynamically-configured component-based adaptations provide the required flexibility to deal with the heterogeneity of the requirements.
– The proposed configuration approach implemented on the *DirectShow* environment allows real time adaptation of video contents providing a good tradeoff between flexibility and performance.

This paper is structured as follows. Section 2 discusses the related work. Section 3 presents the configuration of proxies. Our experiments are then detailed in section 4. Section 5 presents results of a performance evaluation. Section 6 concludes the paper while enumerating some perspectives to this work.

2 Related Work

Many component-based environments have been proposed to support the building of architecture-based applications. These environments exploit software architectures to provide dynamic configuration mechanisms in order to adapt applications according to execution constraints [3], [15], [13]. However, very few projects have validated the effectiveness of dynamic configuration with resource intensive applications such as multimedia applications.

Kon [14] describes a framework which allows configuring proxy nodes (called Reflectors) in order to adapt the distribution of multimedia data. However, the main focus is on the adaptation of network routing. Blair [4] describes a framework to adapt multimedia applications for mobile environments. Some proposed scenarios (related to video transcoding) are similar to ours, but their implementation relies on a CORBA platform and they don’t report any performance evaluation.

Many projects have addressed the issue of the adaptation of multimedia data delivery according to the application context. A first class of solutions addresses this issue by modifying the servers, the network protocols or the encoding formats used to deliver multimedia data. Sisalem [22] extends servers in order to adapt the emitted streams according to clients’ requirements. In [18], McCanne uses multi-layered encoding and transmission, which consists in dividing the video into several cumulative layers, each corresponding to an increment of quality. By selecting the appropriate number of layers, receivers can control the amount of video bandwidth they consume, and solve bandwidth heterogeneity problems. However, these solutions only address network bandwidth; adaptations do not consider clients' hardware limitations and software incompatibilities. Moreover, they require modification of the software installed on Internet hosts (clients and servers).

The alternative solution uses intermediate nodes (proxies) inside the network to make additional treatments on media streams. These entities can be deployed for example by ISPs across the Internet, by network operator or individual users in their private networks. Multimedia content is adapted dynamically so that it matches the variable network or host capacities, without requiring modification on the end machines. Fox [7] presents the advantages of infrastructural proxies and proposes design principles to effectively address heterogeneity problems. Various research works have been made in this area [1], [20], [23]. Most of existing works addressed only adaptations of discrete media, such as HTML pages and images [10][16]. Some projects proposed gateways to adapt video streams. For example, VGW [1] can transcode RTP video streams from high bit-rate MJPEG format into 128 Kbps H.261 video streams which are more suitable for MBone sessions. In [24], an MPEG specific proxy-based content adaptation gives more priority to I-frames than P and B-frames by selecting frames to drop when congestion occurs. A RTP to HTTP gateway, described in [12], interconnects a multicast network with the World Wide Web, and enables Web client to receive video streams from multicast conference sessions. However, all these projects were proposed to solve a particular problem and focused on a specific encoding format or protocol conversion.

Our objective is to provide proxies with the flexibility required to implement any of these adaptations. Moreover, such a flexibility aims at providing content based adaptation of video stream in order to take into account both QoS problems and user requirements. In this vein, we propose to use a component-based framework which allows dynamic configuration of a context-specific adaptation proxies.

3 Dynamic Configuration of Adaptation Proxies

3.1 Proxy Implementation

Our goal is to dynamically configure adaptation proxies. We provide a proxy configuration language to define the adaptation and also a flexible configuration programming interface allowing deployment and reconfiguration of proxies. We aim at providing a flexible solution of configuration which allow deployment and configuration of proxies on several multimedia environments. We experiment the implementation of dynamic configuration on the Microsoft's *DirectShow 8.0* environment. This environment provides programmers with an efficient toolkit for building multimedia applications. It is based on the COM component model 8 and it provides abstractions for the manipulation of multimedia data, such as filters, pins and filter graphs.
A filter is the basic building block in *DirectShow*. It is a software component offering a specific multimedia-related functionality, such as capturing, encoding/decoding and rendering multimedia data. Using inheritance techniques, programmers can build additional filters. Several filters can be combined (i.e. interconnected) in order to form a filter graph that represents a particular configuration. The interconnection of several filters is possible thanks to pins, which are filters' input and output ports.

3.2 Components

All needed functions for networking and multimedia data processing are provided by separate basic components. A component is characterized by a set of input and output stream interfaces and a processing unit. Moreover, each *DirectShow* component (Filter) provides one or more configuration interfaces which allow configuring the component. These basic components are the following:

- Networking components: They implement standard Internet protocols used for multimedia streaming, such as HTTP[1] and RTP.
- Decoder/Encoder components: They aim at compressing/uncompressing the data into a chosen intermediate format (e.g. RGB24, YUY2). We provide different encoder/decoder components which can be used to encode/decode standard multimedia format such as H261, H263, MPEG1, MPEG2 or MJPEG.
- ***Transformer components:*** Transformer components implement the basic content based adaptation code. For instance, a transformation component can receive as input a video stream in YUY2 format, resize it and deliver the modified video as output. Each transformer component provides a very basic adaptation on a stream in an intermediate format. Complex stream transformations can be built by

[1] We had to implement networking components for HTTP as they were missing in DirectShow.

combining several basic components. Below are examples of transformer components that we implemented:

- *Image-scaling components* resize video frames, which is useful to adapt a stream for devices with limited display capacities. They are sometimes required to enable transcoding operations, for example MPEG videos may be transmitted in any size while H.261 videos require predefined sizes such as CIF, QCIF or SQCIF.
- *Color-space-scaling components* reduce the number of entries in the color space, for example from 24 to 12 bits, gray-scale or black-and-white.
- *Data insertion components* can be used to insert an image or a text in a video. We used such components to integrate commercials, subtitles in video and textual notifications.
- *Mixer Component* allows building of a mixed video stream resulting from several input sources. The resulting video stream is an *(NxM)* matrix. Each element of this matrix results from an image-scaling adaptation of a particular stream.
- *Multiplexors/Demultiplexors* are used to aggregate/separate audio and video data in a multimedia stream. For instance, an MPEG Multiplexor allows merging an MP3 audio and an MPEG-1 video in a MPEG2 stream. In the presentations of our scenarios, we sometimes omitted these components since our primary focus is on video adaptations.
- *Duplicator components* are used to replicate an output media stream. Duplication is useful when a stream has different targets with different requirements. Duplicators are used in the videoconferencing application further considered.

Data insertion, mixer, duplicator and some networking components were not provided by the original *DirectShow* framework. These additional components and also the configuration manager code are built on top of this framework to make possible instantiation of proxies performing content-based adaptations.

3.3 Adaptation Sessions

A session is instantiated as a graph of interconnected components that implements the adaptation process. The adaptation process is built from the basic components presented above (receivers, decoders, transformers, encoders …). These components are configured and interconnected together to achieve the transformation of the multimedia stream. The configuration of sessions provides the flexibility required by the adaptation process to fulfill the application needs.
The adaptation process is split up into several steps. The input stream is decoded into an intermediate representation, and then transformed and delivered to the encoder, which produces an adapted stream in output. During this process, adaptations can be applied at different levels in the data path. Fig 1 describes the configuration of a session at a high-level.

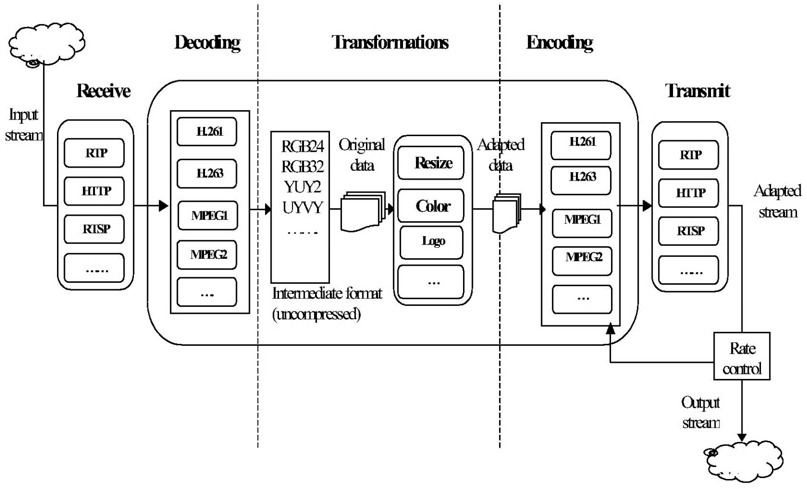


Fig. 1. Video transcoding scheme

Multimedia data is generally transmitted with application-level protocols such as, HTTP, RTP, etc. When configuring a session, an appropriate component[2] is chosen to receive a media stream from the network and to deliver it to the appropriate decoder. At this level, the appropriate decoder component is configured with the intermediate format in which uncompressed data will be represented. This intermediate format allows us to perform additional treatment on data that cannot be performed in a compressed format. Support for multiple intermediate formats allows us to make an optimized configuration to perform these effects (for example, resizing an image in YUY2 format is faster than in RGB format).

At the intermediate level, the data can be transformed in various ways by combining transformer components together in the session's configuration. Changing the interconnections between these transformer components allows customizing the adaptation process according to the requirements.

At the encoder level, an encoder is selected and configured to offer the best-suited data rate that matches network and receiver's states and capacities. The target data rate is obtained by modifying the rate of encoded frames or by degrading the encoding quality. The obtained stream is sent using the protocol used by the network target independently from the protocol used to receive the original data from the server.

3.4 APSL: Adaptation Proxy Specification Language

In order to help the definition of a session, we propose an XML-based specification language called APSL, which allows describing several QoS parameters and User requirements such as input video format, network capabilities, terminal capabilities,

[2] In fact, one or several networking components may be necessary to receive video streams. As described further in the paper, a videoconference adaptation session may rely on several networking components to receive several input streams.

connection protocols, etc. An APSL specification may be composed of the following definitions:

- **INPUT** : defines a list of members. Each member describes a particular input source of the proxy. Attributes of an input member are : PROTOCOL, USER, TERMINAL, and DATA.
- **OUTPUT**: defines a list of output destinations members. Each member describes a particular adapted target streams. OUTPUT members are also defined using the same INPUT member attributes.
- **PROCESS**: defines the proxy architecture. We use a directed graph model to define the adaptation process on proxies. Each graph node represents a particular component. Components are bound using input/output PIN connections. The Document Type Definition of APSL is detailed in Fig 2.

```
<!ELEMENT APSL (INPUT, OUTPUT, PROCESS)>  ← APSL element list
definition
<!ELEMENT INPUT (MEMBER+)>← INPUT member list definition
<!ELEMENT OUTPUT (MEMBER+)>      ←OUTPUT member list definition
<!ELEMENT MEMBER (PROTOCOL, USER, TERMINAL, DATA)> ←
INPUT/OUTPUT member definition
<!ATTLIST MEMBER ID ID #REQUIRED>
<!ELEMENT PROTOCOL (NAME, DESCRIPTION, ARGUMENT*)>
<!ELEMENT USER (LOGIN, PROPERTIES)>
<!ELEMENT LOGIN (#PCDATA)>
<!ELEMENT PROPERTIES (#PCDATA)>
<!ELEMENT ARGUMENT (#PCDATA)>
<!ELEMENT DESCRIPTION (#PCDATA)>
<!ELEMENT NAME (#PCDATA)>
<!ELEMENT TERMINAL (CPU, DISPLAY, NETWORK)>
<!ELEMENT DISPLAY (XSIZE, YSIZE, COLORDEPTH)>
<!ELEMENT CPU (#PCDATA)>
<!ELEMENT NETWORK (#PCDATA)>
<!ELEMENT XSIZE (#PCDATA)>
<!ELEMENT YSIZE (#PCDATA)>
<!ELEMENT COLORDEPTH (#PCDATA)>
<!ELEMENT DATA (TYPE, CODEC+)>
<!ELEMENT TYPE (#PCDATA)>
<!ELEMENT CODEC (#PCDATA)>
<!ELEMENT PROCESS (COMPONENT*)>  ← PROCESS element definition
<!ELEMENT COMPONENT (PIN+)>
<!ATTLIST COMPONENT
  ACTION (RESIZE | DUPLICATOR | H.261Encoder | MPEGEncoder |
H.263Encoder | MIXER) #REQUIRED ID ID #REQUIRED>
<!ELEMENT PIN (#PCDATA)>
<!ATTLIST PIN ID ID #REQUIRED
LINK IDREF #IMPLIED
DIRECTION (OUTPUT | INPUT) #REQUIRED>
```

Fig. 2. APSL DTD.

3.5 Proxy Configuration Architecture

A proxy can be configured through an APSL specification which describes the initial configuration, including the proxy's process architecture and the members' attributes. This APSL specification is interpreted by an **APSL Engine** which is responsible for the instantiation of the associated adaptation session.
At a lower level, a **Configuration Manager** provides all the functions required to instantiate and manage adaptation sessions. The API of the Configuration Manager is invoked by the APSL Engine in order to instantiate a session based on an APSL specification. The Configuration Manager implementation directly relies on DirectShow/COM.
However, an adaptation session may have to be dynamically adapted in order to respond to variations of the execution constraints (e.g. available resources). For this reason, the API of the Configuration Manager is exported to allow direct management of the adaptation session. It allows visiting and adapting the component-based architecture of a session, or modifying the attributes of the session's components.
The overall structure of the proxy environment is shown in Fig 3.

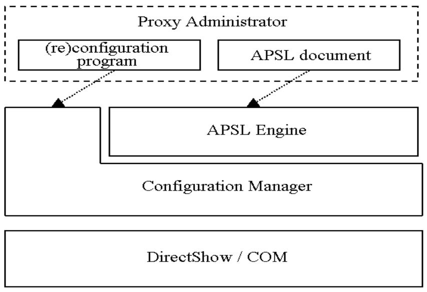


Fig. 3. Overall structure of the proxy environment.

The administrator of the proxy can configure an adaptation session by providing an APSL document which describes the required adaptation. He can also directly use the Configuration Manager API to configure the required adaptation session. And finally, this Configuration Manager API can be used to reconfigure an adaptation session, i.e. to modify it in response to variations in the execution conditions.

4 Experiments

Our objective is first to validate the approach described above with the implementation of realistic scenarios and second to show that it can be efficiently instantiated, thus combining flexibility and performance.
We modeled an experimental environment involving several multimedia applications and mobile handheld devices. The platform is based on PC workstations (PIII 700Mhz-256MB for VoD scenario, PIV 1800 Mhz-512MB for Videoconferencing scenario) interconnected with an Ethernet Local Area Network (Ethernet 100 Mbps)

and mobile PDA (IPaq, ARM Processor 200 MHz, 32 MB RAM, Windows CE 3.0) devices connected through a 802.11 Wireless access points. We experimented with two applications: a video on demand service and a videoconferencing system. The VoD application relies on a web server which hosts several MPEG movies encoded with high quality parameters. The videoconferencing application distributes real-time video streams using VIC [17], an application from University of California, Berkley. The client side includes standard applications offering basic multimedia players; on the PDAs, we used PocketTV [19] for streaming MPEG-1 movies using HTTP and VVP [25] for real-time H.261 streaming.

4.1 VoD Scenario: Adaptation for Hardware/Software Capabilities of the PDA

In the first scenario, we consider the video on demand application for mobile handheld devices (PDAs). Due to their limited processing, display and network capacities, PDAs are only able to efficiently render streams with specific properties (frame size, colors, quality factor and encoding format). To deal with such hardware limitations, we configure a dedicated adaptation proxy. When the client sends an HTTP request to the proxy (using the proxy URL instead of the original URL), the VoD adaptation proxy parses it in order to extract the client properties and it invokes the configuration manager API to instantiate a session according to the received properties (the HTTP request fields give the following client's properties: Accept-encoding, color depth and frame size). Fig 4 gives the composition of the session which adapts an original MPEG stream into an MPEG stream with smaller resolution and color depth in order to fit PDA's display capacities.

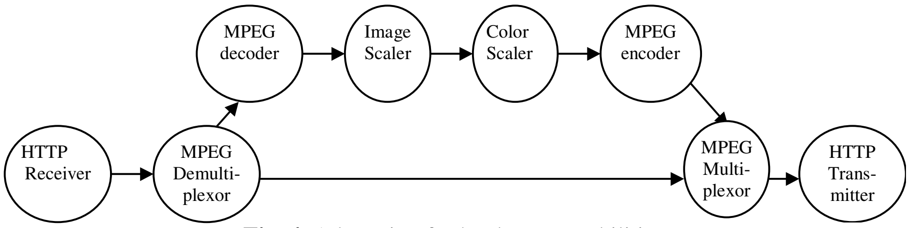


Fig. 4. Adaptation for hardware capabilities

The instantiated session includes two networking components for receiving and transmitting the HTTP streams. Between them, six additional components are inserted. First, an MPEG demultiplexor separates the MPEG audio and video into two streams, an MP3 audio stream and an MPEG-1 video stream. The video stream is handled by an MPEG-1 decoder component, which uncompresses data into YUV video frames. Then, an image-scaler component resizes video frames to QCIF (176*144), transformed by a color down-scaler component into 16 gray-scale colors. Notice that the quality factor of the encoder can also be dynamically adjusted. A similar scenario is used to illustrate adaptation for software capabilities on the PDA. In this context, the proxy transcodes the original stream into H.261 and forwards it to the client.

In addition to the transcoding operation, a third party may want to integrate other services such as the insertion of commercial advertisement or personalized subtitles in the video content.

4.2 Adapting VIC/VVP Videoconferencing Application: Video Stream Mixing and Bandwidth Adaptation

Videoconferencing applications often involve more than two participants, each with its own encoding format and terminal capabilities. The scenario that we implemented focuses on the following problems:

– *A client machine requires a high bandwidth to receive multiple streams and a high processing capacity to decode and synchronize them before display.*
– *On the other hand, VVP which is the VIC version for PDAs does not provide support for multiple streams visualization. We modified the conference architecture (without any modification of VVP/VIC) in order to introduce a proxy conferencing server which receives a stream from each participant, mixes the streams in a single video stream which is sent to all participants.*

VIC and VVP can be used in a multicast mode or as peer-to-peer applications. When used with multicast mode, VVP receives all incoming video streams but can only display one at the same time (due to the limited display capacity of the PDA). The objective of our adaptation is first to reduce the used bandwidth (by emitting a single stream from the proxy to the client machine), and second to allow display of all participants' videos on the VVP user interface.

The scenario that we consider describes a multiparty videoconference between five terminals: A and B are two workstations running VIC with H.263, C and D are also two workstation running VIC with H.261 CIF resolution, and E is a PDA running VVP to just receive video in H.261 and QCIF resolution. The conference starts between A, B, C and D. The participant connected via PDA "E" joins the conference at a later time. For each Workstation participant, the proxy opens one incoming and one outgoing stream. The incoming stream contains the media of that user and the outgoing stream is the result of mixing all media streams into one stream, requiring less bandwidth and less computing on the client. Fig 5 describes the configuration graph resulting from parsing APSL configuration of the proxy.

For each incoming stream, the proxy instantiates an RTP source filter and the appropriate decoder component. Streams received from A and B are resized into QCIF. A central mixer component receives the three videos streams and produces a mixed video stream in CIF size (4 QCIF quarters, one empty). As participants use two different encoding formats, the mixed video is duplicated into two streams with a duplicator component. The first one is encoded in H.263, duplicated again into two streams and sent to participants A and B. The second one is resized to QCIF, encoded in H.261 and sent to participant C and D. When user E joins the conference, the proxy requests a new receiving branch (dotted bag on the left side of the figure). As this participant requires H.261 encoding, the proxy inserts a duplicator after the H.261 encoder and a new RTP transmitter is created to send the mixed stream to E (dotted bag on the right side of Fig 5).

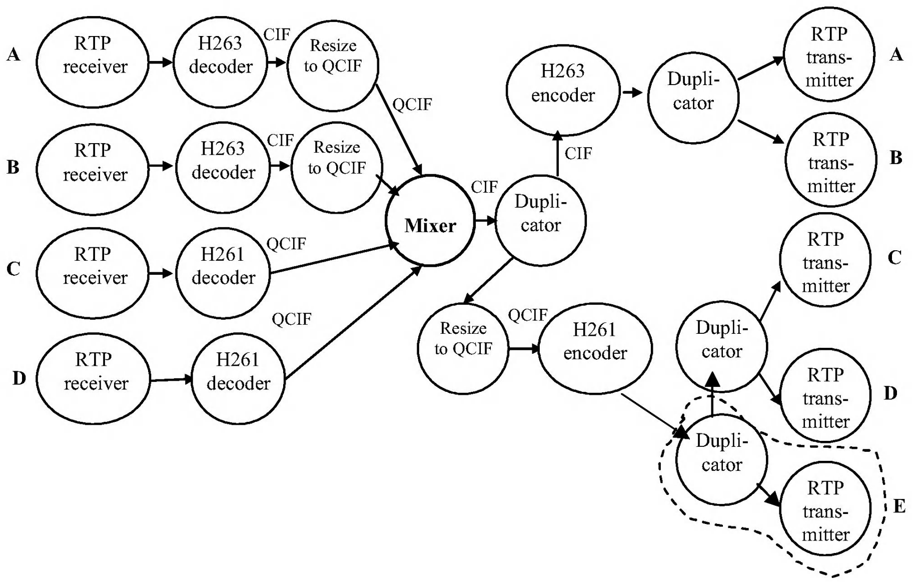
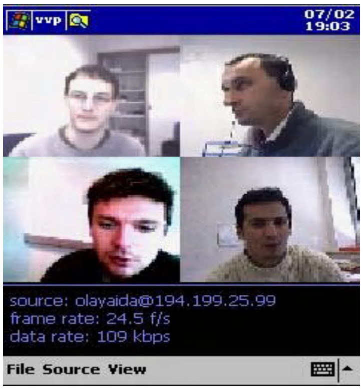


Fig. 5. Dynamic configuration of a videoconferencing proxy

The use of the duplicator components optimizes the configuration in order to prevent redundant tasks. Rather than serving each user independently (e.g. with per-user encoders and transformers), the configuration manager looks for an existing output stream providing the same properties as those requested by the arriving participant. Fig 6 shows a screen shot of the adapted VVP videoconferencing application in which 4 video streams are mixed and rescaled in order to be displayed on the PDA.



source: olayaida@194.199.25.99
frame rate: 24.5 f/s
data rate: 109 kbps

Fig. 6. VVP adapted user interface

4.3 APSL Specification for Videoconferencing Adaptation Scenario

In order to configure the videoconferencing proxy, we used an appropriate APSL specification (cf. Fig 7) which defines for each end-user terminal, the video format, the network connections, and screen size.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE APSL SYSTEM "C:\inria\Sardes\MAD\code\MM\APS\apsl.dtd">
<APSL>
        <INPUT>
                     D=
                 OTOCO
                     A       T       A
                   D   C    T O      C   en    D   C    T O
                  OTOCO
               U
                    O      U          O
                      O     T                    0      O     T
                U
               T        A
                   C  U   DA(   ron  A        C  U
                   D      A


                        CO  O  D    T           s    CO  O  D    T
                    D      A
                      T   O
                T        A
               DATA
                   T         deo     T
                   COD  C            COD  C
                DATA

                        D=    2

                        D=

                        D=

          UT
        OUT  UT
                        D=


                        D=

                        D=

                        D=

         OUT  UT
           OC
              CO    O     T  D=       ACT  O   =
                              D=   2  D     CT  O   =  OUT  UT         =
                              D=       D     CT  O   =   N  UT         =
               CO    O     T
              CO    O     T  D=    2  ACT  O   =
                              D=       D     CT  O   =   N  UT         =    2
                              D=       D     CT  O   =  OUT  UT         =
               CO    O     T
              CO    O     T  D=       ACT  O   =
                              D=       D     CT  O   =   N  UT         =
                              D=       D     CT  O   =   N  UT         =    2
```

```
                         D=     D   CT O  =  N  UT      =
                         D=     D   CT O  =  N  UT      =
                         D=     D   CT O  = OUT  UT       =
          CO   O    T
         CO   O    T  D=     ACT O  =   U        TO
                         D=   0  D   CT O  =  N  UT      =
                         D=      D   CT O  = OUT  UT      =
                         D=   2  D   CT O  = OUT  UT      =
          CO   O    T
         CO   O    T  D=     ACT O  =    2       co
                         D=      D   CT O  =  N  UT      =
                         D=      D   CT O  = OUT  UT      =
          CO   O    T
         CO   O    T  D=     ACT O  =
                         D=      D   CT O  =  N  UT      =
                         D=      D   CT O  = OUT  UT      =
          CO   O    T
         CO   O    T  D=     ACT O  =    2       co
                         D=      D   CT O  =  N  UT      =
                         D=      D   CT O  = OUT  UT      =
          CO   O    T
         CO   O    T  D=     ACT O  =   U        TO
                         D=  22  D   CT O  =  N  UT      =
                         D=  2   D   CT O  = OUT  UT      =
                         D=  2   D   CT O  = OUT  UT      =
          CO   O    T
         CO   O    T  D=     ACT O  =   U        TO
                         D=  22  D   CT O  =  N  UT      =
                         D=  2   D   CT O  = OUT  UT      =
                         D=  2   D   CT O  = OUT  UT      =
          CO   O    T
      OC
A
```

Fig. 7. APSL specification for VIC/VVP videoconferencing adaptation.

5 Performance Evaluation

We evaluated the benefits of dynamic video content adaptations for the performance of the PDA. Evaluation is performed for two application settings:

Table 1. Performance of the PDA with the VoD application

Format	Size	Maximum frame rate (fps)
MPEG1	640*480	3.8
MPEG1	320*240	9.2
MPEG1	176*144	21.4
H.261	176*144	24.5

The video on demand application. A video file is available on a Web server and accessed from the PDA. The video format is MPEG-1 and the video size is 640*480 (we write it MPEG-1/640*480). The evaluated proxy adaptations respectively convert the initial stream into MPEG-1/320*240, MPEG-1/176*144 and H.261/176*144. The video is always encoded at the same frame rate and data rate (25 frames/s, 250 Kbits/s).

The videoconferencing application. It distributes a video stream in H.261-CIF (352*288) at 25 frames/s and the best quality factor. The evaluated proxy adaptation converts 4 initial streams into one mixed H.261-QCIF (176*144) stream with a lower quality factor (85 %).

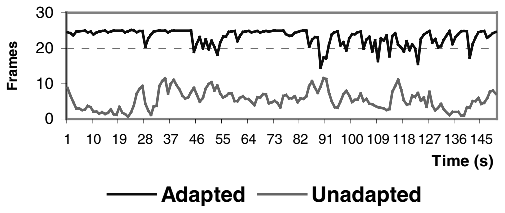


Fig. 8. Performance of the PDA with VVP application

On Table1 and Fig 8, we observe that the PDA can hardly display a large size video because it has to decode and resize the video frames to display them. As a PDA does not have sufficient capacities to perform these operations in real time, frames are displayed at a very low rate. However, when the size decreases, the frame rate increases accordingly, in particular for the QCIF size (176x144), which is probably the best-suited size for handheld devices such as PDAs.

In Table1, we also observe that the frame rate increases when we change the encoding format from MPEG to H.261, which decoding is less demanding.

Power Consumption of the PDA

Power consumption on the PDA, i.e., energy consumption, is an important problem for mobile devices with intrinsic power limitations. Indeed, unlike desktop PCs with permanent power supply, mobile devices such as PDAs have limited power autonomy. Several research works have studied the relationship between power consumption and multimedia applications [7, 21]. Analysis and measurements have demonstrated existing dependencies with the size, color depth and encoding format of data. In order to measure the power consumption on the PDA, we used the notification given by the PDA's battery driver when the battery level changes by ±10%. We therefore run the multimedia application for a long period (up to one hour) and measured the average period at which the battery level changes. We performed these measurements with the VoD application and the proxy adaptations described in the previous section. The results are given in Table 2.

We observe that power consumption also depends on the video size and its encoding format. When the proxy is configured to reduce the video size and to use a cheaper encoding format, displaying the video requires less treatments and consequently it consumes less energy.

Table 2. Power consumption of the PDA with different formats and sizes

Format	Size	10% Power consumption
MPEG1	640*480	every 12 minutes
MPEG1	320*240	15 minutes
MPEG1	176*144	19 minutes
H.261	176*144	24 minutes

Network Bandwidth

Proxy adaptations can also be beneficial in order to save network bandwidth. In order to evaluate this, we used the videoconferencing application and we measured on the PDA the data bit-rate and the packet loss. Data bit-rate gives the amount of data transferred on the network. Packet loss can be caused by network congestion or by PDA's processor overload. Lost packets due to processor overload are delivered to the PDA but discarded as they could not be processed in time. These discarded packets also result in wasted network resources as they are discarded after reaching the PDA.

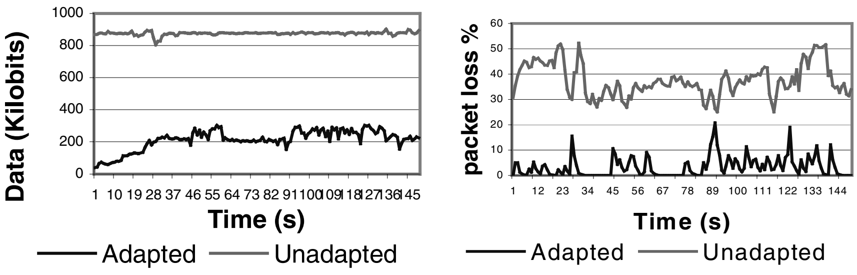


Fig. 9. Impact of adaptation on network traffic

Results are given in Fig 9; in both measurements (data rate and packet loss), the upper curve corresponds to a transmission without proxy adaptation (i.e. using a multicast data communication scheme). On the left side of the figure, we observe that the adapted stream consumes less bandwidth because resizing the video reduces considerably the amount of data transmitted on the network.

On the right side of the figure, we observe that without adaptation, packets are lost at a high rate varying between 30 and 50% (due to PDA's processor overload). This results in a displayed frame rate lower than 11 frames/s (Fig 9) and thus a poor quality presentation. With proxy adaptation, packet loss is kept under 10% and frames are displayed at a rate close to the original.

Performance of the Proxy

The ability of a proxy machine to support one or several sessions which adapt multimedia streams in real time is a critical issue. To evaluate this, we measured the processor load on the proxy in function of the number of sessions (Fig 10). Each session is composed of networking components, an MPEG decoder, a resize component (from 320*240 to CIF) and an H.261 encoder.

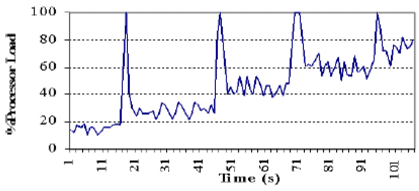


Fig. 10. CPU load on the proxy

Results show that a session consumes on the average 13 % (on a PIII 700 Mhz with 256 MB) of the processor CPU resource. Notice here that these results depend on the number of used codecs, the decoding/encoding formats and the treatments applied to the video in the session. We consider here a very common adaptation of the VoD application, i.e. when we adapt a video stream for a PDA (MPEG to H.261 conversion, and resize to 176*144). We also observe a peak when the proxy creates a new session. This peak corresponds to the creation and the configuration of the session. These results seem acceptable as we used a medium range PC to run the proxy software. In addition, for a large number of clients, the load of the proxy could be balanced between several machines as proposed in [4].

6 Conclusion and Perspectives

In this paper, we reported on an experiment which consisted in evaluating the benefits of dynamic content-based adaptations for distributed multimedia applications. Adaptations are used to face the increasing heterogeneity of today's distributed computing environment: heterogeneity of the hardware, the software and the user preferences. Adaptations are transparently performed on network intermediary nodes called proxies. Finally, adaptations are dynamically configured according to runtime constraints, thanks to a component-based middleware. We used *DirectShow* as an implementation base layer for conducting this evaluation. lessons learned can be summarized as follows:

- Proxy-based approach allow an implementation of adaptations without reconsidering the applications installed on end-user machines. We have implemented application scenarios that demonstrate this possibility without any modification on reused software such as Web servers, videoconferencing tools, or video viewers.
- Dynamically-configured content-based adaptations using APSL provide the required flexibility to deal with the diversity of environment parameters. Our application scenarios (Vod and videoconference) experimented in different

execution environments (PCs on a LAN, PDAs on a WLAN) demonstrate this flexibility.
– *DirectShow* can be extended to provide a well-suited support for dynamic configuration and reconfiguration of adaptation proxies. Its performance allows real time adaptation of video contents. We therefore provide a good tradeoff between flexibility and performance.

We are currently working on the extension of our framework for supporting session establishment protocols such as SIP [21] or H.323 [12] sessions.

At the moment, proxy configurations have to be programmed using the *DirectShow* API. We implemented a configuration manager which dynamically configures the proxy according to a unique APSL configuration file. However, It would be interesting to deploy the adaptation process on several sites and use several APSL files. We made few experiments with dynamic reconfiguration (during execution) of the proxy according to available network and terminal resources. However, we lack a deeper evaluation of dynamic reconfiguration of several proxies cooperating to take into account adaptation. Such experiments would require to integrate a monitoring service in our framework.

References

1. E. Amir, S. McCanne, Z. Hui. An Application Level Video Gateway. Proc. of ACM Multimedia'95, San Francisco, Nov 1995 .
2. Baldonado, M., Chang, C.-C.K., Gravano, L., Paepcke, A.: The Stanford Digital Library Metadata Architecture. Int. J. Digit. Libr. 1 (1997) 108–121.
3. L. Bellissard, S. Ben Atallah, F. Boyer, and M. Riveill. Component-Based Programming and Application Management with Olan. In Proceedings of Workshop on Distributed Computing Systems, pages 579–595, May 1996.
4. G. Blair, A. Andersen, L. Blair, G. Coulson, and D. S. Gancedo, "*Supporting dynamic QoS management functions in a reflective middleware platform*," IEE Proceedings – Software, 2000.
5. Microsoft DirectX (version 8.0): Microsoft DirectShow, Online Documentation: http://msdn.microsoft.com/directx/.
6. A. Fox, S.D. Gribble, Y. Chawathe, E.A. Brewer, and P. Gauthier. Cluster Based Scalable Network Services. In Proceedings of the Sixteenth ACM Symposium on Operating Systems Principles, October 1997.
7. A. Fox, S.D. Gribble, Y. Chawathe, and E.A. Brewer. Adapting to Network and Client Variation Using Active Proxies: Lessons and Perspectives. IEEE Personal Communications, 5(4):10–19, Aug 1998.
8. A. Gordon, A. I. Gordon. *The COM and COM+ Programming Primer*. Prentice Hall, March 2000.
9. C.J Hughes, J. Srinivasan, and S.V. Adve. Saving Energy with Architectural and Frequency Adaptations for Multimedia Applications. Proceedings of the 34th International Symposium on Microarchitecture, Dec 2001.
10. IBM Inc. Internet Transcoding for Universal Access, Sep. 2000. http://www.research.ibm.com/networked_data_systems/transcoding/index.html.
11. ITU-T Recommendation H.261: Video codec for audiovisual services at p x 64 kbit/s. Geneva, 1990, revised at Helsinki, Mar. 1993.
12. M. Johanson, An RTP to HTTP video gateway. In Proceedings of the Tenth International World Wide Web Conference, Hong Kong , May 2001.

13. F. Kon, M. Román, P. Liu, J. Mao, T. Yamane, L. C. Magalhães, R. H. Campbell, Monitoring, Security, and Dynamic Configuration with the DynamicTAO Reflective ORB, Proceedings of the IFIP/ACM International Conference on Distributed Systems Platforms and Open Distributed Processing, Middleware'2000.
14. F. Kon, R. H. Campbell, K. Nahrstedt, Using Dynamic Configuration to Manage A Scalable Multimedia Distribution System, Computer Communications, 2000.
15. J. Magee, J. Kramer, M. Sloman. Constructing Distributed Systems in Conic. IEEE Transactions on Software Engineering, 15(6):663–675, June 1989
16. A. Maheshwari, A. Sharma, K Ramamritham et P. Shenoy. TranSquid: Transcoding and Caching Proxy for Heterogenous E-Commerce Environments, Proceedings of the 12th IEEE Workshop on Research Issues in Data Engineering (RIDE '02), San Jose, California, Feb 2002.
17. S. McCanne and V. Jacobson. VIC: A exible framework for packet video. Proc. of ACM Multimedia'95, Nov 1995.
18. S. McCanne, V. Jacobson, M. Vetterli. Receiver-Driven Layered Multicast. In SigComm'96, Stanford, CA, Aug 1996.
19. MPEG Movie Player for PocketPC, (http://www.pockettv.com), 2000.
20. R. Rejaie, H. Yu, Mark Handley, D. Estrin. Multimedia Proxy Caching Mechanism for Quality Adaptive Streaming Applications in the Internet. Proceedings of the Conference on Computer Communications (IEEE InfoCom), Mar 2000.
21. J. Rosenberg, H. Schulzrinne, G. Camarillo, A. Johnston, J. Peterson, R. Sparks, M. Handley, and E. Schooler, «SIP: session initiation protocol,» RFC 3261, Internet Engineering Task Force, June 2002.
22. D. Sisalem, H. Schulzrinne. The loss-delay based adjustment algorithm: A TCPfriendly adaption scheme. Proc. NOSSDAV'98, Jul 1998.
23. J. Smith, R. Mohan, C. Li. Transcoding Internet Content for Heterogeneous Client Devices. IEEE Conference on Circuits and Systems, Monterey, Jun 1998.
24. M. Hemy and al. MPEG system Streams in Best-Effort Networks. Packte Video'99, Cagliari, Italy, April 1999.
25. M.Thorson. VIC viewer for PocketPC, Available Online: http://www.oncoursetech.com/video/default.htm), Apr 2001.
26. W. Yuan, K. Nahrstedt, X. Gu. Coordinating Energy-Aware Adaptation of Multimedia Applications and Hardware Resource. Proceedings of the 9th ACM Multimedia Multimedia Middleware Workshop), Oct 2001.

MAFIA: A Multicast Management Solution for Access Control and Traffic Filtering

Krishna N. Ramachandran and Kevin C. Almeroth

Department of Computer Science
University of California
Santa Barbara, CA 93106, USA
{krishna, almeroth}@cs.ucsb.edu

Abstract. Recently, multicast has seen only marginal wide-scale deployment. One of the main reasons is the lack of support for security and traffic management. Although there has been some recent work, these efforts have not emphasized the critical need to deploy security features side-by-side with management solutions. In this paper, we propose MAFIA, a multicast management solution with the specific aim of strengthening multicast security through multicast access control, multicast traffic filtering, and the prevention of DoS attacks. MAFIA achieves these tasks by making use of information about multicast group memberships available at different locations in a network. We have also designed various deployment solutions for MAFIA. We have implemented one such solution using the GNU/Linux operating system.

1 Introduction

Although IP multicast is an efficient technology for the delivery of multimedia, only a small percentage of end users receive multimedia through multicast streams. This has mainly to do with the lack of global deployment of multicast. Several reasons have been cited for slowing deployment, chief among them being concerns with multicast security and multicast management [1].

The need for security solutions is only now receiving attention from the engineering and research communities [2]. Unfortunately, the need for management solutions has received little attention [1],[3]. This is in spite of the importance of management solutions for purposes of multicast access control and traffic management. Access control is needed so that multicast access can be controlled on per host and per multicast source/group basis. For example, in an enterprise it may make sense to allow only some privileged hosts to send and receive traffic from a multicast group, whereas all other hosts are restricted to only receiving traffic from that group. Traffic management is important for reasons of efficient bandwidth utilization, quality of service, pricing, and security.

Some solutions have been proposed for multicast access control and multicast traffic filtering. For example, research has been done to control multicast access through encryption and the selective distribution of the keys used for encryption

[4],[5]. However, such solutions rely on complex cooperation between the participants of a multicast group and function at layers above the network layer. In the case of multicast traffic filtering, we are aware of one solution [6] that relies on a proxy-based approach to unicast multicast datagrams to end receivers. Their approach does not scale well for two reasons. One, it relies on unicast delivery of packets to end receivers. Two, it requires changes to host software. In fact, we are not aware of any firewall — commercial or experimental — that natively supports multicast traffic filtering. In reality, most firewalls are configured to drop UDP packets and therefore, also block multicast traffic. To allow multicast traffic through firewalls, tunneling techniques have been proposed [7],[8]. However, these techniques do not address the real issue, which is to ensure that harmful multicast traffic does not enter or leave the protected enclave.

For the quicker adoption of multicast, multicast management solutions are needed that can 1) control multicast group membership, 2) filter multicast traffic flowing in and out of the enterprise, and 3) prevent multicast denial of service attacks using multicast access control as a prevention technique. In this paper, we propose such a management solution, called MAFIA. There are two main challenges in the design of MAFIA. One, MAFIA needs to accommodate the "open" IP service model that multicast is based on. In such a model, the identity of an end host is not maintained by routing entities. This makes group membership control difficult because the identity of a host is needed to control its membership to a group. Two, MAFIA needs to be deployed such that it intrudes as little as possible on the normal operation of the network. By this we mean that end hosts should not be aware of the existence of such a solution. Consequently, they need not cooperate with MAFIA for it to function. Also, routers should undergo little change to support MAFIA. Accommodating these goals comes with some tradeoffs. We discuss the associated tradeoffs when we evaluate the deployment options for MAFIA against a number of factors such as ease of deployment, flexibility in terms of functionality, and routing state overhead.

The remainder of this paper is organized as follows. In Section 2, we reason why multicast management should complement security specific solutions to improve multicast security. Section 3 presents the requirements for MAFIA. In Section 4, we propose the MAFIA architecture and also discuss various options for its deployment. In Section 5, we evaluate the deployment options and in addition, discuss our implementation of MAFIA. Finally, we conclude in Section 6.

2 Multicast Security and Multicast Management

This section describes in detail why solutions to manage multicast need to be deployed side-by-side with security-specific solutions. We start first by broadly classifying multicast security as follows:

1. prevention of *Data Attacks*: the protection of data exchanged between hosts. Data attacks compromise the confidentiality and integrity of data.

2. prevention of *Control Attacks*: the protection of control information exchanged between multicast routing entities. Control attacks compromise the multicast routing state stored in routers.

Data can be protected using encryption. However, in the absence of access control, encryption alone cannot prevent an *edge-receiver attack* [9], an attack in which a multicast receiver joins an encrypted transmission to simply waste bandwidth or to record the encrypted traffic. If the traffic is in fact recorded, it could be decrypted in non-realtime by leveraging easily available computing power. This compromises data confidentiality. Furthermore, in the absence of access control, encryption cannot prevent an *edge-sender attack* [9], an attack in which a malicious host sends bogus packets to interfere with the successful delivery of group traffic to other receivers. For example, suppose some participants in a multicast group are in a video conference. A malicious participant or outsider can transmit bogus traffic to this group and garble the legitimate traffic in the group. Therefore, host access control needs to be used along with techniques such as encryption to protect data.

Prevention of control attacks can be partially achieved using encryption technologies such as IP-Sec [10]. With encryption, routing entities exchange control information over a secure channel this making it almost impossible for a malicious routing entity to inject bogus routing state. However, just the encryption of control traffic does not prevent denial of service attacks against routing entities. For example, a receiver, by subscribing to a large number of multicast groups, can waste bandwidth and overload routers with excess Protocol Independent Multicast (PIM) forwarding state. As another example, consider attacks launched by the RAMEN or Sapphire worms. These attacks, launched from end hosts, resulted in routers becoming overloaded with large amounts of bogus Multicast Source Discovery Protocol (MSDP) state. Therefore, just the encryption of control traffic is not sufficient to prevent such attacks, as these attacks were launched from end hosts whose access to multicast could not be controlled.

Even if multicast access control can be achieved, a sometimes overlooked problem is that UDP is the cause of some security breaches and is often blocked [6]. Blocking of UDP traffic may be too stringent a requirement for enterprises where the potential savings with the use of multicast far outweigh the threat, if any, with its use. Two solutions to minimize the threat with UDP are as follows:

– limit the use of multicast to only trusted hosts and groups. This can be done by controlling access with the use of *multicast security policies*. A multicast security policy defines which groups and hosts are considered safe. These policies can be enforced at the above mentioned control points.
– filter traffic flowing into the enterprise using state gathered from multicast routing protocols in accordance with the multicast security policies.

Clearly, it follows from the above discussion that management of multicast in an enterprise — through access control and the multicast traffic filtering — is needed along with security specific solutions to improve the overall security of a multicast deployment.

3 MAFIA Requirements

In this section, we discuss the requirements for MAFIA in detail. Briefly, the requirements are: (1) Multicast Access Control, (2) Multicast Packet Filtering, and (3) Prevention of DoS Attacks.

3.1 Multicast Access Control

Multicast access control can be broadly classified as *host access control* and *designated router access control*. Host access control controls which host can be a member of a certain multicast group. Controlling the membership behavior of a group of hosts on a subnet to subnet basis is achieved through designated router access control. For host access control, exact host-to-group associations are needed. For designated router access control, designated-router-to-group associations are needed. These two associations i.e. host-to-group associations and designated-router-to-group associations, are available at two distinct locations in the network, we call the *Last Hop Control Point* and the *Central Control Point*. The two points are shown in Fig. 1. The two access control functions are further defined below:

- **Host Access Control.** Since a host can be either a sender or receiver in a multicast group, host access control can be of the following two types:
  1. *Receiver Access Control*: The reception of multicast traffic on per (S,G), and per host basis can be controlled. Receiver access control can be very useful in bandwidth control and the prevention of edge-receiver attacks.
  2. *Source Access Control*: Source behavior can be controlled on per group and per host basis. Source access control can be very useful in the prevention of edge-sender attacks.
- **Designated-Router Access Control.** Having designated router access control is useful in the following cases:
  1. the last hop control points lie in a different administrative domain.
  2. host access control is not implemented or not necessary on the last hop.
  3. to prevent denial of service attacks launched from different subnets. At the granularity of each last hop control point, a distributed attack would not be detectable. To detect such attacks, a global view of the entire network is needed, which can be obtained by looking at designated router membership behavior.

  Since designated routers act on behalf of receivers and sources, the two resulting access control types are *Proxy-Receiver Access Control* and *Proxy-Source Access Control*. With proxy-receiver access control, the reception of multicast traffic on per (S,G), and per subnet basis can be controlled. With *proxy-source access control*, notifications of new sources sent by designated routers to the local rendezvous point (RP) can be controlled.

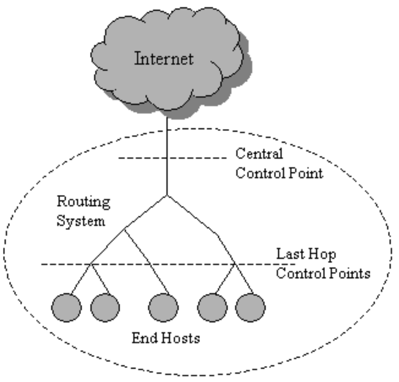


Fig. 1. Protocol regulation points in a network-hierarchy.

3.2 Packet Filtering

UDP traffic is generally blocked by network administrators. One of the reasons is that UDP's connectionless nature can be exploited for performing "port-spoofing" attacks. This problem is not specific to multicast communication. It applies to unicast as well. An effective mechanism to prevent UDP based attacks is to filter malicious packets at the firewall using the multicast policy.

3.3 Prevention of Multicast DoS Attacks

Multicast protocols are vulnerable to DoS attacks. Some attacks result from flawed protocol implementations [11]. However, most of the easily exploitable problems are due to poor protocol specifications. For example, MSDP, a protocol used to advertise the actively transmitting multicast sources, exchanges MSDP Source Active (SA) messages that carry advertisements using a *flooding* mechanism. Flooding of SAs makes MSDP inherently unscalable by design. Attacks by the RAMEN and Sapphire worms are examples of how MSDP's flooding mechanism can be exploited [2]. Using IGMP, it is extremely easy for an end-host to launch edge-sender and edge-receiver attacks. Another consequence of such attacks is the overloading of PIM routers because of the large amount of PIM state created for delivering unwanted traffic.

The problem with DoS attacks can be most effectively solved using a dual approach. One is to limit the use of multicast to only trusted hosts and groups. This can prevent internally launched IGMP, PIM, and MSDP DoS attacks. The second approach is filter bogus packets that result from DoS attacks launched from external networks. For instance, MSDP attacks launched from external networks can be prevented by filtering bogus MSDP SAs.

4 MAFIA Architecture

From our discussion of the MAFIA requirements in Section 3, it follows that two separate functional modules are needed: one to filter UDP packets (requirement 1) and the other to control multicast access (requirement 2). We call the two modules the *MAFIA Packet Filter* and the *MAFIA Access Controller* respectively. The Packet Filter is co-located with the protected enclave's firewall at a central control point (see Fig. 1). The access controller, on the other hand, is situated in the interior of the protected enclave. Requirement 3 from MAFIA is necessary to prevent IGMP, MSDP, and PIM DoS. IGMP edge-sender and edge-receiver attacks are prevented by the MAFIA access controller. Its detailed operation is explained later in this section. PIM DoS attacks are prevented as a consequence of preventing the IGMP attacks. In addition, preventing an IGMP edge-sender attack will also prevent the launch of MSDP DoS attacks from the inside of the protected enclave. This is because containing the number of groups a sender can transmit traffic to will automatically limit the number of MSDP SA messages generated by the local RP. However, externally launched MSDP attacks can still affect the MSDP peer in the protected enclave if the influx of bogus SAs from the outside is not prevented. Since, the MAFIA Packet Filter is co-located with a firewall, the filtering of SAs is also done by the MAFIA packet filter.

Figure 2 illustrates the conceptual view of the MAFIA architecture. It shows a third module, the MAFIA policy server. The policy server maintains the multicast policy. Updates to the policy are always done at the policy server. The updates are then mirrored at the access controller and the packet filter to reduce the latency involved in serving an access request.

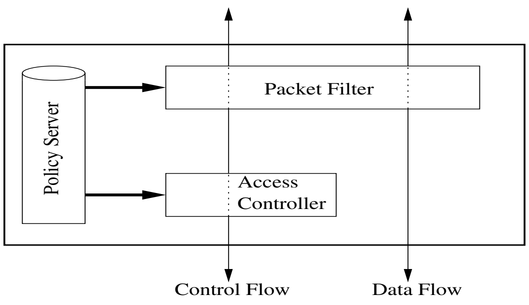


Fig. 2. MAFIA architecture.

This rest of this section describes the architecture of the MAFIA access controller and the MAFIA packet filter.

4.1 MAFIA Access Controller

The MAFIA access controller implements the four types of access control discussed in Section 3.1. For host access control, host-to-group associations are needed. These are available only at the last hop control points. On the other hand, for designated router access control, designated-router-to-group associations are needed. This information is available at the centralized control point. As these two types of associations are available at two distinct locations in the network hierarchy, the MAFIA access controller is composed of two separate modules present at each of these two locations. We call these modules the MAFIA Last Hop Control Point (MLHCP) and the MAFIA Centralized Control Point (MCCP). Figure 3 illustrates the placement of the MLHCP and MCCP in a network. Although the MAFIA Access Controller is made up of the MLHCP and the MCCP, their architecture details and deployment considerations are discussed separately. This is because both modules function independent of each other and as a result do not affect each other's operation in any way.

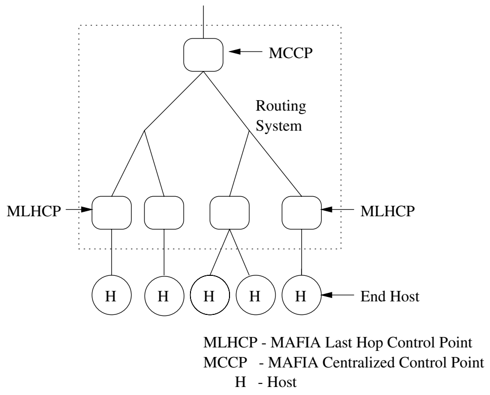

MCCP
Routing System
MLHCP
MLHCP
H
H
H
H
H
End Host
MLHCP - MAFIA Last Hop Control Point
MCCP - MAFIA Centralized Control Point
H - Host

Fig. 3. MAFIA access controller.

MAFIA Last Hop Control Point (MLHCP). The MLHCP implements receiver and sender access control. IGMP membership reports are used to implement receiver access control. When the MLHCP receives an IGMP report, it uses the locally cached multicast policy to decide if the requested access is permitted. For sender access control, the arrival of a multicast datagram is the trigger to authorize a transmission. This is because a multicast sender can send traffic to a group without ever joining the group.

The MLHCP can be deployed either actively or passively, depending on whether the MLHCP is also a routing entity. The MLHCP deployed with the designated router is an active MLHCP, as it performs routing functions. Router vendors provide support for host access control at the designated router through static ACLs (access control lists). However, the flexibility offered by this is limited as additional tasks such as inspecting packet payloads or maintaining state between multiple packets cannot be easily done on routers.

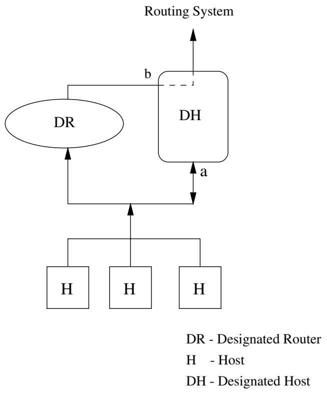


Fig. 4. Designated host as MLHCP.

Figure 4 illustrates the passive solution. Here, the designated router ignores (using ACLs) IGMP reports received from all hosts except ones received from the *designated host* — a dual network interface host that acts as a proxy for all other hosts in the subnet. Interface *a* of the designated host listens to all IGMP reports generated on the last hop subnet. Interface *b* receives all PIM Register messages generated by the designated router. When a host sends an IGMP membership report expressing interest in receiving traffic from a group, the report is received by the designated router and the designated host. As the designated router is configured to ignore all reports except ones from the designated host, it ignores the report. When the designated host receives the membership report, it checks if the host is permitted to perform the requested operation. If the requested operation is permitted, the designated host in turn generates a membership report with the same information as contained in the original report. Since this report is generated by the designated host, the designated router now accepts it and initiates the creation of the distribution tree.

For sender access control, as the MLHCP receives all PIM register messages, it checks whether the operation is permitted by the policy. If the PIM register message is not permitted, the MLHCP simply drops the message. An alternate configuration for this interface is to operate in snooping mode. In snooping mode, the interface listens only to the PIM register messages. When the MLHCP functions in this mode, it cannot prevent messages from reaching the upstream RP. Therefore, to counteract the effect of the PIM register message, the MLHCP masquerades as the RP and originates a PIM unregister message towards the designated router. When the designated router receives the PIM unregister message, it ignores any PIM register messages that the RP may generate. Consequently, it does not forward any data towards the RP. The disadvantage with the snooping configuration is that the MLHCP cannot prevent unauthorized PIM register messages from reaching the RP. Therefore, if a host randomly sends data to a large number of multicast groups — like in a RAMEN worm attack — a large number of PIM register messages will reach the RP. This may launch a MSDP SA flood.

MAFIA Centralized Control Point (MCCP). The MCCP performs designated router access control. It implements proxy-receiver access control by filtering PIM Join messages. Proxy-sender access control is implemented by filtering PIM Register messages. In considering MCCP's deployment, it helps to classify the MCCP based on the role it plays in multicast routing as follows:

- **Active MCCP:** An active MCCP is also a multicast routing entity and therefore takes part in the creation and maintenance of distribution trees. Figure 5 shows one possible deployment solution where the MCCP is implemented in a multicast router. Router vendors already provide some support for controlling designated router behavior by way of access control lists (ACLs). However, the flexibility offered by such a solution is limited. For instance, certain signature-based attack detection techniques [2] require that state be maintained between packets. Such flexibility is not offered by ACLs.
- **Passive MCCP:** A passive MCCP is not a multicast routing entity. However, it receives every protocol message destined for upstream routers. It can be a dedicated system that performs complex tasks such as inspecting packet payloads, maintaining state between multiple packets, detecting attack signatures, and packet monitoring. Figure 6 illustrates a passive MCCP deployment. In this deployment, packets are filtered before they reach upstream routers. The MCCP can also be deployed in snooping mode. In snooping mode, the MCCP cannot filter packets. Therefore, it reacts to an unauthorized request by sending a protocol message that counteracts the request. So, for PIM Join messages, the snooping MCCP will send a PIM Prune message to the upstream PIM router. For a PIM Register message, the snooping MCCP will send a PIM Unregister message to the designated router that sent the register message. A snooping MCCP, however, is not completely effective in preventing PIM flooding attacks. This is because if a large number of unauthorized requests are sent in a short period of time, PIM routers will get overloaded, albeit temporarily, with large amounts of state.

4.2 MAFIA Packet Filter

The MAFIA packet filter does two types of packet filtering:

UDP Filtering. UDP filtering is simple UDP flow filtering to ensure that UDP packets that flow through the firewall match certain criteria. If the UDP packets do not match the given criteria, they are dropped. The criteria for UDP packet

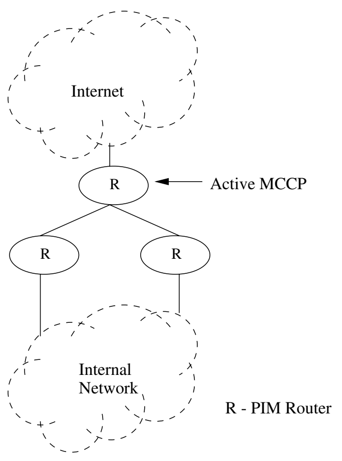


Fig. 5. Active MCCP.

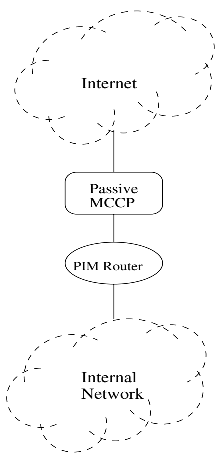


Fig. 6. Passive MCCP.

filtering is specified as part of the multicast policy. The simplest criteria is to ensure a multicast datagram in an incoming or an outgoing stream carries a destination multicast address that corresponds to some "live" multicast distribution tree. MAFIA keeps track of "live" trees by tracking PIM Join messages and corresponding PIM Prune messages. In the normal case, multicast routers generally never forward packets that do not belong in a distribution tree. However, experience tells us that malfunctioning routers erroneously forward such packets. This simple criteria will ensure that bogus packets are dropped. More complex criteria can also be used to filter packets. For example, one criteria would be to drop all multicast datagrams destined to well-known ports. Such a criteria can be effective in preventing UDP port spoofing attacks. Signature-based schemes can also be applied to filter malicious UDP packets with unique signatures.

MSDP-SA Filtering. MSDP SA filtering is done by looking up the multicast policy and determining which multicast groups are permitted by the policy. Only SAs for "joinable" groups are let through the firewall and the remaining are dropped. As with UDP filtering, signature based schemes can be used to filter MSDP SAs. For instance, the RAMEN worm has a unique signature pattern of a large number of SAs with increasing class D addresses originating from the same source. Other SA filtering schemes proposed in [2] can also be applied for more effective filtering.

5 Evaluation

In this section, we evaluate the MAFIA architecture discussed in Section 4. Our goal is to evaluate the architectures against various evaluation metrics. First, we present the methodology used in our evaluations.

5.1 Methodology

The performance of MAFIA depends on the following factors:

– **MAFIA System Configuration:** Performance of MAFIA ultimately depends on its hardware and software configuration. Factors include processor speed, amount of memory available, network card capabilities, and operating system used.
– **Multicast Group Characteristics:** Performance depends on characteristics such as number of groups, sources, and receivers.
– **Traffic Characteristics:** The traffic characteristics depend on the group characteristics and the rate at which each multicast source transmits.
– **Multicast Policy:** If the multicast policy is complex and restrictive in terms of allowing multicast access, the load on MAFIA is greater.
– **Link Bandwidth:** Link bandwidth may be low enough for MAFIA to operate effectively even under maximum utilization. On the other hand, MAFIA may not be able to handle high bandwidth traffic.

We considered evaluating MAFIA through simulations. However, trying to accommodate the above factors in a simulation environment would be difficult and the results would not lead us to any particularly non-obvious conclusions. Therefore, instead of presenting empirical results, we instead focus on an evaluation of the various ways MAFIA can be deployed. To this end, we limit our evaluations only to the MAFIA access controller. This is because the access controller, which is composed of the MLHCP and the MCCP, can be deployed in more than one way and each deployment option offers some interesting tradeoffs. The MAFIA Packet Filter, on the other hand, can be deployed in only one way.

5.2 Evaluation Criteria

We use the following criteria for our evaluations:

– **Ease of deployment:** We evaluate the ease with which the various architectures can be deployed.
– **Flexibility:** We evaluate the flexibility offered by an architecture in terms of the range of features (functionality) an architecture can support.
– **Traffic Rates:** We evaluate the capability to handle high traffic rates.
– **Routing state:** We evaluate scalability in terms of how much routing state needs to be maintained by an architecture.

5.3 Evaluation Results

MLHCP. We evaluated the active and passive MLHCP deployment options discussed in Section 4.1. The passive MLHCP is easier to deploy as it is deployed on a dedicated system and therefore requires no changes to router software. Furthermore, deployment can be easily done using “off-the-shelf” commodity hardware and software. The active MLHCP, on the other hand, is deployed on the router, which means that the router software needs to change to support the MLHCP. The passive MLHCP also offers more flexibility than an active MLHCP. This is because the passive MLHCP does not perform any routing functions and hence can perform more complex tasks such as maintaining state between multiple packets, packet logging, and traffic analysis.

However, a passive MLHCP cannot easily handle very high traffic rates (of the order of gigabits per second). This is because commodity hardware and software cannot easily scale to higher traffic rates. To overcome this problem, in [12], the authors propose traffic splitting architectures to scale commodity hardware and software to handle high traffic rates. The problem with such architectures, as the authors themselves acknowledge, is that they are difficult to implement. The active MLHCP, on the other hand, can handle high traffic rates because the traffic rates it handles after all depends on the capability of the router itself.

With respect to routing state maintained at the designated router, the active MLHCP results in no state being maintained for unauthorized requests. This is because an active MLHCP filters an access request before it reaches the designated router. As the passive MLHCP cannot filter unauthorized requests, state is maintained at the designated router, albeit temporarily, for unauthorized requests.

In summary, the passive MLHCP is a more flexible architecture and is easier to deploy. However, the passive MLHCP cannot handle high traffic rates easily and also results in more state being maintained at the designated router as compared to the active MLHCP.

MCCP. As with the passive MLHCP, the passive MCCP is more flexible and is easier to deploy. However, unlike the passive MLHCP, the passive MCCP does not result in state being maintained at upstream routers for unauthorized requests. This is because all PIM messages are filtered before they reach upstream routers.

The problem with high traffic rates is more serious at the passive MCCP. This is because the passive MCCP now has to filter requests coming from several last hop subnets. Consequently, the passive MCCP can easily become overloaded even at moderate link usage. Traffic splitting [12] could be used to alleviate the problem with high traffic rates. However, this adds complexity to the system.

In summary, the passive MCCP is easier to deploy, more flexible and maintains no more state than what results from an active MCCP. On the other hand, the active MCCP performs better at higher traffic rates as it is deployed on a router.

5.4 Implementation

We have chosen to implement MAFIA as a combination of the passive MLHCP and the passive MCCP. Both architectures offer better flexibility than their active counterparts. Moreover, they can be deployed with no changes to router software. This means that the implementation of MAFIA is not tied to any one router vendor's product.

The MLHCP and the MCCP have been implemented on the GNU/Linux operating system using the netfilter and iptables frameworks[1]. Using these frameworks, GNU/Linux offers a comprehensive packet filtering capability that is open-source, well tested, and widely deployed. Netfilter uses a loadable kernel module (LKM) called a *match* to filter packets. For MAFIA, we implemented two match modules called *MSDP match* and *PIM match*. The PIM match is used to filter PIM Register and PIM Join messages. The MSDP match is used to filter MSDP Source Active messages. The MAFIA packet filter uses the MSDP match and the UDP match (already existing in netfilter) to do MSDP and UDP packet filtering. The PIM match is used by the MCCP and the MLHCP. The MCCP uses the PIM match to filter PIM messages before they reach upstream routers. The MLHCP uses the PIM match to filter PIM messages originated by the designated router. In addition to the PIM match, the MLHCP also uses the IP match module to filter IGMP reports on the last hop. For authorized requests the MLHCP uses the libnet[2] packet-generation tool to generate IGMP reports.

We tested our implementation of MAFIA using a netfilter enabled GNU/Linux system. We used a packet generating tool written using libnet and libpcap[3] to generate our test traffic. The packet generator uses information in its configuration file to randomly send a mixture of UDP, PIM Join, PIM Register, and MSDP SA packets to the Linux system. We used a restrictive multicast policy to create the appropriate netfilter rules on the GNU/Linux system. All packets sent by the packet generator that do not match the netfilter rules were dropped. This test confirms the correct operation of our MAFIA implementation.

6 Conclusions

The lack of multicast management adversely affects multicast security. A testament to this is the recent increase in the number of denial of service attacks against multicast. Moreover, multicast management enables network administrators to manage their multicast deployments for purposes of administrative control and efficient resource utilization.

In this paper, we have proposed MAFIA, a multicast management solution that addresses three requirements: multicast access control, multicast data and control packet filtering, and denial of service attack prevention. We have looked at various ways MAFIA can be deployed. In addition, we have evaluated each deployment option against various factors such as ease of deployment, flexibility, routing state overhead, and its capability to handle high traffic rates. As a result, network designers need to consider the tradeoffs associated with each deployment option before deploying MAFIA in the network. We have implemented MAFIA using the netfilter architecture in the GNU/Linux operating system. We plan

[1] http://www.netfilter.org
[2] Libnet packet assembly tool, http://www.packetfactory.net/libnet
[3] Libpcap packet capture tool, http://sourceforge.net/projects/libpcap

to offer the netfilter extensions written for our implementation in an upcoming release of netfilter.

References

1. K. Sarac and K. Almeroth. Supporting Multicast Deployment Efforts: A Survey of Tools for Multicast Monitoring. *Journal of High Speed Networking–Special Issue on Management of Multimedia Networking*, March 2001.
2. P. Rajvaidya, K. Ramachandran, and K. Almeroth. Detection and Deflection of Denial of Service Attacks against the Multicast Source Discovery Protocol. *UCSB Technical Report*, May 2003.
3. E. Al-Shaer and Y. Tang. Toward integrating IP multicasting in internet network management protocols. *Computer Communications*, 24(5-6):473–485, 2001.
4. C.K. Wong, M. Gouda, and S. Lam. Secure group communications using key graphs. In *ACM SIGCOMM*, pages 68–79, 1998.
5. I. Chang, R. Engel, D. Kandlur, D. Pendarakis, and D. Saha. Key management for secure internet multicast using boolean function minimization techniques. In *IEEE Infocomm'99*, pages 689–698, 1999.
6. K. Djahandari and D. Sterne. An MBone proxy for an application gateway firewall. *IEEE Symposium on Security and Privacy*, 1997.
7. R. Finlayson. The UDP Multicast Tunneling Protocol. Internet Engineering Task Force (IETF), draft-finlayson-umtp-*.txt, September 2002.
8. D. Chouinard. SOCKS V5 UDP and Multicast Extensions to facilitate multicast firewall traversal. Internet Engineering Task Force (IETF), draft-ietf-aft-mcast-fw-traversal-*.txt, November 1997.
9. T. Hardjono and G. Tsudik. IP multicast security: Issues and directions. *Annales de Telecom*, 2000.
10. *IP Security Protocol (ipsec).* http://www.ietf.org/html.charters/ipsec-charter.html.
11. *Spoofed IGMP Report Denial of Service Vulnerability.* http://online.securityfocus.com/bid/5020/info.
12. C. Kruegel, F. Valeur, G. Vigna, and R. Kemmerer. Stateful intrusion detection for high-speed networks. *IEEE Symposium on Security and Privacy*, May 2002.

A Heuristic Algorithm for the Multi-constrained Multicast Tree

Wen-Lin Yang

Department of Information Technology
National Pingtung Institute of Commerce
No.51, Ming-Sheng East Road, Pingtung City,Taiwan
wly@npic.edu.tw

Abstract. Multicasting is an important communication mechanism for implementing real-time multimedia applications, which usually require the underlying network to provide a number of quality-of-service (QoS) guarantees to users. In this paper, we study the problem concerning how to find a feasible multicast tree in which all the multicast paths satisfy the given set of QoS constraints. This problem is referred as the multi-constrained multicast tree (MCMT) problem. Based on tabu-search technique, a heuristic algorithm named MCMTS for the MCMT problem is proposed in this study. According to the experimental results, the probability of finding a feasible multicast tree by our MCMTS method is more than 99% if one exists. Furthermore, the MCMTS is also shown to be a simple and highly efficient method. As a result, it would be a practical approach for developing multicast routing protocol.

1 Introduction

For multimedia applications involved in real-time communications, they usually require the underlying network to provide a number of quality-of-service (QoS) guarantees to users. For a network with a set of QoS constraints, a routing problem studied in this paper is to find a multicast tree such that any path between a pair of source and destination nodes satisfies the given set of QoS constraints simultaneously. This routing problem is also referred as the multi-constrained multicast tree (MCMT) problem.

In the past, the MCMT problem did not receive much attention. Most of previous research on finding a multicast tree considers only one or two QoS constraints, like delay and jitter [8,9,10,11]. As far we know, only two methods have been published [6,13] recently for the MCMT problem. In paper [6], an algorithm named MAMCRA is proposed for the MCMT problem. Their algorithm begins with finding a set of shortest paths from the source node to all destinations. By removing the overlap of paths, a multicast tree is then obtained. However, the problem finding a path between any two nodes is a NP-complete problem [3], and it is referred as the multi-constrained path (MCP) problem in literature. A number of heuristic algorithms have been proposed for it [1,4,5,15,16].

In paper [13], a heuristic based on genetic algorithm called MCMGA is proposed. The goal of this method is not just to find a feasible multicast tree, but also to minimize the cost of the tree. The first step of MCMGA is to generate k shortest paths for each pair of source and destination. Then, a large of number of multicast trees must be maintained for generic operations. Since k could be a large number and many multicast trees must be kept for every generation, the MCMGA is not just too complex but also very inefficient for large-scale networks with a large set of destinations.

For networks with tight QoS constraints, the number of feasible multicast trees could be very few, especially when the number of QoS constraints increases. Hence, a routing algorithm for finding a least-cost multicast tree would be too expensive to be practical for routing protocol implementation. In this paper we are only concerned about how to find a feasible multicast tree efficiently. The resulting multicast tree is not necessary to be a least-cost tree. It has been noticed that all multicast routing protocols are developed based on simple multicast routing algorithm [7], where simplicity and ease of implementation should be the most important criterion when evaluating multicast routing algorithms. Hence, the goal of this study is to develop a simple and efficient algorithm for the MCMT problem.

Our heuristic algorithm proposed in this paper is named the multi-constrained multicast tree algorithm based on Tabu-search technique [2] (MCMTS). The MCMTS method begins with finding a good spanning tree in which a multicast tree is embedded. Since the resulting multicast tree may be infeasible, some multicast paths may violate the QoS constraints. A path replacement procedure guided by tabu-search strategy is then used to improve those infeasible multicast paths. Our MCMTS algorithm is represented in section 3. The results of numerical simulations are given in section 4.

2 The Multi-constrained Multicast Tree Problem

Consider a network that is modeled by a directed graph $G(V, E)$, where V is a set of nodes and E is a set of links. Each link $(u, v) \in E$ is associated with k positive additive QoS parameters, such as delay, delay jitter, data loss rate, etc. The value of QoS parameter i is represented by $w_i(u, v)$, where $i = 0, 1, ..., (k-1)$. Since the network could be asymmetric, $w_i(u, v)$ may be not equal to $w_i(v, u)$ for some QoS parameter i. For a path P and a QoS parameter i, we use the path constraint $W_i(P)$ to represent the summation of $w_i(u, v)$ on each link (u, v) on the path P. That is, $W_i(P) = \sum_{(u,v)\in P} w_i(u, v)$. In this paper, we also use the notation $(S \rightarrow j)$ to represent the path from S node to node j.

Let D be a subset of V ($D \subset V$ and $|D| = m$), and denote a group of destination nodes of a multicast tree. Given a set of nodes D which does not contain the source node S, and k QoS constraints C_i, $0 \leq i \leq (k-1)$, the goal of our MCMT problem is to find a multicast tree T such that the following conditions are hold:

Assume P_j is the path from source S to any node j in the destination group D, $W_i(P_j) \leq C_i$, for all i and j, where $0 \leq i \leq (k-1)$, and $0 \leq j \leq (m-1)$.

3 The MCMTS Algorithm for the MCMT Problem

A simple and efficient heuristic algorithm named MCMTS based on tabu search strategy is developed in this section. The basic idea of our MCMTS method can be outlined as follows:

(a) First of all, a 'good' spanning tree T rooted at the source node must be determined. The spanning tree covers all the nodes in the given network, and a multicast tree M that covers all the nodes in the set D is embedded in T. A method for determining 'good' spanning tree is presented in section 3.1.
(b) Based on the multicast tree M obtained in step (a), for each node u in the set D, all the QoS path constraints are checked. If M is a feasible solution, the procedure stops. Otherwise, go to step (c).
(c) For any destination node u, whose path from source violates QoS constraints, an efficient tabu-search based procedure is applied to modify the path from source to u. The procedure stops after a given number of iterations. This method is presented in section 3.2.
(d) If the multicast tree M obtained from step (c) is still infeasible, we collect all the destination nodes whose paths do not satisfy the given QoS constraints. Let this collection be Γ.
(e) For any node u in Γ, a fix procedure is used to find a feasible path from source to u. The fix procedure is an optimal or heuristic algorithm for solving the multi-constrained path (MCP) problem. Since the simulation results presented in section 4 show that Γ is a small set, the running time spent on these MCP problems is small.

In general, our MCMTS algorithm contains two parts: a modified Prim's algorithm for determining a good spanning tree, and a tabu-search based procedure for improving the multicast tree embedded in a spanning tree.

3.1 The Modified Prim's Method

The well-known Prim's algorithm designed for finding the minimum spanning tree is modified in this section to determine a 'good' spanning tree T for the given network. A multicast tree M covers all the destinations is embedded in T. The modified Prim's method is presented in Fig. 1. During the construction of spanning tree T, an attribute vector $Y(u)$ is computed and saved for any new node u added into T. $Y(u)$ is defined as follows:

$Y(u) = [\alpha_0, ..., \alpha_{k-1}]$, $\alpha_i = W_i(P)/C_i$, $W_i(P) = \sum_{(x,y)\in P} w_i(x,y)$,

$0 \leq i \leq (k-1)$, where P is the path from source node s to node u and $P \in T$.

Hence, for a link $(u,v) \in T$, assume $Y(u) = [\alpha_0, ..., \alpha_{k-1}]$ and $Y(v) =$

$[\beta_0, ..., \beta_{k-1}]$, then $\beta_i = \alpha_i + w_i(u,v)/C_i$, $0 \leq i \leq (k-1)$. If a path $(s \to u)$ is feasible, then every element of the attribute vector $Y(u)$ must be equal to or less than one. For any node u with attribute vector $Y(u)$, a cost value $\Psi(u)$ is defined as follows: $\Psi(u) = \sum_{i=0}^{k-1} \alpha_i$. In each iteration of the 'While' loop in Fig. 1, only one node is selected to add into a partially built spanning tree T. The selection strategy is that the node with the smallest cost value Ψ has the highest priority to be selected.

An example of 4-node network is given in Fig. 2 to illustrate the modified Prim's method. In Fig. 2, c_0 and c_1 are two given QoS constraints. When $T = \phi$ and $F = s$, there are two nodes, 'a' and 'b', can be added into set F. For node 'a', the cost value $\Psi(a)$ is equal to 0.5 since the attribute vector $Y(a) = [1/4, 1/4]$. For node 'b', the cost value $\Psi(b)$ is equal to 1.5 because of $Y(b) = [1, 1/2]$. Hence,link (s,a) has higher priority than link (s,b) to be selected to add into T, which is shown in Fig. 2(b). The desired 'good' spanning tree T is given in Fig. 2(c).

```
 1. Given a network $G(V,E)$ and a source $s$;
 2. Let $F = \{s\}$, $T = \phi$, and $Y(s) = [0,..,0]$;
 3. Let $Q = \{v|(u,v) \in E, u \in F$ and $v \notin F\}$;
 4. While($F \neq V$){
 5.   For each node $v \in Q$ {
 6.    For each node $u \in F$, if the link $(u,v) \in E${
 7.     Compute attribute vector $Y(v)$ based on $Y(u)$ and $w_i(u,v)$,$0 \leq i \leq$
        $(k-1)$;
 8.     Let $Y(v) = [\beta_0, .., \beta_{k-1}]$, and a cost value $\Psi(v) = \sum_{i=0}^{k-1} \beta_i$;
 9.     Keep the link $(u,v)$ with the smallest value of $\Psi$;
10.   }}
11.   $T = T \cup (u,v)$, $F = F \cup \{v\}$;
12. }; Output the spanning tree $T$;
```

Fig. 1. The modified Prim's algorithm

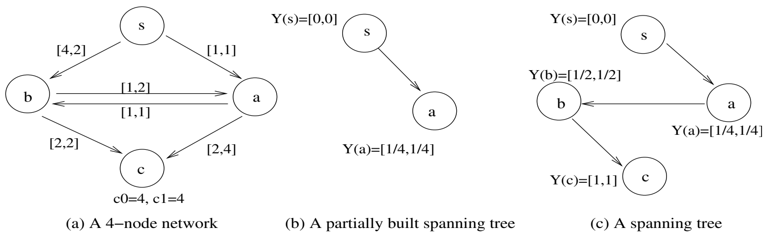


Fig. 2. An example for generating spanning tree.

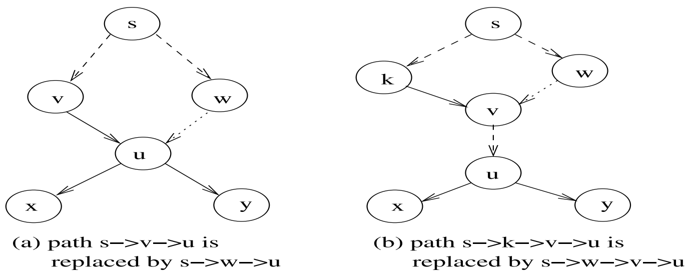

s
v
w
u
x
y
(a) path s—>v—>u is replaced by s—>w—>u
s
k
w
v
u
x
y
(b) path s—>k—>v—>u is replaced by s—>w—>v—>u

Fig. 3. The path rebuilding process.

3.2 The Tabu-Search Based Procedure

In the multicast tree M embedded in the spanning tree T found by the modified Prim's algorithm, some paths from source to destinations may not satisfy the given QoS path constraints. These infeasible paths are then improved by our tabu-search based procedure presented in Fig. 4.

Consider an infeasible path $p = (s \rightarrow v \rightarrow u)$ shown in Fig. 3(a) where s is the source and u is a destination, the goal of our tabu-search based procedure is to find a new path $\bar{p}$ that can reach destination u. Since $\bar{p}$ may be infeasible, the decision about whether p is replaced by $\bar{p}$ is based on the equation (A) given in the following: at node u, assume that

(a) $Y(u) = [\alpha_0, .., \alpha_{k-1}]$, $\Psi = \sum_{i=0}^{k-1} \alpha_i$ for path p;
(b) $Y'(u) = [\beta_0, .., \beta_{k-1}]$, $\Psi' = \sum_{i=0}^{k-1} \alpha_i$ for path $\bar{p}$;
(c) $\Psi' < \Psi + \theta$, $0 \leq \theta < k$, ...(A)

When θ is zero and the equation (A) is hold, the path $\bar{p}$ is said to be better than path p, and p can be replaced by $\bar{p}$. However, this condition may be too strict for our tabu-search based procedure. To avoid the searching path trapped in a local optimal, our heuristic procedure may accept a new path $\bar{p}$ worse than path p by setting θ value greater than zero. Hence, a path $\bar{p}$ can replace a path p even though Ψ' is greater than Ψ. In our simulations presented in section 4, the value of θ is set to be 0.5.

In Fig. 3(a), the new path $\bar{p} = (s \rightarrow w \rightarrow u)$ satisfied equation (A) is found at the destination u. Note that a link (w, u) must satisfy the following conditions: $(w, u) \in E$, $(w, u) \notin T$ and w cannot be in the sub-tree rooted at node u to avoid creating a cycle. However, in Fig. 3(b), no acceptable path can be found at the destination u. The ancestor nodes of u are then tested sequentially in order to construct a desired new path $\bar{p}$. In Fig. 3(b), for example, a new path is built based on node v, which is an ancestor of the destination u. In this paper, node u in Fig. 3(a) and node v in Fig. 3(b) are all referenced as a "branch point" on the upward path from u to s. In Fig. 3(b), if a new path $\bar{p}$ based on a "branch point" v is constructed successfully, the attribute vectors of v and its downstream nodes must be recomputed. For any destination node u, if any element of the attribute vector $Y'(u)$ is greater than one, the new path $\bar{p}$ is still infeasible.

The above path replacement procedure is implemented in an iteration loop, which is developed based on the tabu-search strategy [2], and is shown at lines $5 \sim 23$ in Fig. 4. A circular queue is maintained in our procedure and served as a tabu-list for memorizing the links having been considered recently for constructing a new path (see line 12 in Fig. 4). For example, two links (w, u) and (w, v), which are in Fig. 3(a) and Fig. 3(b) respectively, should be stored in the tabu-list after a new path is built. Since only links not in the tabu-list are considered for rebuilding, our procedure can avoid revisiting some links that have been visited recently, and has better chance to explore some unvisited solution space.

```
 1. Given a network $G(V,E)$, a source node $s$ and a set of destination
    nodes $D$, $D \subset V$;
 2. Find a 'good' spanning tree $T$ based on the modified Prim's algorithm
    given in Fig. 1. A multicast tree $M$ is in $T$;
 3. Let linked list $R = \{u|p = (s \to u), u \in D$, $p$ violates at least one QoS
    constraints.$\}$;
 4. Initialize a circular queue $Q$ to serve as a tabu-list; $I = 0$;
 5. While($R \neq \phi$ and $I < ITERATIONS$) {
 6.    Let $u$ be the first element of $R$; $R = R - \{u\}$;
 7.    Assume the infeasible path be $p = (s \to u)$; Let $v = u$ ;
 8.    While($v \neq s$){
 9.      Assume $w = v \to parent\_node$;
10.      $Z = \{w'|w \neq w', (w',v) \in E, (w',v) \notin T\}$;
11.      Find $w' \in Z$ such that $(w',v) \notin Q$ and $w'$ is not in the sub-tree
    rooted at $v$ {
12.        $Q = Q \cup (w',v)$;Build a new path $\bar{p} = (s \to w' \to v \to u)$;
13.        Compute $Y'(v)$ and $\Psi'(v)$ based on the new path $\bar{p}$;
14.        If($\Psi'(v) < \Psi(v) + \theta$){
15.          $T = T - (w,v)$; $T = T \cup (w',v)$;
16.          Update $Y$ for $v$ and its downstream nodes in $M$;
17.          $R$=$R - \{u|u \in D, u$ is in the subtree rooted at v; the path $p$ from
    $s$ to $u$ satisfies all QoS constraints$\}$;
18.          goto next;
19.        } }
20.      $v = v \to parent\_node$;
21.    }
22. next:   $I = I + 1$;
23. }
24. If($R \neq \phi$){call a fix procedure to find feasible paths from $s$ to each
    node in $R$;}
25. Output the multicast $M$ embedded in $T$;
```

Fig. 4. The MCMTS algorithm

3.3 The Optimal Algorithm and Fix Procedures

Recall that the multi-constrained path (MCP) problem is concerned about how to find a feasible path between two given nodes, so that a set of QoS constraints can be satisfied simultaneously. Hence, for the MCMT problem of a network with m destinations, a multi-constrained multicast tree can be obtained by finding a multi-constrained path for each destination. As a result, the MCMT problem can be solved optimally if the corresponding MCP problem is solved optimally.

For a multicast tree determined by our heuristic procedure presented in Fig. 4, it is possible to have a small set of destinations whose multicast paths are infeasible. At line 24 in Fig. 4, a fix procedure is then called to find a set of multi-constrained paths for those destinations. The fix procedures used in this study can be the optimal or heuristic algorithm [15,16] developed for the MCP problem. In fact, based on the simulations conducted in section 4, the set of paths that must be determined by the fix procedure is very small. That is the reason why our MCMTS procedure is very efficient when it is compared to the optimal algorithm for the MCMT problem.

3.4 Time Complexity

The time complexity of the MCMTS algorithm is determined by the following terms: $O(n^2)$ for the modified Prim's procedure and $O(m * n * ITERATIONS)$ for the loop, where $O(n)$ is required at lines 16-17 in Fig. 4. The time complexity of line 24 in Fig. 4 is $O(d^h)$ if the heuristic algorithm proposed in paper [15] is used as the fix procedure, where d is the maximum number of node-degree and h is the maximum number of hops between source and any destination. Hence, the time complexity of our MCMTS is $O(n^2 + m * n * ITERATIONS) + O(d^h) * \delta$ where $\delta \ll m$. Obviously, the executing efficiency of our MCMTS method is determined by the value of δ.

4 Experimental Results

In this section, we have several sets of experiments for comparing executing performance and solution quality between the optimal algorithm and the MCMTS heuristic proposed in this paper for solving the MCMT problem. The optimal algorithm is extended from the optimal algorithm proposed for the MCP problem [14]. Two network topologies, random graph and mesh, are simulated in this study. All the simulations are done with the following experimental parameters: P4 2.0 GHz CPU, 512MB RAM, Linux OS, and programs are developed by C++. For all benchmarks, the values of QoS parameters assigned on each link in the network are randomly selected from the range $0 \sim 100$. In this study, each data is measured based on 1000 runs. For each run, a network configuration with different QoS assignments on each link is randomly generated. As for the values of QoS path constraints, they are assigned in such a way that feasible multicast trees can only exist in around 90% of 1000 network configurations constructed for the simulations. All networks considered in this study are asymmetric.

Table 1. Simulation results for random graphs with fix procedures

$*$ Results for 100-node random graphs. Two QoS constraints: $C_0 = C_1 = 320$.

Fix procedure	#destinations	#tabu_tree /cpu(secs)	#optimal_tree /cpu(secs)	run_time_ratio	solution quality
BB_optimal	50	846/4.95	846/170.47	2.90%	100.0%
	40	870/5.46	870/180.47	3.03%	100.0%
	30	883/4.31	883/100.63	4.28%	100.0%
	20	911/3.6	911/71.81	5.01%	100.0%
	10	958/3.1	958/34.11	9.09%	100.0%
	Average			4.86%	100.0%
TS_heuristic	50	830/7.39	834/182.3	4.05%	99.52%
	40	843/7.1	849/141.77	5.01%	99.29%
	30	897/4.08	899/98.48	4.14%	99.78%
	20	902/4.48	904/71.76	6.24%	99.78%
	10	943/3.15	944/36.15	8.71%	99.89%
	Average			5.63%	99.65%

Table 2. The simulation results for 8x8 meshes with fix procedures

$*$ Results for 64-node meshes. Two QoS constraints: $C_0 = C_1 = 560$.

Fix procedure	#destinations	#tabu_tree /cpu(secs)	#optimal_tree /cpu(secs)	run_time_ratio	solution quality
BB_optimal	32	880/110.2	880/987.5	11.16%	100.0%
	25	905/129.6	905/902	14.37%	100.0%
	19	937/62.2	937/610.5	10.19%	100.0%
	12	963/55.7	963/482.2	11.55%	100.0%
	6	972/33.1	972/191.2	17.30%	100.0%
	Average			12.91%	100.0%
TS_heuristic	32	899/11.2	911/1062.3	1.05%	98.68%
	25	904/6.8	912/829.7	0.82%	99.12%
	19	918/8.1	927/577.7	1.4%	99.03%
	12	959/5.4	966/418	1.28%	99.28%
	6	975/4.0	977/161.7	2.49%	99.80%
	Average			1.41%	99.18%

A random graph generator [7], which was modified from Waxman's graph generator [12], was used to create links interconnecting the nodes. In order to imitate real networks more closely, the average degree of nodes in each network is set to be four [7].

4.1 With Fix Procedures

Two fix procedures are used in this study. One is the branch-and-bound based optimal algorithm proposed in paper [14], another one is the heuristic algorithm proposed in paper [15]. They are referred as BB_optimal and TS_heuristic in

Table 3. The simulation results on three QoS constraints

Networks	$C_0/C_1/C_2$	Fix_procedure	#destinations	run_time_ratio	solution quality
100-node random graph	320/320/320	BB_optimal	50	5.33%	100.0%
			40	5.02%	100.0%
			30	7.99%	100.0%
			20	9.17%	100.0%
			10	15.41%	100.0%
			Average	8.58%	100.0%
		TS_heuristic	50	16.07%	98.09%
			40	26.37%	97.74%
			30	16.00%	98.91%
			20	21.29%	99.14%
			10	31.91%	99.32%
			Average	22.32%	98.64%
8x8 mesh	560/560/560	BB_optimal	32	22.91%	100.0%
			25	19.84%	100.0%
			19	27.79%	100.0%
			12	16.71%	100.0%
			6	18.98%	100.0%
			Average	19.84%	100.0%
		TS_heuristic	32	3.70%	96.26%
			25	3.96%	95.59%
			19	3.93%	97.73%
			12	4.07%	98.11%
			6	7.21%	98.98%
			Average	4.57%	97.33%

Table 4. The simulation results for random graphs without fix procedures

∗ The number of nodes is 100. At most 3 QoS constraints: C_i.

$C_0/C_1/C_2$	#destinations	run_time_ratio	solution quality
320/320/-	50	2.0%	96.2%
	40	2.5%	96.0%
	30	2.7%	96.2%
	20	4.0%	98.0%
	10	7.9%	99.2%
	Average	3.8%	97.1%
320/320/320	50	3.7%	88.6%
	40	4.3%	92.5%
	30	5.4%	94.2%
	20	8.2%	93.3%
	10	15.4%	97.2%
	Average	7.4%	93.2%

Table 5. The simulation results for 8x8 meshes without fix procedures

$*$ The number of nodes is 64. At most 3 QoS constraints: C_i.

$C_0/C_1/C_2$	#destinations	run_time_ratio	solution quality
560/560/-	32	0.74%	90.1%
	25	0.44%	93.4%
	19	0.82%	93.3%
	12	0.76%	95.3%
	6	1.11%	97.8%
	Average	0.78%	94.0%
560/560/560	32	0.92%	71.5%
	25	0.82%	76.9%
	19	1.14%	81.9%
	12	1.74%	86.4%
	6	2.24%	93.3%
	Average	1.37%	82.0%

Table 1 and 2 respectively. For performance comparisons, several terms shown in Table 1 $\sim$ 5 are defined as follows:

run_time_ratio= CPU time required by the MCMTS procedure in 1000 runs / CPU time required by the optimal procedure in 1000 runs.

solution quality= the number of feasible trees found by the MCMTS procedure in 1000 runs / the number of feasible trees found by the optimal procedure in 1000 runs.

#tree_tabu= the number of feasible trees found by the MCMTS algorithm.

#tree_optimal= the number of feasible trees found by the optimal algorithm. In Table 1, the *run_time_ratio* is around 5% on average when the BB_optimal and TS_heuristic are used as the fix procedures. While in Table 2, the *run_time_ratio* is 12.9% when the BB_optimal is used as the fix procedure, and it is only 1.4% when the TS_heuristic is used as the fix procedure. These data show that the number of destinations, whose multicast paths cannot be determined by our MCMTS procedure and are then found by fix procedures, is very small. As a result, our heuristic can perform very efficient. For example, in Table 1, it takes around 0.0074 (7.39/1000) seconds to solve a 100-node random network in which 50% of nodes are destinations.

We also notice that in Table 2 the executing speed is greatly improved when the fix procedure is implemented with the TS_heuristic instead of the BB_optimal. This is because that the average number of hops between any two nodes in an 8x8 mesh is large enough to slow down the executing speed of the BB_optimal, since its time complexity is $O(d^h)$ where h is the number of hops between a pair of source and destination and d is the maximum degree of nodes in the network. As for the performance, the solution quality of our MCMTS procedure with a fix procedure based on TS_heuristic is more than 99% on average in Table 1 and 2. That is, compared to the optimal algorithm, the probability of finding a feasible multicast tree by our heuristic procedure is very high. As shown in Table 3, when three QoS constraints are considered, the solution quality of our MCMTS procedure is slightly decreased to around 98.6% for random graphs and 97.3% for meshes respectively.

4.2 Without Fix Procedures

In this section, we have two sets of simulations to show the performance of our MCMTS procedure when it is executed without implementing any fix procedure. For random graphs in Table 4, the average solution quality is around 97.1% when two QoS constraints are required. However, it decreases to 93.2% when three QoS constraints are required. The same phenomenon is also hold for meshes in Table 5. For the same values of QoS constraints, a network becomes tight when the number of QoS constraints increases. Obviously, for tight networks, it is not easy for our MCMTS algorithm with no fix procedure to find an acceptable path for replacement. As a result, its performance degrades. Although the probability of finding a feasible multicast tree decreases, this heuristic method is still very efficient, and is a practical approach for developing routing protocols, especially when the executing speed is the major concern for the underlying network.

5 Conclusions

In this paper, we have presented a heuristic algorithm called MCMTS for the MCMT problem. The experimental results show that the probability of finding a feasible multicast tree for a given network based on our heuristic is more than 99% if one exists. Furthermore, the MCMTS is also shown to be a simple and highly efficient method. As a result, it would be a practical approach for developing multicast routing protocol.

Acknowledgments. This work was supported by the National Science Council, Taiwan, R.O.C., NSC-91-2213-E-251-001.

References

1. Shigang Chen, Klara Nahrstedt, "On Finding Multi-constrained Paths," The Proceedings of the ICC'98 Conference, IEEE, pp. 874–979, 1998.
2. F. Glover, M. Laguna, Tabu Search, Kluwer Academic Publishers, 1997.
3. J.M. Jaffe, "Algorithms for finding paths with multiple constraints," Networks, vol. 14, pp. 95–116, 1984.
4. Turgay Korkmaz, Marwan Krunz, "A Randomized Algorithm for Finding a Path Subject to Multiple QoS Constraints," The Proceedings of the IEEE Global Telecommunications Conference, IEEE, pp. 1694–1698, 1999.
5. Turgay Korkmaz, Marwan Krunz, Spyros Tragoudas, "An efficient algorithm for finding a path subject to two additive constraints," Computer Communications, vol. 25, pp. 225–238, 2002.

6. Fernando Kuipers, Piet Van Mieghem, "MAMCRA: a constrained-based multicast routing algorithm," Computer Communications, vol. 25, pp. 802–811, 2002.
7. Hussein Salama, "Multicast Routing for Real-Time Communication on High-Speed Networks," Ph.D. dissertation, Dept. of Electrical and Computer Engineering, North Carolina State University, 1996.
8. H.F. Salama, D.S. Reeves, and Y. Viniotis, "Evaluation of Multicast Routing Algorithms for Real-Time Communication on High-Speed Networks," IEEE Journal on Selected Areas in Communications, 15(3), pp. 332–345, April 1997.
9. G.N. Rouskas, I. Baldine, "Multicast routing with end-to-end delay and delay variation constraints," IEEE Journal on Selected Areas in Communications, 15(3), pp. 346–356, April 1997.
10. S. Verma, R.K. Pankaj, A. Leon-Garica, "QoS based multicast routing algorithms for real time applications," Performance Evaluation, vol. 34, pp. 273–294, 1998.
11. Z. Wang, J. Crowcroft, "Quality-of-service routing for supporting multimedia applications," IEEE Journal on Selected Areas in Communications, 14(7), pp. 1228–1234, 1996.
12. B.M. Waxman, "Routing of multicast connections," IEEE Journal on Selected Areas in Communications, 6(9), pp. 1617–1622, 1988.
13. Jan-Jiin Wu, Ren-Hung Hwang, Hseueh-I Lu, "Multicast routing with multiple QoS constraints in ATM networks," Information Sciences, vol. 124, pp. 29–57, 2000.
14. Wen-Lin Yang, "Solving the MCOP Problem Optimally," The Proceedings of the 27th Annual IEEE Conference on Local Computer Networks (LCN 2002), IEEE, pp. 100–109, 2002.
15. Wen-Lin Yang, "Exact and Heuristic Algorithms for Multi-Constrained Path Selection Problem," Advances in Multimedia Information Processing-PCM2002, Lecture Notes in Computer Science, Vol. 2532, Springer-Verlag, pp. 952–959, 2002.
16. Xin Yuan, Xingming Liu, "Heuristic Algorithms for Multi-Constrained Quality of Service Routing," The Proceedings of the IEEE INFOCOM Conference, IEEE, pp. 844–853, 2001.

The Performance of PoolODMRP Protocol

ShaoBin Cai and XiaoZong Yang

Harbin Institute of Technology, Department of Computer, Mail-box 320 Harbin Institute of Technology, Harbin, 150001
csb@ftcl.hit.edu.cn

Abstract. The PoolODMRP is proposed as an extension of PatchODMRP[1]. The PoolODMRP reduce the local recovery scope of PatchODMRP to one-hop with help of pool nodes. In the PoolODMRP, the nodes, which are not forwarding nodes, however, can receive data packet, are defined as pool nodes to collect route information from their received data. When a forwarding node detects that there is a link failure between it and one of its upstream nodes, it sends out a Local Query. After receiving Local Query, a neighbor pool node checks whether it connects to the sources defined in the Local Query packet by the route information collected form data packets. If it does, the neighbor answers with a Local Reply. The repairing forwarding node collects the information in its received Local Reply, and informs the oldest answered neighbor pool node to be a forwarding node. In this way, one-hop local recovery is realized in PoolODMRP. The simulation results show that the Pool has a similar data deliver ratio with that of ODMRP [2] and PatchODMRP; its local recovery control overhead is 19.2% of that of PatchODMRP; its control overhead is only 41.2% of that of PatchODMRP, and 23.6% of that of ODMRP; its data overhead is only 64.7% of that of PatchODMP, and 79.7% of that of ODMRP.

1 Introduction

The multicast protocols are the inevitable result of the applications of the Mobile Ad Hoc Network, which can be rapidly deployed without any fixed infrastructure in the hostile environment [3][4][5]. The typical application areas of the ad hoc network, which include battlefields, emergency search and rescue site, require lots of one-to-many and many-to-many communications. Compared with multiple unicasts, the multicast makes full use of the inherent broadcast property of wireless communication, and minimizes the link bandwidth consumption, sender and router processing, and data delivery delay [6].
The already proposed multicast protocols [1][2][7][8][9][10][11] for the ad hoc network can be classified into two categories: tree-based protocols and mesh-based protocols. In the tree-based schemes, a single shortest path between a source and a destination is selected out for the data delivery. MAODV (Multicast Ad-hoc On-Demand Distance Vector), AMRoute (Ad hoc Multicast Routing)[7] and AMRIS (Ad hoc Multicast Routing protocol utilizing Increasing idnumberS)[8] are typical tree-based schemes. In a mesh-based scheme, multiple paths are selected for the data delivery. ODMRP (On Demand Multicast Routing Protocol)[2][9][10], PatchODMRP

[1] and CAMP (Core-Assisted Mesh Protocol)[11] are typical mesh-based schemes. In these protocols, tree-based protocols are generally more efficient than mesh-based protocols, but they are not as robust against topology changes as the mesh-based schemes. The recent studies show that the mesh-based schemes generally outperform the tree-based schemes, and they also show that ODMRP outperforms CAMP both in its protocol efficiency and data delivery ratio [12]. And, the further study shows that PatchODMRP outperforms ODMRP by its two-hop or three-hop local recovery [1]. If we can reduce the scope of local recovery of PatcchODMRP further, then we can reduce its control overhead, and improve its performance greatly.

In the PoolODMRP protocol, the Pool nodes, which are not forwarding nodes, however, can receive data packet, collect route information from their received data packets to provide route information for local recovery. With the help of pool nodes, the PoolODMRP protocol realizes its one-hop local route recovery.

The rest of the paper is organized as follow. First, we give an overview of the ODMRP in section 2. Secondly, we introduce the PatchODMRP in section 3. Thirdly, we describe the PoolODMRP protocol in section 4. And then, we present the simulation results in section 5. Finally, we draw a conclusion in section 6.

2 Introduction of ODMRP Protocol

In ODMRP, multicast routes are established and updated by the source on demand. When a multicast source has data packets to send without knowing the routes, it periodically floods a Join Query to the entire network to refresh the membership information and update the routes. When a node receives a new Join Query packet, it firstly stores the source address and the unique identifier of the packet in its Message Cache to detect duplicate; it secondly inserts or updates its Routing Table with the upstream node address in the packet as the next node for the source node; it last decreases the TLL value of the packet by one, and rebroadcasts the packet updated by its address as upstream address when the TTL value is still greater than zero.

A multicast member creates and broadcasts a Join Reply to its neighbors after receiving Join Query packets. After receiving a Join Reply, a node checks whether one of the next address of the received Join Reply matches its own address. If it does, the node marks itself as a forwarding node, and then broadcasts its own Join Reply packet built upon its Routing Table. By the method described above, the Join Reply is propagated by each forwarding node until it reaches the multicast source via the selected path, and the routes from the source to its receivers are constructed or updated by the procedure.

After the forwarding meshes have been founded, the sources broadcast their data to their neighbors. If a node is a forwarding node, then it rebroadcasts its received unduplicated data packet, else it discards its received data packet. By the relay of the forwarding nodes, the data reach all receivers.

In the ODMRP protocol, the group membership information is maintained by soft states. There are lists of the sources of the multicast groups in the members of the multicast groups. The members refresh or update the list by their received Join Query packets. If a node doesn't receive a Join Query packet from a source for a certain amount of time, it deletes the source from its source list. So, when a node wants to join the multicast group as a source, it sends Join Query packets periodically. When a node wants to join the group as a member, it replies the Join Query packet that it received with Join Reply. When a node wants to leave the group, it simply stops sending Join Query or Join Reply.

3 Overview of PatchODMRP Protocol

PatchODMRP extends ODMRP by two-hop or three-hop local recovery, and prolongs the network-wide flooding period to reduce the control overhead of route maintenance. It sets up its forwarding mesh as ODMRP does. But, in the PatchODMRP, each forwarding node uses forwarding table to store the route information from Join Reply. Therefore, it knows which nodes are its upstream nodes, for which sources it relay data, and the distance between it and the sources.

In the PatchODMRP, each forwarding node detects the statuses of its neighbors by the BEACON of MAC layer [13] to know whether the link between it and one of its upstream nodes is broken. If the link is broken, then the forwarding node floods ADVT packet to do local recovery. The ADVT includes which node sends out the ADVT packet, includes how many times the ADVT packet can be relied, includes which groups the forwarding node belongs to, includes which sources it relays data for, and includes the distances between it and the sources.

When a node receives an ADVT packet, it first records the route information gotten from the ADVT packet. And then, it checks whether it belongs to the same groups that the repairing forwarding node belongs to, checks whether it relay data for the same sources the repairing forwarding node relay for, and checks whether it is nearer to the sources than the repairing forwarding node does. If all these conditions meet, the node answers the ADVT with a PATCH packet. Otherwise, it decreases the Hop-count of ADVT by 1, and relays the packet when its Hop-count is greater than 0. The PATCH arrives at the repairing forwarding node by the reverse way of ADVT transmission, and marks the nodes on the way temporary forwarding nodes. After receiving the Patch packets, the repairing forwarding node selects the shortest patch, and informs these temp forwarding nodes of the result. The nodes off the shortest path aren't temp forwarding again.

Fig 1 describes how the PatchODMRP works. The multicast mesh is made up of three fractions (Fig1A): a source (A), a receiver (E), and five forwarding nodes (B, D, F, G, I) marked gray. Node B is the upstream node of node D. When node D can't exchange BEACON with node B, it floods out its ADVT packet. After receiving the ADVT packet, its neighbors (C, H, E) update their Route Tables, and relay the ADVT packet (Fig1B). When the forwarding nodes (B, F), which meet the three conditions above, receive an ADVT packet, they answer with a PATCH packet. The nodes C and H are marked as temp forwarding nodes by the PATCH packet (Fig1C). After receiving PATCH packets, the repairing forwarding node D selects the shortest path, and informs all these answered forwarding nodes the result, and the node H, that doesn't on the shortest path, isn't a temporary forwarding node again (Fig1D).

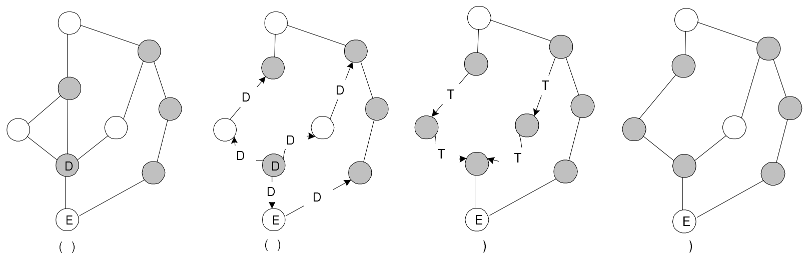


Fig. 1. An example of PatchODMRP

4 The PoolODMRP Protocol

In the PatchODMRP, local recovery is performed by two-hop or three-hop scope local flood [1]. If there is a wanted forwarding node two-hop away, the local repair can be completed by marking the node, which is one-hop away from the repairing node, a temp forwarding node. If a node knows from its received packets not only which sources it connects to but also the distance between it and the sources, then we can finish the local repair in one-hop instead of two-hop. And the control overhead of local recovery can be reduced greatly.

In order to realize the one-hop local repair in the PoolODMRP, the pool nodes, which are not forwarding nodes, however, can receive data packets during the data transmission, are defined to collect route information from its received data. Three kinds of new data structure are defined to record the route information that is used for local recovery: Upstream Table (Fig2A), Pool Table (Fig2B) and Candidate Table (Fig2C). And, three new kinds control packets are defined to perform local recovery: Local Query packet (Fig2D), Local Reply packet (Fig2E), and Local Notify packet (Fig2F).

Firstly, we introduce the Upstream Table, which exists in the forwarding node, and the Pool Table, which exists in the Pool node. Both of them are updated by data packets, and are used to record the route information gotten from the data packets. Their Upstream Node field records the address of the node from which the node received data packet; their MG ID field presents which group the data packet belongs to; their Source field presents what nodes originate the data packet; their Count field presents the shortest distance between the source node and the node; their Timer field presents when the data packet is received. The Birth field of the Pool Table presents when the entry is inserted in the table. Each subentry of these tables has a lifetime. When a subentry can't be updated by data packet in time, it expired, and is deleted. When all subentries of an entry are deleted, the entry is deleted too.

Secondly, we introduce the Candidate Table, which temporarily exists in the local repairing forwarding node. When the forwarding node, which is doing local repair, sends out its Local Query, it creates its Candidate Table that exists for a period time. The field of Candidate Table is from the corresponding field of Local Reply packets. When the repairing node receives a Local Reply from one of its neighbor pool nodes, it inserts the pool node address, the list of the MG ID, and the birth time of the local Reply packet in the corresponding fields of its Candidate Table.

Upstream Node	MG ID	Source	Count	Timer
		Source	Count	Timer
	MG ID	Source	Count	Timer
		Source	Count	Timer

Fig. 2A. Upstream Table

Upstream Node	Birth	MG ID	Source	Count	Timer
			Source	Count	Timer
		MG ID	Source	Count	Timer
			Source	Count	Timer

Fig. 2B. Pool Table

Pool node	List of MG ID	Birth	Lifetime

Fig. 2C. Candidate Table

Local Query	List of {MG ID, Source, Count}	LQ Address

Fig. 2D. Local Query

Local Reply	List of MG ID	LQ Address	Birth	Pool Address

Fig.2E. Local Reply

Local Notify	List of MG ID	LN Address

Fig. 2F. Local Notify

Thirdly, we introduce the Local Query packet, Local Reply packet and Local Notify packet. Their LQ Address is the address of the repairing forwarding node. The List of {MG ID, Source, Count} of Local Query packet presents which group the repairing forwarding node belongs to, for which sources the repairing forwarding nodes relay data, and the distance between these sources and the repairing forwarding node. The List of MG ID of both Local Reply and Local Notify present which groups the repairing forwarding node belongs to. The Pool Address of the Local Reply is the address of the pool node that creates the Local Reply. The Birth field of the Local Reply presents the time when the pool node has connections with these required groups. The LN Address of Local Notify is the address of a pool node that will change to a forwarding node.

The PoolODMRP setups up its forwarding mesh by the same way as the ODMRP does too. After setting up its forwarding mesh by the network-wide flooding, the source begins to send out its data packets. When a node receives a data packet, it inserts or updates its Upstream Table or Pool Table by the arithmetic 1 (Appendix 1). If the node is a forwarding node and the data is unduplicated data, then the node relays the packet, else the node discards the packet. So, the nodes collect information used for one-hop local recovery during the data transmission.

In the PoolODMRP, the BEACON signal of the MAC layer is used by a forwarding node to check which nodes are its neighbors. When it finds an upstream node of its Upstream Table isn't reachable, it determines whether it can connect to all the lost sources by other upstream nodes. If it can't, then it forms its Local Query according to the corresponding entry, and sends out the Local Query packet. When one of its neighbors receives a Local Query packet, it firstly deletes the entry, which matches the LQ Address of the Local Query, in its Pool Table; it secondly uses the arithmetic 2 (Appendix 2) to determine whether it creates and sends out a Local Reply to answer the Local Query. If it does, then the node connects to the sources wanted by the repairing forwarding node, and it is not farther to these sources than the repairing forwarding node does. Therefore, when it sets itself a forwarding node, there will be no loop in the new route.

When the repairing forwarding node receives a Local Reply, it inserts a new entry in the Candidate Table. After a period of time since its receiving its first Local Reply, the node selects the oldest pool node that has the earliest birth time, and sends out a Local Notify to inform the oldest pool node to be a forwarding node. When a pool node receives a Local Notify packet, it sets itself a forwarding node, and acknowledges the Local Notify packet.
Fig 3 describes how the PoolODMRP protocol works. The ad hoc mesh shown in picture 3A consists of 4 fractions: a source (A), a receiver (E), five forwarding nodes (B, D, F, G, I) marked gray, and three Pool nodes (C, H, E) marked gray grid. When the link between node B and node D breaks because of the movement of node B, the node D can't exchange BEACON with node B. The node D sends out a Local Query (fig.3B). When its neighbors (C, H, E) receive the Local Query, they check whether they have at least one wanted upstream node. The pool nodes (C, H), which can receive data from node A, send out their Local Reply to answer the Local Query (fig.3C). After receiving the Local Reply, node D selects the node C out as the oldest pool node and sends out its Local Notify (fig.3D). The node C sets itself a forwarding node as soon as it receives the Local Notify, relays its received data packet, and acknowledges the Local Notify (fig.3E).

5 The Performance Analyses

The reason why we use one-hop local recovery is that we want to reduce the control overhead of local recovery. So, we first set up a mathematic model and compare the local control overhead of both PoolODMRP and PatchODMRP. Firstly we assume that a local repairing forwarding node averagely has L neighbors, M of which have connections with the wanted forwarding nodes, and has N wanted forwarding nodes two-hop away. When C_{pool} is used to represent the local recovery overhead of PoolODMRP, the cost of C_{pool} is defined in the equation (1). When C_{patch} is used to represent the two-hop local recovery overhead of PatchODMRP, the cost of is C_{patch} defined in (2). And, the huge difference between C_{patch} and C_{pool} can be seen in the (3).

C_{pool} =1 (one-hop flooding)+M (the number of pool nodes which answer the Join Query)+1 (Local Notify)+1 (ACK for Local Notify), (1)

C_{patch} =1 (one-hop flooding)+L (the number of nodes which relay the ADVT)+N (the number of forwarding which answer the ADVT)+M (the number of pool node which relay the PATCH)+(1+M) (control packet used for confirming the shortest path). (2)

$$C_{patch} - C_{pool} = L+N-2 \tag{3}$$

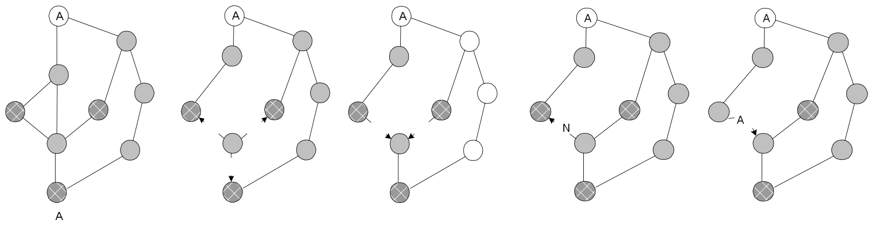


Fig. 3. An example of PoolODMRP

Except the smaller scope of local recovery, PoolODMRP is different from PatchODMRP in other two aspects: the lifetime of the forwarding nodes and the lifetime of the temp forwarding nodes. According to the current PatchODMRP speciation [1], the lifetime of forwarding nodes is set as 3 times as large as the length of the Join Query interval, and the lifetime of temp forwarding nodes is set as 1/3 times as large as the length of the Join Query interval. In the PoolODMRP, there aren't temp forwarding nodes, and the nodes founded in the local recovery are set normal forwarding nodes. The lifetime of forwarding nodes is set as 4/3 times as large as the length of the Join Query interval. The reasons why we reduce the lifetime of the forwarding node not only to reduce the data overhead, but also to reduce unnecessary local recoveries caused by the movement of the too old forwarding nodes. The shorter lifetime of the forwarding node results in not only the lower data overhead and the fewer local recoveries but also the lower data delivery ratio. In order to overcome the shortcomings of the shorter forwarding node lifetime, we use normal forwarding node instead of the temporary forwarding node in the local recovery, and select the most stable route, which is established by the oldest pool node, instead of shortest path in the local recovery of PatchODMRP.

5.1 Simulation Model

GloMoSim [14] is used here to realize the simulation of the PoolODMRP protocol. In the simulation, 50 wireless mobile nodes, which move around over a square (1000m$\times$1000 m), form an ad hoc network. The radio transmission power of the mobile nodes is 15dBm. There are two three-member multicast groups in the ad hoc network, one of which has one source, and the other of which has two sources. During the 500s simulation period, Nodes move according to the "random waypoint" model without pause time, and the multicast sources generate 512-byte data packets with constant bit rate (CBR) of ten packets per second. In order to evaluate the performance, we use three metrics here [15]:

Data Packet Delivery Ratio: The percentage of data packets correctly delivered to multicast receivers.

Number of Data Transmissions per Data Packet Delivered: This metric reveals the path efficiency.

Number of Control Packets per Data Packet Delivered: This metric represents control overhead of each data packet.

5.2 Numerical Results

The most important factor, which inflects the performance of PoolODMRP, is its Flood Period, which is the period between two floods. If the Flood Period is longer, the less flooding route discovery operations decrease control overhead, but, the more local unrecoverable link failures may decrease the data delivery ratio. So, we firstly test the impact of the Flood Period on the data delivery ratio and the local recovery ratio. Fig 4 shows the local recovery ratio of PoolODMRP against the changing nodes' max speed. The local recovery ratio is individually 94.9%, 97.5% and 91.4% when the value of the Flood Period is 12s, 18s and 24s. Fig 5 shows the average packet delivery ratio of PoolODMRP against the nodes' max speed. The average data delivery ratio is individually 93.5%, 95.2% and 90.4% when the value of the Flood Period is 12s, 18s and 24s.

According to the theory, the data delivery ratio of the protocol should decrease when the value of Flood Period increases or the nodes' max speed rise. But, the simulation results aren't consistent with the theory because of other factors that affect the data delivery ratio. One of them is the suitable prolonged Flood Period; the other is the suitable movement of the nodes. Because the longer Flood Period decreases the number of network-wide flood, there are less control packets, which contend the scare wireless bandwidth with the data packets. So, when the longer Flood Period doesn't create too many local unrecoverable link failures, it can help data delivery. To illustrate suitable movement benefiting data delivery ratio, let's consider a multicast group composed of nodes A, B and C. When the link between node A and node B breaks as the result of the movement of node B, the node C can't relay data for node B because of it can't move close enough to node A and B with the max speed being 5m/s. But, the node C can relay data for node B because of it can move close enough to node A and B with the max speed being 10m/s. Although, the increase of nodes' max speed decreases the data delivery ratio for most time, it may occasionally increase the data delivery ratio.

Form fig4 and fig5, we can see that both the local recovery ratio and data delivery ratio are highest when the value of the Flood Period is 18s. So, in the flowing experiments, we set the value of the Flood Period of both PatchODMRP and PoolODMRP 18s. And, we set the Flood Period of ODMRP 3s according to the specification of ODMRP [2].

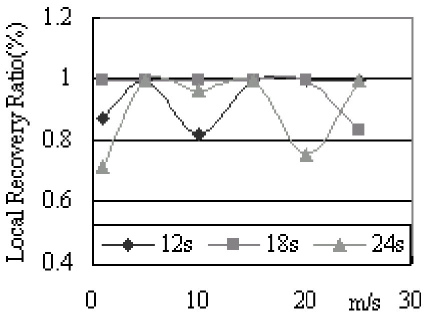


Fig. 4. The relationship between local recovery ratio and the nodes' max speed

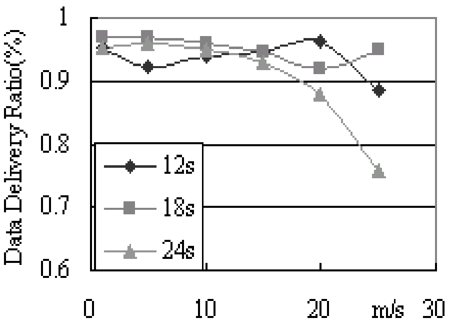
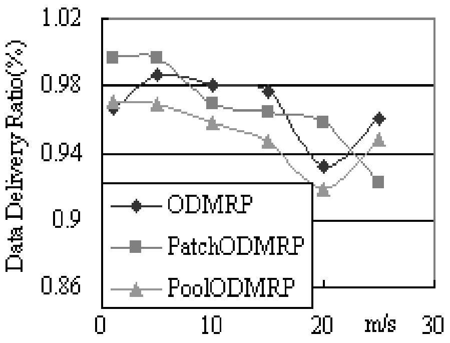


Fig. 5. The relationship between data delivery ratio and the nodes' max speed

Fig 6 shows the packet delivery ratio of the three protocols against the changing nodes' max speed. The average data delivery ratio of the ODMRP, PatchODMRP and PoolODMRP is respectively 97.1%, 96.7% and 95.2%. The data delivery ratio of PoolODMRP is similar to that of ODMRP and PatchODMRP. According to the definition of both ODMRP and PoolODMRP, the route of ODMRP is more frequently updated because of its lower flood frequency, and there are more forwarding nodes in the forwarding mesh of ODDMRP at most time because of its longer lifetime of its forwarding nodes. Therefore, if there is no local recovery in the PoolODMRP, it is clear that the data delivery ratio of PoolODMRP should be much lower than that of ODMRP. But, the local recovery of PoolODMRP efficiently maintains its forwarding mesh, and PoolODMRP gets a satisfied data delivery ratio. Compared with PatchODMRP, the shorter forwarding nodes' lifetime of PoolODMRP result in its lower data delivery ratio.

Fig 7 shows the average local control overhead of both PatchODMRP and PoolODMRP. The local control overhead is the cost created in the local route recovery procedure. Form the simulation results, we can see that the average local control overhead of both protocols is individually 0.267 and 0.045, and the local control overhead of PoolODMRP is only 19.2% of that of PatchODMRP. The smaller local recovery scope of PoolODMRP results in its less local control overhead.

Fig 8 shows the control overhead of the three protocols against the changing node's max speed. From the simulation results, we can see that the average control overhead of ODMRP, PatchODMRP, and PoolODMRP is respectively 0.928, 0.53 and 0.218. The control overhead of PoolODMRP is only is 23.4% of that of ODMRP, and 41.2% of that of PatchODMRP. The control overhead of PatchODMRP and PoolODMRP are both made up of the global control overhead, which are created by the network-wide flooding, and the local control overhead used for the local route recovery. PatchODMRP and PoolODMRP have the similar global control overhead according their definition. So, the lower control overhead of PoolODMRP mainly results from its lower local control overhead. Compared with ODMRP, the longer Flood Period of PatchODMRP and PoolhODMRP results in their lower control overhead.



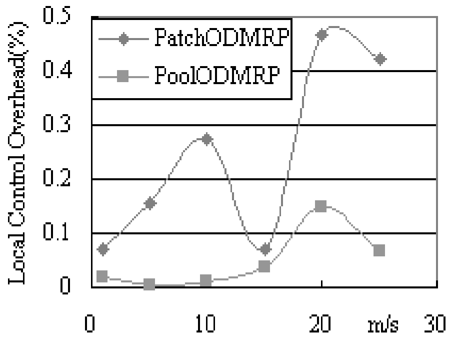


Fig. 6. The relationship between data delivery ratio and the nodes' max speed

Fig. 7. The relationship between local control overhead and the nodes' max speed

Fig9 shows the data overhead of the three protocols against the changing node's max speed. The average data overhead of ODMRP, PatchODMRP and PoolODMRP are respectively 2.153, 2.654 and 1.716. The data overhead of PoolODMRP is 79.7% of that of ODMRP, and is 64.7% of that of PatchODMRP. The ODMRP and PatchODMRP have the same lifetime of forwarding nodes, so they have the similar data overhead. Since there are repaired routes in the forwarding mesh of PatchODMRP, there are more forwarding nodes in the forwarding mesh of PatchODMRP than that in the forwarding mesh of ODMRP. Therefore, the PatchODMRP has higher data overhead. Compared with ODMRP and PatchODMRP, the forwarding node lifetime of PoolODMRP is shorter. The longer the lifetime of forwarding node, the more forwarding nodes take part in data delivery, the higher data overhead. So, the data overhead of PoolODMRP is lower than that of ODMRP and PatchODMRP.

6 Conclusion

In this paper, the PoolODMRP protocol is proposed as a new on-command ad hoc multicast protocol. In the PoolODMRP, the Pool node, which is not a forwarding node, however, can receive data from forwarding nodes, is defined to collect route information from its received data packet. Compared with PatchODMRP, it adopts pool nodes to reduce its local recovery scope, it adopts shorter forwarding node lifetime system to reduce the data overhead, and it adopts normal forwarding node lifetime system in the local recovery to improve its data delivery ratio. The simulation results show that the data delivery ratio of PoolODMRP is similar to that of both ODMRP and PatchODMRP. But, its local recovery control overhead is 19.2% of that of PatchODMRP; its control overhead is only 41.2% of that of PatchODMRP, and is 23.4% of that of PDMRP; its data overhead is only 64.7% of that of PatchODMP, and is 79.7% of that of ODMRP. Therefore, PoolODMRP outperforms both ODMRP and PatchODMRP both in control overhead and data overhead.

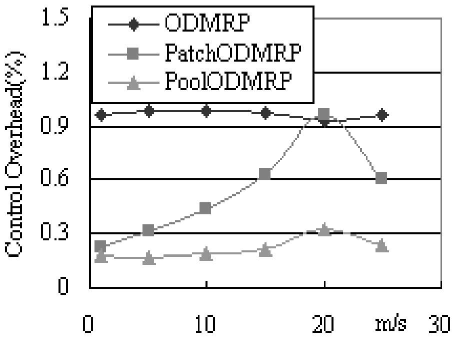


Fig. 8. The relationship between data delivery ratio and the nodes' max speed

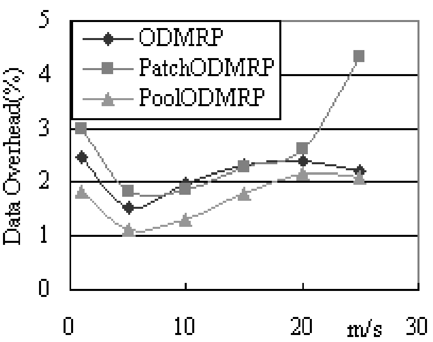


Fig. 9. The relationship between local control overhead and the nodes' max speed

References

1. *Meejeong Lee; Ye Kyung Kim*. "PatchODMRP: an ad-hoc multicast routing protocol" Information Networking, 2001. Proceedings. 15th International Conference on, 2001
2. Sung-ju. Lee, Willian Su, Mario Gerla, "On-Demand Multicast Routing Protocol (ODMRP) for Ad Hoc Networks", Internet Draft, draft-ietf-manetODMRP-02.txt, July 2000.
3. Mobile Ad-Hoc Network (MANET) Working Group [Online]. Available http://www.ietf.org/html.charters/manet-charter.html.
4. D. Bertsekas, R. Gallager, Data Network, second edition, Prentice-Hall, Englewood Cliffs, NJ, 1992, pp. 404–410.
5. C. Perkins, P. Bhagwat, Highly dynamic destination sequenced distance-vector routing (DSDV) for mobile computers, ACM SIGCOMM, October 1994.
6. Sanjoy Paul. "Multicasting on the Internet and its Applications", Kluwer Academic publishers, 1998.
7. E. Bommaiah, M. Liu, A. Mcauley, and R. Talpade, "AMRoute: Ad-hoc Multicast Routing Protocol", Internet-draft, draft-talpade-manet-amroute-00.txt, Aug, 1998.
8 C.W. Wu, Y.C.Tay, and C.-K.Toh, "Ad hoc Multicast Routing protocol utilizing Increasing id-numberS (AMRIS) Functional Specification", Internet-draft, draft-ietf-manet-amris-00.txt, Nov 1998.
9. Sung-ju Lee, Willian Su, Mario Gerla, "On-Demand Multicast Routing Protocol", In proceeding of IEEE WCNC'99. New Orleans, LA, Sep, 1999.
10. Sung-ju Lee, Willian Su, Mario Gerla, "Ad hoc Wireless Multicast with Mobility Prediction", In proceeding of IEEE WCNC'99. New Orleans, LA, Sep, 1999.
11. J.J. Garcia-Luna-Aceves, E. L. Mdrguga, " The Core-Assisted Mesh Protocol", IEEE Journal on selected Areas in Communications, vol.17, no. 8, Aug.1999.
12. Sung-ju Lee, Willian Su, Mario Gerla, and Rejive Bagrodia, "A Performance Comparison Study of Ad Hoc Wireless Multicast Protocols", In Proceeding of Inforcom' 2000, 2000.
13. IEEE Computer Society, LAN MAN Standards Committee. Wireless LAN Medium Access Protocol (MAC) and Physical Layer (PHY) Specification, IEEE Std 802.11-1997 IEEE, New York, 1997.
14. Wireless Adaptive Mobility Lab. DEPT of Comp. SCI, UCLA " GloMoSim: A Scalable Simulation Environment for wireless and wired Network System:, http://pcl.cs.ucla.edu/projects/domains/glomosim.html.
15. S. Corson and J. Macker. "Mobile Ad Hoc Networking (MANET); Routing Protocol Performance Issues and Evaluation Considerations", RFC 2501, IETF, Jan. 1999, Available at http://www.ietf.org/rfc/rfc2501.txt

Appendix 1 The Arithmetic 1 for Inserting or Updating Upstream Table and Pool Table

```
If there is a entry matches last-address of the data packet
  {If there is a subentry matches MG ID of data packet
            {If there is a subentry matches the source of data packet
             {If the Count of the subentry is larger than the Count of data packet
             {Replace the Count field of the subentry with the Count of the data
                      packet;}
              Update the received time of the data;}
             Else {inserts a subentry to recorder the Source, Count, and the received
```

```
                time of the data packet;}}
            Else {inserts a new subentry to record the MG ID, Source, Count, and the
                received time of the data packet;}}
    Else {insert a new entry to record the Upstream Node, Birth (only for Pool Table),
            MG ID, Source, Count, and the received time of the data packet;}
```

Appendix 2 The Arithmetic 2 for Finding a Matched Upstream Node in the Pool Table

```
For each entry of the table
  {If the Upstream Node of the entry doesn't match the LQ address of Local Query
        {For each {MG ID, Source, Count} of the list of {MG ID, Source, Count}
            {If the MG ID of the {MG ID, Source, Count} can find a matched
                subentry in the entry
            {If the Source of the {MG ID, Source, Count} can find a matched
                secondary subentry in the subentry
              {If the Count of the {MG ID, Source, Count} is larger than
                    that of secondary subentry
              {Mark that the {MG ID, Source, Count} is satisfied;}}}}
        If all {MG ID, Source, Count} of the list of {MG ID, Source, Count} is
              marked satisfied
        {Create the Local Reply, whose Birth is filled with the Birth of the entry;
        Answer the Local Query with Local Reply};
        Break;
        }
    }
```

M3G: A Mobile Multicast Multimedia Gateway for Seamless IPv4/IPv6 Transition

Yassine Hadjadj Aoul, Daniel Negru, Abdelhamid Nafaa, and Ahmed Mehaoua

CNRS-PRiSM Lab., University of Versailles
45, av. des Etats Unis 78035 – Versailles - France
Tel: +33 1 39 25 40 56
Fax: +33 1 39 25 40 57
{yana, dan, anaf, mea}@prism.uvsq.fr

Abstract. A growing interest in third generation wireless IP network and service technologies, push up the demand on IPv6 transition. Most of these services require mobility, multicast and multimedia service supports. However, the transition to such a NG IP networks and services is slowed down because of the richness and complexity of the applications and the integration of legacy systems. Existing IETF and academic proposals permit to solve several transition problems but they don't cope with the multimedia or wireless aspects of some next generation IP services. In the other hand, most transition solutions are limited to LANs and couldn't scale well. Therefore, this article describes the architecture and the implementation of an integrated multi-functional IP gateway for IPv4/IPv6 network transition with multimedia, multicast and mobility service management. This gateway, named M3G, has been partially implemented and validated on a Linux-based testbed with MPEG4 video streaming applications. Nevertheless, the main goal of this work is to achieve a transparent establishment and a smooth control of unicast (e.g. VOD) and multicast (e.g. TV broadcasting) multimedia communications between heterogeneous fixed and mobile hosts belonging to dense IPv4 and IPv6 domains.

1 Introduction

The transition from the former version of IP to its new shape is certainly not the least task to consider. Even though IPv6 has a strong base and is easily extensible, the migration is facing difficulties. Therefore, there is a critical need to deploy a solution that would allow IPv4 existing services to be accessed by IPv6 terminals.

The solution proposed in here takes into consideration this aspect. Not only is the heterogeneous problem taken into account at all levels, but also many important features of IPv6 are exploited, from mobility to multicast, including multimedia and scalability. The fruition would be a Mobile Multicast IPv4/IPv6 Gateway for Multimedia Applications working in a heterogeneous environment. Its main role is to manage connectivity between a native IPv4 domain and a native IPv6 one, all in respect to multimedia applications constraints.

Existing proposals address the unicast case. In addition, MTP [6] provides an multicast IP translation but present several limitations. Ours provides an IETF fully compliant solution through (1) an IPv4/IPv6 addressing mapping, (2) an RTSP protocol for real-time streaming, (3) a SAP/SDP [2][3][4] protocols for transparent media access, (4) a scalable multicast sessions managements as well as (5) a support for mobility.

First, we will present the context and motivations. Next, we will focus on our proposal and especially on the multicast transition, multimedia, mobility and scalability parts. Then, we will develop the structural aspect of our gateway, from the architectural to the implementation point of view. Finally, we include a short analysis of our gateway's performances.

2 Context and Motivations

There is a growing demand on multicast media services due to the will of bandwidth saving. This particularity is very appreciable, especially in the wireless networks. Those will need IPv6 technologies for many reasons the most important being the lack of addresses. Also, most of applications communicate only with IPv4 hosts and can't communicate with IPv6 hosts. So, there is a critical need to deploy a transitory solution in order to allow an IPv4 existing service to be accessed by new IPv6 terminals.

2.1 Transition to IPv6 and Multimedia Context

[11] [12] [13] IPv6 is an important part of the future infrastructure of the Internet, as it offers improvements over IPv4. The current shortage of IPv4 addresses presents a large problem for the Wireless industry, which has recently offered Wireless Internet access requiring new demands for IP addresses. Therefore, IPv6 has become the target protocol for the forecoming wireless networks and infrastructures.

The key transition objective is to allow IPv6 and IPv4 hosts to interoperate [14]. A second objective is to allow IPv6 hosts and routers to be deployed in the Internet in a highly diffused and incremental fashion, with few interdependencies. A third objective is that the transition should be as easy as possible for end-users, system administrators, and network operators to understand and carry out.

Concerning multimedia applications, for now, the multicast solution has not been fully adopted in an IPv4 context at a network layer level. On the contrary, IPv6 has been designed with the support of multicast.

The deployment of coherent multimedia multicasting services over IP networks involves frequently some out of band signaling protocols behind the RTP [10] media streams. These protocols (e.g. SAP/SDP, RTSP and others) provide essentially an abstraction to initiation stage and transparent access to media through negotiation, configuration and synchronization.

2.2 Mobile IPv6 and Multicast Issues

Mobility is an integrated part of IPv6; it has been designed to make full support for wireless mobility. The main problem, though, is in the combination of multicast and mobility at a terminal level first, but also at a network one afterwards, for multimedia applications, all this according to QoS characteristics and security constraints.

Providing these services are difficult due to (1) frequent changes of mobile host location and group membership and (2) the stringent QoS requirements of multimedia applications. If a conventional multicast routing protocol is used in wireless mobile networks, several problems may be experienced since existing multicast routing protocols assume static hosts when they construct the multicast delivery tree. To overcome the difficulties, several multicast routing protocols for mobile hosts have been proposed. Although the protocols solve several problems inherent to multicast routing proposals for static hosts, they still have problems such as non-optimal delivery path, tunnel convergence problem, datagram duplication, overheads resulting from frequent reconstruction of a multicast tree, scoping problem, etc....

The current IETF mobile-IP specification proposes two approaches for supporting multicast service to mobile hosts [10]: foreign agent-based multicast (referred to as remote-subscription) and home agent-based multicast (referred to as bi-directional tunneling) [17].

In foreign agent-based multicast, a mobile host has to subscribe to multicast groups whenever it moves to a foreign network. It is very simple scheme and does not require any encapsulations. This scheme has the advantages of offering an optimal routing path and non-existence of duplicate copies of datagrams. However, when mobile host is highly mobile, its multicast service may be very expensive because of the difficulty in managing the multicast tree. Furthermore, the extra delay incurred from rebuilding a multicast tree can create the possibility of a disruption in multicast data delivery.

In home agent-based multicast, data delivery is achieved by unicast mobile IP tunneling via home agent. When a home agent receives a multicast datagram destined for a mobile host, it encapsulates the datagram twice (with the mobile host address and the care-of address of the mobile host) and then transmits the datagram to the mobile host as a unicast datagram. This scheme takes advantage of its interoperability with existing networks and its transparency to foreign networks that a mobile host visits. However, the multiple encapsulation increases the packet size, and the datagram delivery path is non-optimal since each delivery route must pass through a home agent. Furthermore, if multiple mobile hosts that belong to the same home network visit the same foreign network, duplicate copies of multicast datagrams will arrive at the foreign network.

3 M3G: Proposal of a Mobile Multicast Media Gateway for IPv4/IPv6 Transition

Our proposal of a Mobile Multicast Media Gateway is based on four principal objectives. First, enabling the transition between an IPv4 domain and an IPv6 one, as well from the unicast as from the multicast point of view. Next, accessing multimedia applications with the support of multicast and signaling processes. Then, presenting the mobility aspects and the interactions with multicasting and multimedia constraints. Finally, developing the important issue of scalability, especially for multicast session management.

3.1 IPv4/IPv6 Translation

The first objective of the M3G is to enable IPv4/IPv6 transition between heterogeneous networks, and, above all, to support multicast streams. This task is realized through different points. As a first step, an addressing mapping is obligatory. Afterwards, a solution for multicast signaling translation is proposed in order to avoid the manual setup of the gateway present in the MTP [6] proposal.

Address mapping. The gateway consists of a physical entity connected to two different subnets and implementing a double stack IPv4/IPv6. Its main role is to translate and route packets between a native IPv4 domain and a native IPv6 one.

Our translation is based on the NAT-PT [5] proposal for unicast communications. Unlike SIIT [1], which particularity is to use an "IPv4 mapped address", the unicast mapping used by our NAT-PT solution adds a 96-bits prefix for all communications before the unicast IPv4 address. Owing to this prefix, the packets will be routed through the M3G gateway. This implies the recalculation of the checksum.

The classical mapping technique seen above can not be applied to multicast addresses. So we use a dedicated one to that case, based on the proposition of MTP [6] and named "IPv4-compatible" IPv6 multicast group address.

IGMP/MLD signaling. In order to join IPv4 group by IPv6 hosts, we need interaction between the two domains. The extension is done through a simple mechanism that won't drop IGMP [**8**] packets sent with a TTL=1 but will translate them to MLD [**9**] packets also with a TTL=1. The translation is based on the shape of the packet since IGMP and MLD have the same functionalities.

Translating IGMPv2 packet to MLD. IGMPv2 fields are translated to MLD fields as follows:
– Type: Takes the value of the IGMP Type.
– Code: This field doesn't exist in IGMP packet. However, it must be initialized to zero by the sender.
– Checksum: Must be recalculated by the gateway.
Maximum Response Delay: Takes the value of the IGMP Maximum Response Time.
– Reserved: This field doesn't exist in IGMP header. It's a reserved field and must be initialized to zero by the sender.
– Multicast Address: The value of this field represents the "IPv4-compatible" IPv6 multicast group address of the IGMP Group Address field.

Translating MLD packet to IGMPv2. MLD fields are translated to IGMPv2 fields as follows:
– Type: Takes the value of the MLD Type.

– Maximum Response Time: Takes the value of the Maximum Response Delay MLD field if it is lower than 0xff, it is set to 0xff otherwise.
– Checksum: Must be recalculated by the gateway.
– Group address: Is extracted from the IPv6 address if it's a "IPv4-compatible" IPv6 multicast group address of the MLD Multicast Address field, we must have a pool of IPv4 multicast addresses in our gateway otherwise.

The other MLD fields are ignored by the gateway.

We explain below in a more detailed way our solution for the communication from IPv4 multicast media server to IPv6-only mobile hosts. (See section (4.2): Implementation issues).

3.2 Multimedia Support

Industrials actors are interested in multimedia support for their products. However, the classical approaches consider principally the transition problems and avoid multimedia aspects. In this way, we integrate the multimedia dimension in our transition proposal, which treats principally the constraints linked to multicast multimedia applications. So, we first discuss in this section about multimedia services access problems and next, about the efficient multimedia support.

Media Services Access. The primary objective of our gateway design, once an environment for multimedia applications is set up, is to propose solutions for the deployment of heterogeneous multimedia applications, going from real-time to streaming, videoconferencing, video surveillance, radio broadcasting, etc… all of these according to applications exigencies and QoS requirements.

In order to allow a transparent access to multimedia streams, M3G supports the commonly used signaling protocols (e.g. RTSP, SAP, and SDP). Moreover, the M3G architecture is enough flexible to enable other application-level protocols intended to support incoming multimedia services. It should be noted that the majority of multimedia applications induce the use of signaling protocols for session initiation and description. Basically, the signaling protocols provide an abstraction to session parameters and automate the access to multicast groups, especially for SAP/SDP.
Currently, M3G supports SAP/SDP and RTSP protocols for both multicast and unicast multimedia streaming scenarios. The translation of their associated packets is achieved through the tracking of their standardized port numbers at the Translator module (see Fig. 3).

Efficient Multimedia Support. For a more efficient support of multicast multimedia sessions and for scalabilities reasons (see section 4.4), we add in M3G the support for MLD and IGMP protocol. This will permit routers to discover the presence of multicast listeners and then to send packets only if there is at least one listener. Thus, M3G does the translation only if it's necessary. Thereby, the active multimedia sessions won't be slowed down by useless packets.

Also, we pass through heterogeneous domains' limitations by creating an interaction between MLD and IGMP protocols (see sections 3.1 and 4.2 for more details). Hence, we permit the creation of multicast multimedia sessions between heterogeneous domains.

3.3 Mobility Support

The M3G has been developed with mobility support in mind. One of the primary objectives was to integrate wireless applications, starting at a LAN level and then extending to subnets. No major difficulties are encountered on the wireless LAN part, though. The expansion to several subnets and the apparition of mobility aspects, such as a mobile node willing to receive multicast sessions' flows while moving from a place to another, represents one of the most innovative aspects, especially on the convergence of mobility and multicast.

Mobility Purposes. The increasing demand for mobility in the Internet has created the need for a routing protocol that allows a host to roam in the network. Mobile IP is a solution that enables an IP–host to leave its home link while transparently maintaining all of its present connections and remaining reachable to the rest of the Internet. Mobile IPv4 has been standardized by the IETF and Mobile IPv6 is currently an Internet draft.

An IP address identifies the link on which the host resides. If a host moves to a different link without changing its IP address, there is no information in its IP address about the new point of attachment. Existing routing protocols are therefore not able to deliver datagrams to the mobile host correctly, but always route them to its home link. The purpose of Mobile IPv6 [15] is to enable a mobile host to change its point of attachment to the Internet while still maintaining transport–layer connectivity.

The difficult task is to choose an approach for providing multicast traffic to mobile hosts that respects most multimedia applications' constraints.

Interoperation of Multicast and Mobility. The problem considered here is about providing IP multicast to mobile hosts using Mobile IPv6. We do not forget that the source of a multicast traffic comes from an IPv4 server before crossing the gateway and becoming an IPv6 flow. We also suppose that the sender is a fixed node. Thus, no mobility is involved at the source. It means that, for the moment, the focus is done at the receiver side.

Two approaches can be brought here for the support of multicast by mobile nodes as described in section (2.2).

Multicast group registration on home link: a mobile node registers a multicast group on its home link via its home agent. A tunnel between the MN and the HA is set up.

Multicast group registration on foreign link: a mobile node registers a multicast group on its foreign link, via its local multicast router. Its care-of address is used as the source address.

Each of these two approaches gathers advantages and drawbacks. Our solution conveys both, trying to propose an optimized way to deal with multicast flows to mobile hosts. The main requirement for multimedia applications is the delay. We cannot tolerate a long join delay, therefore the multicast group registration should be done on home link through the HA. For that, we propose an extension of the Binding

Update: Multicast Registration BU. In order to optimize this solution and not sending all the work to the HA, we only adopt this solution until the Mobile Node is able to register to its border router for getting the multicast flow through this way. At that time, the MN will inform its HA that it doesn't need to receive multicast packets from it, thanks to a Multicast Deregistration BU. From then, the HA will not take care of multicast packets destined to this MN. They will be received from the nearest designed router on its care-of address.

The main drawback of this solution is the increase of the signaling part, since new BUs and exchange messages are used between the entities.

However, a good point is that security can be involved through this method. First, the MN-HA tunnel provides it and then it is possible to use a pseudo return routability procedure between the MN and the designed router as done for routing optimization in Mobile IPv6 between the CN and the MN. Further work and a deeper look need to be added in this area to fulfill these achievements.

3.4 Scalability Issues

Scalability is an important issue for multicast session management; it has not been forgotten in our solution.

Unlike MTP [6], which handles connections and creation of multicast group locally, M3G doesn't manage any connections. It takes principally care of routing of packets after their translations.

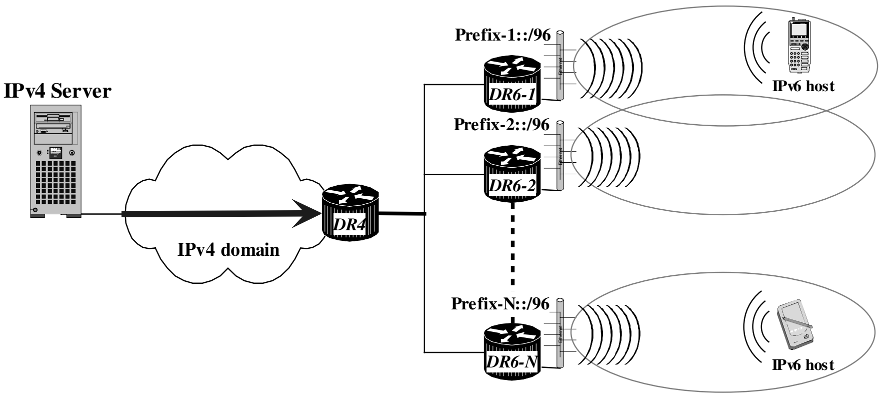


Fig. 1. M3G can be extended almost infinitely by adding M3G boxes. Each M3G box must have an unique prefix (PREFIX::/96) for routing reasons.

Also, we keep DNS information inside the gateway because the DNS has a stateless nature. Thereby, we can limit the use of the gateway and throwback potential bottleneck [7]. This limitation can also be done, by tracking IPv4 SAP/SDP announces in order to avoid redundant costly translations made at application level. Thus, we have to keep the first SAP packet of a given session, after translation, in order to do this translation only once and then send this packet automatically, after the identification of the SAP-4 session.

An optimization is also done by interacting with IGMP/MLD protocols to translate multicast packets only when there is at least one host wanting to join the multicast group.

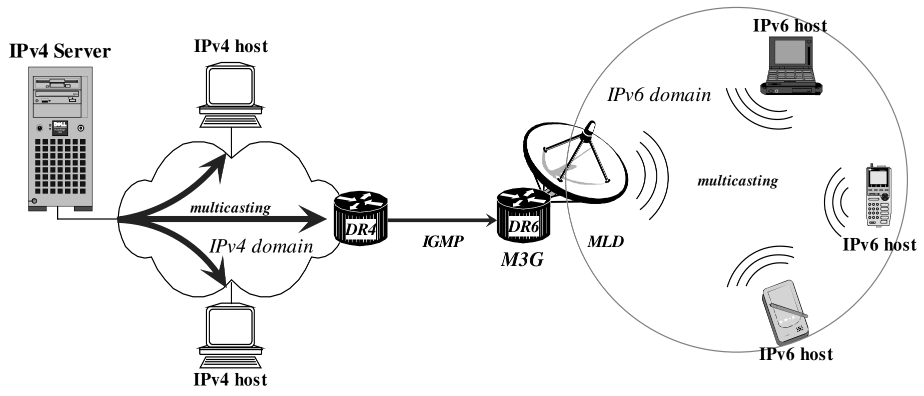


Fig. 2. We consider an IPv6 LAN where some IPv6-only mobile hosts communicate with remaining IPv4 Server in IPv4 domain. We solve the case where IPv6 equipments must talk with IPv4-only domains and vice versa in unicast and multicast. We suppose that no change can be made to the IPv4-only equipments and applications.

4 Gateway's Structure

4.1 M3G Architecture

The communication is performed through our gateway (M3G) for translating IPv4 packets to IPv6 in order to allow a transparent media access to IPv4 multimedia services. We assume that the router behind our gateway, in the IPv4 domain, implements IGMP protocol. As a result, our gateway will solicit a given multicast stream based only on the clients' requests. According to this fact, M3G considers only the active multicast stream and thus, improves the gateway scalability.

4.2 Implementation Issues

Components Specifications

Link Manager. It operates at a MAC level for the transmission of IPv6 packets, according to RFC 2464 [18] and does the filtering with the use of Berkeley Packet Filters (BPF), thanks to which very low-level programming and interaction with the kernel is done.

Network Manager. It initializes the network functionality and handle IP packets to network modules.

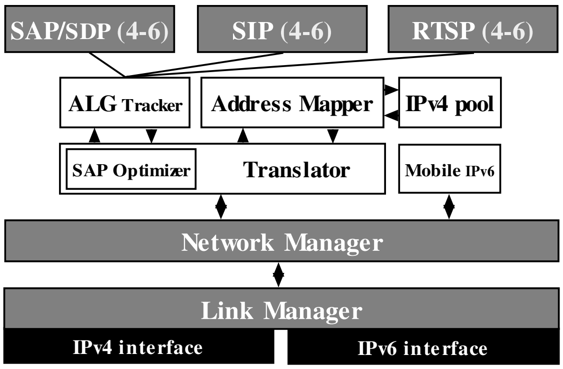


Fig. 3. This section describes in a detailed way the components needed for the whole mechanism and provides an example of the global functioning of the gateway.

Translator. This is the component which translates IPv4 packets into IPv6 ones and vice versa including ICMP and IGMP/MLD packets. The translator includes a "SAP Optimizer" module that permits optimization of communication by sending automatically SAP packets (which have been previously translated).

Mobile IPv6. This module is located at the same level as the translator since Mobile IPv6's signaling process consists of IP level packets. It directly interacts with the translator module by considering IPv6 translated stream. However, it is a complete separate module. It will have to be implemented in the nodes too, either as Home Agent or Mobile Node, depending on what it is desired. It is also possible to set the gateway as a Home Agent.

ALG Tracker. It automatically detects packets concerned with a translation at an application level. Every packet is oriented towards the appropriate module. It will determine by looking into the address field and the port field, if necessary, of translated packets. Therefore, multicast streams will not be translated to the application level and the rapidity of the functioning will increase. Only signaling packets will be translated.

Address Mapper. It maintains each unicast and multicast address pool for IPv4 and IPv6. It also maintains a mapping table which consists of pairs of an IPv4 and an IPv6 address [6]. The translator translates packets between IPv4 and IPv6 domains according to this table. This module is directly connected to the translator, which chooses from it necessary addresses.

SAP/SDP (4-6), SIP (4-6) and RTSP (4-6). These modules do the conversion of packets at the application level. This architecture permits the addition of new modules easily. These signaling packets that need to be converted at the application level are then retransmitted to the desired domain.

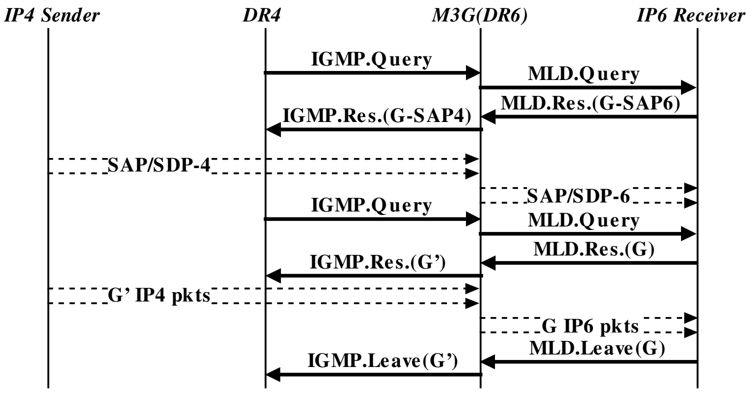


Fig. 4. Explain communication from one IPv4 multicast multimedia sender node to one or more IPv6 multicast receivers nodes based on SAP/SDP signalization.

Global Functioning Example

The communication between the sender and the receiver can be done in two steps:

Joining a multicast group. The translation of IGMP packets allows IPv6 receivers, implementing the MLD protocol, to join naturally multicast session. M3G, in that case, behaves as a designated router (DR6) for IPv6 receivers but in reality, it's the router behind (DR4) who is the designated router for IPv6 receivers.

Thus, IPv6 receivers will receive SAP/SDP signaling flows. The IPv6 receivers will be able to join in the same way another multimedia multicast session.

Leaving a multicast group. When leaving a multicast group, the IPv6 receivers' nodes send MLD Leave request. This request must not be handled at our gateway but it must be translated to IGMP Leave request. So, there is no state at our gateway about multicast receivers.

5 Performance Analysis

M3G's implementation is evaluated over a test platform using various network configurations. We emphasize the scalability and efficiency of our proposal using several MPEG-4 communication sessions. Each session is generated by a server attached to the wired IPv4 network, while the clients are located in the wireless IPv6 network behind an access point (802.11b) attached to the M3G (Fig. 2) entity.

According to this scenario, the server generates RTP sessions and their associated SAP/SDP signaling streams.

In our evaluation, we use an MPEG-4 with different spatial resolutions. This MPEG-4 video configuration, with a reduced complexity, is much suitable for mobile handled terminals. Also, we use different audio streams configuration ranging from MPEG-2 Audio layer 3 (mp3) to MPEG-4 Audio (aac), with different bit rates and sampling rates in order to get more information about the M3G behavior. Based on a predictive coding, these media streams are very sensitive to the delayed packets since the packet with a too large latency will be discarded. In addition to these constraints, the delay variations (jitter) affect directly the perceived quality at client side.

All the components of our platform (i.e. servers, M3G gateway, and handled client) are running on a Linux (kernel 2.4.18) operating system. Both the gateway and the client are patched with USAGI pack for IPv6 support.

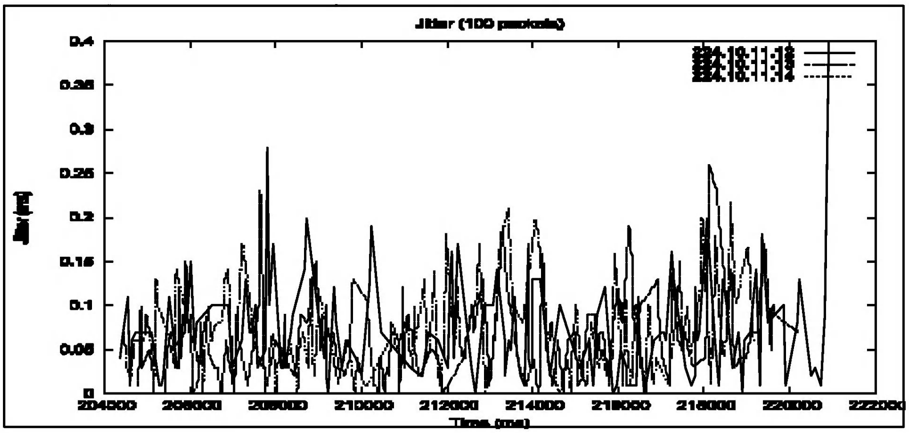


Fig. 5. Is a graphical representation of the jitter in the same environment as described above. We observe that the delay variation (jitter) for the processing of 100 packets remains under an acceptable threshold, the maximum being 0.38 ms, and the average around 0.07ms.

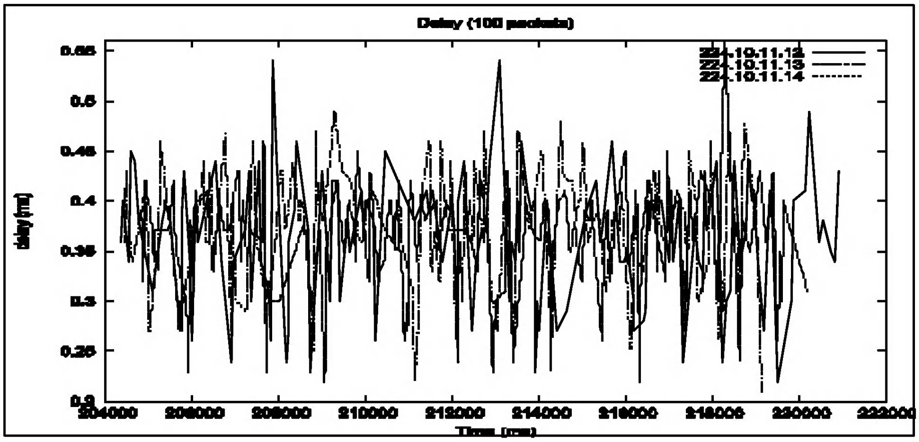


Fig. 6. Represents the delay induced by the treatment process of our gateway for RTP media packets. The Y-axes represents the delay in milliseconds for the processing of 100 packets and the X-axes represents the time scale related to this communication session. Our gateway doesn't introduce such a delay variation, which could lead to a too important packet discarding rate.

For further analyses and details about the performance evaluation of the M3G, we recommend to take a look to the internal report [19].

6 Conclusion

This article provides open issues on the smooth and efficient support of multicast multimedia services involving heterogeneous mobile IPv4 and IPv6 hosts. After analyzing related works on this broad area as well as existing but limited and partial solutions, we have proposed and evaluated an integrated Multicast Media Gateway for IPv4/IPv6 Multimedia Service Transition.

The overall M3G's architecture and associated functional modules have been described and implemented on a Linux-based system for features validation and performance evaluation. Transition issues regarding the user and the signaling control plans have been both addressed in this work.

A proposal on an interaction mechanism between IPv4 and IPv6 clouds for the automation of the access to multicast groups by wireless IPv6 hosts has been developed. Next, an implementation of Application Level Gateways (ALG) has been brought for SAP/SDP and RTSP in order to support both IP multimedia service types: (1) interactive unicast (e.g. VOD) and (2) non-interactive multicast (e.g. TV broadcasting) access modes.

Regarding to the mobility aspect of such services, new approaches and improvements have been added to the signaling plan through a better integration of existing multimedia session management, mobility and multicast signaling protocols. Consequently, transparent establishment and efficient control of hybrid multimedia multicast communications between mobile IPv4-only and IPv6-only clouds are possible while preserving legacy IPv4 networks and services during a smooth IPv6 transition.

References

1. Nordmark, E.: Stateless IP/ICMP Translation Algorithm (SIIT). RFC2765 (2000).
2. Handley, M., Perkins, C., Whelan, E.: Session Announcement Protocol (SAP). RFC2974 (2000)
3. Handley, M., Jacobson, V.: Session Description Protocol (SDP). RFC2327 (1998).
4. Olson, S., Camarillo, G., Roach, A. B.: Support for IPv6 in Session Description Protocol (SDP). RFC3266 (2002)
5. Tsirtsis, G., Srisuresh, P.: Network Address Translation – Protocol Translation (NAT-PT). RFC2766 (2000)
6. Tsuchiya, K., Higuchi, H., Sawada, S., Nozaki, S.: MTP: An IPv6/IPv4 Multicast Translator based on IGMP/MLD Proxying". draft-ietf-ngtrans-mtp-03.txt (2002)
7. Hallin, P., Satapati, S.: NAT-PT DNS ALG solutions. draft-hallin-natpt-dns-alg-solutions-01 (2002)
8. Fenner, W.: Internet Group Management Protocol, Version 2 (IGMPv2). RFC2236 (1997)
9. Deering, S., Fenner, W., Haberman, B.: Multicast Listener Discovery (MLD). RFC2710 (1999)
10. Schulzrinne, H., Casner, S., Frederick, R., Jacobson, V.: A Transport Protocol for Real-Time Applications. RFC1889 (1996)

11. Bradner, S., Mankin, A.: The Recommendation for the IP Next Generation Protocol. RFC1752 (1995)
12. Deering, S., Hinden, R.: Internet Protocol, Version 6 (IPv6) – Specification. RFC1883 (1995)
13. King, S., Fax, R., Haskin, D., Ling, W., Fink, R., Meehan, T., Perkins, C. E.: The Case for IPv6. draft-iab-case-for-ipv6-06.txt (2000)
14. Gilligan, R., Nordmark, E.: Transition Mechanisms for IPv6 Hosts and Routers. RFC2893 (2000)
15. Johnson, D., Perkins, C., Arkko, J.: Mobility Support in IPv6. draft-ietf-mobileip-ipv6-21.txt (2003)
16. Perkins, C.: IP Mobility Support. RFC 2002, Mobile IP Networking Group (1996)
17. Harrison, T., Williamson, C., Mackrell, W., Bunt, R.: Mobile multicast (MOM) protocol: multicast support for mobile hosts. in Proc. Of ACM MOBICOM'97 (1997) 151–160.
18. Crawford, M.: Transmission of IPv6 Packets over Ethernet Networks. RFC2464 (1998)
19. Hadjadj Aoul, Y., Negru, D., Nafaa, A., Mehaoua, A.: M3G: Deploying Multimedia Services for Next Generation Networks. Technical Report PRiSM Lab., University of Versailles, France (2003).

TCP Friendly Rate Adaptation for Multimedia Streaming in Mobile ad hoc Networks

Zhenghua Fu, Xiaoqiao Meng, and Songwu Lu

Computer Science Department, University of California, Los Angeles
{zfu,xqmeng,slu}@cs.ucla.edu

Abstract. Transport protocol for supporting multimedia streaming in mobile ad hoc networks has to cope with the rich dynamics, such as mobility-induced disconnection and reconnection, high out-of-order delivery ratios, channel errors, and network congestion. In this work, we design and implementate ADTFRC, a TCP-friendly transport protocol for ad hoc networks. ADTFRC adapts wireline TFRC protocol to ad hoc networks with improved rate adaptation behavior, the capability of application layer framing and selective retransmission. ADTFRC detects different packet loss behaviors based on end-to-end measurements of multiple metrics. This allows ADTFRC to more accurately gauge the network behavior and achieve higher throughput. Simulations show that the performance of ADTFRC is higher than standard TFRC and TCP with explicit-link-failure-notification (ELFN) support in terms of throughput, rate adaptation behavior and application level quality while still maintaining the TCP-Friendliness property.

1 Introduction

The new generation wireless networks offer much higher transmission rate, e.g., IEEE 802.11b supports up to 11 Mbps transmission rate, IEEE 802.11a can support even up to 54 Mbps. It therefore feasible to run multimedia applications over mobile ad hoc networks. Such applications can include multimedia instant messaging, environmental monitoring, distributed gaming and etc. In this paper, we study protocol design to support multimedia streaming over ad hoc networks. Most of the relevant research efforts are focused on the low-layer design such as service differentiation in MAC QoS-aware routing, admission control and adaptive packet scheduling. Different from these work, we address the transport-layer issues. Specifically, we adapt the popular, slowly responsive congestion control protocol — TCP Friendly Rate Control (TFRC), which is originally proposed for wired multimedia transport, to mobile ad hoc networks.

Early proposals use UDP to carry multimedia streams [20]. Since UDP does not provide congestion control, unresponsive multimedia flows will compete unfairly with other responsive TCP flows. Using TCP in multimedia streaming transport can prevent congestion collapse, however, TCP provides 100% reliability through its retransmission mechanism, which is not necessary for loss-tolerant multimedia streaming. Moreover, TCP halves its transmission rate upon any congestion event; such dramatic rate oscillations are deemed to be detrimental to multimedia applications.

Recent research on multimedia transport design has focused on developing a TCP friendly protocol that does not react to any single congestion event dramatically but slowly adapts to network dynamics [16][18]. A noticeable proposal is TCP Friendly Rate Control (TFRC) [17][19], which calculates a TCP friendly throughput and increases/decreases the transmission rate accordingly.

There are several technical challenges for TFRC to function well in mobile ad hoc networks. It is well known that because ad hoc networks exhibit a rich set of packet loss behaviors, directly applying the congestion control mechanisms of TFRC upon every packet loss typically leads to unsatisfactory performance [1][6][7][5]. Moreover, TFRC has to handle several ad hoc networks specific network events , i.e., mobility-induced disconnection and re-connection, route change induced out-of-order delivery and error/contention-prone wireless transmissions[1]. These events require TFRC to respond differently from congestion control. For example, it makes more sense to ignore a random packet loss incurred by channel errors rather than to multiplicatively decrease the sending rate [3]; and it seems to be more appropriate to periodically probe the network during disconnections for a prompt recovery than to slow down and exponentially increase the retransmission timer [1]. Another challenge for TFRC is to accurately detecting and differentiating all these network events. Packet loss as the only detector used by conventional TCP/TFRC flows has been shown to fail to differentiate all these new events [8].

1.1 Rate Adaptation in ad hoc Networks

In order to effectively adapt the transmission rate, network congestion should be reliably detected. In particular, among all kinds of packet losses, the congestion loss probability needs to be estimated; treating all losses as congestion loss leads to undesirable rate adaptations. Most of the literature on congestion detection for ad hoc networks endorses a network-oriented approach. In this approach, the routers implement a monitoring module and generate explicit notifications to send back to the TCP sender upon various packet losses. Specifically, if mobility triggers network disconnection, an explicit link failure notification (ELFN) will be sent to the sender [1]; if a congestion loss occurs, an explicit congestion notification (ECN) message is generated [23]; if the router observes a packet loss induced by the wireless channel [3][4], an explicit loss notification (ELN) will be sent to the TCP sender. Although such a network-oriented approach is can effectively improve TCP performance, it suffers from several drawbacks when it is used for rate adaptation in ad hoc networks.

First, the failure notifications, such as ELFN, generated by intermediate nodes are directly sent to the sender without the awareness of the receiver. Hence, a single packet loss event can trigger two different reports, one is from intermediate nodes and the other comes from the receiver. Because the two reports can arrive at the sender in an arbitrary order and at different time instances, it is difficult to combine the two observed reports and render a consistent image of the overall network condition [2]. Secondly, it is difficult

[1] Even with link-layer retransmissions of 802.11 MAC, packet loss still occurs due to bursty channel error or MAC-layer contentions.

[2] Even if the measurements are performed at sender side, feedbacks from both receiver and intermediate nodes are still needed, same complication arises when trying to combine the two overlapping observations.

to globally deply monitoring modules at every node because of the heterogeneity of an ad hoc environment.

Based on the above considerations, an end-to-end approach is more desirable for TCP-Friendly rate adaptation in mobile ad hoc networks since it naturally falls into the existing TFRC protocol [17]. An end-to-end approach means that the TCP receiver differentiates network events, decides the occurrence of congestion and sends feedback to the sender. The sender then adjusts the transmission rate accordingly. Compared with the network-oriented approach, such an end-to-end approach is much easier to be implemented and deployed in practice.

We notice a relevant work [24] which uses packet out-of-order to differentiate packet losses due to route changes or network congestion. However, [24] assumes that packet losses can only be caused by route changes or congestion. Our contribution is that we provide a more general and reliable solution to detect network conditions while still preserving the end-to-end principle.

1.2 Main Contributions

The key innovation proposed in this paper is the use of multi-metric joint detection instead of single-metric detection. Because end-to-end measurement data in ad hoc networks are usually highly noisy, frequent false identifications and notifications can happen [8]. How to *robustly* detect events through noisy measurements imposes a challenge to the design. Based on a multi-metric joint detection, we exploit the degree of independence in the measurement noise of each individual metrics, so that the probability of false identification is significantly reduced by cross verification among the multiple metrics.

In addition, we applied the Application Level Framing (ALF) and Partial Reliability techniques to improve the quality of the multimedia streaming perceived by the end user. The resultant ADTFRC protocol is implemented in NS-2 simulator, and the performance of ADTFRC is extensively evaluated.

Our results show that, without compromising the TCP-Friendliness property, ADT-FRC outperforms TFRC and TCP NewReno with ELFN support in terms of throughput, packet loss ratio and smoothness in rate adaptation behavior. By using real MPEG-4 video traces to evaluate the streaming quality at user level, we demonstrate that ADT-FRC significantly improves the streaming quality of TFRC and TCP NewReno with ELFN support.

The remainder of the paper is organized as follows: Section 2 provides an overview of our ADTFRC design. Section 3 describes the detection design. Section 4 presents the ADTFRC protocol implementation issues together with the ALF and partial reliability design. Section 5 is performance evaluations and Section 7 concludes the paper.

2 Design Goal of ADTFRC Protocol

The design of ADTFRC relies on the ideal rate adaptation behavior with full knowledge of the network states in ad hoc networks. We now define the network states to be distinguished and the ideal rate adaptation policies to be applied in each state. *CONGESTION (CONG)*: We define congestion in ad hoc networks as the signal that the offered load exceeds the network capacity. When congestion occurs, usually there are queue building up and the network throughput is reduced becaused of excessive contention delays and collision losses. To deal with congestion, the transport protocol should reduce the sending rate, react the same as the standard TFRC.

CHANNEL_ERR (CHERR): The receiver should not treat random packet loss as congestion event, instead of slow down, the sender should calculate the sending rate as normal.

ROUTE_CHANGE (RTCHG): Because the delivery path between two end hosts can change from time to time, disconnection may happen and it may be too transient to incur retransmission timeout. In this case, the receiver experiences a short burst of out-of-order packet delivery or packet losses. The receiver, again, should not treat it as congestion, instead, the sender should keep the streaming rate unchanged in the next RTT period, waiting for the receiver to feedback more measurement statistics for the new path.

DISCONNECTION (DISC): When the delivery path is disconnected for a long time so that a retransmission timeout happens, the sender should freeze the current congestion window and the retransmission timer instead of exponentially slow down and back off. The sender then performs a periodic probing to prepare for resuming the transmission once a new path is established. When the new path is established, the actions in the case of RTCHG are followed. We notice that such a probing technique is also proposed in [1][2].

So far, we have proposed four network states that need to be identified. In the next section, we will present a multi-metric joint detection algorithm to reliably detect the four network states by using noisy end-to-end measurements. The detection algorithm is implemented at the receiver side, which periodically updates the sender with its current network state estimation through ADTFRC feedback packets.

3 Detection via Multiple Metrics

3.1 Devising End-to-End Metrics

End-to-end measurement has been widely used in transport protocols. In TCP, the round trip time (RTT) is maintained by the sender to calculate the retransmission timeout. Previous work uses delay related metrics to measure the congestion level of the network. For example, [2] and [8] use inter packet arrival delay, and [9] uses RTT to estimate the expected throughput. A challenge in ad hoc networks is that packet delay is not only influenced by network queue length, but it is also susceptible to other conditions such as random packet loss, routing path oscillations, MAC layer contention and etc. These conditions make such measurement highly noisy. Rather than pursuing any single metric that is robust to all dynamics of the network, we devise four end-to-end metrics that tend to be influenced by different conditions so that the noise independence among them can be exploited by multi-metric joint identification.

Inter-packet delay difference (IDD): IDD measures the delay difference between a pair of consecutive packets. It measures the congestion level along the forwarding delivery path by directly sampling the transient queue size variations in the intermediate nodes. However, in an ad hoc network, there are still a number of situations in which

IDD values might give an incorrect estimation of congestion. For example, *IDD* can be influenced by non-congestion conditions such as mobility induced out-of-order packet delivery. We therefore introduce an additional metric STT in the following.

Short-term throughput (STT): STT is a metric used together with IDD to identify network congestion. STT provides observations within a time interval T, and it is less sensitive to short term out-of-order packet delivery than IDD. Therefore, STT is more robust to transient route changes, which can happen very frequently in ad hoc networks. However, using STT alone to detect network congestion can be susceptible to measurement noise introduced by bursty channel error, network disconnections or altering source rate. Therefore, we combine STT and IDD to jointly detect the network congestion. Aside from the IDD and STT which are related to congestion, we also consider the following two metrics for non-congestion state identification.

Packet out-of-order delivery ratio (POR): A packet is counted as being out-of-order if it arrives after a packet that was sent later than it (by the same sender). The receiver records a maximum sending time T_{maxtx} for all the received packets since the establishment of the connection. Every received packet that has a sending time-stamp less than T_{maxtx} is added into POR. POR is intended to indicate a route change event. During the route change period, multiple delivery paths exist. Packets along the new path may catch up, and those along the old path are then delivered out-of-order.

Packet loss ratio (PLR): Within each time interval $[t, t+T]$, we compute this metric as the number of missing packets in the current receiving window. PLR is used to measure the intensity of channel error.

3.2 Detecting the Network States

In the previous section we describe network states that are important for improving TFRC performance in ad hoc networks, and also metrics that can be measured end-to-end. In this section, we study the identification of these states by using the above four metrics. Unless explicitly specification, the default settings for all the simulation results shown in this section are as follows: We use the NS-2 simulator with CMU wireless extension modules. 30 wireless nodes roam freely in a $400m \times 800m$ topology following a *random waypoint* mobility pattern, in which the pause time is zero so that each node is constantly moving. The wireless link bandwidth is 2M bps. IEEE 802.11 and Dynamic Source Routing (DSR, see [1]) are used as MAC and routing layer protocols respectively. One TFRC flow is created with equal packet size 1000 bytes. To introduce congestion, three competing UDP/CBR flows, each with source rate 180K bps, are created within the time intervals of [50,250],[100,200] and [130,170] respectively. The throughput for each UDP flow is 180Kbps. The simulations last for 300 seconds.

Detecting Congestion. To study the relationship between network congestion and IDD/STT, we simulate both static and mobile scenarios. A TFRC flow and three competing UDP/CBR flows are introduced in each simulation and the network is expected to become congested as it becomes overloaded. The first two figures of Fig. 1 show the simulation results. The first is for the static case without channel errors, and the second is for the mobile case in which node mobility speed is 5m/s and channel error ratio is 5%. In both figures, we plot the measured IDD/STT values with respect to the instantaneous

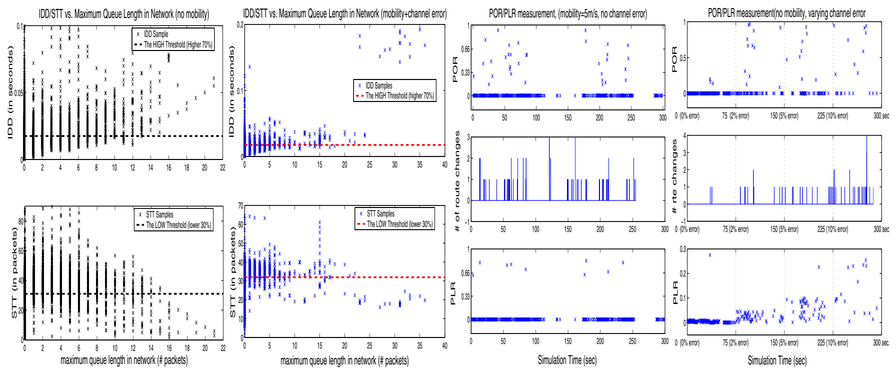


Fig. 1. End To End Metric Measurement. Left two: the IDD and STT measurement w.r.t. instantaneous maximum queue occupation of all wireless nodes in the network. First figure shows simulation with static nodes; second figure shows simulation with mobile nodes (5 m/s). Right two: POR and PLR measurements w.r.t. the number of route changes. The third figure shows simulation with mobile nodes (5 m/s), no channel error. The fourth one shows simulation with static nodes, progressively increasing channel error (0%,2%,5%,10%) during the entire run.

maximum buffer occupation of all nodes in the network, which reflects the network congestion level at the sampling time instance. [3]

In Fig 1, we observe that when the maximum network queue size exceeds half of the buffer capacity (25 packets), IDD is clearly *high* and STT is clearly *low*. We formalize such an observation by defining a value to be HIGH and LOW if it is within the top and bottom 30% of all samples respectively [4]. However, when the network queue size is small (non-congestion case), both IDD and STT vary from LOW to HIGH, with the majority of IDD samples being not HIGH and STT samples not LOW. In the left two figures of Fig. 1, when node mobility is present, the two metrics become much more noisy in non-congestion state (i.e., small network queue).

In the single metric-based detection using either IDD or STT, the noise reduces the accuracy significantly when the network is not congested, especially in scenarios with mobility and channel errors. However, in the proposed joint detection approach, we can use both metrics to *verify* each other to improve the accuracy. Specifically, a congestion state is identified when IDD is HIGH and STT is LOW, otherwise it is identified as a non-congestion state. The following shows why the multi-metric approach has better detection accuracy than the single metric approach.

When the network is congested, let P_1 and P_2 be the probability that IDD is HIGH and STT is LOW respectively. The single metric accuracy is $acc_{idd}(cong) = P_1$ and $acc_{stt}(cong) = P_2$. For the multiple metric case, $acc_{multi}(cong) = P_1 \cdot P_2$. Since the simulations show that $P_1 \simeq P_2 \simeq 1$ (see left two figures of Fig. 1), these three

[3] The maximum buffer size for each node is 50 packets in our simulations.

[4] This threshold was determined empirically from simulation results and real testbed measurements [15].

accuracies are roughly equal in congestion state. On the other hand, when the network is not congested, let P_1' and P_2' be the probability that IDD is still HIGH and STT is still LOW. Similarly, we have $acc_{idd}(non\_cong) = 1 - P_1'$, $acc_{stt}(non\_cong) = 1 - P_2'$ and $acc_{multi}(non\_cong) = 1 - P_1' \cdot P_2'$. Since each noise probability, $0 < P_1', P_2' < 1$, is non-negligible, multiple metrics thus achieve higher accuracy. Combining these two cases, multi-metric identification improves the accuracy in non-congestion states while maintaining a comparable level of accuracy in the congestion state. Therefore, it achieves better identification performance over a variety of network conditions.

The key insight here is that in the non-congestion state, IDD and STT are influenced differently by various network conditions, such as route change and channel error; while in congestion state, they are both dominated by prolonged queuing delay. Thus, the two noise probability P_1' and P_2' become largely independent. Effective verification across multi-metrics is possible as long as these conditions do not co-exist during the measurement time interval. Although this joint identification technique cannot achieve perfect accuracy, it does increase the accuracy significantly as we show in Section 3.3.

Detecting Non-congestion States. If the network state is not congestion, we next seek to detect whether it is RTCHG or CHERR. The third figure of Fig 1 shows data from a simulation run with node mobility speed being 5m/s. A single TFRC flow is created without any competing flows that might cause network congestion. We plot POR and PLR sample values together with the number of route changes in the forwarding path over time. A clear correlation is seen between route change events and bursts of high POR measurement. During the changing period, packets arrive at the receiver from multiple paths and consequently may lose their ordering. Although not all route changes result in out-of-order packet delivery, we only count those observable changes, which would have an impact upon TFRC. POR can be used to identify RTCHG state and PLR can be used for CHERR state.

Moreover, since there is no congestion or channel errors in these simulations, PLR remains stable with a few significant outliers. These anomalies correspond to situations in which packets along the old path are excessively delayed or lost.

In the simulation shown in the fourth figure of Fig 1, nodes are stationary and four channel error rates ($0\%, 2\%, 5\%$ and 10%) are introduced into four identical time intervals (75 seconds). In this case, packet loss is proportional to the channel error rate and the PLR gradually increases as the channel error rate increases. Note that a high channel error rate can also create route change in the network that will in turn result in bursts of high POR measurements. The routing layer interprets any MAC-layer transmission failures (in this case, channel error) as a sign of a broken link and consequently seek to repair/re-establish the delivery path, which may cause route changes.

In conclusion, a burst of high POR sample values is a good indication of a route change and a high PLR is a good indication of a high rate of channel error. It should be noted that the network may be both in a state of high channel error and route change, which can be identified by high values in both PLR and POR.

We next consider disconnection. Disconnection happens when packet delivery is interrupted for non-congestion reasons for long enough to trigger a retransmission time-out at the sender. Multiple network conditions can trigger such a timeout at the sender including frequent route changes, heavy channel error, and network partition after mobility. If the timeout is triggered by congestion, then previous state feedback should reflect the transient queue build up period by increasing IDD and STT measurement at the receiver; if not, the timeout was due to non-congestion conditions in the network. Therefore, a DISC state is identified at the sender if the current state estimation is non-congestion when retransmission timeout is triggered.

Table 1. Metrics patterns in 5 network states. High: top 30% values; Low: bottom 30% values; '*': do not care

	IDD and STT	POR	PLR
CONG	(High, Low)	*	*
RTCHG	NOT (High, Low)	High	*
CHERR	NOT (High, Low)	*	High
DISC	(*, $\approx$ 0)	*	*
NORMAL	default		

Table 1 summaries the metrics patterns in the four network states. Later on we show that such an identification method, combined with a simple sample classification technique, achieves an accuracy above 80% on average in all simulations scenarios.

3.3 Detection Accuracy

We now study the accuracy of congestion identification using the RSD technique. In particular, we compare the single-metric (using only IDD or STT) and multiple-metric (using both) approaches. We run two sets of simulations under non-congested and congested cases (Figure 2). In the first non-congested case, a single TFRC flow is created within the topology. In the second congested case, two competing UDP flows are created as before. In both cases, 1% random channel error is introduced and the mobility speed varies from 0 to 20 m/s. We repeat simulations 50 times at each speed to reduce the impact of random topology factors.

During the simulation, upon each packet loss, we compare the identified network state and the actual network state to determine the accuracy of detection[5]. In particular, if a packet is lost due to network congestion, but the algorithm gives non-congestion estimation, we count it as an incompatible error because this error in detection (and only this one) causes ADTFRC to be more aggressive than a TCP-friendly flow and consequently TCP-incompatible.

Figure 2 shows the percentage of inaccurate identification in both cases. In the single TFRC flow case (the left figure), mobility and channel errors are the dominant reasons for packet loss. The increase in mobility speed reduces the accuracy of the single-metric identification quickly. However, the multi-metric approach results in only 10% to 30%

[5] The real network state is obtained by a global monitor implemented in NS-2 simulator. See [5] for implementation details.

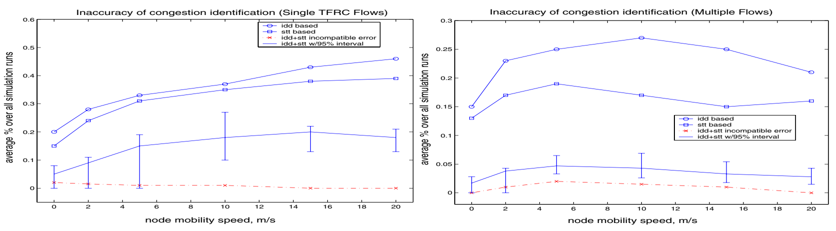


Fig. 2. Identification Accuracy. Left: Percent of inaccurate identifications in a non-congested case, Right: Inaccuracy ratio in a congested case

inaccurate identification. This is achieved by the cross verification between IDD and STT measurements to eliminate false congestion alarms. Meanwhile, the incompatible error remains less than 2%.

In the multi-flow cases (the right figure), congestion happens more frequently. For multi-metric identification, more than 95% accuracy is observed in all simulations with less than 2% incompatible errors. For the single metric approach, accuracy is only about 70% to 80%.

In summary, we have demonstrated that multiple metrics combined with RSD is a feasible approach to detect network events by end-to-end measurements only.

4 ADTFRC Protocol Design and Implementation

We now incorporate the design of Sections 2 and 3 in our ADTFRC protocol to improve the performance of TFRC in ad hoc networks.

4.1 Adaptive Rate Adaptation

ADTFRC seeks to maintain backward compatibility with conventional TFRC. It uses identical connection establishment and connection teardown processes. It estimates the RTT and derives the sending rate in the same way with TFRC. To improve the performance of TFRC in ad hoc networks, ADTFRC makes several extensions at both the sender side and receiver side.

Upon each packet arrival at the receiver, besides the normal operations, values for the four previously discussed metrics are calculated and network states are estimated. In ADTFRC, the congestion probability is calculated based on the outcomes of our multi-metric identification instead of the packet loss events. The receiver then passes this congestion frequency measurement together with state estimations, i.e., CONG, CHERR and RTCHG, to the sender in every feedback packet. Besides the regular feedback of each RTT, the receiver generates *Urgent* state update packet as soon as a congestion event is detected and fedback to the sender immediately. The sender maintains the most recently received state report , and proceeds with normal TFRC operations until either of the following two events happen: the reception of feedback packet, or the re-transmission

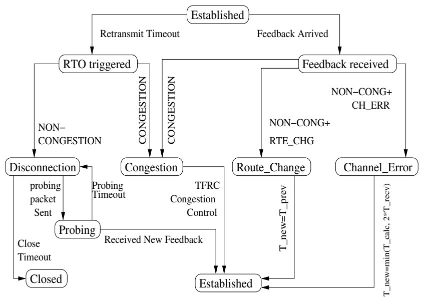


Fig. 3. ADTFRC state diagram for sender in *NS-2* implementation

time out. A modified TFRC state diagram is shown in Figure 3 for the sender. The pseudo code is avaialbe in our technical report [15].

A feedback report or retransmission timeout triggers ADTFRC to take different control actions according to the current network state estimation. In particular, a probing state is introduced to explicitly handle network disconnection. When a non-congestion induced retransmission timeout occurs at the sender, ADTFRC freezes its current transmission state and enters a probing state. The sender leaves the probing state when a new acknowledgement is received or the probing is timed out[6]. The ADTFRC connection is closed after multiple probing attempts fail.

5 Performance Evaluation

In this section, we evaluate the performance of ADTFRC through extensive *NS-2* simulations in terms of its throughput, rate adaptation behavior as well as the application level quality perceived by the end user.

5.1 Throughput Improvement

In the throughput evaluation part, we compare it to TFRC and TCP with ELFN [1] support. Instead of using end-to-end measurements, TCP ELFN collects link state information directly from the network and is expected to be more accurate. It is used as a reference system; a throughput close to ELFN indicates the effectiveness of ADTFRC.

Figure 4 shows the single flow throughput of ADTFRC, TFRC and TCP-NewReno with ELFN support. The simulation parameters for TFRC flows are set as described in section three, and for TCP ELFN flow, we set the packet size to be 1000 bytes and maximum window size to be 8 packets. In all three cases, ADTFRC provides significantly better throughput than TFRC. When nodes are mobile, ADTFRC achieves a throughput improvement from 100% to 800% over TFRC. Furthermore, it is surprising to see that ADTFRC out-performs TCP+ELFN even in a static network where the mobility speed is zero. The reason is because the ACK packet traffic on the reverse path is much heavier in

[6] A similar probing mechanism was proposed by [1]

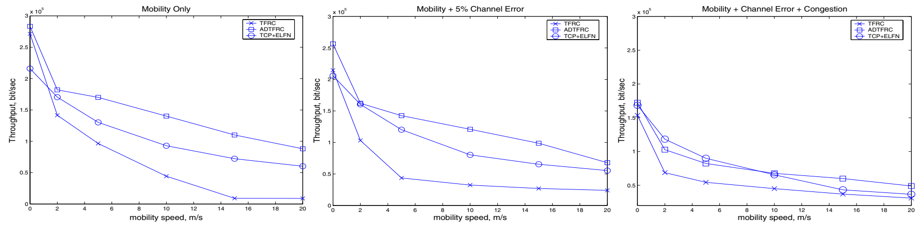


Fig. 4. Performance Improvement of ADTFRC. From left to right:1)mobility only, 2)mobility+5% channel error, 3) mobility+5% channel error + 3 competing UDP/CBR flows

TCP+ELFN than in ADTFRC. Due to the broadcast nature of the wireless link, such ACK flows contend for the channel access with forwarding data flows, introducing additional delay in RTT and resulting in throughput decrease.

5.2 Rate Adaptation

We further measure the rate oscillations experienced at the receiver for each of these three protocols. To effectively support best-effort multimedia streaming, dramatic rate variations are highly undesirable.

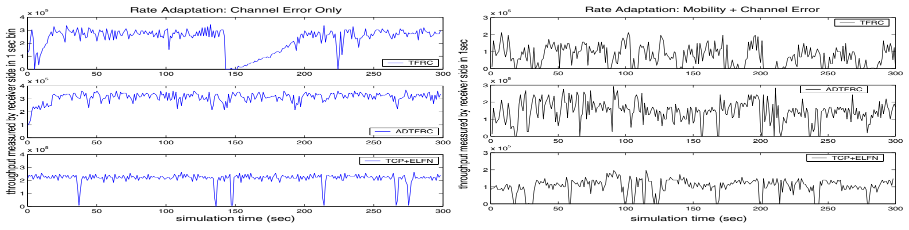


Fig. 5. Smoothness of Rate Adaptation. From left to right:1)Single flow in static ad hoc network, 2% random channel error. 2) Single flow with node mobility 5m/s, 2% random channel error.

In Figure 5 we show the throughput fluctuations at the receiver side in two environment settings. The left figure is for the simulation in static network, with 2% random channel error. It shows that the ADTFRC (the middle one) maintains a more stable transport rate.

When mobility and channel error are both introduced as shown in the right one of Figure 5, the rate variation of ADTFRC becomes much larger. However due to a probing mechanism of ADTFRC, the transmission interruptions are much smaller than the TFRC flow. For TCP+ELFN flow, although its disconnection period is also short, due to its aggressive bandwidth probing mechanism, it again encounters more frequent interruptions than ADTFRC.

5.3 Application Layer Quality

To evaluate the quality improvement perceived by an end host, we use an application layer metric, *client starvation time* that is defined in [26]. This metric characterizes the situation when the client experience a freezing motion or frame skip during the streaming. We use the StarWar video trace [25] encoded with 70Kbps average source rate in MPEG-4 format. The application layer framing and partial reliability is enabled at the two end hosts. The implementation details are again referred to the technical report [15].

Table 2. The (aggregated) client starvation time (in seconds) for single MPEG-4 streaming and two simultaneous flows.

	Single Flow				Aggregated for Two Flows			
	0 m/s	2 m/s	5 m/s	10 m/s	0 m/s	2 m/s	5 m/s	10 m/s
TFRC	14	69	102	178	31	308	1083	4032
ADTFRC	2	15	42	58	4	137	216	367
TCP-ELFN	6	17	39	67	14	187	831	6873

Observe from Table 2, the streaming quality measured by client starvation time is greatly improved by ADTFRC. Compared with standard TFRC or TCP+ELFN, the improvement is especially significant in scenarios when multiple streaming flows run in a mobile network. A detailed trace analysis is available in our technical report [15].

6 Conclusion

In this paper, we explore an end-to-end approach to design a TCP-friendly transport protocol, ADTFRC, to improve the performance of rate adaptation of TFRC in mobile ad hoc network. ADTFRC uses multiple metrics to jointly detect network states in the presence of measurement noises, so that the sending rate can be adjusted accordingly. We also propose the application layer framing to further enhance the streaming quality perceived by the end user. Simulations show that ADTFRC can significantly improves the performance of real-time video streams in a TCP friendly way.

References

1. G. Holland and N. Vaidya, “Analysis of TCP performance over mobile ad hoc networks,” *MOBICOM’99*.
2. P. Sinha, N. Venkitaraman, R. Sivakumar and V. Bharghavan, “WTCP: A reliable transport protocol for wireless wide-area networks,” *MOBICOM’99*.
3. H. Balakrishnan, S. Seshan, E. Amir, and R. Katz, “Improving TCP/IP performance over wireless networks,” *MOBICOM’95*.
4. H. Balakrishnan, and R. Katz, “Explicit loss notification and wireless web performances,” *Globecom’98*.

5. Zhenghua Fu, Xiaoqiao Meng, Songwu Lu, "How bad TCP can perform in wireless ad hoc network" IEEE ISCC (IEEE Symposium on Computers and Communications) 2002, Italy, July 2002.
6. J. Monks, P. Sinha and V. Bharghavan, "Limitations of TCP-ELFN for ad hoc networks," *MOMUC'00.*
7. M. Gerla, K. Tang, and R. Bagrodia, "TCP performance in wireless multihop networks," *WMCSA'99.*
8. S. Biaz and N.H. Vaidya, "Distinguishing congestion losses from wireless transmission losses" *IEEE 7th Int. Conf. on Computer Communications and Networks*, October 1998.
9. L. Brakmo, S. O'Malley, and L. Peterson "TCP Vegas: New techniques for congestion detection and avoidance" *ACM SIGCOMM 1994*
10. V. Jacobson and M.J. Karels, "Congestion Avoidance and Control" *ACM Sigcomm 1988*
11. M. Allman, V. Paxson, and W. Stevens, "TCP Congestion Control" *RFC 2581* April 1999.
12. S. Biaz, N. Vaidya, "Discriminating congestion losses from wireless losses using inter-arrival times at the receiver" *IEEE ASSET* 1999
13. A. Bakre and B. Badrinath, "I-TCP:indirect TCP for mobile hosts", *Proc. 15th International Conf. on Distributed Computing Systems (ICDCS)* May 1995
14. A. Bakre and B. Badrinath, "Implementation and performance evaluation of Indirect TCP" *IEEE Trans. Computers*, Vol. 46, March 1997
15. Z. Fu, X. Meng, S. Lu, "TCP Friendly Rate Adaptation for Multimedia Streaming in Mobile Ad Hoc Networks", *Technical Report of Computer Science Department, UCLA, 2003.*
16. S. Floyd, M. Handley, J. Padhye and J. Widmer, "Equation-Based Congestion Control for Unicast Applications" In *Proceedings of ACM Sigcomm 2000*
17. M. Handley, S. Floyd, J. Padhye and J. Widmer, "TCP Friendly Rate Control (TFRC): Protocol Specification", *RFC 3448, January 2003.*
18. D. Bansal and H. Balakrishnan, "Binomial Congestion Control Algorithms." In *Proceedings of IEEE Infocom, 2001.*
19. D. Bansal, H. Baladrishnan, S. Floyd and S. Shenker, "Dynamic Behavior of Slowly-Responsive Congestion Control Algorithms", In *Proceedings of ACM Sigcomm 2001*
20. H. Schulzrinne, S. Casner, R. Frederick, V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications", *RFC 1889, 1996*
21. S. Raman, H. Balakrishnan, M. Srinivasan, "ITP: An Image Transport Protocol for the Internet", *IEEE/ACM Trans. on Networking*, June, 2002.
22. M. Feamster and H. Balakrishnan, "Packet Loss Recovery for Streaming Video", *12th International Packet Video Workshop*, Pittsburgh, PA, April 2002
23. J. Liu and S. Singh. "ATCP:TCP for mobile ad hoc networks", *IEEE Journal on Selected Areas in Communiations, 19(7):1300–1315* July 2001
24. F. Wang and Y. Zhang, "Improving TCP performance over mobile ad-hoc networks with out-of-order detection and response", *In Proceedings of the third ACM international symposium on Mobile ad hoc networking and computing*, 2002, Lausanne, Switzerland.
25. F. Fitzek and M. Reisslein, "MPEG-4 and H.263 Video Traces for Network Performance Evaluation", *IEEE Network*, Vol.15 No.6, November 2001. Video traces available from the website http://www-tkn.ee.tu-berlin.de/~fitzek/TRACE/pub.html
26. F. Fitzek, M. Reisslein, "A Prefetching Protocol for Continuous Media Streaming in Wireless Environments", *IEEE Journal on Selected Areas in Communications (Special Issue on Mobility and Resource Management in Next Generation Wireless Systems)*, Vol.19, No.6 Pages 2015–2028, October 2001.

BeanWatcher: A Tool to Generate Multimedia Monitoring Applications for Wireless Sensor Networks*

André Lins[1], Eduardo F. Nakamura[1,2], Antonio A.F. Loureiro[1], and Claudionor J.N. Coelho Jr.[1]

[1] Department of Computer Science
Federal University of Minas Gerais – UFMG
Belo Horizonte, MG, Brazil.
{alla,nakamura,loureiro,coelho}@dcc.ufmg.br

[2] Department of Technological Development
Research and Technological Innovation Center – FUCAPI
Manaus, AM, Brazil.
eduardo.nakamura@fucapi.br

Abstract. In this paper we present a new tool called BeanWatcher that allows the semi-automatic generation of multimedia monitoring and management applications for wireless sensor networks. Thus, particularities of multimedia management and wireless sensor networks were taken into account. The architecture of the tool is based on a component model flexible enough to allow the creation of new components and the optimization of the components currently provided. BeanWatcher was designed to offer a development environment suitable for both expert and beginner users allowing them to choose the programming language that better fits the application requirements.

1 Introduction

A Wireless Sensor Networks (WSN) is a special kind of *ad-hoc* network with distributed sensing and processing capability that can be used in a wide range of applications, such as environmental monitoring, industrial applications and precision agriculture [1,2,3]. As a recent practical example, Intel Research and Agriculture and Agri-Food Canada (equivalent to the U.S. Department of Agriculture) are working on a project in which a WSN of motes [4] is used to measure air temperature on a 50-acre vineyard [5] to enhance the growth of grapes and the quality of the wine produced.

Despite their potential applications, such networks have particular features imposed by resource restrictions, such as low computational power, reduced bandwidth and specially limited power source. Current research efforts have followed different lines such as network establishment [6,7], data dissemination [8,

* This work has been partially supported by CNPq grant number 55.2111/2002-3.

9], network and resource management [10,11,12,13]. Therefore, tools to assist designers with the development of management applications in such networks are very useful.

Usually, WSNs are composed of *source* and *sink* nodes [8,9]. Sources are data generators that detect events and provide observations or measurements of a physical phenomena. Sinks are designed to receive data sent by sources. Therefore, such nodes can monitor and act in the network performing some management function. Besides, sinks can act as gateways between the WSN and a infrastructured network. Thus, sinks may also provide an interface to the user allowing a manager to decide and act based on the data provided. This interface can be textual or multimedia becoming a useful tool to network managers.

In this paper, we use a precision agriculture example as our guiding application. In this scenario, an orchard or plantation can be equipped with several sensors gathering different types of data, such as soil temperature, humidity and acidity or alkalinity. A walking employee can receive data from those sensors through a wireless network interface into a portable device that acts as a sink node as depicted in Figure 1. The application in the portable device can be implemented in Java running on a portable platform such as J2ME, or SuperWaba. In this case, a multimedia interface, with graphics, animations and audio/video stream capability could be provided to exhibit data collected from different sensors and to support the network monitoring and management.

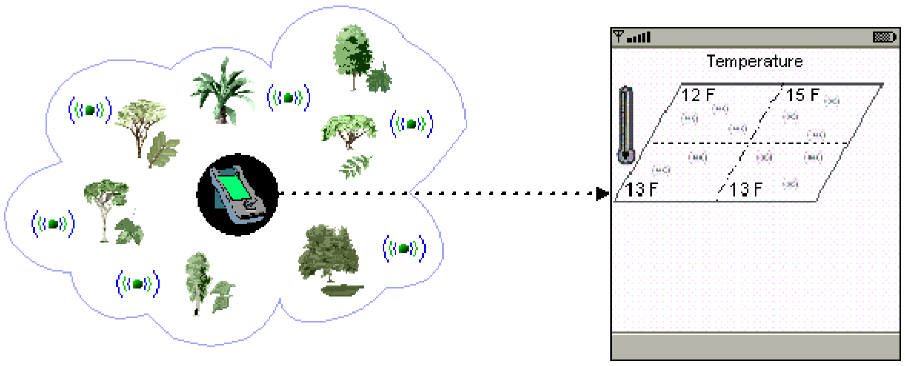


Fig. 1. A WSN example.

The application described above is usually designed to solve a specific issue, not considering a more general model, in which we could have, for instance, code reuse. In this work we present a tool called BeanWatcher that aims the code reuse proposing a standardization to the development of such applications. BeanWatcher allows the development of management applications in different programming languages such as Java, C/C++ and Embedded C. Our tool generates a management application for WSNs in a semi-automatic fashion. These applications are intended to be run on portable and mobile devices acting as a sink node monitoring raw sensory data and multimedia data streams.

Some commercial tools like LabView[1] and HP VEE[2] were designed to develop applications to monitor and act on instruments. However, those tools are proprietary solutions hardly integrated with other tools and languages. Furthermore, those tools do not allow the development of applications to portables devices neither monitoring applications for WSNs. Other related tool is the PECOS Component Environment[3] that allows the development of monitoring and actuation applications. In this paper, we show how it can be used to develop applications running in portable devices.

This paper is organized as follows. Section 2 presents some challenges in the design of multimedia management applications for WSNs. In Section 3 we present the component model used in BeanWatcher. Section 4 presents the BeanWatcher architecture and Section 5 shows how we can use BeanWatcher to develop a simple monitoring application. Finally, Section 6 presents our conclusions and future work.

2 BeanWatcher Challenges

In the following, we discuss the challenges that we must address when developing a multimedia management application for WSNs.

2.1 Multimedia Management for WSNs

Multimedia management faces new challenges in WSNs concerned with provision of scalable quality of service (QoS) through the management of metrics, such as coverage [14,15,16], exposure [17,18], energy consumption [12,13], and application specific metrics (e.g., for target detection, miss detection and false detection ratios). Due to the ad-hoc nature of WSNs, which might be deployed in hostile environments with fairly unpredictable conditions, management must be scalable, self-configurable and adaptive to handle such challenges. A classic approach is the data-centric design of WSNs [8,9], which aims the integration of application-level and network-level operations to provide power-efficient solutions.

In addition, WSNs can be composed of unrelated sensors that measure different physical properties (e.g., temperature, pressure, image, video and audio streams) that are semantically and/or structurally distinct. In such networks, incommensurate data can be understood as different types of media. In this case, multimedia management encompasses resources, theories, tools and techniques to manipulate data provided by different sources (multiple medias) with the goal to extract relevant information.

[1] National Instruments – Labview 6.1. http://www.ni.com/

[2] Agilent – VEE Onelab 6.1. http://www.agilent.com/

[3] PECOS Composition Environment. http://www.pecos-project.org/software.html

From this perspective, data fusion [19,20,21] should be used by multimedia management to combine different medias (e.g., video streams and temperature) to obtain relevant information (e.g., fire detection) or to improve the quality of the data provided (e.g., noise reduction). Therefore, fusion of multimedia data is equivalent to cooperative fusion proposed by Durrant-Whyte [22] where different nodes contribute with different data (e.g., video and audio streams) to accomplish a common task (e.g., intrusion detection). In addition, simple data aggregation schemes [11,23,8] can be used to summarize, organize and retrieve data in a power-efficient manner.

Figure 2 depicts some possible components in a multimedia management application for WSNs. The Stream Organizer receives multiple data streams through the same communication channel, then it organizes and directs them to the appropriate processing module. Three levels of processing are identified. The first level is the Stream Processing, which can perform low level data fusion. The second level is represented by the Feature Extractor which fuses several data streams to obtain relevant features from the environment. The last level is represented by the Decision Making where action plans are formulated in response to a identified situation. According to the application requirements or computational restrictions, the application can encompass only the desired processing levels (stream processing, feature extraction or decision making). This is also depicted in Figure 2 where the user interface is connected to all processing levels. In addition, the user can set properties and parameter related to the components responsible for each level of processing.

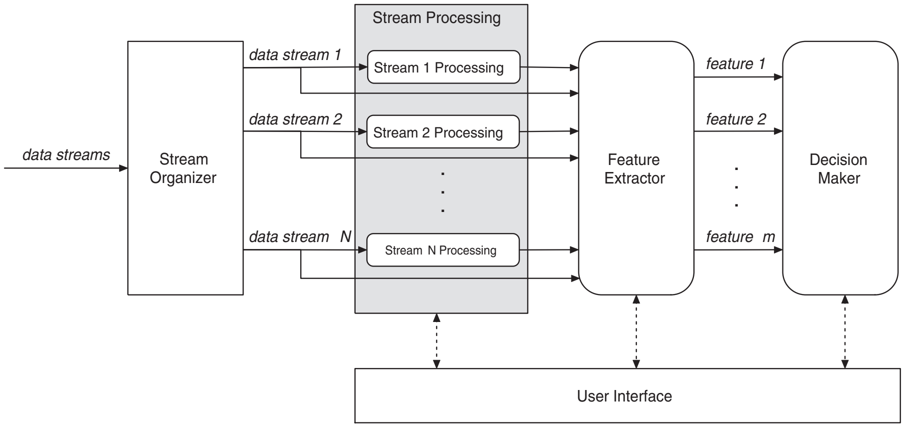


Fig. 2. Possible application involving multimedia management.

Considering our precision agriculture example with different data streams (soil temperature, acidity and humidity), which can be received in any order through the same communication channel. Thus, we need a component to organize the data streams properly. Further processing can be executed on these streams, such as data fusion algorithms like Kalman Filter [24], Particle Filter [25] or the Marzullo function [26] to obtain more accurate values. If desired, these data can be fused again to extract characteristics, like *the soil conditions that are favorable to cultivation of "X"* or *whether the soil is dry*. Yet further data fusion can be applied to obtain decisions, such as *"X" should be harvested in one week* or *irrigation required* or *freezing threat is eminent*. Multimedia fusion (concerned with feature extraction and decision making) is strongly coupled to the application and the type of data provided by sensors. Thus, the reusability of a multimedia fusion component might be limited to a specific domain of application and sensors. In addition, the development of such applications, and a tool to generate them, is still subject to the particularities of WSNs.

2.2 Attending WSN Requirements

Traditionally, system design can be organized in a Logical Layer Architecture (LLA) were the system is divided in different abstraction levels ranging from a physical layer, that deals with computing devices, to an application layer, that deals with business requirements. Usually, the LLA model is used in a bottom-up approach. However, WSNs are application-driven, and as discussed in [10], a top-down approach is preferable since once the business issues are understood, the requirements of lower levels become clearer.

Clearly, different applications tend to present distinct features and restrictions. Thus, it is not feasible to provide a unique Application Programming Interface (API) that is self-contained. Instead, we should think of providing small software components that represent elementary functionalities useful for various applications.

Regarding the applications illustrated in Figure 2 and using the top-down approach, we consider first the elements of the user interface that are related to multimedia management applications for WSNs. As we consider physical measurements acquired by sensors, it is reasonable to provide visual components appropriate to display common measurements, such as, temperature, pressure, acidity and humidity. Thus, BeanWatcher provides some visual components (e.g., thermometer, speedometer, gauge, and valued maps) to cover different types of sensory data.

The next steps should try to identify features in data streams. As mentioned before, this task depends on the application objectives, and, thus, the tool does not provide special components for them, since its reusability is restricted. For stream processing, there are some popular fusion algorithms that can be applied such as Kalman Filter, Particle Filter or Marzullo function. The current version of the tool provides two components that implement the Kalman Filter and the Marzullo function.

At the lower levels we need a component to receive data streams and to properly organize them. To get data from sensors we implement a communication component that receives and sends data without worrying about their semantics. Again, as we might have different types of sensors, we cannot provide a unique stream organizer. Besides, same sensors (data streams) can be semantically distinct in two different applications.

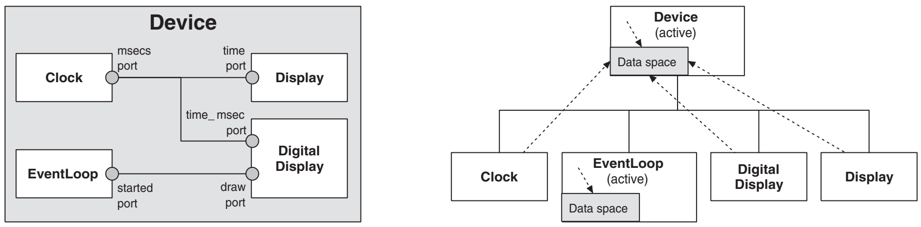
In summary, our API comprises elements for visual display, low level data fusion algorithms and communication operations. We chose to use a component model to implement this API and to generate the applications using our tool. The adoption of a component model allows us to take advantage of code reusability.

3 Component Model

BeanWatcher adopts the PECOS component model proposed by Genssler et al. [27], and provides a communication component to allow the monitoring of remote applications. In this section, we briefly describe PECOS and how it is used in BeanWatcher.

3.1 PECOS Component Model

The PECOS component model aims the design of embedded systems, more specifically field devices [27], which are executed directly by the instruments. PECOS is divided into two sub-models: structural and execution. Structural sub-model defines the entities included in the model, their features and properties. The execution sub-model defines the semantics of the components execution. An example of this model for a clock application is depicted in Figure 3, which is further discussed.



(a) Structural sub-model. (b) Execution sub-model.

Fig. 3. PECOS component model for a clock application.

Structural Sub-model. There are three main entities in PECOS structural sub-model: components, ports and connectors. Each component has a semantics and a well-defined behavior. Components form the model kernel and are used to organize both data and computation of the generated application. In the example shown, the components are *device*, *clock*, *display*, *eventloop* and *digital display*.

Ports provide an interaction mechanism among components. Output ports are connected to input ports through connectors as illustrated in Figure 3a, that shows the output ports *msecs* and *started*, and input ports *time*, *time_milsecs*, and *draw*. Connectors describe a data sharing a relationship between two ports and are represented by lines connecting them.

Components in PECOS can be primitive and composed. A primitive component can be passive, active or an event. Passive components cannot control their execution, and are used as part of the behavior of another component being executed synchronously. In Figure 3a, the passive components are *clock*, *display* and *digital display*. Active components control their execution, which is triggered by a system request. In Figure 3a, the active components are *device* and *eventloop*. Event components are similar to active components, but their executions are triggered by an event. A composite component is built using connected sub-components, but their internal sub-components are not visible to the user. In addition, a composite component must define a scheduling with the exact execution order of its sub-components. In Figure 3a, the composite component is the *device*.

Ports can be input, output or input/output ports. Input ports just receive data from other components. Output ports just send data to other components. Input/Output ports are bi-directional receiving and sending data from and to other components.

PECOS also defines properties and a parent component. Properties are component's meta-data such as memory usage or execution time. The structure of a component generated by the model is always hierarchical where the top component is always a composite component (parent).

Execution Sub-model. PECOS provides a sub-model for the execution of applications, that shows how data are synchronized among components running in different threads and describes their semantics.

Problems of data synchronism may happen in PECOS. For instance, suppose there are two active components connected to each other through a port. Both can read and write data simultaneously by different operations. To solve this problem, active and event components have a private data space where they can update unconditionally and periodically a private data that can be synchronized with a parent component. In Figure 3b we can see the private data space in the *device* and *eventloop* components.

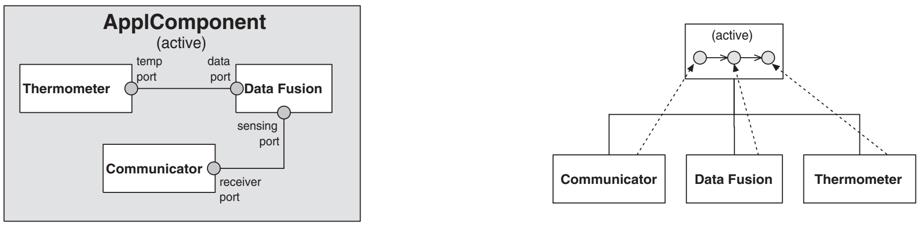
Due to this need for synchronization, active and event components have two possible behaviors: execution and synchronization. The execution behavior defines the actions performed when the component is executed. The synchronization behavior specifies how the private data space is synchronized with the parent component (arrows in Figure 3b). The execution semantics obeys the following simple rules:

– Execution behavior of a passive component is executed by a thread of its parent component;
– Synchronization behavior is executed by a thread of its parent component;

– Active and event components execute their sub-components using a control thread;
– Each component has a scheduler for its children.

3.2 PECOS in BeanWatcher

BeanWatcher adopts PECOS as its component model adding a communication component to support remote monitoring applications for wireless sensor networks. As an example, consider a WSN and a temperature application with three components: one to present the temperature, one to perform data fusion and one for communication, as depicted in Figure 4a.



(a) Structural sub-model. (b) Execution sub-model.

Fig. 4. Using PECOS in BeanWatcher.

In the structural sub-model of BeanWatcher, every component that monitors data provided by the WSN (e.g., a thermometer) is an active component since it just receives data that is presented to the user. Also the *ApplComponent* is always an active component. Components used as alarm indicators are event components. Internal components (used by a parent component) are passive. In Figure 4a, the *Data Fusion* and *Communicator* components are passive. In addition, we added a *show* functionality to the components because the generated remote applications show raw or pre-processed data from the WSN.

The execution sub-model of BeanWatcher adopted a pipeline execution, i.e., only one component can be executed at a time. This is illustrated in Figure 4b where *ApplComponent* executes sequentially its sub-components.

BeanWatcher gives at least two benefits to the user by adopting PECOS. First, the component model allows the code reuse. Second, PECOS specifies that the interaction of two components must be done through input and output ports; this interaction simplifies the understating of the application that is constructed by connecting input ports to output ports.

4 BeanWatcher Architecture

Figure 5 shows the BeanWatcher architecture that has three different modules: repository, processing and presentation.

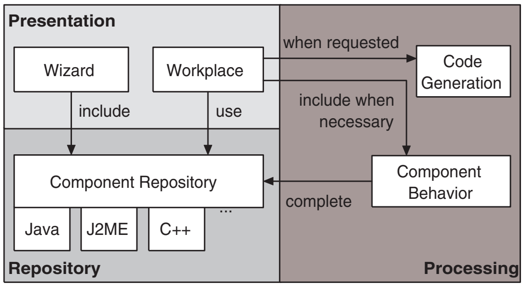


Fig. 5. BeanWatcher architecture.

The repository includes all components generated by a wizard that can be used to develop the application. It is important to mention that when a new target language is added to the tool, the components currently in the repository are automatically replicate to that target language so the user does not have to generate them again through the wizard. The repository uses the component model discussed in Section 3.2.

The presentation module specifies the interface provided by BeanWatcher to the user and it is composed of the workplace and the wizard. The workplace is the application building area and is divided into component edition and code edition. The component edition allows the user to choose and place the components according to the application requirements, adding a connector when two or more components demand it. The code edition is used to implement the behavior and show the functionality of the application being created. Furthermore, it allows the re-implementation of the components in the application.

The wizard enables the creation of new components extending the repository. To create a new component it is necessary to label the component and specify its ports. However, the behavior and functionalities must be implemented in the code edition.

The processing module performs the code generation and updates the components repository with new behavior implementations. The code generation obeys the application built by the user on the workplace. To have the complete application, the behavior of the composite component must be implemented.

In summary, the architecture depicted in Figure 5 presents some benefits to the user. First, it generates all the repetitive code to the user through the *code generation* unit. Second, as we mentioned in Section 2, many applications require unique features, so new components can be created through the *wizard* to attend unpredicted requirements; and the behavior of new or current components can be (re)implemented in the *workplace*. Third, it allows code reuse: once components are implemented the user can make it available in the *component repository* for future applications. Finally, the *component repository* allows the development of application in several target languages, such as Java, J2ME and C++.

5 Application Development Using BeanWatcher

In this section, we present an example using BeanWatcher.

5.1 The Three-Step Generation Process

Every application generated by BeanWatcher has a composite component called *ApplComponent*, i.e., every component added to the workplace will be a sub-component of *ApplComponent*. To build an application, the user must go through the three steps: building, implementation and generation (Figure 6).

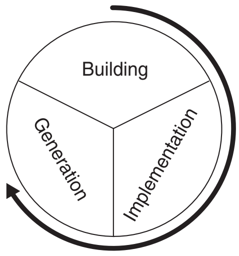


Fig. 6. The three steps to create a BeanWatcher application.

In the building phase, the user chooses one or more components to be added to the component edition area in the workplace. Thus, the added component becomes a sub-component of the *ApplComponent*. If necessary, a connector can be added to allow interactions among components.

Once the application is built, it is necessary to implement the behavior and the visualization of the *ApplComponent*. This is done in the code edition area in the workplace. All the functionalities and features of this component are generated automatically and are chosen by the user (e.g., implementation language). At this point the user can modify the behavior of the sub-components selecting the desired sub-component and editing it in the code edition area.

When the user saves the application, the code of each sub-component in the component edition area, the *ApplComponent*, and a default presentation code of BeanWatcher are automatically generated, including repetitive code. After that, the user can compile and run the generated application.

5.2 Developing an Application

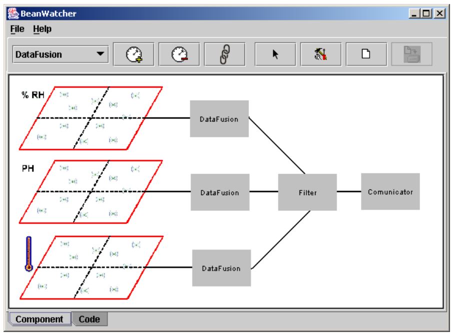
The application presented in this work is for precision agriculture, where a WSN is used to monitor the soil conditions of a orchard of strawberries. Sensors will be used to capture the soil temperature, humidity and acidity of the orchard. For the sake of simplicity, our application will be restricted to stream processing using the Kalman Filter, and no feature extraction nor decision making will be performed by the application (see Figure 2).

Initially, we chose the components from the menu (temperature map, data fusion and communicator). The temperature map is an active component that shows the temperature provided by the sensors and represents the area delimited by the sensors and is divided into four quadrants that show the respective fused values; if the user wants the quadrants can be zoomed in. The data fusion is a passive component based in the Kalman Filter.

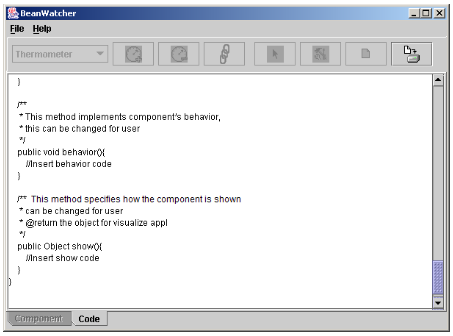
The same procedure is repeated using the humidity and acidity maps that will show the other fused measures.

The communicator must be added to get the data streams and it is a passive component that allows the communication between *sink* and *sources*. Finally, a filter component is added to organize the received streams (this component is implemented by the designer).

After the addition of the components above, it is necessary to connect them. *Connector* allows the data forwarding from the communicator to the filter, and then to data fusion, and so on. The application available in the component edition area is shown in Figure 7a.



(a) Workplace.

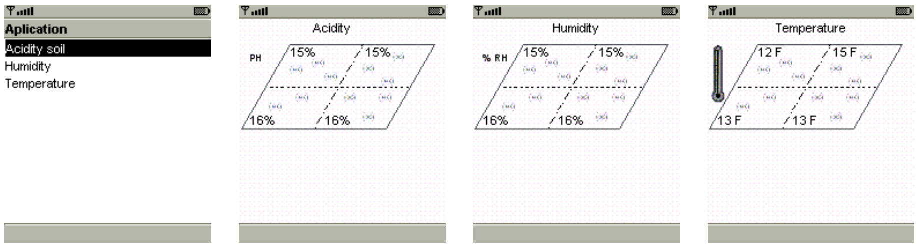


```
 }

 /**
  * This method implements component's behavior,
  * this can be changed for user
  */
 public void behavior(){
    //Insert behavior code
 }

 /**  This method specifies how the component is shown
  * can be changed for user
  * @return the object for visualize appl
  */
 public Object show(){
    //Insert show code
 }
}
```

(b) Code edition area.

Fig. 7. Development environment of BeanWatcher.

Once the application is built, the next step is the implementation. For the filter component, we considered that packets sent by sensors indicate the geographic position, the type of data (temperature, acidity or humidity) and the associated measurements. Thus, the filter only directs the data to the appropriate output port. Also, the behavior and *show* of the *ApplComponent* need to be implemented. This is done through the code edition area shown in Figure 7b. Finally, after saving the project we have the application ready. This example was executed in a simulator[4] and can be visualized in Figure 8. Note that we can add different graphical elements to the *show* method improving the presentation of the application.



(a) Menu. (b) Acidity. (c) Humidity. (d) Temperature.

Fig. 8. Execution of the application.

The application depicted in Figure 8 is presented as a menu where the user selects the type of data to be collected. All the three measurements are presented in a map, which is divided into four quadrants. The values represent the fusion of the measurements provided by the sensors at each quadrant. This approach is used for temperature, humidity and acidity streams.

6 Conclusions and Future Work

BeanWatcher is a powerful and flexible tool that adopts a component model aiming the development of management and monitoring applications for WSNs. BeanWatcher was designed to allow users with limited programming experience to develop a wide variety of monitoring applications. In addition, it supports several target languages.

Currently we are working on an extension module that uses PECOS to develop the applications that are executed into the sensor nodes so we can develop applications to both *sink* and *sources*. Thus, once data fusion and management components are added to the BeanWatcher repository, they can be used to build

[4] The simulator is included in the J2ME Wireless Toolkit freely available at http://java.sun.com/j2me/.

applications to *sink* and *sources* nodes so data fusion and management can be done in-network.

Finally, another module is being developed to generate applications based on rules, i.e., if the *sink* node is able to receive different sensory data, the application will show such data according to a set of rules so the user do not need to interact with the application.

As a future work we plan to implement a data fusion component to combine different types of media, such as video and audio streams to accomplish a more important task such as intrusion detection in a closed environment. However, as stated in Section 2, the reusability of such components is restricted to specific application domains and sensors.

References

1. Estrin, D., Girod, L., Pottie, G., Srivastava, M.: Instrumenting the world with wireless sensor networks. In: International Conference on Acoustics, Speech, and Signal Processing, Salt Lake City, USA (2001)
2. Pottie, G.J., Kaiser, W.J.: Wireless integrated network sensors. Communications of the ACM **43** (2000) 51–58
3. Estrin, D., Govindan, R., Heidemann, J., Kumar, S.: Next century challenges: Scalable coordination in sensor networks. In: Proceedings of the Fifth Annual International Conference on Mobile Computing and Networks (MobiCom'99), Seattle, Washington, USA, ACM Press (1999)
4. Hill, J., Culler, D.: Mica: A wireless platform for deeply embedded networks. IEEE Micro (2002) 12–24
5. Baard, M.: Wired news: Making wines finer with wireless. [online] available: http://www.wired.com/news/wireless/0,1382,58312,00.html, access: March 2003 (2003)
6. Schurgers, C., Tsiatsis, V., Ganeriwal, S., Srivastava, M.B.: Topology management for sensor networks: Exploiting latency and density. In: 2002 ACM Symposium on Mobile Ad Hoc Networking & Computing (MobiHoc'02), Lausanne, Switzerland (2002)
7. Chen, B., Jamieson, K., Balakrishnan, H., Morris, R.: Span: An energy-efficient coordination algorithm for topology maintenance in ad hoc wireless networks. Wireless Networks **8** (2002) 481–494
8. Krishanamachari, B., Estrin, D., Wicker, S.: The impact of data aggregation in wireless sensor networks. In: Proceedings of the International Workshop of Distributed Event Based Systems (DEBS), Vienna, Austria (2002)
9. Intanagonwiwat, C., Govindan, R., Estrin, D.: Directed diffusion: A scalable and robust communication paradigm for sensor networks. In: Proceedings of the 6th ACM International Conference on Mobile Computing and Networking (MobiCom'00), Boston, MA, USA, ACM Press (2000) 56–67
10. Ruiz, L.B., Nogueira, J.M., Loureiro, A.A.F.: Manna: A management architecture for wireless sensor networks. IEEE Commmunications Magazine **41** (2003) 116–125
11. Zhao, Y.J., Govindan, R., Estrin, D.: Computing aggregates for monitoring wireless sensor networks. In: Proceedings of the 1st IEEE International Workshop on Sensor Network Protocols and Applications (SNPA'03), Anchorage, AK, USA (2003)

12. Zhao, J., Govindan, R., Estrin, D.: Residual energy scans for monitoring wireless sensor networks. In: IEEE Wireless Communications and Networking Conference (WCNC'02), Orlando, FL, USA (2002)
13. Mini, R.A.F., Nath, B., Loureiro, A.A.F.: A probabilistic approach to predict the energy consumption in wireless sensor networks. In: IV Workshop de Comunicação sem Fio e Computação Móvel, São Paulo, SP, Brazil (2002)
14. Tian, D., Georganas, N.D.: A coverage-preserving node scheduling scheme for large wireless sensor networks. In: Proceedings of the 1st ACM International Workshop on Wireless Sensor Networks and Applications, Atlanta, Georgia, USA, ACM Press (2002) 32–41
15. Chakrabarty, K., Iyengar, S.S., Qi, H., Cho, E.: Grid coverage for surveillance and target location in distributed sensor networks. IEEE Transactions on Computers **51** (2002) 1448–1453
16. Meguerdichian, S., Koushanfar, F., Potkonjak, M., Srivastava, M.: Coverage problems in wireless ad-hoc sensor networks. In: Proceedings of IEEE Infocom 2001. Volume 3., Anchorage, AK, USA (2001) 1380–1387
17. Megerian, S., Koushanfar, F., Qu, G., Veltri, G., Potkonjak, M.: Exposure in wireless sensor networks: Theory and practical solutions. Wireless Networks **8** (2002) 443–454
18. Meguerdichian, S., Slijepcevic, S., Karayan, V., Potkonjak, M.: Localized algorithms in wireless ad-hoc networks: Location discovery and sensor exposure. In: Proceedings of the 2001 ACM International Symposium on Mobile Ad Hoc Networking & Computing, Long Beach, CA, USA, ACM Press (2001) 106–116
19. Luo, R.C., Yih, C.C., Su, K.L.: Multisensor fusion and integration: Approaches, applications, and future research directions. IEEE Sensors Journal **2** (2002) 107–119
20. Brooks, R.R., Iyengar, S.S.: Multi-Sensor Fusion: Fundamentals and Applications. Prentice Hall, New Jersey, USA (1998)
21. Hall, D.L.: Mathematical Techniques in Multisensor Data Fusion. Artech House, Norwood, Massachusetts, USA (1992)
22. Durrant-Whyte, H.F.: Sensor models and multisensor integration. International Journal of Robotics Research **7** (1988) 97–113
23. Dasgupta, K., Kalpakis, K., Namjoshi, P.: Improving the lifetime of sensor networks via intelligent selection of data aggregation trees. In: Proceedings of the Communication Networks and Distributed Systems Modeling and Simulation Conference, Orlando, FL, USA (2003)
24. Brown, R.G., Hwang, P.Y.: Introduction to Random Signals and Applied Kalman Filtering. 2nd edn. John Wiley & Sons, New York, NY, USA (1992)
25. Arulampalam, S., Maskell, S., Gordon, N., Clapp, T.: A tutorial on particle filters for on-line non-linear/non-gaussian bayesian tracking. IEEE Transactions on Signal Processing **50** (2002) 174–188
26. Marzullo, K.: Tolerating failures of continuous-valued sensors. ACM Transactions on Computer Systems (TOCS) **8** (1990) 284–304
27. Genssler, T., Christoph, A., Schulz, B., Winter, M., Stich, C.M.., Zeidler, C., M ller, P., Stelter, A., Nierstrasz, O., Ducasse, S., Arevalo, G., Wuyts, R., Liang, P., Schonhage, B., van den Born, R.: Pecos in a nutshell. Technical report, The Pecos Consortium (2002)

Managing Services Quality through Admission Control and Active Monitoring

Solange Lima, Paulo Carvalho, Alexandre Santos, and Vasco Freitas

Universidade do Minho, Dept. de Informática, 4710-057 Braga, Portugal
{solange,paulo,alex,vf}@uminho.pt

Abstract. We propose a lightweight traffic admission control scheme based on on-line monitoring which ensures multimedia services quality both intra-domain and end-to-end. The AC strategy is distributed, service-oriented and allows to control QoS and SLS without adding complexity to the network core. For each service class, AC decisions are driven by rate-based SLS control rules and QoS parameters control rules, defined and parameterized according to each service characteristics. These rules are essentially based on systematic on-line measurements of relevant QoS and performance parameters. Thus, from a practical perspective, we discuss and evaluate methodologies and mechanisms for parameter estimation. The AC criteria is evaluated as regards its ability to ensure service commitments while achieving high network utilization. The results show that the proposed model provides a good compromise between simplicity, service level guarantee and network usage, even for services with strict QoS requirements.

1 Introduction

The deployment of multimedia services in the Internet has been fostering the adoption of QoS models and related traffic control mechanisms in order to handle different applications QoS requirements while using network resources efficiently. Class of service (CoS) networks, such as Diffserv, are a step forward in pursuing this objective, where flows with similar characteristics and service requirements are aggregated in the same class. Controlling the admission of flows sharing a class allows to support new traffic flows conveniently without compromising existing QoS commitments. An admission control (AC) strategy should consider three vectors: (i) assurance level; (ii) control complexity; and (iii) network resources usage. Overprovisioning is currently the most common way to provide QoS guarantees in network backbones. Although for some ISPs overprovisioning is an attainable solution, in general, it is either not available or a solution too expensive. In our opinion, some degree of overprovisioning is recommended so that the AC process can be relaxed and simplified. The level of service guarantee to be provided is closely related to the complexity of the underlying traffic control strategy. In fact, either using centralized or decentralized AC approaches, the provision of guaranteed services, e.g. for hard real-time traffic, involves controlling the state and load of traffic aggregates in the core nodes [1,2,3]. These solutions tend to require significant network state information and, in many cases, changes in all network nodes [3]. Furthermore, as they are closely tied to network topology and routing, their complexity increases with the network dynamics. The provision of predictive services, e.g. for soft real-time traffic, allows a flexible AC, reducing AC control information and overhead. In this context, measurement-based AC (MBAC) solutions involving all network nodes [4], or only edge nodes [5,6,7] (EMBAC) have deserved special attention. However, while leading to more efficient resource utilization, QoS degradation may occur. The need to control elastic traffic, for more efficient network utilization, has also been discussed and implicit AC strategies have been defined [8].

A further step towards a generic and light AC model oriented to multi-service networks, able to operate both intra-domain and end-to-end has been proposed in [9]. The present work details the main components of this model (section 2) focusing on its implementation (section 3). The tuning and performance of the AC model is evaluated as regards the underlying measurement process and effectiveness of AC rules (section 4).

2 The Admission Control Model

The AC model proposed in [9] considers: (i) the control of distinct network services and assurance levels, to handle different application QoS requirements and traffic profiles; (ii) the operation intra-domain and end-to-end, controlling both the QoS levels in a domain and the sharing of the existing SLS between domains to fulfill end-to-end QoS requirements. As explained in [9] ingress routers perform explicit or implicit AC depending on the application type and corresponding traffic class, while egress routers perform edge-to-edge on-line QoS monitoring and SLS control. QoS Monitoring measures relevant parameters for each service (service metrics) reflecting service availability from each ingress. SLS Control monitors the usage of downstream SLSs at each egress to ensure that traffic to other domains does not exceed the negotiated profiles and packet drop will not occur due to an indiscriminate traffic conditioning (TC) process. QoS monitoring statistics, SLS utilization and associated parameters are then sent to the corresponding ingress routers to update an Ingress-Egress service matrix used for distributed AC and active service management.

Explicit and Implicit AC. Explicit flow AC is oriented to applications able to signal the network with their traffic profile and QoS objectives, e.g. streaming applications. In this case, the AC decision requires two initial verifications (see Fig. 1): (i) SLS Utilization Control checks if the downstream SLS can accommodate the traffic profile of the new flow; (ii) QoS Control checks if, for the corresponding egress node and service, the domain QoS metrics, the SLS QoS parameters agreed with the downstream domain and the previous measures (if any) fulfill the application QoS requirements.

Each AC decision is based on a service dependent AC equation and thresholds defined to ensure specific service guarantees. When a flow is accepted in the domain, the notification may be generated either locally (local admission) or remotely (end-to-end admission). The *end-to-end case* [9] is viewed as a repetitive and cumulative process of admission control and available service computation, performed at ingress nodes. At each domain the ingress node decides if a flow can be accepted, and if so the domain service metric values are added to the flow request to inform the downstream domain of the service available so far. Using the incoming and its own measures each domain performs AC. This solution leads to a generic AC model, which can be applied both to source and transit domains.

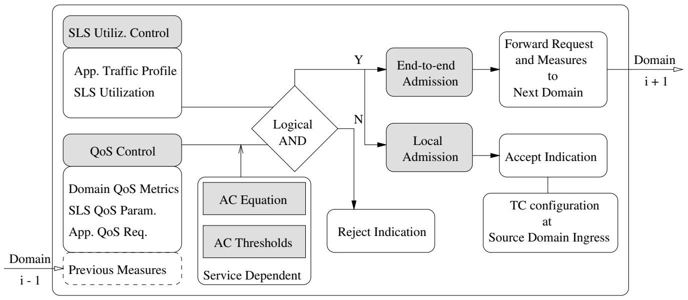


Fig. 1. Admission Control Criterion

Implicit flow AC, oriented to elastic applications which do not use signaling, use implicit detection of flows [8]. This type of AC, likely to be implemented only in the source domain, will be restricted to SLS information and QoS monitoring.

3 Model Implementation

3.1 Definition of Service Classes and SLS

A differentiated services architecture needs to be supported by an adequate traffic classification strategy [10,11]. As initial policy, we treat TCP and UDP traffic separately, being UDP traffic further divided according to the applications QoS requirements stringiness. As result, three initial service classes were defined (see Table 1). Service Class 1 (SC1), supported by EF PHB, provides a high QoS performance service guarantee and is oriented to streaming applications imposing hard real-time constraints. Due to the high priority treatment this class requires in each network node, which may starve low priority classes, the access to the corresponding service is tightly controlled. SC1 AC criterion will follow a conservative schema, with TC giving a severe treatment on excess traffic (see section 3.2). Service Class 2 (SC2), supported by AF PHB, provides a predictive type of service with low delay, low loss and minimum bandwidth guarantee

Table 1. Definition of Service Classes

Serv.Class	Serv.Level	Traffic type	PHB	AC	Policing	Scheduling
SC1	guaranteed	UDP (hard RT)	EF	explicit & conserv.	drop excess	strict priority
SC2	predictive	UDP (soft RT)	AF	explicit & flexible	3 color marker	WRR
SC3	best-effort	TCP	BE	implicit & relaxed	3 color marker	WRR

and is oriented to streaming applications with soft real-time constrains. SC2 AC criterion will be less conservative, taking more advantage of statistical multiplexing. TC will act on non-conformable traffic using a three-color marker (TSW3CM). In a first set of experiments, only AF1x is considered. Service Class 3 (SC3) provides best-effort service to adaptive TCP applications. Detailed classification rules for TCP differentiation will be considered in the future taking the remaining AF classes. SC3 AC criterion will be implicit and relaxed. The service classes are implemented resorting to class-based queuing with priority weighted round-robin scheduling.

As regards SLS definition, both SLA template defined in [9] and QoS parameters upper bounds for common applications defined in [12] have been considered here to support SLS instantiation.

3.2 Admission Control Criterion

Establishing an admission criterion consists of defining the rules by which flows are accepted or rejected. Usually AC criteria are parameter-based, measurement-based or follow an hybrid scheme combining both. Parameter-based AC algorithms, oriented to flows requiring a guaranteed service, tend to be conservative leading to low utilization for bursty traffic. Measurement-based AC (MBAC) algorithms [4,5] are less conservative, taking advantage of statistical multiplexing of traffic to increase network utilization at an eventual cost in QoS degradation. In this way, MBAC is more suitable for flows requiring a predictive service.

In the proposed model, the AC criterion is essentially measurement-based[1] controlling both the QoS in the domain and the downstream SLS utilization. This leads to the specification of two types of rules: (i) rate-based SLS control rules; and (ii) QoS parameters control rules.

Rate-based SLS Control Rules. Let I be the set of ingress nodes, i.e. $I = \{I_1, I_2, ..., I_N\}$ and E the set of egress nodes in a domain, i.e. $E = \{E_1, E_2, ..., E_M\}$. For each egress $E_m \in E$ with $1 \leq m \leq M$, one or more SLSs can be in place, one per service type and per downstream domain. At this point, it is assumed a single mapping between a service class within the domain and a downstream SLS for the corresponding service type. As each SLS has a specified negotiated rate, a rate based Measure-Sum (MS) algorithm is applied to control SLS utilization. Thus, for each ingress $I_n \in I$ with $1 \leq n \leq N$, the equation used to verify if a new flow can be admitted takes both rate estimate and flow traffic description, , i.e.

$$\rho_s + r_j \leq \beta_s R_s \tag{1}$$

In (1) ρ_s is the current measured load or estimated rate of flows using SLS_s of $E_m \in E$ considering all the ingress-egress estimated rate for class i going through E_m, i.e. $\rho_s = \sum_{k=1}^{N} \rho_{i,k}$; r_j is the rate specified by the new flow j; β_s is the utilization target for the SLS (with $0 < \beta_s \leq 1$); and R_s is the rate defined in SLS_s.

QoS Parameters Control Rules. When controlling QoS levels in a domain, the QoS parameters under control and corresponding thresholds can vary depending on

[1] Some degree of overprovisioning is considered for stringent QoS classes so that simplicity and flexibility of MBAC can be useful to control these classes too.

each service class commitments, the statistical properties of the traffic and degree of overprovisioning. A new flow is accepted or rejected by checking the controlled parameters $P(i) = \{p_1, p_2, ..., p_p\}$ of class i against the corresponding pre-defined threshold t_p, i.e. $\{\forall p \in P(i) : p_p \leq t_p\}$, in which t_p can be affected by a safety margin to the QoS parameter bound. Tuning these limits, making them useful and realistic indicators of the overall QoS status is a fundamental aspect for AC, as shown in section 5.

Following the service definition provided in section 3.1, AC for SC1 and SC2 uses both type of rules defined above. For SC1, a more conservative criterion is taken, considering the worst-case scenario (flow peak rates, concurrent AC taking place at other ingress nodes and optimistic measures), larger safety margins and tighter thresholds. Table 2 summarizes the AC criteria illustrating the type of QoS parameters under control for SC1, SC2 and SC3, which are then defined in Table 3. Recall that, due to the nature of TCP traffic, where a flow has not a pre-defined rate, Equation (1) is applied as a threshold for the estimated rate. The estimation mechanisms for the parameters under control and the time granularity used in the estimation is discussed in the section 3.3.

From an *end-to-end* perspective, whenever a flow AC request specifies end-to-end QoS requirements, QoS parameter control rules need to be extended to accommodate an additional verification. For each specified parameter, this verification weights the corresponding QoS parameter estimate/bound in the domain, its negotiated value with the downstream domain and the cumulative value computed so far.

Table 2. Admission Control Criteria

	Flow Inputs		Network Inputs	SLS Util. Control		QoS Parameter Control	
Class	T.Desc.	QoS	Measures	Parameter	Method	Parameter	Method
SC1	peak rate	if any	load, IPTD, ipdv, IPLR	rate	MS	IPTD, ipdv, IPLR	thresh.
SC2	mean rate	if any	load, IPTD, IPLR	rate	MS	IPTD, IPLR	thresh.
SC3	n.a.	n.a.	load, IPLR	rate	thresh.	IPLR	thresh.

3.3 Online Monitoring

In our study, the objective of on-line monitoring is twofold: (i) it allows SLS auditing in the domain and (ii) it provides inputs for the AC decision module, which being measurement based, requires a realistic view of the network status. The systematic use of on-line monitoring for traffic load and QoS estimation, while allowing an active service management avoids the common per application intrusive traffic and the initial latency of EMBAC approaches. Furthermore, the effect of cross-traffic and other internally generated traffic (e.g. routing, management, multicast traffic) is implicitly considered.

The problematic of monitoring involves the definition of metrics, measurement methodologies and timing decisions. ITU-T work on QoS in IP networks and particularly IETF IPPM have defined a set of standard QoS and performance metrics and have proposed measuring methodologies for them [13,14]. Several tools useful for measuring the SLS metrics have also been developed and tested [12,15]. Taking these inputs into account, on-line monitoring implementation options are discussed below.

Table 3. Controlled QoS Parameters

Rate Parameters	
Throughput ρ (bps)	$\rho_i = \sum bits\_received_i / \Delta t_i$
Utilization U (%)	$U_i = \rho_i / C$
Delay Parameters (ms)	
IP Transfer Delay ($IPTD$)	$IPTD_i(pkt) = (t_E(pkt) - t_I(pkt)$
Mean IPTD ($\overline{IPTD}$)	$\overline{IPTD_i}(\Delta t_i) = \sum (t_E i - t_I i)/pkts\_received_i(\Delta t_i)$
Inst. Packet Delay Var. ($ipdv$)	$ipdv_i = (IPTD_{i,pkt_k}) - IPTD_{i,pkt_{k-1}})$
Mean ipdv ($\overline{ipdv}$)	$\overline{ipdv_i}(\Delta t_i) = \sum \lvert ipdv_i \rvert / pkts\_received_i(\Delta t_i)$
Maximum IPTD	$max(IPTD_i(pkt), \Delta t_i)$
Minimum IPTD	$min(IPTD_i(pkt), \Delta t_i)$
IPTD Alarm ($Alarm\_IPTD$)	$Alarm\_IPTD_i = \overline{IPTD_i}(\Delta t_i) - \overline{IPTD_i}$
Loss parameters (%)	
IP Loss Ratio (IPLR)	$IPLR_i = pkts\_lost_i / pkts\_sent_i$
Mean IPLR ($\overline{IPLR}$)	$\overline{IPLR_i}(\Delta t_i) = pkts\_received(\Delta t_i)/pkts\_sent_i(\Delta t_i)$

QoS and Performance Metrics. Several edge-to-edge QoS and performance parameters have been identified to be controlled at each egress node. They are specified for an given ingress-to-egress pair (I_n, E_m), class i and time interval Δt (see Table 3).

Measurement Methodology. A measurement methodology can be either passive, active or combinations thereof. Passive measurements are carried out on existing traffic and are particularly suitable for troubleshooting; active measurements inject extra traffic (probing) in the network for measurement purposes allowing to check QoS and SLS objectives in a more straightforward way. Probing brings an additional advantage when measuring edge-to-edge performance and QoS. As specific packets are injected in the network containing timestamping and sequencing data, delay and loss estimations are simplified. Obtaining these estimates combining link-by-link measures is not an efficient and easy solution. However, as probing is an intrusive process, its impact on the network load needs to be minimized. In the proposed model, in-band probing is used per class and not per application which is a clear advantage over other EMBAC approaches as overhead is reduced. For each class, the parameters in Table 3 are estimated and controlled, resorting to passive and active measurements. Comparing the outcome of both approaches allows to assess and tune the probing process (see section 5.1).

Parameters Estimation. Apart from the measurement methodology, there are several measurement mechanisms which can be used for parameter estimation. In particular, *Time-Window* (TW), *Point Sample* (PS) and *Exponential Averaging* (EA) mechanisms are commonly an option due to their simplicity [16,4]. Although these mechanisms are usually applied to a single node, we have applied them to edge-to-edge measurements. For *SLS utilization control*, the class traffic load is the estimated parameter, obtained resorting to the estimation mechanisms described above in order to assess which one more closely reflects the real network behavior (see section 5.2). In order to obtain a statistically meaningful number of samples, following [4], $T/S \geq 10$ with $S > 100 * L/C$, where S is the sampling period controlling the measurement sensitivity, T is the window size controlling the mechanism adaptability, L is the packet size (bits) and C the trans-

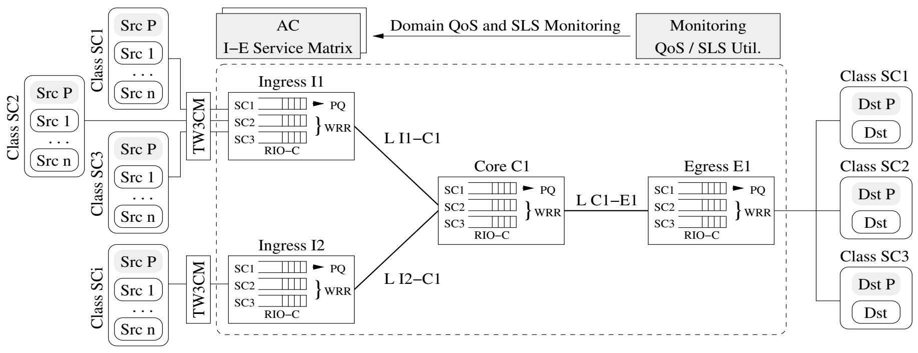


Fig. 2. Simulation topology

mission rate. As our estimates are edge-to-edge, dimensioning S considers the one-way delay instead of C. As in [16], for each sampling period S, independently of which estimation method is in use, the parameter average is evaluated. For *QoS control*, the QoS parameter average in Δt ($\Delta t = S$) is used as estimate.

4 Simulation Scenario

To test the proposed AC model in a multi-class domain, a simulation prototype was devised and set up based on NS-2 platform. The main objectives of the experiments are threefold. First, we intend to assess the active measurement methodology as a whole. Both probing patterns, probing periodicity and probing ability to capture each class behavior are studied. Second, a comparison of the estimation mechanisms TW, Avg_PS and EA is carried out in order to evaluate which one provides the closest estimate to reality, tuning S and T timing parameters. Third, the proposed AC criterion is evaluated as regards its ability to ensure service commitments are not violated, while assessing both network utilization and QoS safety margins.

Simulation topology. The simulation topology is illustrated in Figure 2. The network domain consists of ingress routers I_1, I_2, a core router C_1 and an edge router E_1. While I_1 multiplexes three types of sources, each type mapped to a different class, I_2 is used to inject concurrent traffic. This allows to evaluate concurrency in distributed AC and assess cross traffic impact. The domain internodal links capacity is 34Mbps, with a 15ms propagation delay. L_{C1-E1} link works as a bottleneck in this network topology. Access links have been configured so that intra-domain measurements are not affected. In each node, each class queue is 150 packets long. The scheduling discipline follows an hybrid Priority Queuing - Weighted Round Robin (PQ-WRR) and the active queue management mechanism is RIO-C. The PQ-WRR(2,1) discipline gives to the highest priority class (SC1) a strict priority treatment with a tight limit on a pre-defined rate (10% of the bottleneck link), whereas the remaining class queues are served with a 2 to 1 proportionality. At network entrance, each traffic class is policed using a TSW3CM.

Source models. Generically, three source models have been considered: Constant Bit Rate (CBR) sources, Exponential on-off (EXP) and Pareto on-off (PAR) sources. PAR

Table 4. Source Parameter Configuration

Class	Protocol	Src Type	Src Parameters	Inter. t	Hold. t
SC1	UDP	CBR_{SC1}	(r=100kbps, l=128B)	0.4-2s	60s
	UDP	EXP_{SC1}	(r=200kbps, l=128B, on=off=500ms)	0.4-2s	60s
	UDP	PAR_{SC1}	(r=200kbps, l=128B, on=off=500ms, α=1.5)	0.4-2s	60s
SC2	UDP	$CBR_{SC2,3}$	(r=0.5Mbps, l=512B)	0.4-2s	120s
and	UDP	$EXP_{SC2,3}$	(r=1Mbps, l=512B, on=off=500ms)	0.4-2s	120s
SC3	UDP	$PAR_{SC2,3}$	(r=1Mbps, l=512B, on=off=500ms,α=1.5)	0.4-2s	120s
SC3	TCP	FTP App.	(r=unspecified, l=512B)	0.4-2s	120s
Probing	UDP	CBR_P	(r=1.6kbps (2pkts/s), l=100B)	1 src	sim. dur.
	UDP	POI_P	(r=1.6kbps, l=100B, Poisson)	1 src	sim. dur.
	UDP	EXP_P	(r=3.2kbps, l=100B, on=off=250ms)	1 src	sim. dur.

sources with $1 < \alpha < 2$ under aggregation will allow to generate traffic exhibiting long-range dependence [10]. As this property has a significant impact on queuing behavior and on the nature of congestion leading to unexpected QoS degradation, larger safety margins may be needed. SC1 comprises UDP traffic with small to medium peak rate and packet sizes, as usually generated by some real-time streaming applications, such as VoIP. SC2 comprises UDP traffic with higher peak rate and packet sizes. Initial tests consider UDP traffic in SC3. The flow arrival process is Poisson with exponentially distributed interarrival and holding times. Table 4 details the parameters choice. As regards probing, three in-band source types have been defined. In [17], two packets per second according to a Poisson distribution are used as a probing scheme to assess loss and delay between any two network measurement points. Here, the adequacy of this pattern is tested and compared with both CBR and Exponential on-off probing. The use of EXP-like sources prevent possible synchronization among probing and other events in the IP network [17].

Service and AC Configuration. Table 5 illustrates the main parameters used to configure the AC rules, for controlling both SLS utilization and domain QoS levels. Three downstream SLSs have been considered, one per service class, with a negotiated rate (R_s) defined according to the traffic load share intended for the corresponding class in the domain. The MS algorithm that rules SLS utilization has specific utilization target (β_s) values depending on how conservative the AC decisions must be. For instance, a β_s=0.75 corresponds to impose a safety margin of 25% to absorb load fluctuations and optimistic measures. This value can be viewed as a degree of overprovisioning. The AC thresholds for QoS control in the domain are set taking into account the domain topology dimensioning, queuing and propagation delays, and perceived QoS upper bounds for common applications and services [12].

5 Simulation Results

In this section, the results are discussed as regards: (i) the probing process (ii) the estimation mechanism and (iii) the AC criteria. The results were obtained running multiple simulations of about 8 minutes, discarding results from an initial convergence period.

Table 5. Service Parameter Configuration

Service Class	SLS Rate R_s (% share)	Util. Target β_s	QoS Parameter	Threshold
SC1	3.4Mbps (10%)	0.75	IPTD,ipdv,IPLR	35ms,1ms,10^{-4}
SC2	17.0Mbps (50%)	0.90	IPTD,IPLR	50ms,10^{-3}
SC3	13.6Mbps (40%)	1.00	IPLR	10^{-1}

5.1 Evaluation of the Probing Process

Generically, the probing process is assessed as regards its ability to capture classes behavior realistically, and used for monitoring network status. Initial tests consider three distinct probing sources (CBR, POI, EXP) to measure the parameters defined in Table 3. For each class, the probing measuring outcome was cross-checked against the corresponding measures using the class real traffic.

Despite the traffic type in each class, the results obtained are consistent and similar for the probing sources considered. Our major findings are: (i) probing can be successfully used to evaluate both IPTD and mean IPTD. This is true both for capturing the shape and scale of these QoS metrics. Figure 3 shows an almost perfect match for the mean IPTD. This is valid even for probing rates as low as two packets per second. (ii) for the test conditions, probing is inappropriate for measuring both ipdv and IPLR, as Figure 3 shows. As ipdv is a consecutive packet measure, probing gaps lead to higher measures as consequence of queue occupancy variations. For IPLR, as loss is typically seen as a rare event and probing packets are marked as high priority (green) packets, the probing method is again inadequate. The exception to this behavior, occurs when traffic in a class suffers heavy loss (over 10%). This mis-behavior can be reduced resorting to higher probing rates and using red-marked probe traffic. However, the overhead introduced may be prohibitive, in particular, if it is performed in-band. Due to its particular rate characteristics probing rate cannot be directly compared to class throughput. For bandwidth monitoring, specific probing techniques must be used [18,19].

5.2 Evaluation of the Estimation Mechanism

In a first instance, the evaluation method consists of determining on how close an estimate is to the real traffic load. In this way, the rate estimation of each service class using the estimation mechanisms TW, Avg_PS and EA is compared (see Figure 4), taking Avg_PS estimates as reference. This is because Avg_PS represents the real aggregate mean rate in a pre-defined interval S.

Usually, TW leads clearly to over estimation of the metric and, in practice, it can be a very conservative method. In special, when the window T is reinitialized upon a new flow admission and for short flow interarrivals, the estimate increases steadily as the departure of flows is not taken into account. The ratio between flow duration and T is studied in detail in [4]. However, this method allows to consider in advance the weight the new admission might have. This also occurs with EA, where the estimation is artificially increased when a new flow is admitted. This estimation method is controlled by the weight parameter γ, where for $\gamma = 0.25$ a closer match is achieved (Fig. 4).

As far as AC is concerned, there are other aspects to consider: (i) the estimate is due to be used during a time interval T and (ii) the estimate needs to reflect, and somehow foresee, the network behavior trends. The tests on AC criteria show that Avg_PS allows to achieve high network utilization without service violations, for CBR traffic. However, for EXP and PAR traffic, all services have suffered disruption. EXP/PAR traffic fluctuations and a particularly low estimate results in over acceptance. When this happens, in the next estimation period, the AC rate and QoS control rules will stop the new flows' entrance, however, degradation occurs during the lifetime of the existing ones. This effect can

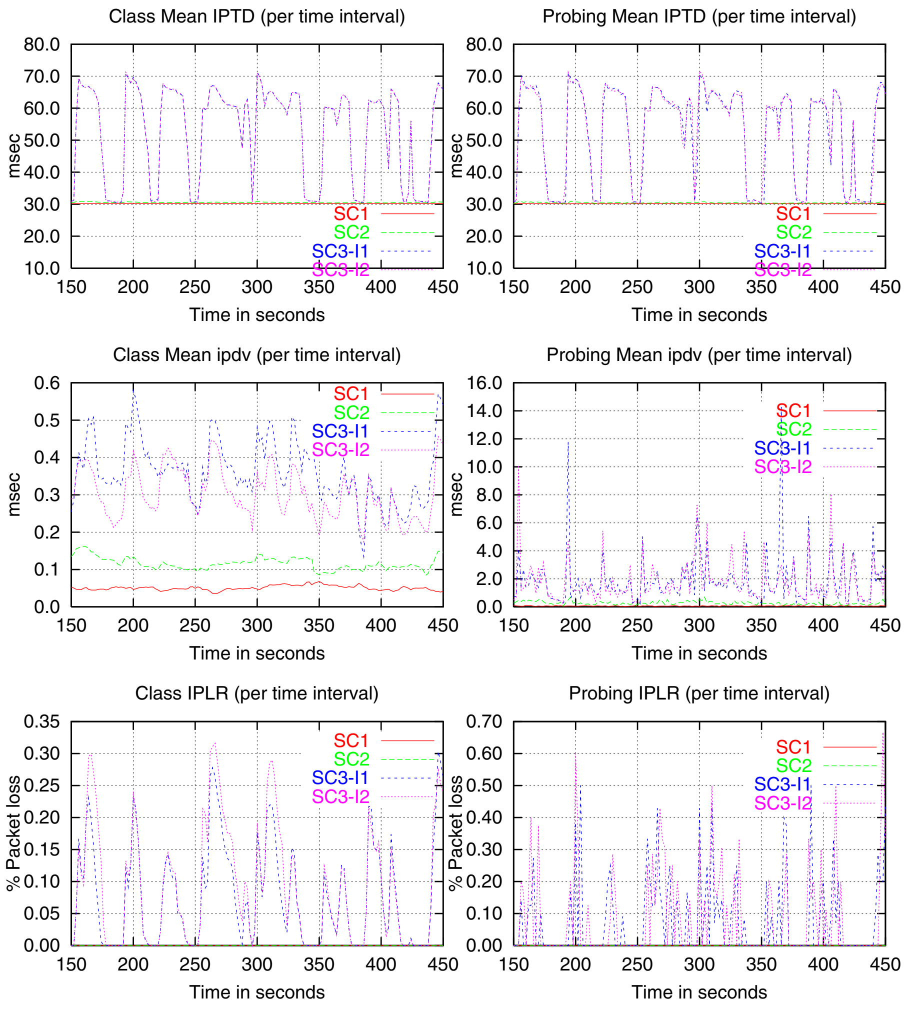


Fig. 3. Class (left) and Probing (right) Mean IPTD, Mean ipdv and IPLR (time interval of 2s).

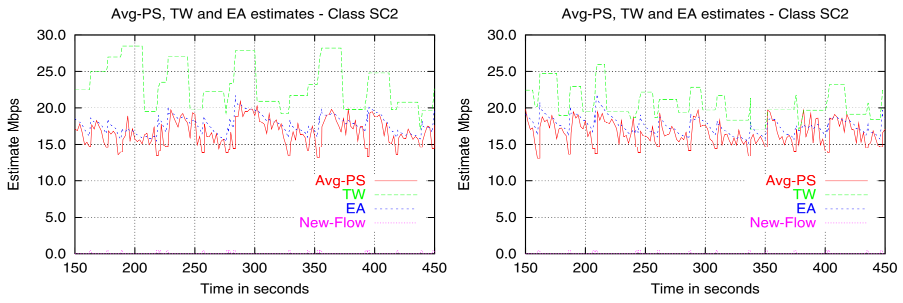


Fig. 4. Estimation mechanisms for (T=10; S=1; $\gamma = 0.25$; flow (EXP, 500kbps, i.a.t.=0.4s, h.t.=120s) (a) reseting T (b) without reseting T on flow admittance.

be reduced with TW and EA due to their initial accounting on the flow rate. In fact, an immediate increase on the estimate, reflecting the impact of the new flow will have, allows a more adaptive and conservative AC.

5.3 Evaluation of the AC Criteria

While the active monitoring process is being tuned, passive measurements are used to evaluate the proposed AC criterion so that results are not misleading. The Avg_PS mechanism (with $S = 2s$) which represents the real average values for the parameters under control is used. In these experiments, concurrent SC3 traffic is injected through ingress I2. As expressed in Table 5, SC1 traffic is blocked whether the sum of the rate estimate ρ_s and the flow's peak rate r_j is above 75% of the class share (see Equation 1), or any of the QoS controlled parameters exceed the pre-defined thresholds. For SC2, a safety margin of 10% (β_s=0.9), was defined and the flow mean rate is used instead. SC3 does not uses safety margin but controls IPLR. The tests are performed under high demanding conditions with a flow interarrival of 400ms.

Table 6 summarizes the results obtained for each class and each source type as regards: (i) the average of concurrent active flows; (ii) the percentage of packets exceeding the pre-defined delay bounds; (iii) the total loss ratio; (iv) the new utilization target proposal, for which no packet QoS violations occur and (v) the new average of concurrent active flows. The results obtained show that while for CBR traffic there is no loss and reduced delay violations, for EXP and PAR sources an increasing packet loss is clearly noticed. Although, the AC rules are effective in blocking new flows when QoS degradation or an excessive rate is sensed, the effect of previously accepted flows persists over the next intervals while they last. This over acceptance is caused by traffic fluctuations combined with the type of estimation mechanism used (without prevision). To minimize this, more conservative estimates or larger safety margins are needed. We have explored this last option for EXP traffic, where new safety margins leading to no packet QoS violations were established. Fig. 5 (b) shows that, under the new defined margins, the AC criteria achieves good network utilization. In this tuning process, we found that, in SC3, IPLR is a difficult parameter to control. Even when the total loss is below the defined

Table 6. AC Test Results

Class	Src Type	#act_flows	%pkts_viol:(IPTD;ipdv)	%IPLR	New Util.Target	new #act_flows
SC1	CBR_{SC1}	26.0	(0.41 ; 0.14)	0.0	0.55	19
	EXP_{SC1}	27.0	(0.33 ; 0.08)	2.2		
	PAR_{SC1}	26.5	(0.30 ; 0.08)	2.9		
SC2	CBR_{SC2}	31.5	(0.05 ; n.a)	0.0	0.75	27.5
	EXP_{SC2}	34.5	(0.07 ; n.a)	1.5		
	PAR_{SC2}	34.0	(0.08 ; n.a)	1.1		
SC3	CBR_{SC3}	16.0+16.0	(n.a. ; n.a)	0.7	0.80	13.5+14.5
	EXP_{SC3}	15.5+17.5	(n.a. ; n.a)	17.4		
	PAR_{SC3}	16.5+19.0	(n.a. ; n.a)	20.2		

upper bound, the Mean IPLR per interval may exceed the corresponding threshold. This may be due to the low weight in the scheduling discipline which serves the corresponding queue. The results also shows that SC1 can be particularly affected by the scheduling mechanism. While PQ is suitable and commonly used to handle in-profile high priority traffic, if the aggregate rate exceeds the maximum rate allowed by the scheduler, the class is severely punished. This is evident through an increase of IPTD and IPLR (see Fig. 5 (a)), which stems from a head-of-line blocking while waiting for the scheduling cycle to be completed. This stresses the need of having a tight control on this class and a wider safety margin.

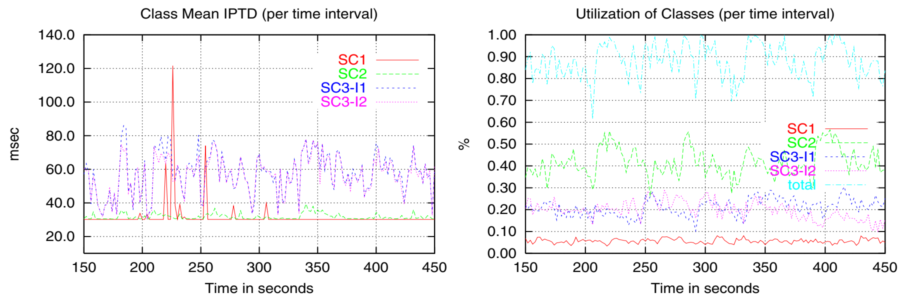


Fig. 5. (a) Mean IPTD (for EXP sources and initial utilization target) (b) Utilization for the new utilization target (considering the bottleneck capacity of 34Mbps)

6 Conclusions

In this paper we discussed and evaluated a distributed AC model for the management of multimedia services quality in class-based networks. We evaluate the two main components of the model: the monitoring process, which controls QoS and SLS parameters, providing inputs to AC; and the AC criteria which guide the AC decisions. Parameter estimation methodologies and mechanisms are compared and tuned. Both probing patterns and periodicity were assessed as regards its ability to capture the behavior of the different classes. The results show that probing is a good solution to measure IPTD, but did not capture ipdv and IPLR behavior properly. The evaluation of the proposed AC criteria as regards its ability to ensure service commitments shows that using proper AC rules and safety margins the simplicity and flexibility of this measurement-based AC approach can be successfully used to manage service quality.

References

1. B. Teitelbaum, S. Hares, L. Dunn, R. Neilson V. Narayan, and F. Reichmeyer. Internet2 QBone: building a testbed for differentiated services. *IEEE Network*, 13(5):8–16, September/October 1999.
2. Zhi-Li Zhang et al. Decoupling QoS Control from Core Routers: A Novel Bandwidth Broker Architecture for Scalable Support of Guaranteed Services. In *SIGCOMM'00*, 2000.
3. I. Stoica and Hui Zhang. Providing Guaranteed Services Without Per Flow Management. In *ACM SIGCOMM'99*, October 1999.
4. S. Jamin, S. Shenker, and P. B. Danzig. Comparison of measurement-based call admission control algorithms for controlled-load service. In *INFOCOM (3)*, pages 973–980, 1997.
5. L. Breslau et al. Endpoint Admission Control: Architectural Issues and Performance. In *ACM SIGCOMM'00*, 2000.
6. C. Cetinkaya, V. Kanodia, and E. Knightly. Scalable Services via Egress Admission Control. *IEEE Transactions on Multimedia*, 3(1):69–81, March 2001.
7. V. Elek, G. Karlsson, and R. Rnngren. Admission Control Based on End-to-End Measurements. In *IEEE INFOCOM'00*, 2000.
8. R. Mortier and I. Pratt and C. Clark and S. Crosby. Implicit Admission Control. *IEEE Journal on Selected Areas in Communications*, 18(12):2629–2639, December 2000.
9. S. Lima and P. Carvalho and A. Santos and V. Freitas. A Distributed Admission Control Model for CoS Networks using QoS and SLS Monitoring. In *ICC'2003 – IEEE International Conference on Communications*, May 2003.
10. S. Lima and M. Silva and P. Carvalho and A. Santos and V. Freitas. Long-Range Dependence of Internet Traffic Aggregates. *Lecture Notes on Computer Science*, (2345), May 2002.
11. A. Bak, W.Burakowski, F. Ricciato, S. Salsano, and H. Tarasiuk. Traffic Handling in AQUILA QoS IP Networks. In *QoFIS'01*, September 2001.
12. S. Leinen and V. Reijs. D9.7 – Testing of Traffic Measurement Tools, September 2002.
13. Tijani Chahed. IP QOS Parameters. *TF-NGN*, November 2000.
14. V. Paxson, G. Almes, J. Mahadavi, and M. Mathis. Framework for IP Performance Metrics, IETF RFC2330, 1998.
15. CAIDA Tools. http://www.caida.org/tools/.
16. S. Floyd. Comments on measurement-based admissions control for controlled-load services. Technical report, LBNL, 1996.
17. F. Georgatos et al. Providing Active Measurements as a Regular Service for ISPs. In *PAM'2001*, April 2001.
18. C. Dovrolis, P. Ramanathan, and D. Moore. What Do Packet Dispersion Techniques Measure? In *IEEE INFOCOM'01*, 2001.
19. C. Dovrolis and M. Jain. 'End-to-End Available Bandwidth: Measurement methodology, Dynamics, and Relation with TCP Throughput. In *ACM SIGCOMM'02*, August 2002.

Pervasive Accounting of Resource Consumption for Wireless Services with Adaptive QoS

Paolo Bellavista, Antonio Corradi, and Silvia Vecchi

Dip. Elettronica, Informatica e Sistemistica - University of Bologna
Viale Risorgimento, 2 - 40136 Bologna - ITALY
Phone: +39-051-2093001; Fax: +39-051-2093073
{pbellavista, acorradi, svecchi}@deis.unibo.it

Abstract. Wireless communications and device miniaturization are enabling pervasive scenarios where the limited resource availability and the wide heterogeneity of access terminals make crucial to provide mechanisms and tools for the dynamic adaptation of Quality of Service (QoS). Particularly in case of very resource-consuming services QoS levels should be downscaled to fit the specific characteristics of the provisioning environment and of the current access device. In this context, a challenging aspect is to account clients not only for the generated network traffic, but also for the distributed resources involved in the dynamic QoS adaptation. The paper presents an accounting middleware solution that extends the traditional Internet where and when needed, depending on the mobility patterns of the wireless client devices, by introducing flexible mobile components for metering and charging. The middleware exploits the Mobile Agent (MA) technology to achieve dynamicity in metering/charging behavior, locality in the access to metering data, and to enable session-dependent accounting, even in case of temporary network partitioning.

1 Introduction

Telecommunication systems and the Internet are converging towards a pervasive integrated scenario with an enlarging set of very heterogeneous mobile devices, e.g., palm-sized computers, cellular phones and wireless pagers, that should access services anytime and anywhere. The ubiquitous availability of (either wired or wireless) networked computing environments identifies new service scenarios and requires new support solutions [1]. In particular, service provisioning to wireless portable devices should consider the strict limitations on their hardware/software characteristics, their wide heterogeneity, and the limited (and expensive) network resources typical of wireless environments.

The above aspects are particularly crucial when dealing with very resource-demanding services, such as multimedia distribution. Some recent research work starts to recognize the need for middleware components that can dynamically downscale the Quality of Service (QoS) levels of Video on Demand (VoD) flows at the involved wireless access points over the fixed Internet infrastructure, in order to fit the characteristics of the specific provisioning environment, e.g., the screen resolu tion/size of current access devices, the expected average network bandwidth, and the user preferences [2, 3].

In this context, there is an emerging need for accounting solutions allowing the enforcement of QoS-aware charging and pricing strategies, which consider not only the service access duration time, but also the usage of all distributed resources involved in (the possibly adapted) service provisioning. Apart from billing purposes, rich and fine-grained accounting solutions could have the relevant side-effect of limiting the excessive resource demand by greedy users, thus reducing the risk of network/system congestion.

Accounting includes several management activities, from the metering of resource consumption, to the storing and processing of metering data. Metering should take into account several factors at different levels of abstraction, e.g., the type of accessed services, the requested QoS level, the generated network traffic, but also the processor time and the memory possibly consumed at intermediate nodes in case of dynamic QoS adaptation to client characteristics and preferences. Storing keeps metering data on stable storage, either in the locality where resource consumption has been measured or in a remote administration site with the additional transport and security issues. Processing elaborates the raw metering data to obtain concise indicators about service usage. When and how to perform processing may greatly vary depending on the accounting goals: in billing, metering data can be processed on/off-line to enable dynamic/static pricing strategies [4]; when dealing with QoS-aware services over best effort networks, processing can also be in charge of ascertaining the QoS level offered during service provisioning, both to perform QoS management operations and to charge users for the actually supplied QoS level [5].

The design and implementation of QoS-aware accounting systems in mobility scenarios has to face additional challenging issues. Mobility-enabled accounting requires location tracking while clients roam in global environments and coordinating remote resources in different networks, possibly with heterogeneous metering mechanisms. In addition, accounting systems should be capable of organizing charging strategies that may depend on mutual agreements between different service providers and network operators, and on previous user actions in different network localities. Moreover, mobility-enabled accounting should operate independently of possible temporary disconnection and network partitioning.

As already sketched above, the hardware/software limits of several categories of portable devices are pushing towards dynamic infrastructures to support their network connectivity, service access, and QoS differentiation. We claim that the same infrastructures should provide accounting functions to charge access devices for the resources actually consumed in any network locality willing to offer mobile accessibility. For instance, let us consider tourists equipped with Wi-Fi Personal Digital Assistants (PDAs) who like to access information about the historical buildings, monuments, restaurants, and shops in the downtown area where they are. If the area provides IEEE 802.11 service access points, the PDAs can potentially access the very detailed multimedia tourist information which is often already available in Web servers over the fixed Internet; to this purpose, however, this information usually has to be downscaled to fit the specific bandwidth/visualization capabilities of the PDAs. It is necessary to dynamically deploy novel support functions for content adaptation on the involved wireless access points; this infrastructure should monitor and charge tourists also for the resources consumed in the process of dynamically downscaling the service contents.

The paper specifically focuses on solutions for the accounting of wireless devices in mobility contexts, i.e., *pervasive accounting*. We claim that pervasive accounting requires a dynamic and extensible support infrastructure, capable of evolving during service provisioning depending on client mobility. The accounting infrastructure should meter, register and charge resource consumption locally where mobile users/devices move to, without requiring continuous connectivity with remote and centralized accounting managers. The paper discusses and motivates the design of pervasive accounting middleware based on the Mobile Agent (MA) technology, to deploy the support infrastructure only when and where needed at service provisioning time. MAs can install new metering and charging behavior dynamically, can maximize locality in the access to metering data, and can enable accounting even in case of temporary network partitioning.

Along these guidelines, we have designed and implemented a pervasive accounting service on top of the Secure and Open Mobile Agent (SOMA) platform[1]. On the basis of a support layer for the portable metering of resource consumption, the pervasive accounting service is organized in terms of accounting MAs that follow client movements in the different wireless network localities. The co-locality of accounting MAs and their mobile users enable local metering, storing and charging depending on the actual consumption of system/application-level resources at service provisioning time. The pervasive accounting service presented in this paper integrates with our previous research work in multimedia QoS tailoring/adaptation over best-effort networks [6].

The paper is structured as follows. Section 2 presents the running case study of a tourist guide service with wireless accessibility and QoS dynamic adaptation of the provided VoD flows: the Downtown Visitor Assistant (DVA). The DVA scenario points out the requirements for pervasive accounting solutions and makes evident the need for the adoption of highly dynamic implementation technologies, such as MAs, as discussed in Section 3. Section 4 describes the layered architecture of the pervasive accounting service, while Section 5 and 6, respectively, give some implementation insights and show the accounting components at work during DVA provisioning. Related work, conclusive remarks and directions of current research end the paper.

The Downtown Visitor Assistant Case Study

The provisioning of the DVA service to wireless portable devices is a useful scenario to outline the issues related to QoS-aware pervasive accounting. In DVA, tourists are expected to use a standard Web browser running on their PDA while they roam in the city downtown. The DVA service is made available by Service Access Points (SAPs) offering Wireless Local Area Network (WLAN) connectivity; each SAP covers a single district of the downtown area.

The DVA service may be provided to wireless terminals as a set of *service units,* where a service unit is a set of Web pages (and included multimedia presentations) with logically correlated tourist information. For instance, a service unit may be the “virtual visit of the cathedral”, which consists of a start HTML page and of a set of video/audio files; another one may be the “cinema and theater program”, which is

[1] Additional information about SOMA and its downloadable code are available at: `http://lia.deis.unibo.it/Research/SOMA/`

composed of a collection of HTML pages with embedded JPEG/GIF images. The above modeling and de-composition of service contents is also useful for the management of service costs, as illustrated in the following.

Any SAP plays the twofold role of both supporting wireless connectivity to the tourist information available on the wired Internet and adapting dynamically the service provisioning depending on the user/access device. In this scenario, for each service unit, WLAN providers are interested in charging users for two main provisioning costs:

- *Wireless connectivity costs.* They are the costs associated to supporting wireless accessibility to DVA. Accounting these costs should consider the consumption of communication resources between the SAP and the wireless access device, and can be measured in terms of access duration time and generated traffic over the wireless network.
- *Tailoring costs.* They are the costs associated to the QoS adaptation operations performed by the SAP to make the service unit fit the specific access terminal. Accounting these costs requires metering the consumption of resources, such as CPU and memory, for dynamically downscaling service contents to adapt service results to the bandwidth/visualization capabilities of the access devices, e.g., to convert MPEG multimedia flows into a sequence of JPEG fixed images with minor resolution or to simply discard images of unsupported formats.

Tourists accessing DVA should have the possibility to provide personal preferences (user profiles) during the subscription phase; similarly, device profiles should be available to describe the characteristics of all the different supported access terminals. When a tourist enters a WLAN with her wireless device, user/device profiles should be exploited to define how to dynamically adapt the service content and to configure the user-specific charging strategy, i.e., by specifying which metering data are relevant and which are not.

In addition, when tourists request a service unit, the local wireless provider should send them an Estimate of Charge (EoC), i.e., an evaluation of the provisioning costs related to the delivery of the requested service unit, including the communication resources to be provided and the QoS adaptation to be applied. The EoC approximates the actually imposed charge and is statically calculated before service provisioning, and may consequently differ from the final actual charge. Only if the tourist accepts the EoC, the provider goes on with service provisioning.

2 Mobile Agents for Pervasive Accounting

The DVA scenario points out how the merging of mobility, limited devices, wireless communications and the Internet is going to change service provisioning and management. Traditional support solutions for fixed networks do not suit these novel dynamic environments where users, devices and even service components can change their location at runtime.

MAs are a suitable and effective technology to face the challenging issues raised by providing service accessibility to wireless portable devices, first of all because of MA *mobility*. MAs can follow the portable device movements to maintain co-locality with the clients they work for, or can dynamically move close to needed resources and service components to preserve operation locality. In addition, they can keep the

reached execution state and restore it when continuing their execution at the destination host. This is crucial in pervasive accounting for tracking the service session state and the resource consumption history of portable devices connecting to/disconnecting from different wireless network localities.
Pervasive accounting stresses *dynamicity*, as the possibility of modifying and extending the support infrastructure by installing/discarding the needed metering, pricing and charging behavior, in response to runtime evolving requirements. Dynamic distribution/modification of code and dynamic resource binding are very similar in case of both MAs and the QoS-aware support of portable device accessibility [2].
Service provisioning and accounting in the mobility-enabled Internet call for visibility of the location of client users/devices. *Location awareness* is crucial to adapt services to the currently available local resources and to enable accounting strategies depending on local conditions and requirements. Location awareness is typical of the MA paradigm that propagates location visibility up to the application level, thus allowing high-level service management operations and QoS adaptation.
Moreover, the possible MA autonomy from clients simplifies service *personalization*. While following the device movements, MAs can facilitate the tailoring of services to the device profile characteristics and to the user personal preferences. Ad-hoc tailoring MAs can perform client-specific content adaptation at service provision time and should be accounted for their usage of execution resources. For instance, an MA-based infrastructure for service accessibility can follow the change of location of access devices by migrating correspondingly the QoS adaptation logic, even depending on the resource availability in the new network locality; the same infrastructure should account the clients for the local resource consumption related to runtime tailoring operations.
Accounting also calls for *security*, to authenticate mobile users/devices, to charge the service usage in a non-repudiable way and to grant secrecy/integrity in communications. The MA research faced and is facing the complex security challenges inherent to the technology to favor the MA adoption. As a side-effect, several state-of-the-art MA systems provide rich security solutions a pervasive accounting service can significantly benefit from [7, 8]. Finally, pervasive accounting needs *interoperability* to interact, meter and control the consumption of resources and service components available in the new, statically unknown, localities where the clients move at service provisioning time. To face similar problems of interaction with unknown resources, the MA research has promoted interoperable and standard interfaces, e.g., via compliance with CORBA and related standards, such as the OMG Mobile Agent Systems Interoperability Facility (MASIF), and the FIPA specifications, which can favor the openness and portability of MA-based implementations of pervasive accounting services [9].

3 The Pervasive Accounting Service: Outline

We have designed and implemented a pervasive accounting service realized in terms of a layered facility infrastructure that supports the provision-time accounting of the resources actually consumed by mobile users with wireless portable devices.
We already pointed out how pervasive accounting can benefit from the MA technology; our accounting solution operates on top of the SOMA platform to take advantage

of the MA features. SOMA is a Java-based general-purpose middleware for the design, development and deployment of MA-based applications in global, open and untrusted environments [3]. Our accounting solution mainly exploits two SOMA facilities: the monitoring support and the portable service support. The monitoring support can inspect and extract data about the state of the resource engagement at any node hosting the execution of SOMA agents: each SOMA execution environment is equipped with a monitoring module tracing the allocation of different types of resources, at both the system level and the application one [10]. The portable service support consists of a mobile proxy-based infrastructure for service discovery, binding, dynamic QoS tailoring, and delivery of service contents to mobile wireless devices [2].

Our pervasive accounting service consists of three different components: the Metadata Service, the Accounting Coordination Service and the Charging Service.

The *Metadata Service* maintains profiles of registered users, supported access devices, available service components and available pricing policies. It implements a partitioned and partially replicated directory service specialized for profiles; any metadata directory component keeps local copies of profile information and coordinates with other peers to achieve global profile visibility.

The *Accounting Coordination Service* rules both service provisioning and resource usage metering. It configures and coordinates metering/tailoring functions driven by the enforced pervasive accounting strategy: according to the dynamically retrieved service/user/device profiles, it chooses the most suitable QoS adaptation components and commands the monitoring module by setting monitoring configuration parameters to trace the resource consumption also due to the QoS adaptation operations.

The *Charging Service* computes and proposes the EoC to the users during the service setup phase, and, most important, produces the charging reports to bill responsible users for their actual resource usage. The Charging Service locally processes monitoring data by taking into consideration the pricing profile information to apply to the specific service session, and compiles charging report for each wireless network locality visited during the service session.

After this brief sketch of the architecture of our pervasive accounting service, the following section gives implementation insights about the different accounting components and the SOMA facilities crucial for the accounting process.

4 The Pervasive Accounting Service: Implementation Guidelines

To fully understand the implementation of the pervasive accounting service, let us preliminary introduce the concept of locality abstractions. Any kind of interconnected system, from simple intranet LANs to the Internet, can be modeled in SOMA in terms of suitable locality abstractions. Any network node hosts one *place* for MA execution; several places can be grouped into *domain* abstractions, each one usually corresponding to one network locality. For instance, in the DVA scenario one SOMA domain typically models one LAN, together with its local IEEE 802.11 SAP and the corresponding WLAN cell. In each domain, a *default place* is in charge of inter-domain routing functionality.

In the following, we first describe which metering parameters the monitoring module can provide and how. Then, the section details how the Metadata Service handles and exploits different types of profile metadata for accounting purposes, and presents the main implementation guidelines of the services for Accounting Coordination and Charging.

4.1 Monitoring Module

The Monitoring Module (MM) is a local middleware component available on any SOMA place. MM can inspect and make visible a wide set of monitoring indicators about resource consumption, both at the system level and at the application one. At the system level, MM gets information about the processes working on local resources and about their usage of the communication infrastructure. For any process, it can report the process identifier and name, the CPU usage (time and percentage, of both the process and the composing threads) and the allocated memory (both physical and virtual). Network metering data include, for any process, the total number of sent/received UDP packets, of sent/received TCP segments, of TCP connections, and of TCP/UDP packets received with errors. At the application level, MM can collect information about all service components accessed from within the Java execution environment. For any active Java thread, it can detect any invocation of a dynamically definable set of methods and any object allocation/deallocation operation.
To overcome the transparency imposed by the Java Virtual Machine (JVM), MM exploits extensions of the Java technology: the JVM Profiler Interface (JVMPI) [11] and the Java Native Interface (JNI) [12]. In addition, MM integrates with external standard monitoring entities, i.e., Simple Network Management Protocol (SNMP) agents [13]. JVMPI makes possible to instrument dynamically the JVM for debugging and monitoring purposes, and MM exploits it to collect, filter and analyze application-level events produced by Java applications. At the kernel level, MM collects system-dependent monitoring data, by interrogating SNMP agents that export local monitoring data via their standard Management Information Base (MIB). To enable also the monitoring of hosts without any SNMP agent in execution, MM exploits JNI to integrate with platform-dependent monitoring mechanisms, which we have currently implemented for the Windows NT, Solaris, and Linux platforms.
MM processes raw metering data to quantify concisely the resource consumption and the service usage. For instance, the network-related information is used to determine the overall network traffic generated by one user during her service session. The CPU usage and the memory allocation on the SAP currently providing connectivity to the wireless client can represent a rough estimate of the computational costs associated with dynamic QoS downscaling, e.g., when converting a VoD flow included in a DVA service unit from the AVI/WAV formats to the TealMovie proprietary one.
Let us point out that MM can be dynamically tuned by choosing both the type of relevant resources to monitor and the accuracy/frequency of the monitoring probing. This permits to reduce significantly the overhead of the metering functions. For additional details about the monitoring implementation and performance, see [10]. In addition, MM is in charge of locally maintaining the metering data in stable storage supports; these data are then collected and processed by the specialized charging MAs presented in Section 5.4.

4.2 Metadata Service

The pervasive accounting service adopts high-level metadata, provided by the Metadata Service (MS), to describe the entities involved in service provisioning and the corresponding pricing strategies depending on actually consumed resources. MS exploits four types of profiles: user, device, service, and pricing profiles.

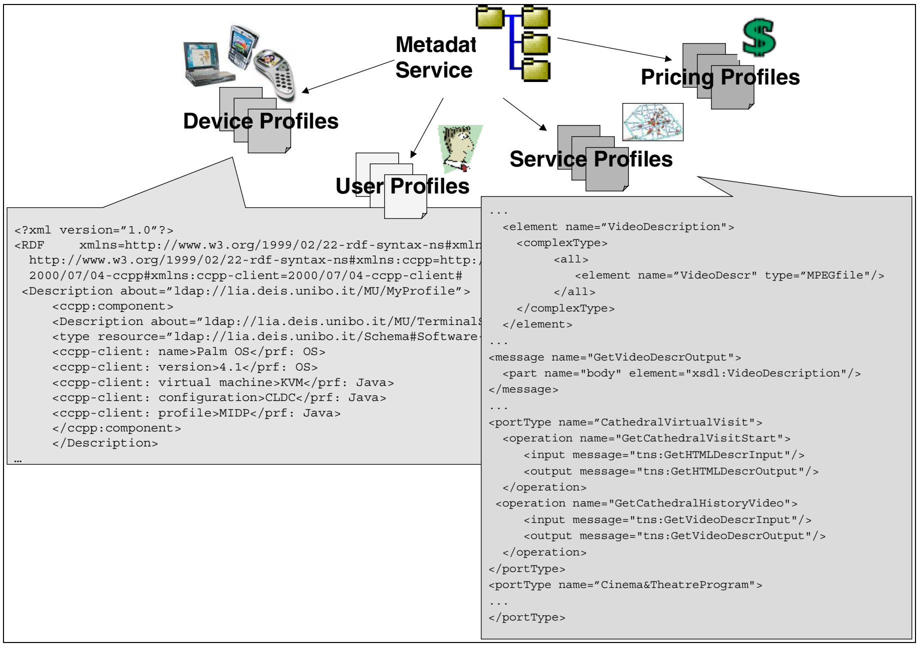
User profiles are generated in the user subscription phase and include information about users and their service preferences: personal data, security-related information (digital identity certificate), credit card number, subscribed services, and class of subscription (gold, silver, and bronze) to the WLAN provider.

Device profiles describe the hardware capabilities (type of processor, quantity of memory, graphic resolution, ...) and software characteristic (operating system, JVM version, supported graphic formats, ...) of the registered wireless access devices. User/device profiles are represented according to the W3C Composite Capabilities/Preference Profile (CC/PP) specification. Fig. 1 shows an example of device profile.

Service profiles describe the service units composing a service, in terms of operations and input/output type definitions. Each service unit implies a sequence of operations (getCathedralVisitStart, getCathedralHistoryVideo, …) consisting of a set of requests/replies messages with specified types and formats (HTML, MPEG-1, …); these operations define the service interface. The service profile describes this interface and plays a crucial role to evaluate the costs of the requested service unit, depending also on applicable pricing/device profiles, and to determine dynamically which kind of adaptation operations are needed, accordingly to the proper device profile. Service profiles are expressed in the Web Services Description Language (WSDL). Figure 1 shows an example of service profile in the case of the virtual visit of the cathedral.

Pricing profiles define the price for any resource consumed during service provisioning and are the basis to enforce charging policies and to determine final user bills. According to the DVA pricing strategy sketched in Section 2, our solution accounts users for resource consumption in terms of connection and dynamic QoS adaptation operations. Two pricing profiles are exploited: the connection pricing, and the adaptation pricing, each of them specifying the cost per unit of the corresponding accountable resources, i.e., access duration time and bandwidth occupation for the connection, CPU and memory for the adaptation. For instance, access duration and CPU are priced in €/s. Costs can vary with the time of the day and with the user class of subscription. The pricing profiles are expressed according to the Resource Description Framework (RDF) format. CC/PP, WSDL and RDF specifications can be found in the W3C web site [14].

We implemented a very simple pricing/tariffing strategy just to verify the feasibility of our approach. However, we believe that this accounting approach, which exploits profiles to describe how to price the service and performs distributed processing of monitoring data to charge users at provisioning time, is suitable for enforcing also more sophisticated and dynamic pricing policies.



```
<?xml version="1.0"?>
<RDF     xmlns=http://www.w3.org/1999/02/22-rdf-syntax-ns#xmln
  http://www.w3.org/1999/02/22-rdf-syntax-ns#xmlns:ccpp=http:/
  2000/07/04-ccpp#xmlns:ccpp-client=2000/07/04-ccpp-client#
 <Description about="ldap://lia.deis.unibo.it/MU/MyProfile">
     <ccpp:component>
     <Description about="ldap://lia.deis.unibo.it/MU/TerminalS
     <type resource="ldap://lia.deis.unibo.it/Schema#Software-
     <ccpp-client: name>Palm OS</prf: OS>
     <ccpp-client: version>4.1</prf: OS>
     <ccpp-client: virtual machine>KVM</prf: Java>
     <ccpp-client: configuration>CLDC</prf: Java>
     <ccpp-client: profile>MIDP</prf: Java>
     </ccpp:component>
     </Description>
…
```

```
...
  <element name="VideoDescription">
    <complexType>
         <all>
            <element name="VideoDescr" type="MPEGfile"/>
         </all>
    </complexType>
  </element>
...
<message name="GetVideoDescrOutput">
  <part name="body" element="xsdl:VideoDescription"/>
</message>
...
<portType name="CathedralVirtualVisit">
  <operation name="GetCathedralVisitStart">
     <input message="tns:GetHTMLDescrInput"/>
     <output message="tns:GetHTMLDescrOutput"/>
  </operation>
 <operation name="GetCathedralHistoryVideo">
     <input message="tns:GetVideoDescrInput"/>
     <output message="tns:GetVideoDescrOutput"/>
  </operation>
</portType>
<portType name="Cinema&TheatreProgram">
...
</portType>
```

Fig. 1. Examples of device and service profiles

4.3 Accounting Coordination Service

The Accounting Coordination Service exploits the decentralized monitoring modules and the Metadata Service profiles to determine how, where and when to perform QoS-aware accounting management. The service is implemented in terms of support components, i.e., the Coordination Mobile Agents (CoMAs), which automatically and dynamically distribute to the involved wireless SAPs.

CoMAs are in charge of configuring the service downscaling and of specializing the resource metering according to the applicable profiles. In particular, the user, device and service profiles drive the decisions about the service management operations necessary for the dynamic QoS adaptation of service contents, while service and pricing profiles impact on the choice of the accountable resources to monitor. CoMAs are organized hierarchically: the CoMA manager decides the adaptation/metering operations to perform in accordance with profile metadata and delegates these actions to more specialized CoMAs, the Tailoring MAs (TMAs) and the Monitoring MAs (MMAs), operating locally to the interested SAPs.

TMAs can carry out different QoS adaptation operations by exploiting local basic filtering functions provided by service adaptation components [2]. Examples of filtering include transcoding of the hypertext representation format (from XML to WML, from HTML to C-HTML, from HTML to HTML without image tags, …), VoD/image format transcoding (from AVI/WAV to MPEG-1, from AVI/WAV to

Divx, from AVI/WAV to the TealMovie proprietary format, from TIFF to JPEG/GIF, …), and VoD/image resolution/size/color-depth decrease.
MMAs set their local monitoring modules to trace the CPU usage and the memory allocation of their local TMAs working for the specific downscale, and to measure the connection costs (service duration time and bandwidth consumption) from the SAP to the wireless client. MMAs exploit the metering functions available on any SOMA place involved in service provisioning. MMAs can also install new metering functions to extend the monitoring module, where and when needed, by exploiting the SOMA support for code mobility.
Any CoMA manager relates to one authenticated user and works as her care-of entity: it is instantiated at the first user access to the service, and obtains the user profile from the SOMA MS component in the domain. Then, the CoMA manager follows the user movements by carrying the user profile with itself. Since the same user can access different subscribed services in different moments and with different devices, only at the request of a new service unit the CoMA manager retrieves the applicable device/service/pricing profiles and determines which TMAs and MMAs are needed in the current access.

4.4 Charging Service

The Charging Service accomplishes two tasks. It exploits the user/service/device/pricing profiles to provide the EoC for the requested service unit, and uses the metering data stored by MMAs and the pricing profiles provided by the MS for the on-line processing of the global service cost report to bill the user with. The service is implemented in terms of Charging MAs (CMAs).
As the CoMA manager, also a CMA instance is dedicated to one authenticated user during a session and works as her care-of entity by following the user movements. When the user requests a service unit, the associated CMA is in charge of evaluating its cost and of proposing the corresponding EoC to the client. The EoC is calculated on the basis of two elements: the estimated adaptation costs necessary for downscaling the requested service unit depending on the applicable user/device/service profiles, and the estimated connection costs for the delivery of the adapted data in the requested service unit. For instance, the middleware chooses the proper tailoring operation in accordance with the device and service profiles, then estimates the cost for this operation and for the transmission of the adapted data on the basis of statistics about previous service sessions, e.g., the conversion cost from AVI to MPEG-1 format is X€ / (s*MB), the transmission cost of MPEG-1 stream with resolution Y is estimated as Z€ / (s*Y).
CMAs also process resource usage data at provision time according to the pricing strategy to compile charging reports. On the basis of the costs of accountable resources indicated in the pricing profiles, CMA works locally to elaborate the resource consumption information and to calculate the cost for the service usage in any visited SOMA domain. When the user leaves a domain, the CMA sends the charging report related to that locality to a central billing authority, which uses the CMA reports to compile the final bill for customer charging.

5 The Pervasive Accounting Service at Work in the DVA Scenario

To clarify how the accounting components interoperate and coordinate in an actual service scenario, we present them at work in the DVA accounting. We model each district of the downtown, i.e., each WLAN, with a SOMA domain, where the IEEE 802.11b Service Access Point (SAP) is provided by a default place and the other tourist information servers in the district correspond to SOMA places, as depicted in Figure 2.

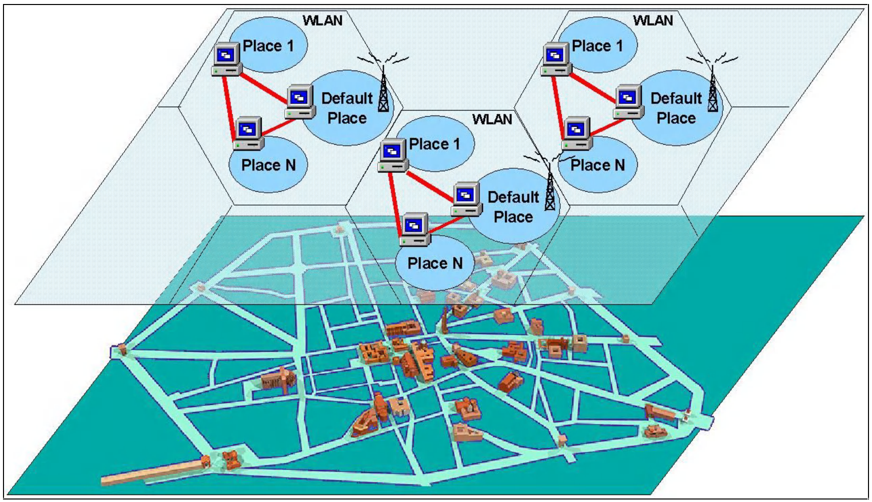


Fig. 2. Downtown district modeling in the DVA service.

Let us imagine a tourist T accessing the DVA service with an IEEE 802.11b-compliant PDA. When T enters a new domain, her entrance triggers a sequence of accounting management actions, as shown in Fig. 3. First, the system ascertains whether T already has a currently open accounting session; to this purpose, the system controls whether T has previously visited another close district/domain. If it is the first access to the service, the SOMA place currently providing wireless connectivity to the client asks MS for the user profile and then associates the tourist with newly instantiated CoMA manager and CMA. Otherwise, the middleware triggers the migration of the previously instantiated CoMA manager and CMA for that user from the last visited district/domain. In the worst case of network partitioning, if the already instantiated CoMA manager and CMA cannot reach the new domain, the system instantiates brand new CoMA manager and CMA for the user. To avoid CoMA and CMA proliferation, when CoMA managers and CMAs are inactive for a defined amount of time, they automatically terminate their execution.
When T requests a service unit, the CMA retrieves the involved user, service, device and pricing profiles (in the example, the profile of the user T, the DVA service profile, the Wi-Fi-enabled device profile, and, supposing T is a "gold" user, the "gold price" profile), and calculates and sends the EoC for the requested service unit, e.g. the virtual visit of the cathedral. If the tourist accept the EoC, the CoMA manager, by exploiting the involved service and device profiles, instantiates the suitable TMA and

MMA. Since T accesses DVA from a PDA with limited screen size and graphic resolution, the required downscaling involves mainly the reduction of image size/resolution to fit the PDA-specific display capabilities. TMA performs these filtering/transcoding operations on the SOMA place with the SAP. MMA configures the local MM component on the same place to meter CPU usage and memory allocation of the selected TMA, and to measure bandwidth consumption to transmit the data composing the requested service unit. After this middleware reconfiguration, service provisioning and usage accounting can start, or continued in case of user roaming from a previously visited SOMA domain.
When T moves to another SOMA domain, the CMA charges locally the tourist service usage by processing data collected and stored in the domain by MM, according to customer category and pricing profiles: if T is a gold user and has visited the domain A from 10 a.m. until 12 a.m., CMA considers resource prices (time, bandwidth, CPU and memory) according to her gold class in the red (highest price) time. CMA creates a report for the visited domain with the cost for the local service usage and sends them to the central administration node, where reports are maintained in stable storage and finally processed to create a total usage-based bill for T.

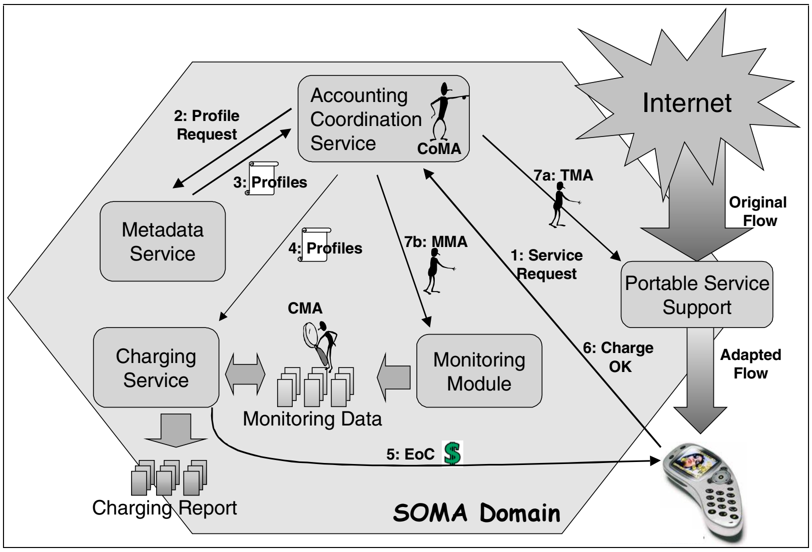


Fig. 3. Interactions of pervasive accounting components in a SOMA domain.

6 Related Work

Accounting is recognized as a relevant and challenging activity in the management of distributed resources, systems and services. IETF and IRTF have established working groups on Authentication Authorization Accounting (AAA) activities to deploy protocols and architectures to provide AAA services for the Internet. The most known and currently adopted solutions are Remote Authentication Dial In User Service (RADIUS) and Diameter [15, 16]. These systems represent general-purpose manage-

ment solutions mainly designed for traditional network infrastructures. They lack specific service management functions for pervasive accounting, and Diameter only faces the mobility issue by supporting mobile IP networks.
By considering the main accounting research activities on QoS and cost sharing aspects, on the one hand, the Market Managed Multiservice Internet project aims at enabling resource management on the basis of dynamically differentiated charging for different QoS levels. The essence of the market-managed approach is bringing the customer systems into the control loop through a mechanism of price tuning and new fare notification: support functions dynamically tune prices depending on client QoS demands; the proposed middleware infrastructure notifies users of the new prices and configures the accounting support accordingly, by exploiting a Java-based SNMP-compliant implementation [17]. On the other hand, the accounting working group of Next Generation Internet specifically focuses on the concepts of cost sharing and reverse charging, to enable an Internet Service Provider (ISP) to account final users for their actually received traffic, even when users are accessing the Internet via other collaborating ISPs [18].
By focusing on the accounting approaches that address also mobility issues, the Charging Accounting and Billing proposal, developed within the framework of the MOBIVAS project [19], is an integrated system for supporting advanced business models for service provisioning in 3G mobile systems. Its architecture allows flexible accounting, intended as the process of dynamic revenue sharing among Value Added Service Providers and home/visited network operators, depending on their mutual relationships and agreements. Another relevant project is MobyDick, which exploits the IPv6 mobility-enabled end-to-end architecture as the basis for its AAA charging infrastructure. In particular, MobyDick interfaces the DiffServ architecture with a dedicated Application-Specific Module that permits to control service access and to account roaming users/devices by adopting solution patterns similar to the mobile IP ones [20].
The above proposals confirm the need for increasing flexibility and dynamicity in pervasive accounting. They represent a significant evolution step in accounting solutions as they recognize the relevance of taking into consideration both user/device mobility and the provisioning of services with differentiated QoS levels. However, they follow the guideline of extending the functions of quite traditional management technologies and do not support the dynamic extensibility of the support infrastructure, the autonomy of accounting operations with regards to temporarily unreachable managers, and the metering of resources in visited networks considering consumed to dynamically adapt the provided QoS level. To the best of our knowledge, our pervasive accounting service is the first MA-based proposal specifically designed and implemented for highly dynamic pervasive scenarios where IEEE 802.11 SAPs provide and adapt service provisioning to Wi-Fi access devices.

7 Conclusions and Ongoing Work

The Internet is likely to offer a pervasive service environment and this requires extending the fixed network infrastructure to support adapted service provisioning to mobile users and portable devices, when and where needed. In this context, effective accounting solutions should consider primarily the resources actually consumed during service provisioning; this is crucial to stimulate a fair resource sharing among users and to leverage the market of services with differentiated QoS levels. The paper shows how the accounting of portable client devices with wireless connectivity and limited hardware/software characteristics can significantly benefit from highly dynamic supports that exploit mobile code programming paradigms and, in particular, the MA technology.

The first encouraging results achieved in using the pervasive accounting service in the DVA case study are stimulating further research work to produce a more usable and complete accounting solution. Issues currently under investigation are non-repudiation, micro-payments mechanisms, and mobile ad-hoc styles of interaction. From the security point of view, our accounting already exploits secure communications channels and encrypted storage for metering data. However, additional work is necessary to guarantee non-repudiability of resource usage and to establish differentiated trust levels between users and accounting components, so to minimize the security overhead when working in trusted environments. By focusing on payment aspects, a challenging issue is how to alleviate problems of fraud in mobile networks, by ensuring that all local bills are paid and by eliminating at the same time the need for inter-operator billing agreements. We are modifying the charging part of the accounting service to support also micro-payments mechanisms that can exploited by user-delegated CoMAs to operate local payments directly at service provisioning time [21]. Finally, we are our solution to account also services cooperatively provided by peer portable devices composing a proximity-based ad-hoc network, without exploiting a SAP infrastructure. This significantly changes the perspective and imposes to face additional challenges, such as how to measure resource consumption over very limited and heterogeneous access devices and where to store and process the accounting data.

Acknowledgement. This work is supported by the Italian MIUR (FIRB WEB-MINDS project) and CNR (IS-MANET project).

References

1. G. G. Richard III, "Service Advertisement and Discovery: Enabling Universal Device Cooperation", IEEE Internet Computing, Vol. 4, No. 5, Sep.-Oct. 2000, pp. 18–26.
2. P. Bellavista, A. Corradi, C. Stefanelli, "The Ubiquitous Provisioning of Internet Services to Portable Devices", IEEE Pervasive Computing, Vol. 1, No. 3, Sept. 2002.
3. P. Bellavista, A. Corradi, C. Stefanelli, "An Integrated Management Environment for Network Resources and Services", IEEE Journal on Selected Areas in Communications, Vol. 18, No. 5, May 2000.
4. L. Da Silva, "Pricing for QoS-enabled Networks: a Survey", IEEE Communications Survey, Second Quarter 2000.
5. P. Bellavista, A. Corradi, S. Vecchi, "QoS-aware Accounting in Mobile Computing Scenarios", 11th Euromicro Conf. Parallel Distributed and Network-based Processing (PDP'03), Genoa, Italy, Feb. 5–7, 2003.
6. P. Bellavista, A. Corradi, "How to Support Internet-based Distribution of Video on Demand to Portable Devices", 7th IEEE Int. Symp. Computers and Communications (ISCC'02), Taormina, Italy, Jul. 1–4, 2002.

7. A. R. Tripathi, N. M. Karnik, K. Manish, V. T. Ahmed, R. D. Singh, "Mobile Agent Programming in Ajanta", IEEE Int. Conf. Distributed Computing Systems (ICDCS'99), 1999.
8. N. Dulay, R. Montanari, C. Stefanelli, "Flexible Security Policies for Mobile Agent Systems", Microprocessors and Microsystems, Elsevier Science, Vol. 25, No. 2, Apr. 2001.
9. IKV++, Grasshopper 2: the Agent Platform, http://www.grasshopper.de/
10. P. Bellavista, A. Corradi, C. Stefanelli, "How to Monitor and Control Resource Usage in Mobile Agent Systems", 3rd IEEE Int. Symp. Distributed Objects and Applications (DOA'01), Sept. 2001.
11. Sun Microsystems – Java Virtual Machine Profiler Interface (JVMPI), http://java.sun.com/j2se/1.4/docs/guide/jvmpi
12. Sun Microsystems – Java Native Interface (JNI), http://java.sun.com/j2se/1.4/docs/guide/jni/
13. J. D. Case, et al., "Simple Network Management Protocol (RFC 1157)", DDN Network Information Center, SRI International, May 1990.
14. W3C - The World Wide Web Consortium – http://www.w3.org/
15. C. Rigney, A. Rubens, W. Simpson, S. Willens, "Remote Authentication Dial In User Service (RADIUS)", IETF RFC 2138, Jan. 1997.
16. Pat R. Calhoun,Tony Johansson, Charles E. Perkins, "Diameter Mobile IP Application", http://search.ietf.org/internet-draft/draft-ietf-aaa-diameter-mobileip-14.txt
17. The Market Managed Multiservice Internet Project, http://www.m3i.org/index.html
18. A. Pras, B. van Beijnum, R. Sprenkels, R. Parhonyi, "Internet Accounting", IEEE Communications, Vol.39, No. 5, May 2001.
19. M. Koutsopoulou, N. Alonistioti, E. Gazis, A. Kaloxylos, "Adaptive Charging Accounting and Billing system for the support of advanced business models for VAS provision in 3G systems", IEEE Int. Symp. Personal Indoor and Mobile Radio Communication (PIMRC'01), San Diego, USA, Oct. 2001.
20. H. Einsiedler, R. L. Aguiar, J. Jaehnert, K. Jonas, M. Liebsch, R. Schmitz, P. Pacyna, J. Gozdecki, Z. Papir, J. I. Moreno, I. Soto, "The Moby Dick Project: A Mobile Heterogeneous ALL-IP Architecture", Int. Conf. Advanced Technologies, Applications and Market Strategies for 3G (ATAMS'01), Cracow, Poland, Jun. 2001.
21. H. Tewari and C. O'Mahony, "Real-Time Payments for Mobile IP", IEEE Communications Magazine, Feb. 2003.

Delay Guaranteed Fair Queueing(DGFQ) in Multimedia Packet Networks

Hyunho Yang[1] and Kiseon Kim[2]

[1] Dept. of Computer and Information, Suncheon Cheongam College,
224-9, Deogweol-dong, Suncheon-si, Jeonnam, 540-743, Republic of Korea.
`hhyang@scjc.ac.kr`
[2] Dept. of Information and Communications,
Kwangju Institute of Science and Technology(K-JIST)
1 Oryong-dong, Puk-gu, Kwangju, 500-712, Republic of Korea
`kskim@kjist.ac.kr`

Abstract. Fair queueing has been an important issue in the multimedia networks where resources are shared among nodes both wired and wireless. Most fair queuing algorithms based on the generalized processor sharing (GPS) emphasize fairness and bounded delay guarantee, while overlooking service differentiation for different service classes of their own delay bounds. In this paper, we propose a new fair queueing scheme, delay guaranteed fair queueing (DGFQ), which guarantees bounded delay for the classes of flows according to their individual delay requirements for multimedia services in the wireless packet networks.

Keywords. Fair Queueing, Bounded Delay, Quality of Service (QoS), Multimedia Network.

1 Introduction

The telecommunication technologies are envisioned to support multimedia services, both error-sensitive and delay-sensitive applications, even over the bandwidth-constrained wireless medium. With this vision in mind, the issue of providing fair and bounded delay channel access among multiple contending hosts over a shared channel resources has come to the fore. To achieve this goal, fair queueing has been a popular paradigm in both wireline and wireless packet cellular networking environments [1].

A series of algorithms are based on generalized processor sharing (GPS) [2] approach. Among them are WFQ [3] also called pack-by-packet GPS (PGPS), self clocked fair queueing (SCFQ) [4], and start time fair queueing (SFQ) [5]. These algorithms guarantee the fairness according to the relative weight of flows. Many other wireless fair queueing algorithms have been proposed as a variation of these algorithms.

In [5], the authors reviewed typical GPS based fair queueing algorithms and proposed a new simple, deadline guaranteeing fair queueing algorithm, i.e., start-time fair queueing (SFQ). Their proposed scheme implemented virtual time function less complex by adopting start tag of currently transmitting packet as virtual time value. In addition, SFQ allocates bandwidth fairly regardless of admission control as well as variation in server rate. Though this scheme improves delay guarantee and fairness bound, the authors overlooked the service differentiation according to the flow characteristics, e.g., guaranteed bandwidth service or best effort service, which is critical to support multiple heterogeneous sessions traffics both realtime and non realtime.

The delay and data decoupled fair queueing (D-FQ) was presented in [7], to mitigate the inherent drawback of GPS based fair queueing algorithm i.e., the inability to deliver independent control over delay and bandwidth guarantees, which results in inflexible bandwidth management and inefficient link utilization [8]. Their approach for independent control for delay and data rate is to sperate scheduling algorithms for real time and non-real time flows. This scheme not only guarantees the delay deadline of real time flows but also improves the performance of non-real time flows in terms of packet delay, however, it should be pointed out that the computational complexity increases due to the dual scheduling architecture, specifically for the explicit-delay-guarantee(EDG) algorithm which support real time flows.

In this paper, we propose a new fair queueing scheme i.e., delay guaranteed fair queueing (DGFQ), guaranteeing bounded delay of multimedia services. Our model is based on GPS algorithm with some modifications to guarantee bounded delay. In detail, we categorized the flows into two classes, i.e., delay guaranteed (DG) and non-delay guaranteed (NG). Then introduce additional weight factor to apply differentiated tagging operation for each class, i.e., set a little earlier start tag to DG class flows than to NG class flows. With this policy we can get better delay performance for DG class flows at the same fairness guarantee without serious increase in computational complexity.

The rest of this paper is organized as follows. Section 2 is devoted to describe the network model being considered. In Section 3, detailed architecture and algorithm of delay guaranteed fair queueing (DFGQ) is given. Then in Section 4, the performance of proposed scheme is evaluated. Finally, we concluded our work in Section 5.

2 Network Model

For the purpose of this work, we basically consider wireless networks supporting multimedia services with guaranteed delay performance. We also expect that this scheme performs similarly for the wired network.

In detail, we consider packet-cellular network with a high-speed wired backbone and small, partially overlapping, shared channel wireless cells. Each cell is served by a base station, which performs the scheduling of packet transmissions for the cell. Neighboring cells are assumed to transmit on different logical channels. Every mobile host in a cell can communicate with the base station, though it is not required for any two mobile hosts to be within range of each other. Each flow of packets are identified by a certain flow index. To concentrate our work on the enhancement of the delay guarantee performance, we excluded channel error to leave as a future work. Thus, we do not make any explicit mathematical assumptions about the error model in our framework. It should be noted that in our model, we do not consider the scenario where a wireless channel is shared across several neighboring cells, which is more complicated and introduces the hidden/exposed station problems [9].

3 Delay Guaranteed Fair Queueing(DGFQ)

In this section, we propose a new fair queueing algorithm, delay guaranteed fair queueing (DGFQ), which provides differentiated bounded delay for each traffic class having different service requirements. This proposal includes algorithm description, analytical properties and implementation in error free channel.

3.1 Algorithm Descriptions

Our proposed model, delay guaranteed fair queueing (DGFQ) basically adopts start-time fair queueing (SFQ) algorithm proposed in [5]. In DGFQ, as is in the SFQ, two tags i.e., a start tag and a finish tag, are associated with each packet. However, unlike WFQ and SCFQ, packets are scheduled in the increasing order of the start tags of the packets. Furthermore, $v(t)$ is defined as the start tag of the packet in service at time t. Finally, we assume that, in SFQ, WFQ or DGFQ scheme, there is a certain interval of time in which all flows are scheduled at least once, we call it *scheduling interval.*

It is desirable to categorize flows into classes to provide QoS for multimedia services, specifically for IP networks. [6]. To adopt this policy, all flows are classified into a number of classes according to their delay bound requirements. The simplest and basic classification is to make two classes, one for delay guaranteed (DG) flows and the rest for non delay guaranteed (NG) flows.

In all GPS based queueing algorithms, the weight parameter ϕ_f for a flow f is defined as

$$\phi_f = \frac{r_f}{\sum_{j \in B} r_j}$$

where B is a set of backlogged flows. It is intuitively evident from the above definition that ϕ_f is strongly dependent on the relative rate of a flow to the total backlogged flows in the system at a certain time instant. Thus, an additional weight parameter is required to control delay performance, hence, QoS, for each group of flows.

In our scheme, we introduce the *service differentiation coefficient*, α $(0 < \alpha \leq 1)$, to handle each flow class differently. When $\alpha = 1$, which is the case for NG class, our proposed scheme is identical to SFQ. By varying α, we can customize delay bound for individual flows i.e., adjust the relative service order of each flow in a scheduling interval.

The complete algorithm is defined as follows.

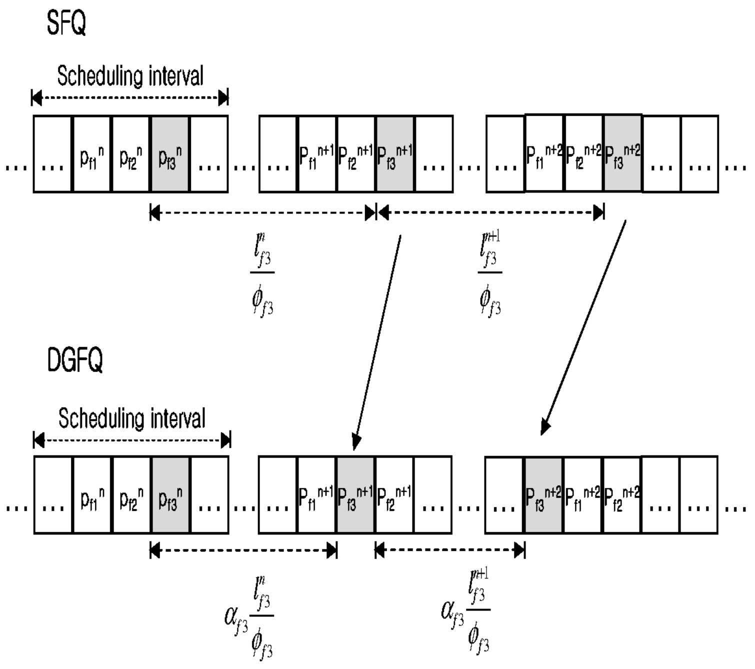

SFQ
Scheduling interval
p_{f1}^n p_{f2}^n p_{f3}^n P_{f1}^{n+1} P_{f2}^{n+1} P_{f3}^{n+1} P_{f1}^{n+2} P_{f2}^{n+2} P_{f3}^{n+2}
$\frac{l_{f3}^n}{\phi_{f3}}$
$\frac{l_{f3}^{n+1}}{\phi_{f3}}$
DGFQ
Scheduling interval
p_{f1}^n p_{f2}^n p_{f3}^n P_{f1}^{n+1} P_{f3}^{n+1} P_{f2}^{n+1} P_{f3}^{n+2} P_{f1}^{n+2} P_{f2}^{n+2}
$\alpha_{f3}\frac{l_{f3}^n}{\phi_{f3}}$
$\alpha_{f3}\frac{l_{f3}^{n+1}}{\phi_{f3}}$

Fig. 1. Comparison of Packet Scheduling Schemes (SFQ v.s DGFQ)

1. On arrival, k^{th} packet of flow f, p_f^k, is stamped with start tag $S(p_f^k)$, computed as
$$S(p_f^k) = \max\{v[A(p_f^k)], F(p_f^{k-1})\} \quad k \geq 1 \tag{1}$$
where $A(p_f^k)$ is the arrival time of packet p_f^k and $F(p_f^k)$ is the finish tag of packet p_f^k, defined as
$$F(p_f^k) = S(p_f^k) + \alpha_f \frac{l_f^k}{\phi_f} \tag{2}$$
where $F(p_f^0) = 0$, ϕ_f is the weight of flow f, l_f^k is the length of packet p_f^k, and α ($0 < \alpha_f \leq 1$) is the *service differentiation coefficient* for flow f. The value of α is 1 for NG class, or appropriate value for DG class.
2. Initially the system virtual time is 0. During a busy period, the system virtual time at time t, $v(t)$, is defined to be equal to the start tag of the packet in service at time t. At the end of a busy period, $v(t)$ is set to the maximum of finish tag assigned to any packets that have been serviced by then.
3. Packets are serviced in the increasing order of the start tags, ties are broken arbitrarily.

As shown in the Fig. 1, in DGFQ, the next packet is scheduled after $\alpha_f \frac{l_f^k}{\phi_f}$, while in SFQ scheme, the next packets of each on going flow scheduled after $\frac{l_f^k}{\phi_f}$. As mentioned before, the range of α is $0 < \alpha \leq 1$, the scheduling time difference i.e., $(1-\alpha_f)\frac{l_f^k}{\phi_f}$, is limited to one packet transmission duration. It is evident from (1) and (2) that the differentiated service time (virtual start time of a packet) can not go before system virtual time of the packet arrival instant. Due to this property, DGFQ maintain the fairness between each flow. On the other hand, it is still another issue to compensate lagged service in the error-pron channel environment, thus exclude from our considerations. Finally, the computational complexity of computing virtual time $v(t)$ in DGFQ is $O(\log N)$ per packet where N is the number of flows in the system.

3.2 Analytical Properties of DGFQ

In this section, the analytical properties of DGFQ are discussed specifically for the fairness guarantee and delay guarantee.

Fairness Guarantee. To derive fairness guarantee of DGFQ, let $W_f(t_1, t_2)$ and $W_g(t_1, t_2)$ be the amount of service received by flows f and g in the time interval $[t_1, t_2]$, moreover, ϕ_f and ϕ_g be the weight of each flow respectively. We need to prove a bound on

$$\left| \frac{W_f(t_1, t_2)}{\phi_f} - \frac{W_g(t_1, t_2)}{\phi_g} \right|$$

for any interval in which both flows f and g are backlogged. We achieve this objective by establishing a lower and an upper bound on $W_f(t_1, t_2)$ and $W_g(t_1, t_2)$ in Lemmas 1 and 2, respectively.

Lemma 1. *If flow f is backlogged throughout the interval $[t_1, t_2]$, then in an DGFQ server*

$$\phi_f(v_2 - v_1) - l_f^{\max} \leq W_f(t_1, t_2) \tag{3}$$

where $v_1 = v(t_1)$ and $v_2 = v(t_2)$ are virtual time of each time instant t_1, t_2.

Lemma 2. *In an DGFQ server, during any interval $[t_1, t_2]$*

$$W_g t_1, t_2) \leq \phi_g(v_2 - v_1) + l_g^{\max} \tag{4}$$

where $v_1 = v(t_1)$ and $v_2 = v(t_2)$ are virtual time of each time instant t_1, t_2.

Since unfairness between two flows in any interval is maximized when one flow receives maximum possible service and the other minimum service, Theorem 1 follows directly from Lemmas 1 and 2.

Theorem 1. *For any interval $[t_1, t_2]$ in which flows f and g are backlogged during the entire interval, the difference in the service received by two flows at an DGFQ server is given as*

$$\left| \frac{W_f(t_1,t_2)}{\phi_f} - \frac{W_g(t_1,t_2)}{\phi_g} \right| \leq \frac{l_f^{\max}}{\phi_f} + \frac{l_g^{\max}}{\phi_g} \tag{5}$$

Theorem 1 demonstrates that DGFQ guarantees the same fairness as in SFQ [5]. This is intuitively evident that DGFQ only adjusts the start tag, hence the scheduling order, in a set of backlogged flows to be serviced in a given time interval $[t_1, t_2]$, to insure earlier service for delay guaranteed flows.

Delay Guarantee. On the contrary to the fairness guarantee, DGFQ shows better performance than SFQ, for the delay guaranteed (DG) flows such as real time videos. Let r_f be the rate assigned to packet p_f^k length of l_f^k. Then finish tag of packet p_f^k, $F(p_f^k)$ is defined as

$$F(p_f^k) = S(p_f^k) + \alpha_f \frac{l_f^k}{r_f^k} \qquad k \geq 1 \tag{6}$$

Start tag of a packet and the system virtual time are defined as before.

As in SFQ, DGFQ provides deadline guarantees when the server capacity is not exceeded. To drive the deadline guarantee, let us formalize the meaning of the term "capacity is not exceed." Let rate function for flow f at virtual time v, denoted by $R_f(v)$, be defined as the rate assigned to the packet that has start tag less than v and finish tag greater than v. Formally

$$R_f(v) = \begin{cases} r_f^k & if\ \exists k \ni [S(p_f^k) \leq v < F(p_f^k)] \\ 0 & \text{otherwise.} \end{cases}$$

Let B be the set of flows served by the server. Then the capacity of a server with average rate C is *not exceeded* if

$$\sum_{n \in B} R_n(v) \leq C \qquad v \geq 0 \tag{7}$$

To derive the delay guarantee of DGFQ servers, we first derive a bound on the work done by an DGFQ server in the virtual time interval $[v_1, v_2]$ in Lemma 3.

Lemma 3. *If the server with parameter C is not exceeded, then the aggregate length of packets that have start tag at least v_1 and at most v_2, and are served in the same busy period, denoted by $\hat{W}(v_1, v_2)$, is given by*

$$\hat{W}(v_1, v_2) \leq C \sum_{n=0}^{n=k-m-1} \frac{l_f^{m+n}}{r_f^{m+n}} + \sum_{n \in B \wedge n \neq f} l_n^{\max} + l_f^k. \tag{8}$$

whenever

$$v_1 = S(p_f^m, r_f^m), \quad v_2 = S(p_f^k, r_f^k),$$

and

$$v_2 - v_1 = \sum_{n=0}^{n=k-m-1} \frac{l_f^{m+n}}{r_f^{m+n}}.$$

where $S(p_f^k, r_f^k)$ *is the start tag of packet* p_f^k *of flow* f *with rate* r_f^k*.*

For brevity, we will denote

$$\theta_f^k = \sum_{n \in B \wedge n \neq f} \frac{l_n^{\max}}{C} + \frac{l_f^k}{C}.$$

Further, we can define $A_e(p_f^k, r_f^k)$, the expected arrival time of packet p_f^k that has been assigned rate r_f^k as

$$A_e(p_f^k, r_f^k) = \max\left\{ A(p_f^k), A_e(p_f^{k-1}, r_f^{k-1}) + \alpha_f \frac{l_f^{k-1}}{r_f^{k-1}} \right\} \tag{9}$$

where $A_e(p_f^0, r_f^0) = -\infty$.

Theorem 2 defines the delay guarantee of DGFQ servers.

Theorem 2. *If the capacity of DGFQ server with parameter* C *is not exceed, then the departure time of packet* p_f^j, D_{DGFQ} *satisfies the following inequality.*

$$D_{DGFQ}(p_f^k) \leq A_e(p_f^k, r_f^k) + \theta_f^k \tag{10}$$

To compare this with SFQ, we can rewrite (10) as follows

$$D_{DGFQ}(p_f^k) \leq A_e(p_f^{k-1}, r_f^{k-1}) + \alpha_f \frac{l_f^{k-1}}{r_f^{k-1}} + \theta_f^k \tag{11}$$

and the departure time of a packet at a SFQ server, given in [5] is

$$D_{SFQ}(p_f^k) \leq A_e(p_f^{k-1}, r_f^{k-1}) + \frac{l_f^{k-1}}{r_f^{k-1}} + \theta_f^k \tag{12}$$

The difference of (11) and (12) is

$$(1 - \alpha_f) \frac{l_f^{k-1}}{r_f^{k-1}} \tag{13}$$

where $0 < \alpha_f \leq 1$. It is clear from the comparison above that, in DGFQ, the departure time of packet p_f^k is earlier by (13) than in SFQ. Hence, the delay bound is more tightly provided than in SFQ.

3.3 Implementation of DGFQ in Error-Free Channel

In this section, we consider about the implementation of the propose algorithm, delay guaranteed fair queueing (DFGQ). In the implementation, both real time and non-real time flows are scheduled based on the proposed packetized DGFQ algorithm.

To concentrate our work on the enhancement of the delay guarantee performance, we excluded channel error. It is still another issue to compensate lagged service in the error-pron channel environment. Thus, we do not make any explicit mathematical assumptions about the error model in our framework.

We assume that a transmission link is composed of time slots with fixed duration. Let L and l_i be a slot size and the packet size of a flow i in bits, respectively. Let us again assume $l_i = nL$ for a flow i, where n is a positive integer. Without loss of generality we assume that a time instant t indicates a slot which lasts during $[t, t + \tau)$, where τ is a slot duration.

Whenever packets (single or burst of packets) from on going flows arrive, one or more time slot are assigned in the *service order table.* The service order table has the following properties:

- it is a queue architecture handling packets from each flow
- its elements contain the following information; flow type, flow index, service tag which is the reserved service time instant for a packet
- its elements are sorted in ascending service tag order and handled as in a single linked list.

Each arrived packet is inserted into the service order table and serviced one by one from the head of table at the start of time slots. Even though the packets are served at the beginning of each time slot, the virtual start time calculated by (1) and (2) is not quantized to a discrete time instant, i.e., the starting points of slots. It is because the virtual service time tag is used as the index key to sort the relative order in the service order table and due to this policy, each packet could be differentiated in service order during a given scheduling period.

4 Performance Evaluation

In this section we evaluate the performance of DGFQ, in the measurement indices i.e., throughput, average delay and maximum delay. First we described the simulation environment then, discussed about the results.

4.1 Simulation Environments

We used simulations method to evaluate our proposed algorithm. For the simulation, we assumed 2Mbps wireless channel which is typical capacity of current wireless mobile networks. We implemented DGFQ scheme in error free channel model to concentrate our evaluation work on the key features of proposed scheme, i.e., delay guaranteed scheduling. In the simulation, we selected average

Table 1. Summary of Traffic Source.

	Voice	Video	Data
Model	ON/OFF (ON:OFF=1:1.35)	Modified MPEG	Poission & Truncated Pareto
Delay Deadline	20 ms	1000/24 ms	-
Data Rate	32kbps	114kbps	300kbps
Relative Weight	0.0016	0.057	0.15

delay, maximum delay and throughput as the performance measures. Detail definitions of these measure are explained in the following Section 4.3. Moreover, we compared these measures for the following three fair queueing schemes; SFQ, DGFQ, D-FQ[7]. SFQ also regarded as a special case of DFGQ where the *service differentiation coefficient* $\alpha = 1.0$. For the case of D-FQ is simulated to get the reference value of the delay performance because it is explicitly guarantee delay deadline with somewhat costly complex system architecture with computational complexity of $O(N_{AW}N_{RT})$ where N_{AW} is the number of allocation windows and N_{RT} is the number of realtime flows [7], and it also meaningful to compare the performance of our simple scheme with computational complexity of $O(\log N)$ where N is the number of flows in the system. Simulation performed for 1000 seconds, further, repeated 20 times to get more refined results.

4.2 Traffic Source Models

In this work we choose the same traffic model used in [7] to make the results be comparable. In the simulation, a voice flow is modelled as ON-OFF signal with ON and OFF duration having exponential distributions. A video flow is modelled by modified MPEG source, where there are three types of frame, i.e., I, B and P frames. Each frame size is determined by a Lognormal distribution with a specified mean and standard deviation. A video source generates 24 frames per second. Data flow is modelled by Poisson arrival with truncated Pareto distributed burst size. Table 1 shows the traffic model parameters used in this simulation, where voice and video traffic are assumed to be real time flow while data traffic is assumed to be non real time flow.

4.3 Results and Discussions

To evaluate the performance of DGFQ, we choose throughput, average delay and maximum delay as performance measures. The definitions and simulation results are given in the following.

Throughput. We used *throughput* as a fairness measure, which is total transmitted packets during the whole simulation duration, say, 1000 seconds. As described in previous Section 3, basically DGFQ does rearrange the service order for the packet from a set of flows according to the individual flow characteristics.

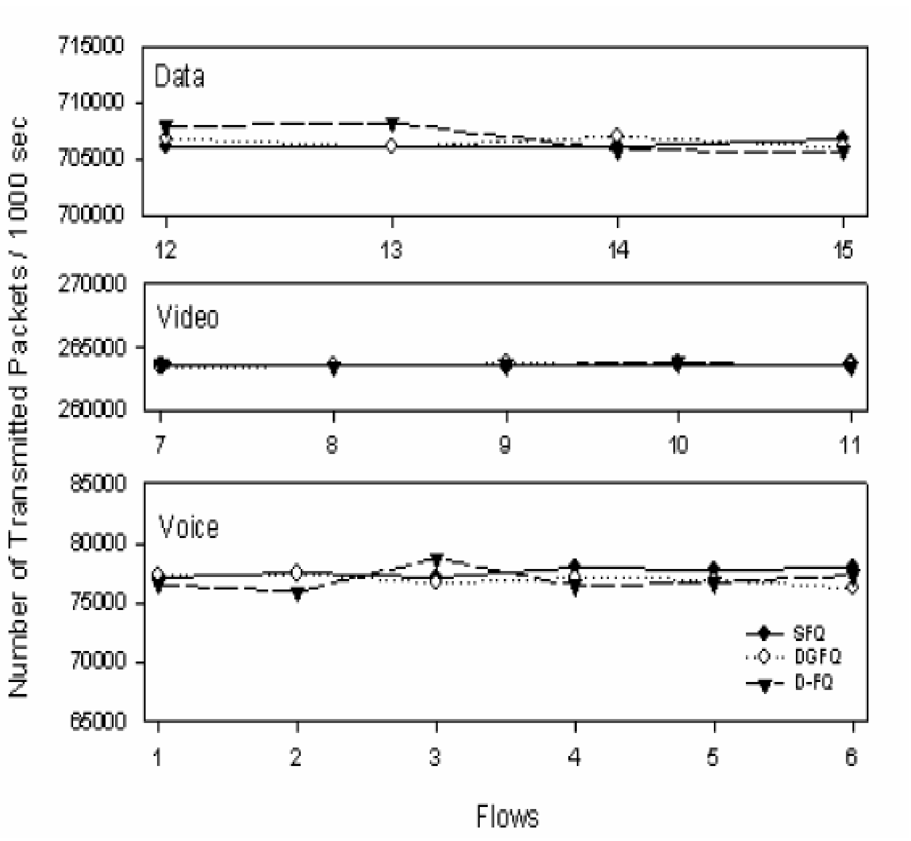


Fig. 2. Total Transmitted Packets for Each Flow by Three Schemes.

This is the only difference from SFQ. Thus conceptually, there is no changes from the results of SFQ, as far as throughput is concerned. Again, fairness among flows is guaranteed analytically by (5). Therefore, as shown in the Fig. 2, it is natural to conclude that there is no difference in throughput for respective flow classes, i.e., voice, video and data.

Average delay. In our work *average delay* is defined as the average time interval between the arrival and departure of a packet for a certain time duration. To compare the average delay of three schemes, we choose 3 classes of 6 real-time video flows, then assign α value for each class, 1.0 (for SFQ), 0.8 and 0.6 respectively.

As shown in the Fig. 3, the *service differentiation coefficient* α is the key parameter to control delay performance. When $\alpha = 1$ our proposed scheme is identical to the start time fair queueing(SFQ) scheme. With varying α we can

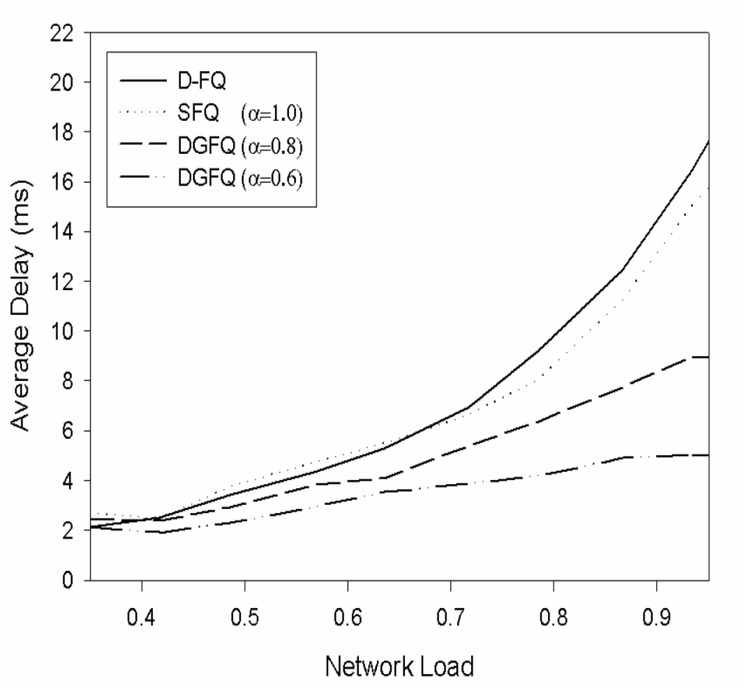


Fig. 3. Average Delay for Some Values of α with Varying Average Utilization.

handle classes of flows, having different delay bound requirements, maintaining the same fairness guarantees of flows. From the figure, our new scheme outperforms D-FQ and SFQ, independent of network load conditions. Specifically, with lower value of α, we can precisely control average delay requirement. Naturally, average delay increases for all scheme with increasing network load.

Maximum delay. The *maximum delay* is another critical performance measure for real time multimedia flows. We define maximum delay as the maximum time interval between the arrival and departure of a packet in the system in a certain duration of time, say, simulation duration. We can get the results simultaneously with average delay from the same simulation. From the figure (Fig. 4), we can conclude that maximum delay could be also controllable with our new scheme, though the performance is not as good as for the average delay case. Specifically, as far as for the low network load conditions are concerned, say below 60 % of full network load, maximum delay performance is comparable to that of D-FQ. On the contrary, for relative high load conditions, say above 80 % of full network load, the maximum delay performance of DGFQ get worse with increasing network load. However, in either case, DGFQ controls the maximum delay more tightly than SFQ algorithm does.

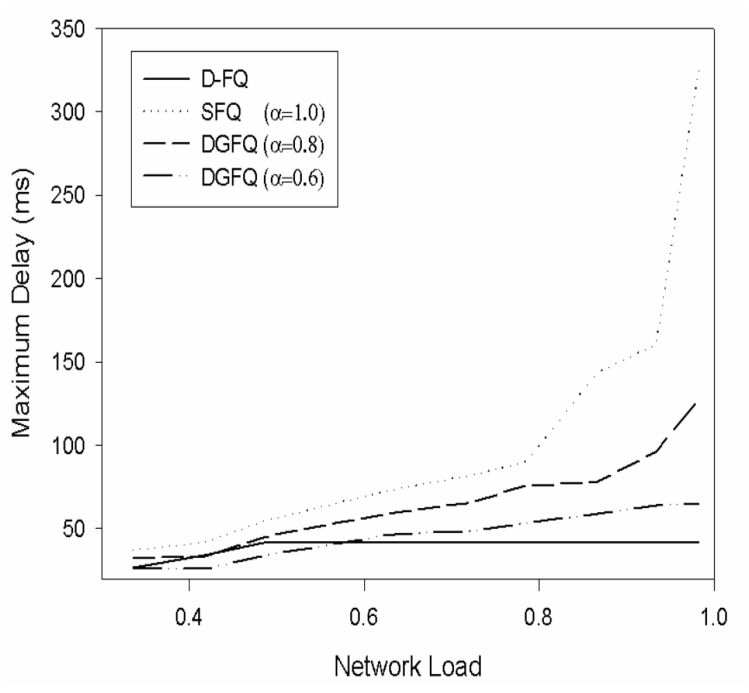


Fig. 4. Maximum Delay for Some Values of α with Varying Average Utilization.

For the numerical details, the target performance measure, i.e., maximum delay bound of 50ms, typical value for the realtime video traffic, is satisfied by DGFQ for the network load up to 60%, when $\alpha = 0.6$, while for SFQ, which is $\alpha = 1.0$ case, meets the target only for the low load condition, say, below 30 % of full network load. D-FQ remains in the satisfied condition all the way by virtue of complex explicit delay guarantee (EDG) algorithm. It is noteworthy that our proposed scheme could manage maximum delay guarantee by simpler mechanism than D-FQ. In addition, delay performance is still under control of α even in the heavy load conditions where traffic congestion could be occured. As shown in the figures (Fig. 4 and Fig. 3) delay is more tightly controlled when $\alpha = 0.6$ than when $\alpha = 0.8$ for high load conditions.

5 Conclusion

We proposed a new delay guaranteed fair queueing scheme, DGFQ. This scheme provides better delay performance for DG class flows than NG class flows at the same fairness guarantee. Moreover, this scheme guarantees bounded delay of multimedia services with simpler algorithm than D-FQ [7] which explicitly guaranteeing bounded delay with rather complex queueing policy. From the simulated results, our proposed scheme outperforms for average delay than other schemes in comparison. Proposed scheme performs well for maximum delay bounds except for the high network load conditions with no degradation of fairness guarantees. With some modifications and performance tuning, our proposed DGFQ scheme will be a good alternatives for the fair queueing algorithm guaranteeing QoS, in the context of delay, for the multimedia wireless packet networks.

Finally, we just consider about error-free wireless channel which is too idealistic to apply our work in the practical systems. So, much more work should be done for the error-prone wireless channel case as a future work.

Acknowledgement. This work is supported in part by BK21 Project, MOE and ERC-UFON, MOST.

References

1. H. Fattah and C. Leung, "An overview of scheduling algorithms in wireless multimedia networks, "*IEEE Wireless Communications*, Oct. 2002, Vol.9 No.5, pp. 76–83.
2. A. K. Parekh and R. G. Gallager, "A generalized processor sharing approach to flow control in integrated services networks: the single-node case , "*IEEE/ACM Transactions on Networking*, Jun. 1993, Vol.1 No.3 pp. 344–357.
3. A. Demers, S. Keshav, and S. Shenker, "Analysis and simulation of a fair queueing algorithm," *Proc. ACM SIGCOMM*, Sep. 1989, pp. 1–12.
4. S. J. Golestani, "A self-clocked fair queueing scheme for high speed applications," *Proc. INFOCOM94*, Apr. 1994, pp. 636–646.
5. P. Goyal, H. M. Vin and H. Cheng, "Start-Time Fair Queueing: A Scheduling Algorithm for Integrated Services Packet Switching Networks," *IEEE/ACM Transactions on Networking*, Oct. 1997, Vol.5 No.5, pp. 690–704.
6. F. cuomo, A. Baiocchi and R. Cauteiier, "A MAC Protocol for a Wireless LAN Based on OFDM-CDMA," *IEEE commun. Magaz.*, Sep. 2000, pp. 152–159.
7. S. Lee, K. Kim, A. Ahmad, "Delay and data rate decoupled fair queueing for wireless multimedia networks," *Proc. GLOBECOM '02*, 2002, vol. 1, pp.946–950.
8. H. Zang, "Service disciplines for guaranteed performance service in packet switched networks," *Proc. of the IEEE*, 1995, vol. 83, no. 10, pp. 1374–1399.
9. V. Bharghavan, A. Demers, S. Shenker, and L. Zhang, "MACAW: A Medium Access Protocol for Indoor Wireless LAN's," *ACM Comput. Commun. Rev.*, Oct. 1995, vol. 24, pp. 215–225.

Mobility Model for Multimedia Mobile Radio Network

Sonia Ben Rejeb[12], Sami Tabbane[1], and Zièd Choukair[2]

[1] Ecole Supérieure des Communications de Tunis
Unité de Recherche en Technologies de l'Information et de la Communication
Route de Raoued, km 3.5, 2083 Cité el Ghazala – Tunis
Sonia.benrejeb@enst-bretagne.fr
Sami.Tabbane@supcom.rnu.tn

[2] Ecole Nationale Supérieure des Télécommunications de Bretagne
Laboratoire d'Informatique des Télécommunications
Technopole Brest-Iroise-Bp 832-29285-Brest Cedex-France
Zièd.choukair@enst-bretagne.fr

Abstract. The aim of this work is to develop a mobility and traffic model for multimedia mobile radio networks. These models are developed according to measures from two considered service areas and they analyzed by simulation. Users issue multimedia calls which are to be managed according to a traffic model. The simulation is used to improve the network management and it ensures that network resources are used efficiently. Indeed, traffic growth in the network needs a continuity in the QoS management between the planning phase and the use phase or operating phase.

Keywords: Multimedia mobile radio network, mobility model, traffic model, system planning.

1 Introduction

In order to offer multimedia services with optimal QoS, the planning of a cellular network is a necessary task which is considered as complex by 3G network operators. Indeed, a bad planning of the network will lead to a bad call quality, a high call dropping rate and a high blocking rate. Furthermore, this causes supplementary costs and a shortage of profit for the operator. In this paper, we develop mobility and traffic models. In order to elaborate these models, we have used measurements collected from a live GSM network (i.e. from operation and maintenance center (OMC meters).

An area includes potential sites to which weights were assigned with real positions of base stations (Figure 3). This area is described by parameters which are the input of the mobility model. This mobility model is based on the study of the users' flows on the roads and at first allow us to determine the transition rate, among different cells. These also give the probability that a user transits towards another cell, and the handover rates.

The paper is organised as follows: the second section presents the state of the art. The development mobility model for system dimensioning is examined in section III. The environment and simulation models are presented in section IV. Simulation results and analysis are discussed in the fifth section. Finally, we finish by conclusion and some perspectives.

2 State of the Art

Users' mobility is a major characteristic of mobile radio systems. It allows the users to benefit from network service in all the service area and to communicate while they move in the network. Two principal functionalities of mobile radio network ensure this mobility: itinerancy management [1] and intercellular handover.

There are different causes for intercellular handover initiation. In this work, we concentrate on the handovers due to changes of cells during a multimedia call. Other reasons may justify handover such radio quality [2] maintenance will not be considered.

Each handover modifies the distribution of the traffic in the network. In this work, we study the influence of users' mobility on the system performances. Several models of cellular traffic mobility were presented in the literature. A constant density of users for macro-cellular systems is considered in [3]. The limits of this approach for micro or pico-cellular systems are shown in [4]. Other works [5] propose solutions with pseudo-aleatory moves.

Another approach is based on stochastic models which represent in a microscopic way the movements of each user. Such approaches are analyzed by simulations with discrete events. A vehicles' mobility simulator is presented to determine the localization update rates in [6], such a simulator is used to study micro-cellular networks performances in [7].

A deterministic model is based on vehicles flows is presented for highway networks considering at a dimension [8]. This model is restricted to study in a macroscopic way the total dynamic total of the system. The users' densities and their incoming and starting rate at the highway are described according to their site and time. A cyclic network example is studied from a similar model [9].

A simplified version of mobility model described in [10] and based on the graph of street consists an adapting the available road traffic to calibrate the model. It allows to study systems with two dimensions. The speeds of the different classes of users are constant in the whole system and the moves of the mobiles are independent. The lack of statistical data on users mobility is a major problem for mobility analysis. To calibrate the model, the only information they have is vehicles' flows per day in the network of streets.

Our study take place in the previous context, it is focused on the development of a mobility model considered for four traffic models (voice, video, ftp and www). The mobility model is adapted to urban areas. It is calibrated by measurements carried out from the system at the existing sites and road networks.

To dimension the system and analyze its performances, the users' mobility must be considered in four selected traffic models. These models do not distinguish among neither the different mobiles nor their movements but model macroscopic mobility in the system. retained. In this study retained models are based on the users' transitions rate (cell input/output calls) among cells as mobility parameters.

The mobility model allows to analyze the total dynamic of the system, to calibrate the capability of the system and to regulate the provision of the resources thereafter.

3 The Mobility Model

Users in an urban area move especially along the streets. Thus, we define the graph of streets RG (X'' , R) to model the mobility of users in the streets.

In this graph, a street is represented by a ridge $r_{kt}=(x''_k,x''_t)$ which binds the two ends of the street. A street with double direction is represented by two ridges $r_{tk}=(x''_t,x''_k)$ and $r_{kt}=(x''_k,x''_t)$. We attribute to each ridge r_{tk} a weight ϕ_{tk} which represents the user flows in the street (figure 1). The streets graph is thus composed of a set of nodes $X''=\{x''_t\}$ at the intersections of streets connected by directive ridge of set $R=\{r_{tk}=(x''_t,x''_k)\}$.

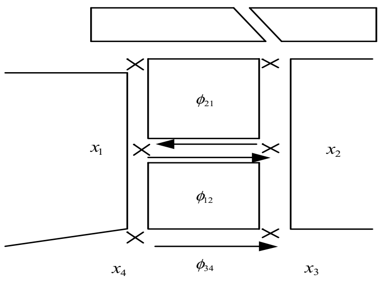


Fig. 1. Example of streets graph in an urban area

3.1 Users' Flows

Users flows among the cells network allow to determine our model parameters. If a street is covered by several cells, we distribute the flows of the users in the street on covering cells.

The streets are assigned to the different areas of cover of the BTS. A street which is inside an area of covering is influenced by each of covered cells. We assume that a user in such a street is connected with a certain probability to one of the covering cells.

The total flow ϕ_{tk} in a street $r_{tk}=(x''_t,x''_k)$ (where x''_t and x''_k are covered by a number M_{tk} of cells) is divided in sub-flows $\phi^{(j)}_{tk}$, where $\phi^{(j)}_{tk}$ the sub-flows attributed to the cell *j*. This distribution can be expressed by:

$$\phi^{(j)}_{tk}=\phi_{tk}.\pi^{(j)}_{tk} \quad \text{with} \quad \sum_j \pi^{(j)}_{tk}=1 \tag{1}$$

The distribution of flows is illustrated for example in figure 2 with a street which is covered by two cells. The flow ϕ_{12} allotted to (x''_1,x''_2) is distributed for example on cells x'_n and x'_p covering: $\phi_{12}=\phi^{(n)}_{12}+\phi^{(p)}_{12}$.

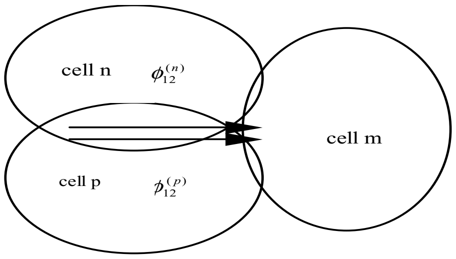


Fig. 2. Distribution of users' flows on different covering cells

In the following, we consider the case where: $\pi_{tk}^{(j)} = \frac{1}{M_{tk}}$ and the distribution of users' flows is as follows:

$$\phi_{tk}^{(j)} = \frac{\phi_{tk}}{M_{tk}} \tag{2}$$

Mean flow of users on road r_{tk} is given by: $\phi_{tk} = \frac{n_{tk} v_{tk}}{l_{tk}}$.

Where : M_{tk} : the cells number covering road r_{tk} ,
n_{tk} : the users' number on road r_{tk} ,
l_{tk} : the roads length r_{tk} ,
v_{tk} : the users' speed on road r_{tk} .

3.2 Users' Flows among Cells

Let us determine user flows among network cells. Users' flows ϕ_{tk} in the streets of the network are distributed on different cells of the network. $\phi_{tk}^{(j)}$ introduced values before take account into of this distribution. In order to determine flows between cells, we must defining the place where the users which moved from cell x'_j towards cell x'_n under take handover. We assume in the following that the mobile under take this handover as late as possible, before leaving the area definitively covering the cell x'_j . Consequently, the flow of users Θ_{jn} from the cell x'_j towards cell x'_n is given by the sum of the flows $\phi_{tk}^{(j)}$ in the streets crossing the limit covering area of x'_j towards x'_n :

$$\Theta_{jn} = \sum_{t \in j, k \in n, k \notin j} \phi_{tk}^{(j)} \tag{4}$$

This sum contains all the streets which connect a node x''_t covered by cell x'_j to a node x''_k which is covered exclusively by cell x'_n (but not the cell).

3.3 Transition Rates among Cells

With users' flows Θ_{jn} among cells, users' transition rates between cells α_{jn} can be given for micro-cellular systems, by taking account different axes of mobility of the users. We deduce the users transition rate from cell x'_j towards cell x_n of users' flows Θ_{jn} and users' number N_j in the cell x'_j according to:

$$\alpha_{jn}=\frac{\Theta_{jn}}{N_j} \tag{5}$$

For constant density $\sigma_{tk}^{(j)}$ in a street $r_{tk}=(x''_t,x''_k)$ with length l_{tk}, the number of users $N_{tk}^{(j)}$ in connection with cell x'_j is given by $N_{tk}^{(j)}=\sigma_{tk}^{(j)}l_{tk}$. The users' density $\sigma_{tk}^{(j)}$ is given by $\sigma_{tk}^{(j)}=\frac{\phi_{tk}^{(j)}}{v_{tk}}$, where v_{tk} is the speed of the users in the street.

The number of users N_j in cell *j* which is given by the sum of the users who are in streets which are covered by cell *j* and which are attached to the cell:

$$N_{tk}=\sum_{r\in j}\sigma_{rs}^{(j)}l_{rs} \tag{6}$$

The users' transition rate from x'_j towards x_n, can then be calculated by:

$$\alpha_{jn}=\frac{\Theta_{jn}}{N_{tk}}=\frac{\sum\limits_{t\in j,k\in n,k\notin j}\phi_{tk}^{(j)}}{N_{tk}} \tag{7}$$

3.4 Global Transition Rates from Cell j towards Its Neighbors

The parameter α_j expressing global transition rates from cell *j* towards its neighbors is linked with α_{jn} by the following relation:

$$\alpha_j=\sum_n\alpha_{jn}=\sum_n\frac{\sum\limits_{t\in j,k\in n,k\notin j}\phi_{tk}^{(j)}}{N_{tk}} \tag{8}$$

3.5 Intercellular Handovers Rate

Using the transition rate among cells α_j in the system, we calculate the intercellular handover rates by the use of service laws such as those for voice, video, www, ftp which we note μ_{trafic}, μ_{trafic} being μ_{trafic}, μ_{video}, μ_{www}, μ_{ftp}.
Taking into account user mobility, service rates are given by the following equation:

$$\mu_j=\mu_{trafic}+\alpha_j \tag{9}$$

3.6 Traffic Load

Total load of traffic ρ_j in cell *j* is given by the following equation:

$$\rho_j = \frac{\lambda_j + \lambda_j^H}{\mu_{trafic} + \alpha_j} \tag{10}$$

Where: $\lambda_j^H = \sum_k \lambda_{kj}^H$ is the incoming handovers rates at cell j since the whole of adjacent cells and λ_j represents the incoming new call rates for each traffic model.

Balancing the Flows

The incoming handovers rates in a cell j are given by the following flow equation which express a balance between:

$$\lambda_j^H = \sum_k \alpha_{kj} [(1 - P_{bk}) \frac{\lambda_k}{\mu_{trafic} + \alpha_k} + (1 - P_{dk}) \frac{\lambda_k^H}{\mu_{trafic} + \alpha_k}] \tag{11}$$

Where, P_{bk} the blocking probability of new calls, P_{dk} the probability of dropping handover << 1.

4 The Environment Model - Simulation Model

With some key indicators (Tables 1, 2, 3) from measures carried out from the system. At first the simulation (figure 4) allows us to define mobility and traffic characteristics of users in the area of the two considered configurations (Figure 3), then, to obtain QoS statistics and a signalling load for network configuration and a given list of parameters (figure 10).

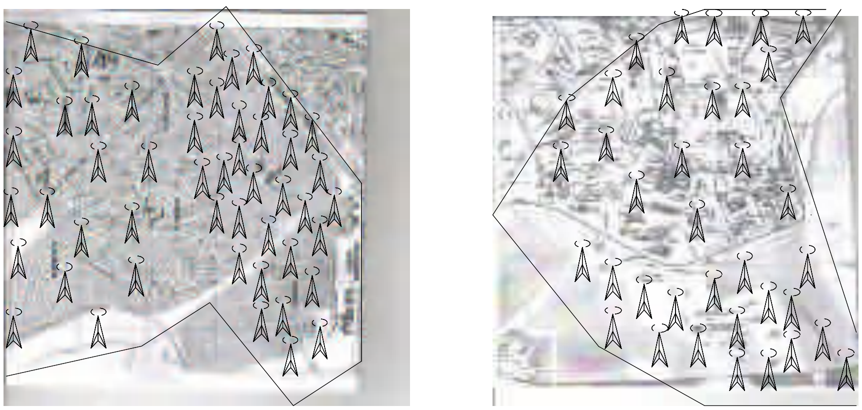


Fig. 3. left: Configuration 1 (inspired from Hached Tunis, scale: 1/12000, 50 cells, 1BTS/cell)_ right: Configuration 2 (inspired from Sfax center, scale: 1/10000, 36 cells, 1BTS/cell.

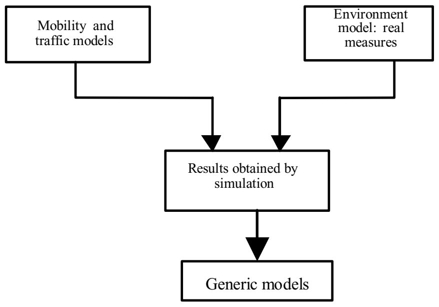

Mobility and traffic models
Environment model: real measures
Results obtained by simulation
Generic models

Fig. 4. Generic models generating process

4.1 Mobility Model Calibration

Before applying the mobility model in the considered configurations, it is necessary to adapt the highway traffic values available for our application in order to calibrate the model.

The operators can adjust the different weights of the highway network by measures in the existing networks and extrapolate them for new networks.

Based on the configurations 1 and 2, the mobility model presented above and considering an approach calibrated in [10, 11, 12, 13, 21, 22], the simulation model to be designed is then formed by a set of some real input parameters given in Tables 1, 2 and 3.

Table 1. Domain configuration

Parameters	Conf. 1	Conf. 2
Measure period	12am to 1pm	12am to 1pm
Configuration radius	1.7 km	2 km
Number of cells	50	36
Number of base stations	50	36
Capacity of cell	28 channels/cell	45 channels/cell

Table 2. Measurements from network

Parameters	Conf. 1	Conf. 2
Road lengths	20 m to 1.7 km	20 m to 2.6 km
Number of users in the streets	20 to 1500 users	20 to 1700 users
Load traffic voice	7.787 Erl	6.009 Erl

Table 3. Assumptions on domains

Parameters	Conf. 1	Conf. 2
Mobile speed in the streets	3 to 60 km /h	3 to 60 km /h
Call blocking probability/cell	1%	

The first phase of measures consists on defining key performance indicators. Those also come from measures carried out from the system.

In this study, the key indicators to be adjusted and optimized are taken from two configurations. Configuration 1 which consists of an area of 1.7 km and covered by 50 cells, and the other with an area of 2 km and covered by 36 cells positioned as shown in Figure 3. In each of these two configurations, we consider one BTS per cell, roads, avenues narrow streets,...

Assumptions

- A quasi regular distribution of cells is chosen in order to dissociate the mobility aspects from the cell distribution aspects [10],
- the cell radius is: $R_{cellL} = \frac{d_{BTS2-BTS1}}{2}$,
- the mobile users in a cell are served by a base station (BTS),
- one site can gather many BTS,
- streets are represented by a line r_{tk},
- the mobiles speed in a road r_{tk} is v_{tk},
- the roads lengths are l_{tk},
- the users' number on the roads is n_{tk},
- the BTS representation in the map is done according to deployment data,
- one cell contains c channels for voice, c'channels for video, c''channels for the web.

Base stations are located near the streets and cover their associated cell. The mobility of the users is along the streets and is represented by flows. The network is assumed to be cell-based with support for diverse traffic types. The goal is to estimate the requirements for resources from mobiles which are currently neighboring cells and which might potentially move to the current cell. This estimate is then used to appropriately reserve resources in the current cell for potential handover connections.

The environment model is represented as follows:

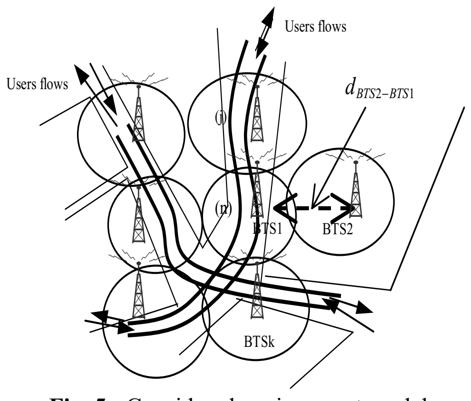


Fig. 5. Considered environment model

Handovers modify the traffic distribution. So, we must consider user mobility in the measures and the whole performance analysis.

In order to perform system measures and to analyze its performance, users' mobility must be considered in the analytic traffic models used. These models distinguish neither the different mobiles nor their movements, but if we want to model we have to consider the macroscopic mobility in the system [10]. The traffic models of this study are based on the users' mobility parameters, such as transition rate (input/output calls) among cells (equation 7). The proposed parameters are considered for the two configurations to design an optimized mobility model (figure 6).

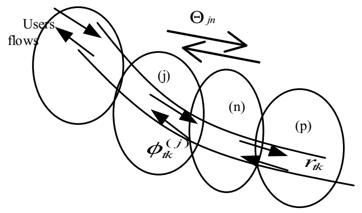


Fig. 6. Mobility model used in the streets of the considered environment

5 Results

According to the calibrated mobility and traffic models, users' flows on the roads allow the determination of the users' flows (input/output calls) among cells of the network (equation 7). These values lead us next to deduce handover rates as well as traffic loads in a cell to evaluate the global load traffic in the global system.

Figure 7 represents the calculation (equations 2, 4) of the user flow variation for the two considered configurations (Conf.1 and Conf.2) according to the lengths of the roads. The results presented show that the user flows evolve almost with the same profile for the two configurations. Their values increase when roads length does the same until a certain limit and after decrease quickly. For those two configurations, user flows are important for roads which are located at dense areas (for example for roads whose lengths vary between 0.08 and 1.3 km) and the flows reach 1000 users per second for each cell. Then we notice that there is a size of road for which the flow of users is maximum. These roads correspond to high traffic loads. Users' flows values reach a maximal value of 1.3 users/cell/s for road length equal to 0.52 km in the configuration 1 (1.1 users/cell/s for roads length equal to 1.33 km in the configuration 2). Beyond those two maximal values (roads length >1.44 km) the flows of the users decrease and tend towards minimal values (respectively 1 user/cell/s for Conf.1 and 0.7 users/cell/s for Conf.2). Users' flows reduce for roads which are longer than a certain value which could be out of the dynamic area of the two configurations: roads far from the crowded area of each configuration.

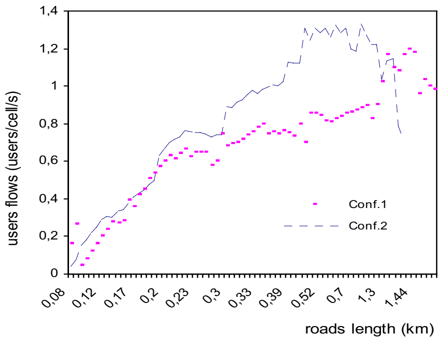


Fig.7. Users' flows $\phi_{tk}^{(j)}$ according to roads length l_{tk}

According to the results shown in Figure 7, the flows of users can be modelled by the following equation:

$$\phi_{tk}^{(j)}(l_{tk})=(\mu+\delta\log(l_{tk}))\ users/cell/s \tag{12}$$

Where: $\mu=1$, $\delta=0.911$

User flows calculation, based on Figure 7, leads to the density evaluation of users per unit area (km^2). The results are presented in Figure 8 and show that the density is maximal for medium user speeds. In configuration 1, the density value equals 0.07 users/km^2 for a speed between 12 and 27 km/h. This value becomes much more important (0.13 users/km^2) in the second configuration. Beyond those values (>27 km/h) density values decrease and tend towards a minimal value equal to 0.02 users/km^2. We conclude that the density increases for users moving with medium speeds (near dense service areas where roads are not too long (Figure 7)) and reduces if they move with more important speeds (since they are far from dense service areas). Indeed, results of Figure 8 comply with those shown in Figure 7.

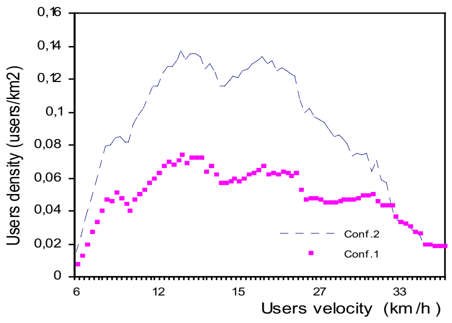


Fig. 8. Users' density $\sigma_{tk}^{(j)}$ according to users' speed v_{tk}

According to the results obtained in figure 8, the density of users can be modelled by:

$$\sigma_{tk}(v_{tk})=\frac{1}{\sqrt{2\pi\zeta}}\exp\left(\frac{-(v_{tk}-\gamma)}{2\xi}\right)\ users/km^2 \tag{13}$$

Where: $\gamma=15$, $\xi=30.69$

The transition flows among cells deduced from equation 4 are presented in Figure 9. For example, in configuration 2, and with a speed varying between 6 and 33 km/h, transition flows increase with the speed. They reach a constant maximal value of 0.04 users/cell/s in the configuration 2 for a speed equal to 27 km/h. Beyond those values, transition flows among cells are considered to be those for fast vehicles which move from the current cell to the destination cell in a regular manner. They go through big streets, avenues, located generally far from crowded service area centers (Figure 7). Their transition flow should be low according to the regularity traffic. For low speed traffic flows, vehicles move generally with less order and less regularity through cells. These last vehicles move in a dense area where users'density is high (Figure 8) such as for small streets, representative area, administrative area,…However, this less regular traffic is characterized by low speeds which make the transition flows rather low.

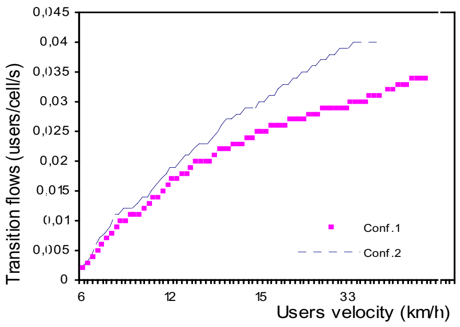


Fig. 9. Transition flows between cells $\Theta_{tk}^{(j)}$ according to users' speeds v_{tk}

The transition flows, according to the speeds of the users, (Figure 9) are given by:

$$\Theta_{tk}^{(j)}(v_{tk})=\tau(\sqrt{v_{tk}}-\varepsilon)\ users/cell/s \tag{14}$$

Where: $\tau=9e^{-3}$, $\varepsilon=2.44$.

We notice that the curves of the two configurations have almost the same profile for the two configurations which are considered in urban areas, and then have almost the same characteristics.

Indeed, we can conclude that these models could be applied to other similar service areas. The generic formulas extracted for each figure remain valid and they can be re-parameterized according to parameters such as: period of measurement, service area radius, distance between base stations sites,…

Mobility and traffic models are then applied to the considered configurations (figure 10). They will allow us to evaluate the different performance parameters such as[24]: traffic load, handover rate, grade of service (GoS),...

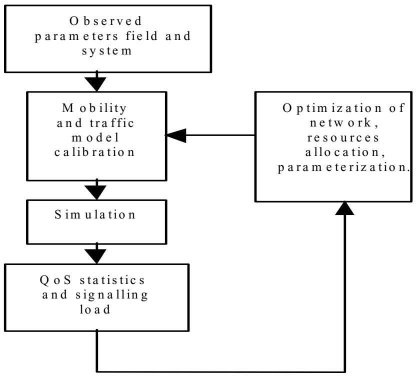

Observed parameters field and system
Mobility and traffic model calibration
Optimization of network, resources allocation, parameterization.
Simulation
QoS statistics and signalling load

Fig. 10. Control process to drive resource allocation

6 Conclusion and Future Work

In this study, we have proposed mobility model which can be used to model the different uses met in the considered environment. The mobility of the users is considered for four models of traffic retained from our bibliography such as : voice, video, www and ftp.

To dimension the system and analyze its performances well, the mobility model is adapted to the urban areas. It is calibrated by measurements carried out from the system from existing areas and road networks. It allows to analyze the globally dynamic of the system, calibrate system capability and regulate the provision of resources.

This mobility model is based on the study of the user flows on the roads and allow us at first to determine the transition rate between different cells. These also give the probability that a user transits towards another cell, and the handover rates. It is developed in order to evaluate handover rates among cells for the different type of traffic, and to open perspectives considering them for more precise multimedia traffic load analysis as well as to facilitate the allocation of radio resources for network operators.

The proposed mobility model also open perspectives to determine at cell level or at system level the thresholds beyond which a call can be rejected and then know the limitations in terms of resource reservation: handovers rate, probability call dropping, assured traffic, grade of service…These parameters will then be used as QoS criteria on the acceptance of new calls. The idea is then to develop new mechanisms of allocation of resources, whose objective is to maintain, as much as possible the QoS end to end criteria similar to the threshold predefined for GSM network.

References

1. Tabbane, S.: Location management methods for third-generation mobile systems. IEEE Communication Magazine, August 1997.

2. Tabbane, S.: Réseaux mobiles. Edition Hermes 1997.
3. Thomas, R., Gilbert.H, Mazziotto.G.: Influence of the moving of the mobile stations on the performance of a radio mobile cellular network. Proc. Of the third nordic seminar on land mobile radio communications, September 1988.
4. Nanda, S.: Teletraffic models for urbain and suburban microcells : cell sizes and handoff rates. IEEE Transactions on vehicular technologiy, Vol.42, No.4, November 1993.
5. Foschini, G.J., Gopinath.B, Miljanic.Z.: Channel cost of mobility. IEEE Transactions on vehicular technology, Vol. 42, No.4, November 1993.
6. Seskar. I, Maric,V., Holtzmann.J, Wasserman, J.: Rate location area updates in cellular environement. 42 th IEEE Vehicular technology conference 1992.
7. Steel, R. and al.: Teletraffic performance of GSM 900/DCS 1800 in street microcells. IEEE Communications magazine, March 1995.
8. Leung, K.K.: Traffic models of wireless communication network. IEEE Journal on selected areas of communications, Vol 12, No.8, October 1998.
9. Montenegro. G, and al. Timedependent analysis of mobile communication trafic in a ring shaped service area with non uniform vehicule distribution. IEEE Transaction on vehicular technologiy, Vol.41, No.3, August 1992.
10. Bathelt, A. : Grafical models for analysis of design mobile wireless networks. PhD thesis ENST, 14 April 1998.
11. Tunisia Telecom: Network planning mobile radio system ,Tunis-Hached and Sfax cities. Technical Report, May 2002.
12. Ben Rejeb, S., Tabbane, S., Choukair, Z.: Network Planning for Mobile Radio Network. Technical Report, SUPCOM-ENSTB, May 2001.
13. Ben Rejeb, S., Choukair, Z.., Tabbane, S.: Mobility Model for Multimedia Mobile Radio Network. Technical Report, ENSTB-SUPCOM, August 2002.
14. Chandra, A., and al.: Characterisation of Mobility Patterns based on Cell Topography in a cellular Radio System. IEEE (ICPWC'99) 428-432.
15. Yeng, K.H., and al.,: On the modeling of www request arrival. ICPP Workshop (1999) 248-253.
16. Cheung, J.C.S., and al.: Network planning for third generation mobile radio systems. IEEE Communication Magazine, 32(11). November (1994) 54-59.
17. Thomas, R., and al.: Influence of the moving of the mobile stations on the performance of a radio mobile cellular network. Proceeding of the Nordic Seminar on Digital Land Radio Mobile Communications, 1998.
18. Ajmone, M.M., and al.: Performance analysis of cellular mobile communication networks supporting multimedia services. Mobile networks and applications 5 (2000) 167-177.
19. Lagrange, X. : Performance analysis of hierarchical cellular network. PhD thesis ENST, 11 Mai 1998.
20. Valois, F.: Modeling and evaluation of performance hierarchical cellular networks, PhD thesis University of Versailles, 14 Januray 2000.
21. ETSI.: Universal Mobile Telecommunications System (UMTS); Selection procedures for the choice of radio transmission technologies of the UMTS. UMTS 30.03, version 3.2.0, Avril 1998.
22. Dawood, A.M., Ghambari, M.: Universal Mobile Telecommunications System (UMTS); Selection procedures for the choice of radio transmission technologies of the UMTS. IEEE Transactions on multimedia, Vol. 1, No.1, March 1999.
23. Lesieur, L., Lefebvre, J.: Analyse-Mathématique. Armand Colin, Collection U- Paris. 1967.
24. Ben Rejeb S., Choukair, Z., Tabbane, S.: Resource Allocation and QoS Management for cellular Multimedia Network. Proc. GRES'03, (February 2003) 3-13.

On Reciprocal Altruism and Its Application to QoS

Nupur Kothari, Vartika Bhandari, and Dheeraj Sanghi

Department of Computer Science & Engineering,
Indian Institute of Technology Kanpur,
Kanpur, India
{nupur.kothari, vartika.bhandari}@alumni.cse.iitk.ac.in,
dheeraj@cse.iitk.ac.in

Abstract. With the growing usage of the Internet for resource-intensive applications e.g., streaming multimedia, best-effort service has ceased to be adequate. The result has been a spurt of proposals for guaranteed quality-of-service (QoS). Since network resources can be but limited, supply very often falls short of demand, thereby leading to competition to secure available resources. The factors that come into play in such situations, are network mechanisms, as well as complex economic behavior. A framework for providing cost-effective QoS needs to address both these aspects. Selfishness as a guiding motivation for action is widely observed in nature and has also been applied to QoS in the form of approaches based on competitive game theory. However, as has been observed in the natural world, long-term selfish motives may also give rise to apparently altruistic actions. This notion is well-captured in the phenomenon of Reciprocal Altruism, and has been modeled in game theory as the Iterated Prisoner's Dilemma. We propose Reciprocal Altruism as the guiding principle for a QoS framework that allows for cooperation between otherwise competing flows, leading to long-term benefit for all. We also present simulation results to validate the notion that cooperation can lead to better end-user experience.

1 Introduction

With the growing usage of the Internet for resource-intensive applications e.g., streaming multimedia, best-effort service has ceased to be adequate. An emphasis on guaranteed Quality-of-Service (QoS) has emerged, propelled by the increasing volumes of traffic with stringent QoS requirements. The existing Internet infrastructure is not equipped to handle such services. Hence various frameworks e.g., IntServ [1], DiffServ [2] etc. have been proposed to provide QoS. While these address the mechanisms required to provide QoS guarantees, the policies for resource allocations made, are beyond their purview. By the basic tenets of economics, network users need to pay for any guarantees sought. With limited availability of network resources, intense competition can emerge to secure the above guarantees. When the peak-rate of data transmission is significantly different from the average rate, it becomes difficult to maintain consistent quality without the risk of over-allocation and consequent over-expenditure. This may also lead to under-utilization of resources.

Real time applications like streaming video, VoIP, online gaming, etc., often exhibit variable traffic rates, with the peak rate significantly higher than the average rate. This leads to time-varying bandwidth requirements. Variable bandwidth requirements may also be imposed by the use of protocols like Split-and-Merge [3]. The perceived quality of the transmitted content is of great importance in such applications. While it is not an easy task to map network QoS to user QoS, yet it has been observed that perceived quality is affected by such factors as Packet Loss Ratio (PLR) and the loss burst lengths (i.e. number of consecutive packets lost). The relationship between perceptual quality and PLR has been studied in numerous works e.g. [4] and [5]. Voice traffic exhibits extreme loss sensitivity with a PLR of over 2% causing appreciable voice degradation. For video streams, the loss of different kinds of packets have different end-effects, depending on the encoding. As an example, in MPEG video, certain frames are critical and as long as they are not lost, it may be possible to achieve acceptable quality even with PLRs of the order of 20%. These observations can form the basis of a notion of *survival* for multimedia flows over discrete time intervals corresponding to the transmission of a Group of Pictures (GOP). In order to ensure good quality, survival rates should be high over the flow duration, even with fluctuating data rates. At the same time, network utilization and economic concerns need to be taken into account.

Attempts have been made to partially address these issues by devising dynamic provisioning and pricing schemes to allow for adaptive QoS negotiation. Some techniques for enforcing conformance of encoder output to the pre-decided traffic envelope have also been proposed [6]. An alternative approach is to retain a simple provisioning and pricing structure with reasonably long-term service agreements, and accommodate transient bursts within this framework, so that flows are able to survive such burst phases. We propose such an approach wherein flows can go in for moderate resource allocations and cooperate with each other to tide over transient bursts and maintain consistent end-quality. Our proposal envisages a cooperative game theory based approach, as opposed to earlier proposals for application of competitive game theory, as in [7], [8] and [9], amongst others.

2 Characteristics of Video Traffic

There are generally three types of frames in encoded video. In the case of MPEG-4 encoding , they are referred to as the Intra Frame (I-Frame), Predictive Frame (P-Frame), and Bidirectional Frame (B-Frame). An I-Frame is the most basic frame used in video compression and stores all the data required to display the frame. A P-Frame is smaller in size as it builds upon the previous I or P-Frame and only stores the difference. A B-Frame is built upon two frames, one I or P before it and one I or P after it. Therefore the loss of a P-Frame or an I-Frame would affect all P and B-Frames dependent on it. In fact a notion of Group of Pictures (GOP) exists wherein a GOP is a block of frames comprising one or more I Frames followed by P or B-Frames. It is largely self-sufficient in that inter-GOP dependence is either non-existent or restricted to a single B-Frame. Hence, losses in one GOP do not really affect the quality of other GOPs. These observations about video traffic open up the possibility of looking at QoS as a per-GOP phenomenon.

3 Reciprocal Altruism

Altruism is defined as social behavior that benefits an unrelated individual while being detrimental to the individual displaying altruism. Motivations for altruistic behavior may be varied. However, it has been observed that often apparently altruistic behavior is actually motivated by self-interest. This is termed as *Reciprocal Altruism* [10]. This kind of behavior is characterized by the acceptance of short-term costs/losses in the expectation of a long-term benefit based on reciprocation of the *altruistic* gesture by the current beneficiary.

Reciprocal Altruism has been observed in numerous organisms e.g. vampire bats (*Desmodus rotundus*) [11]. Vampire bats have a very high metabolic rate and hence starve to death if unable to find food for two days running. As such, it has been observed that a bat that was unable to find food solicits the same from a roost-mate and is often helped based on the expectation that if the benefactor ever fails to find food, the current beneficiary would be willing to help in return.

Reciprocal Altruism can sustain itself as a long-term behavioral model only if the gain to beneficiary is much more than the cost to the benefactor. This does happen in the case of vampire bats, since bats with large *margin-to-death* donate food to a starving bat who is close to death. *Reciprocal Altruism* has been modeled in Game Theory as a non-zero sum game, where it is possible to come up with a situation where all parties stand to gain.

3.1 Iterated Prisoner's Dilemma

The Iterated Prisoner's Dilemma (IPD) is a classic example of a non-zero sum game. The two-person Prisoner's Dilemma (PD) is characterized by two prisoners being interrogated in isolation about a joint crime. Each prisoner has the choice of either confessing (defection) or maintaining silence (cooperation). The payoff matrix is as shown in Figure 1. The payoff here denotes the years of imprisonment they will receive. As can be seen, if both cooperate, they can get away with a minimal sentence. However, since each has no means of knowing what the other will do, the most obvious choice would be to defect, and this is also the Nash

A \ B	Cooperate	Defect
Cooperate	1, 1	20, 0
Defect	0, 20	10, 10

Fig. 1. Payoff Matrix for Prisoner's Dilemma showing the possible jail terms (A's term, B's term) prisoners A and B may get (cooperate *implies* be silent, defect *implies* testify)

equilibrium for this game. The IPD extends the PD into a multi-move game. In this case, the prisoners can derive from previous history to decide their next move. The Alternating IPD is a variant in which the participants do not make their next move simultaneously but in turn.

3.2 Reciprocal Altruism as an Alternating IPD

Reciprocal Altruism has been modeled in game theory as an Alternating IPD. Various strategies for determining the next move in the Alternating IPD have been studied. Axelrod and Hamilton [12] ran a tournament whereby they attempted to determine the best strategy, which in their case was simple Tit-For-Tat (TFT) with an initial cooperating move. Another mechanism for determining an optimal strategy is an evolutionary game, in which individual strategies evolve till only the most evolutionarily stable one(s) remain. When the environment is noisy, a misunderstanding may arise between individuals asking for/giving help. In such a case, a strategy like TFT fails to achieve cooperation. Hence introduction of an element of clemency is required.

4 Reciprocal Altruism as a QoS Paradigm

The great intra-GOP dependence between frames implies that the post-reconstruction quality of received frames belonging to a GOP is inextricably interlinked with each other. The inter-GOP independence indicates that packet loss in one GOP has no effect on the end-quality of other GOPs. This suggests the treatment of a GOP as an atomic unit having either acceptable or unacceptable quality from the end-user view point. Borrowing terminology from the biological world, we therefore introduce a notion of *survival* wherein a flow either survives or dies for a particular GOP. *Survival* corresponds to having acceptable end-quality, whereas *death* corresponds to the contrary. In this section we formalize this notion and propose a paradigm for Reciprocal Altruism amongst flows that would lead to higher survival rates for the participant flows.

We define certain notions that shall be used throughout this paper.

Definition 1. (Renewal Time) *Renewal Time(T_r) is the time span over which the effect of loss of crucial packets lingers. After this time has elapsed, the flow can renew itself to its original quality.*

Definition 2. (Instantaneous PLR) *Instantaneous PLR (PLR_i) is defined as the ratio of the number of packets lost (lp_i) to the total number of packets received (tp_i) over a time span (t_i) of duration T_r.*

Definition 3. (Survival) *As long as the Instantaneous PLR remains below a certain threshold (PLR_S), the perceived end-quality is acceptable over that time span (t_i). This condition is termed as survival during the time span t_i.*

Definition 4. (Survival Rate) *Survival Rate (σ) for a flow (over the consecutive time intervals $t_1, \cdots, t_N$ of length T_r may be defined as*

$$\sigma = \sum_{i=1}^{N} S_i/N \ \textit{where}\ S_i = \begin{cases} 0, & \textit{if}\ PLR_i \geq PLR_s; \\ 1, & \textit{otherwise;} \end{cases}$$

The basic premise of our proposal is that *consistent end-quality may be quantified by the survival rate of a flow over its entire duration*, and each flow would seek to maximize this rate. However a sudden burst of traffic can lead to high losses (PLR$_i$) and resultant failure to survive (flow death) in t$_i$. The core concept behind this notion of survival is that the perceptual quality of the reconstructed video at the receiver's end should conform to some minimum level. Given a particular GOP, its reconstruction becomes extremely difficult either if an I Frame packet is lost or if the PLR becomes rather high. In order to avoid this, a flow may seek help from other flows that have low PLR. These flows may decide to temporarily lend their resources at some (but not critical) expense to themselves, in expectation of future reciprocation. At the same time, we recognize a concept of *Posthumous Donation (PD)* whereby a flow that has already incurred heavy packet losses may decide against sending further packets for the current GOP (i.e. the current T_r) and instead grant assistance to another flow in need. Such action would be motivated by the fact that the current GOP is already beyond reconstruction. Hence it is advisable to garner good-will for the future. Notions similar to PD have been proposed earlier to avoid sending useless packets [13].

5 Mechanisms for Cooperation

It is possible to incorporate *Reciprocal Altruism* into QoS scheduling policies via two broad categories of approaches.

A possible mechanism for cooperation is to deploy intelligent agents that act on behalf of each flow. These agents would monitor flow and network state. Based on decision parameters specified by the flow and the current state, these would make decisions regarding obtaining/providing help. There are numerous issues

related to inter-flow communication etc. that need to be studied to implement such a mechanism. This has not been looked at in this paper.
A more simple mechanism comprises the assumption of the task of enforcing cooperation by a centralized authority (say the ISP). Flows may specify their survival thresholds and other requirements. The centralized authority (which would have to be trusted) thereafter decides on behalf of each flow about the need to seek help as well as the feasibility of admitting a request for help. Such a mechanism would have the advantage of having lower overheads as all the functionality would be built into the service discipline deployed by the service-provider. This paper presents results for a preliminary implementation of such a centralized scheme, that assumes complete trust and willingness-to-help.

6 Validation by Simulation

We present a preliminary validation of our proposal by simulating simplistic schemes for centralized imposition of reciprocal altruism. We analyze the effect of the same on flow performance.

6.1 Simulation Model

We consider a scenario wherein flows contend for bandwidth on a single link. We first consider a situation where there are only two flows. The topology corresponding to this scenario is depicted in Fig. 2. Links l_1 and l_2 are of 3.5 Mbps capacity each. The bottleneck link is l_3. Flows f_1 and f_2 each have a certain reserved bandwidth allocation a_1 and a_2 respectively over l_3. It is assumed in all simulations that $a_1 + a_2 = \mathit{capacity}(l_3)$. We assume lossless links. The traffic corresponds to MPEG-4 traces of various movies [14] as listed in Table 1, sent over UDP/IP. A constant packet size of 200 bytes (data+headers) is considered. We assume the use of the TOS field of the IP header to mark the type of frame (I, P or B) to which each packet belongs.

The simulator used is ns-2 [15] into which we have built our own scheduling module. We assume a simple scheduling policy loosely modeled on Round Robin scheduling, wherein each flow gets assigned a quantum equal to its legitimate bandwidth share over a certain time interval. The quantum decreases with each packet sent. At the end of the interval, the remaining quantum is flushed, and quantum is re-assigned for the next interval. Bandwidth unused by a flow may be used by the other if it has a packet to send. The time interval for quantum assignment coincides with the *renewal time* T_r defined in Section 4. The value of T_r is set to 0.45s which is the approximate display time of one GOP (IPBBPBBPBB) in the MPEG-4 traces used. However, this is only an approximate measure employed for preliminary validation and needs to be replaced by a mechanism that conforms exactly to GOP boundaries.

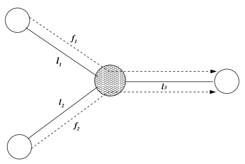


Fig. 2. Simulation Topology 1

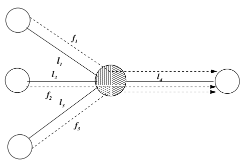


Fig. 3. Simulation Topology 2

Table 1. Traffic Characteristics of MPEG-4 Traces Used

Flow	Content	Encoding	Mean BR	Peak/Mean
f_1	*Star Wars IV* Video	MPEG-4	1.9e+05	6.81
f_2	*Jurassic Park* Video	MPEG-4	7.7e+05	4.37
f_2	*The Firm* Video	MPEG-4	2.9e+05	6.96

We have built in *Reciprocal Altruism* atop this strategy by tracking intra-T_r PLRs and estimating the need for seeking help, as well as the feasibility of granting it. The architecture of the scheme is depicted in Fig. 4. The scheduler maintains per-flow queues with two occupancy limits defined, viz. LB and UB. LB is the lower bound and as long as the queue length remains below it, all incoming packets are enqueued. As soon as the length exceeds LB, an arrival burst is assumed. If the length is equal to UB, the packet is immediately dropped. However, when the queue length lies between LB and UB, if PLR is within acceptable limits, the packet may be dropped, unless it is an I frame packet. In that case, the flow attempts to accommodate it by dropping a lower priority packet (P/B Frame) from the queue. If the PLR is already hovering at dangerous levels, the flow may seek help from other flows in the form of a promise for an extra transmission turn(s). If such a promise is forthcoming from any other flow, the packet is retained, in the belief that the backlog shall soon return to normal levels (i.e. $< LB$). It is to be noted that a promise is not binding, in that the current PLR levels of the benefactor flow are checked prior to actually passing on its turn(s). The turn is passed on only if PLR is still within acceptable limits, thereby avoiding situations in which an earlier mis-estimation might lead to critical losses for the benefactor flow. The value of PLR_S is set to 0.2. This corresponds to a figure often used as a loose upper bound on acceptable video losses. At PLR > 0.2, the end-quality generally becomes unacceptable even with error-correction etc. However, it is to be noted that the value of PLR_S may be different for different flows depending on how stringent their quality requirements are, and may vary significantly from the value used in this paper. The values of UB and LB used for the simulations presented here are 40 and 100 respectively. The pseudo-code for the entry terminal and service terminal procedures are shown in the next page.

We have also looked at a buffer-based scheme for implementing *Reciprocal Altruism*, as a reference for comparison with the estimation-based strategy. This scheme utilizes a buffer to obtain a lookahead of one T_r and makes advance decisions on exchange of help for this upcoming interval.

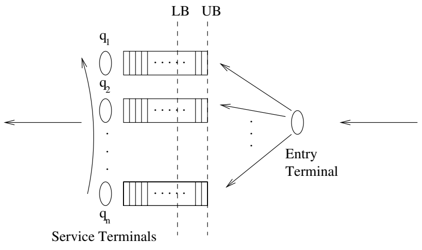


Fig. 4. Estimation Based Scheme

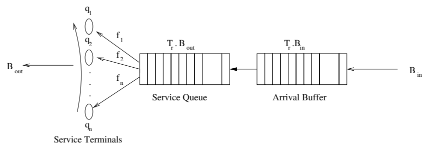


Fig. 5. Buffer Based Scheme

We have also performed basic simulations for three-flow interactions by simulating the topology depicted in Fig. 3, wherein flows f_1, f_2 and f_3 contend for bandwidth on bottleneck link l_4. Links l_1, l_2 and l_3 are of 3.5 Mbps capacity each. Allocations a_1, a_2 and a_3 are all equal and $a_1 + a_2 + a_3 = capacity(l_4)$.

6.2 Results and Analysis

We present here the results obtained from simulation. Figs. 6 and 7 depict the survival rates for all participant flows (each having equal allocation) with variation in bottleneck link bandwidth for the topologies in Figs. 2 and 3 respectively. All flows correspond to runs of the MPEG-4 trace of *Star Wars IV* with varyingly staggered start times. All three schemes, viz. non-altruistic, altruistic-estimation-based and altruistic-buffer-based are simulated, and all flows are allocated equal shares of the link bandwidth. It may be seen that the estimation-based scheme consistently tends to perform well in terms of having higher survival rates. It significantly out-performs the other two schemes at low link bandwidths. However at higher link bandwidths, the buffer-based scheme performs rather well, especially in the three-flow case.

Since the above results point to similar trends for 2-flow as well as 3-flow interactions, it seems that the results for 2-flow interaction are indicative of general trends in n-flow interaction. Hence further simulations have only been carried out for the two-flow case. Besides, only the non-altruistic scheme and the estimation-based scheme have been considered.

Figs. 8 and 9 show the distribution of bytes received and bytes sent per interval of duration T_r for Flow 0 over a total duration of approximately 3500s for MPEG-4 traces of Star Wars IV staggered by 90.0s for the non-altruistic and estimation-based schemes respectively. The bottleneck link bandwidth is 0.6 Mbps and both flows have allocations of 0.3 Mbps each (that also corresponds closely to their mean rates). As may be clearly seen, in the former case, the

```
handle_arrived_packet(packet p){
    f= flow_id(p);
    plr=pkts_drpd_sofar_thisTr/(pkts_drpd_
    sofar_thisTr+pkts_sent_sofar_thisTr);
    if((bytes_drpd_sofar_thisTr>0.1*
    quantum_allocated) and
    (plr>death_thresh(f))
&&(frametype(p)!=I))
        posthumous_donation(f)=yes;
    if(length(flowqueue(f))>UB){
        drop_packet(p);
        return;
    }
    if(length(flowqueue(f))>LB){
        if(plr>=0.7*death_thresh(f))
            alert=1;
         if(frametype(p)==I){
          \\Giving priority to I
          sacrificial_lamb=look_for_some_
          PorB_packet(flowqueue(f));
          if(sacrificial_lamb_not_found)
              alert=1;
            else if(alert==0)
              drop_packet(sacrificial_lamb);
        }
        else if(alert==0){
            \\If no alert then drop
            drop_packet(p);
            return;
        }
if(posthumous_donation(f)==yes)
            \\if dead no alert
            alert=0;
        if(alert){
            seek_help(f);
            if(!help_obtained){
                drop_packet(p);
                return;
            }
        }
enque_packet(p,flowqueue(f));
}

seek_help(flow f){
    for each flow g!=f{
        if(posthumous_donation(g)==yes){
            help_obtained=1;
            return;
        }
        else{
            plr=pkts_drpd_sofar_thisTr/
(pkts_drpd_sofar_thisTr
+pkts_sent_sofar_thisTr);
            if(((plr/death_thresh(g))*
            (length(flowqueue(g)/LB)<0.3)and
            (length(flowqueue(g))<0.8*LB)and
            (plr<0.8*death_thresh)and
            (not_already_giving_help)){
                help_obtained=1;
                return;
        }
}

service_flowqueue(flow f){
    if(giving_help(f))
        service(beneficiary(f));
    else
        if(!empty(flowqueue(f))
            service(f);
    turn=next_flow;
}

revolution(){
    service_flowqueue(turn);
}
```

number of bytes sent per interval of duration T_r remains close to the allocation. In the latter case, it may be seen that the flows are able to send at rates higher than their allocation a greater number of times, consequent to their cooperation. For non-altruistic scheduling, the number of time-intervals in which bytes sent were greater than allocation was 2800, whereas for altruistic scheduling it was 3097, i.e. an increase of around 10%.

Fig. 10 depicts flow behavior trends for a wide range of bottleneck link bandwidths, as well as varying proportion of allocation of the same to the flows. The figure essentially illustrates the survival rate against the individual allocations of both flows. Though in a real world situation, flows may not actually go in for such a widely varying allocation spectrum, it is useful to study the same.

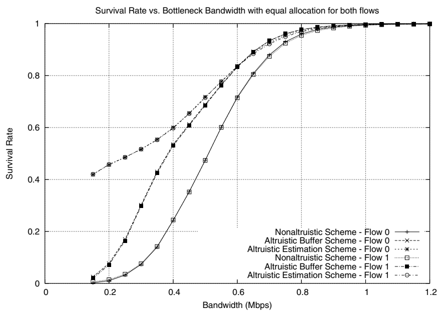


Fig. 6. Survival Rates for 2 Flow Scenario

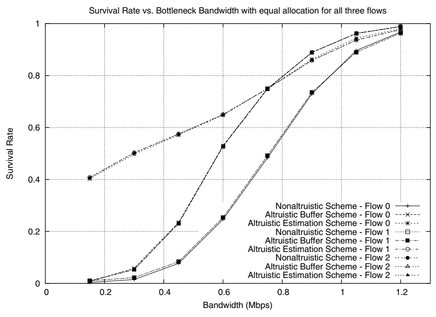


Fig. 7. Survival Rates for 3 Flow Scenario

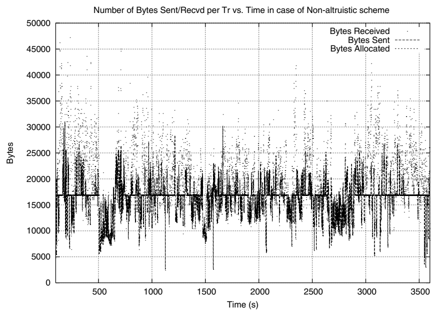


Fig. 8. Distribution for Non-Altruistic Scheme

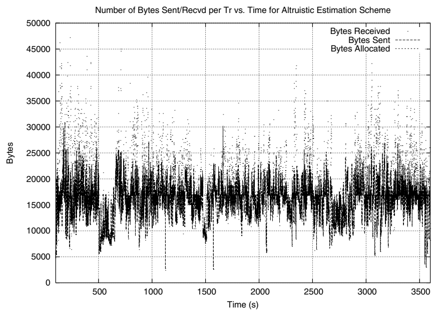


Fig. 9. Distribution for Altruistic Estimation Scheme

Fig. 11 depicts the net number of deaths saved for Flow 0 over a wide allocation spectrum for Flows 0 and 1 (varying bottleneck link bandwidth divided between the two flows in varying proportions). Once again it is indicated that the net improvement is greater when allocations are lower. Fig. 12 depicts the survival rates of both flows versus the ratio of allocation to mean rate of one flow, for bottleneck link bandwidth of 0.6 Mbps respectively. This bandwidth corresponds to the situation in which the total available capacity is just enough to allow traffic at the mean rates of both flows to pass through. The intent is to determine the degree of incentive a flow might have to go in for a much lower or higher allocation than its mean rate. At a bottleneck capacity of 0.6 Mbps it is seen that, at an allocation lower than the mean rate, there is significant increase in survival rate due to altruism. At allocations much higher than the mean rate, the performance gain gradually diminishes. At the mean rate, we find that there is significant improvement in performance and the new survival rate lies in a very desirable range of above 80%. So it seems that there is sufficient motivation for everyone to seek allocations close to the mean rate, as then the post-altruism survival rates are fairly reasonable. Fig. 13 depicts the percentage of flow deaths saved on using the estimation-based altruistic scheme

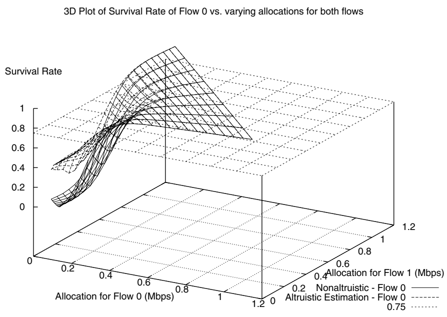


Fig. 10. Variation in Survival Rate with Allocations

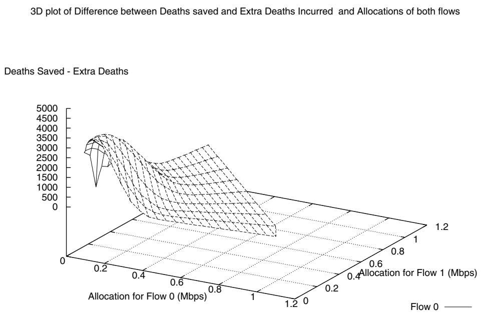


Fig. 11. Variation in Net Deaths Saved with Allocations

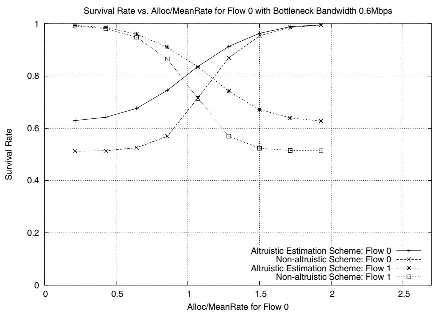


Fig. 12. Survival Rate vs. Allocation/Mean for Flow 0(0.6 Mbps Link)

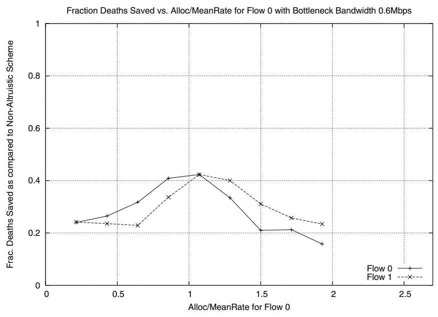


Fig. 13. Fraction of Deaths Saved vs. Allocation/Mean for Flow 0(0.6 Mbps Link)

for bottleneck link bandwidth of 0.6 Mbps. It may be seen that the maximum percentage is obtained close to the mean rate allocation. Fig. 14 shows survival rates for a situation in which one flow cheats i.e. though it has a true PLR_S of 0.2 (on which actual survival depends), it advertises a different PLR_S (on which help decisions are made). The figure clearly depicts that the cheating flow gains no advantage in terms of survival rate by advertising a higher or lower PLR_S. However, if one looks at the average PLR obtained by this flow in time intervals where it survived (Fig. 15), one finds that its PLR significantly reduces at lower advertised death thresholds. Thus the flow can gain in terms of better end-quality during periods of survival. It is therefore advisable to employ game strategies in real-world situations where complete trust may not be assumed.

Fig. 16 depicts survival rates versus allocation for one flow, in the case when the two flows correspond to two different MPEG-4 traces viz., *Jurassic Park* and *The Firm*. We obtain trends similar to Fig. 12 which show that the results obtained hold even for interacting flows with different traffic characteristics. Fig. 17 depicts survival rates versus allocation for one flow, keeping the allocation

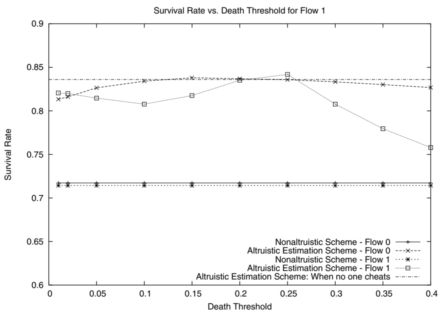


Fig. 14. Survival Rates with one cheating flow

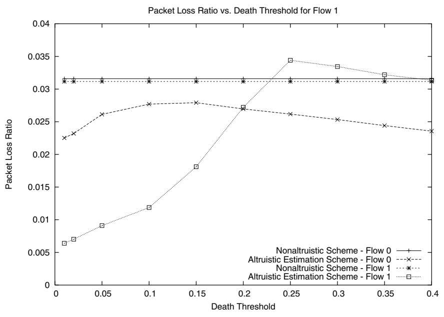


Fig. 15. Avg. PLR over all Periods of Survival with one cheating flow

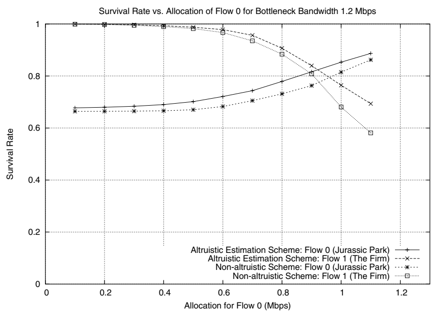


Fig. 16. Survival Rate vs. Allocation/Mean for Flow 0 (1.2 Mbps Link)

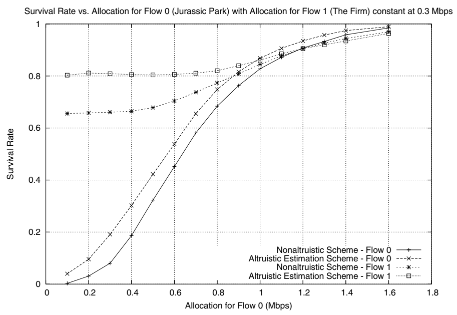


Fig. 17. Survival Rate vs. Allocation for Flow 0 (0.3 Mbps to Flow 1)

for the other flow constant. The flow with a constant allocation of 0.3 Mbps corresponds to the MPEG-4 trace for *The Firm* while the other corresponds to the MPEG-4 trace for *Jurassic Park*. We once again find that though the flow for *Jurassic Park* does obtain some performance improvement due to altruism at allocations lower than its mean rate, yet they are not significant enough to be an incentive for deliberate under-allocation. Besides, we find that the survival statistics of the other flow (which has gone in for a fair allocation) do not deteriorate due to altruism. Rather, it also sees a performance improvement.

7 Conclusions

This paper presents *Reciprocal Altruism* as a paradigm for providing QoS to multimedia flows. It also introduces a notion of per-GOP survival. A preliminary investigation into the feasibility of the same has been undertaken and a simplistic centralized mechanism has been simulated. The results obtained so far are indication of the potential of this paradigm. They serve to validate the concept of *Reciprocal Altruism* as a QoS Paradigm. The simplistic estimation-based scheme described here vindicates the possibility of coming up with lightweight on-the-fly estimation strategies that have no foreknowledge of traffic characteristics. However, there is need to come up with a comprehensive architecture of a service discipline for centralized imposition of reciprocal altruism. Such a discipline would need to handle multi-flow interactions and allow for various game strategies (e.g. TFT, Pavlov etc.) instead of assuming complete trust. One would also need to look at efficiency issues in the implementation, as well as the interplay of multimedia flows with best-effort traffic. The possibility of an agent based approach also merits investigation. Such an approach would be particularly useful to large organizations which could then define their own custom policy, based on economic concerns. Another major issue is that of looking at multiple points of contention along flow-paths in an integrated manner, and basing decisions thereof. A comprehensive investigation into these issues needs to be undertaken.

References

1. Braden, R., Clark, D., Shenker, S.: Integrated services in the internet architecture: an overview. Internet RFC 1633 (1994)
2. Blake, S., Black, D., Carlson, M., Davies, E., Wang, Z., Weiss, W.: An architecture for differentiated services. Internet RFC 2475 (1998)
3. Liao, W., Li, V.O.K.: The split and merge protocol for interactive video-on-demand. IEEE Multimedia Magazine **Oct.-Dec.** (1997) 51–62
4. Verscheure, O., Frossard, P., Hamdi, M.: User-oriented qos analysis in mpeg-2 video delivery. Journal of Real-Time Imaging (special issue on Real-Time Digital Video over Multimedia Networks) **5** (1999) 305–314
5. Feamster, N., Balakrishnan, H.: Packet loss recovery for streaming video. In: Proceedings of 12th International Packet Video Workshop. (2002)
6. Verscheure, O., Frossard, P., Boudec, J.Y.L.: Joint smoothing and source rate selection for guaranteed service networks. In: Proceedings of IEEE Infocom (2). (2001) 613–620
7. Shenker, S.: Making greed work in networks: A game-theoretic analysis of switch service disciplines. IEEE/ACM Transactions on Networking **3** (1995) 819–831
8. Park, K., Sitharam, M., Chen, S.: Quality of service provision in noncooperative networks with diverse user requirements. Decision Support Systems **28** (2000) 101–122
9. Garg, R., Kamra, A., Khurana, V.: A game-theoretic approach towards congestion control in communication networks. ACM SIGCOMM Computer Communications Review **32** (2002)
10. Trivers, R.: The evolution of reciprocal altruism. Quarterly Review of Biology **46** (1972) 35–57
11. Wilkinson, G.S.: Food sharing in vampire bats. Scientific American **262** (1990) 76–82
12. Axelrod, R., Hamilton, W.D.: The evolution of cooperation. Science **211** (1981) 1390–1396
13. Wu, J., Hassan, M.: The issue of useless packet transmission for multimedia over the internet. Computer Communications (2003)
14. Fitzek, F.H.P., Reisslein, M.: Mpeg-4 and h.263 video traces for network performance evaluation. IEEE Network **15** (2001) 40–54
15. NS: ns notes and documentation. (http://www.isi.edu/nsnam)

A Novel DCA Scheme for Resource Sharing in Cellular Environments with Heterogeneous Traffic

Vahid Tabataba Vakili and Arash Aziminejad

Department of Electrical Engineering,
Iran University of Science & Technology (IUST),
Narmak, Tehran, 16844, IRAN
arash1971@yahoo.com

Abstract. In cellular networks it is crucial to be able to use the available radio spectrum as efficiently as possible while providing a certain level of Quality of Service (QoS) for users. Emergence of miscellaneous services has dramatically increased the complexity of this problem by creating a heterogeneous traffic environment. In this paper an efficient resource allocation scheme has been proposed for cellular networks with multimedia traffic, which combines classical resource borrowing concept with a novel inter-cell resource sharing scheme between different classes of traffic. By assuming the heterogeneous offered traffic to be a combination of audio and video traffic types, it will be shown that the proposed resource allocation scheme is capable of significantly improving audio teletraffic performance of the system without imposing additional expense upon video QoS performance.

Keywords: Resource allocation, Heterogeneous traffic, Channel borrowing assignment, Teletraffic performance

1 Introduction

The design of cellular networks is impressively being influenced by the continuous growth in traffic volume and the emergence of new diverse services in mobile communications. The introduction of novel applications such as data delivery and real-time multimedia in 3G and 4G systems will not only significantly increase the traffic on wireless networks, but also will create a heterogeneous traffic environments [1, 2]. To effectively utilize the bandwidth resource, a number of bandwidth assignment schemes have been proposed in order to transmit multimedia traffic in wireless networks [3, 4].

Generally speaking, a real-time video call requires considerably larger amount of bandwidth than an audio call and in contrast to both audio and video traffic, data traffic does not have real-time requirements. Furthermore, traffic characteristics of audio and video calls (e.g. call arrival rate and call holding time distribution) are different. In practice, bandwidth resources are pre-assigned to each type of traffic and the idle capacity allocated to video or audio traffic will be temporarily used for transmitting packet switched data according to a procedure which leaves QoS performance of video and audio calls intact [5, 6]. Keeping this preemptive priority of audio and video calls over data traffic in mind, in this research an efficient resource-sharing scheme between audio and video calls has been proposed, which is based upon intra-cell (for audio traffic) and inter-cell (between audio traffic and video resources) resource sharing. Since video call requests arrival rate is considerably lower comparing to audio call arrival rate and also as a video call requires higher bandwidth for transmission, the resource borrowing between audio and video calls (cross-borrowing) will be performed unilaterally, i.e. only overflowed audio calls will borrow idle video channels once by using a threshold type decision policy it has been ascertained that QoS performance of the system against video traffic will not be degraded by this resource borrowing. A criterion has been analytically derived for this purpose and numerical results obtained from extensive simulations conducted on a as real as possible cellular environment indicate that Heterogeneous Channel Borrowing Assignment with Unilateral Cross-Borrowing (HCBA-UCB) is able to reduce the audio call blocking probability considerably at the expense of only negligible increase in video call blocking. Also it will be shown that by allowing a video resource to be simultaneously borrowed by multiple audio calls on the condition that video call blocking performance requirement is not violated, teletraffic performance of HCBA-UCB can be further enhanced.

2 General Assignment Strategy in HCBA-UCB

Initially all of the audio and video channels will be divided between the cells according to a pre-defined uniform Fixed Channel Assignment (FCA) reuse pattern and it is assumed that after this pre-assignment of resources each cell will possess N_A audio and N_V video nominal channels. Resource assignment process in HCBA-UCB follows the following algorithm:

```
if (T-State = 0)  /*Voice-type traffic*/
{
          Min = No. of the first audio channel
          Max = No. of the last audio channel
}
else if (T-State = 1)  /*Video-type traffic*/
{
          Min = No. of the first video channel
          Max = No. of the last video channel
}
/*Local FCA assignment of A or V channels to
corresponding call demands*/
for (m = Min to Max)
```

```
{
          if (S[m][i][j] = -1 & C[D[m]][i][j] = 1 & Ch-
              Lock[m][i][j] = 0)

          /*Assign channel m to the call request*/

             Ch-No = m

             S[m][i][j] = User No.

             Ch-Lock[m][i][j] = 1

}

/*Borrowing between A-channels in adjacent cells*/

if (T-State = 0)

          Ch-No = Borrow (i, j)

          if (Min <= Ch-No <= Max)

             S[Ch-No][i][j] = User No.

             /*Borrowed channel will be locked in donor
             cell and near co-channels*/

          else

          /*Unilateral cross borrowing between A-traffic
          and V-channels in  the home-cell*/

             Ch-No = CrossBorrowing (i, j)

             if (Min V-channel No. <= Ch-No <= Max V-
                 channel No.)

                S[Ch-No][i][j] = User No.

                /*Borrowed V-channel will be locked in
                the home-cell*/

             else

                The Audio call demand is blocked
```

According to above-mentioned resource assignment policy, if a new video call request arrives in a cell and finds all its N_V nominal video channels busy, it will be blocked. In case of a new voice call arrival, if all the N_A nominal audio channels are busy, HCBA-UCB will try to borrow a free and unlocked audio channel from one of the six neighboring cells by activating *Borrow* function. The borrowing process will be accomplished based on Borrow First Available (BFA) scheme [7]. If no appropriate audio channel is found in the neighboring cells to be borrowed, HCBA-UCB tries to cross-borrow one of the nominal video channels by activating *CrossBorrowing* function. The crucial question in permitting cross borrowing to take place is whether or not the pre-defined video call blocking probability requirement (P_B) will be violated by the cross-borrowing operation. If no video channel is found to satisfy the cross-borrowing criterion in home cell, the audio call request will be blocked. To further illustrate this algorithm, A few variable names should be defined. Assume the location of cells be represented by their integer coordinates (i, j) in a two-dimensional array of hexagonal cells, the m-th Compact Pattern (CP) for uniform FCA initial assignment strategy is represented by the 3D array *C [m][i][j]*: [8]

$$C_{ij}(m)=\begin{cases}1 & \text{if cell } (i,j) \text{ belongs to the } m-th\ CP\\ 0 & \text{otherwise}\end{cases}$$

Array *S [k][i][j]* denotes the occupancy state of channel k in cell (i, j) as:

$$S_{ij}(k)=\begin{cases}\text{User's No.} & \text{if channel } k \text{ is being used in cell } (i,j)\\ 0 & \text{otherwise}\end{cases}$$

One-dimensional array *D [k]* indicates the CP that has been allocated to channel k and it will be filled by the initial FCA assignment according to its reuse pattern [8]. Finally, *Ch-Lock [k][i][j]* expresses locked or unlocked state [7] of channel k in cell (i, j).

$$Ch-Lock_{ij}(k)=\begin{cases}1 & \text{if channel } k \text{ is being locked in cell } (i,j)\\ 0 & \text{otherwise}\end{cases}$$

3 Unilateral Cross-Borrowing

The unilateral cross-borrowing of video traffic resources by overflowed audio call requests in a cell can be stated in a simple way that a video channel can be cross-borrowed to attend an audio call request, if sufficient number of free video channels remain after cross-borrowing so that the video call blocking in that cell does not exceeds its QoS requirement, say P_B. One way of approaching this problem is to use a multi-server queuing system model with queue size of zero (loss model), i.e. if no server is available at the moment that a new call originates, that call is blocked immediately. Each type of calls has different traffic parameters but all traffic sources are modeled as Poisson processes. The goal here is to find a control policy

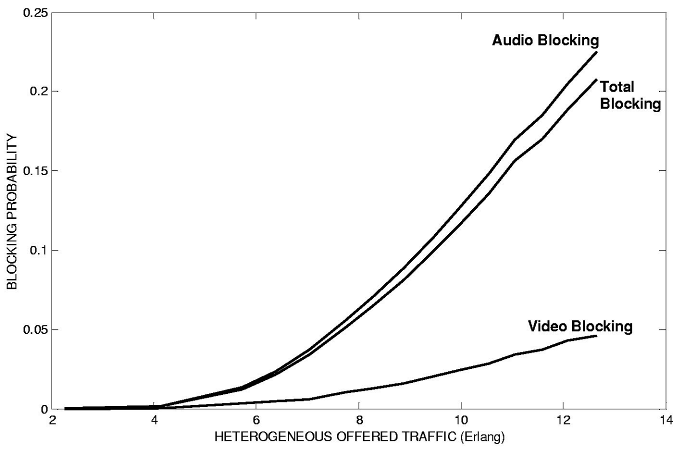


Fig. 1. Average audio, video and overall blocking probabilities for uniform FCA strategy versus heterogeneous offered traffic (Erlangs)

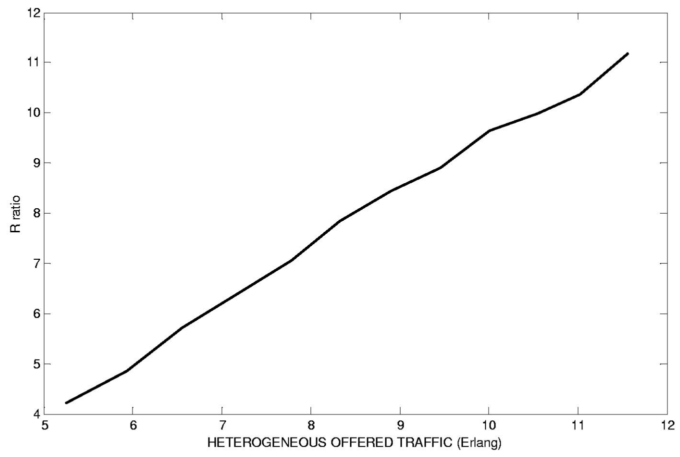


Fig. 2. Ratio of video call generation probability to audio call generation probability (R) for different values of average heterogeneous offered traffic (Erlangs) in the case of applying uniform FCA strategy

maximizing audio channel utilization while guaranteeing QoS requirement for video traffic and in general case this is accomplished using stochastic optimization techniques [9]. The main problem with this procedure is its mathematical complexity, which in some cases turns out to be completely untractable. In this research by adopting an approach similar to one outlined in [10], the analytical complexity of solution procedure has been considerably reduced.

Suppose new call arrivals for both audio and video traffic follow Poisson processes with mean arrival rates λ_A and λ_V and call servicing processes are also Poisson with mean servicing rates μ_A and μ_V, respectively.

If in the time origin (t = 0) n_0 video channels out of total N_V nominal video channels assigned to an arbitrary call are busy carrying video calls, then the probability that there will be n active video channels due to video traffic after time t in this cell is a transient state probability of a $M/M/N_V/N_V$ queuing system [11]. If the offered traffic load to the cell is not too heavy, this probability can be approximated by the transient state probability of a $M/M/\infty/\infty$ queuing system, $Q_n(t, n_0)$, according to the following equation: [10]

$$P_n(t,n_0) = \frac{Q_n(t,n_0)}{\sum_{i=0}^{N_V} Q_i(t,n_0)} \tag{1}$$

It can be shown that: [10, 11]

$$Q_n(t,n_0) = \begin{cases} \frac{1}{n!} e^{(\beta-\mu_V nt)} \gamma^{n_0} \sum_{i=0}^{\infty} \frac{(n_0+i)!}{i!(n_0-n+i)!}(-\beta)^i \gamma^{i-n}, n \le n_0 \\ \frac{1}{n!} e^{(\beta-\mu_V nt)} \sum_{i=0}^{\infty} \frac{(n+i)!}{i!(n-n_0+i)!}(-\beta)^{n-n_0+i} \gamma^i, n > n_0 \end{cases} \tag{2}$$

Where

$$\begin{aligned} \beta &= [(\lambda_V / \mu_V)(1-e^{\mu_V t})] \\ \gamma &= (1-e^{-\mu_V t}) \end{aligned} \tag{3}$$

If the goal is to cross-borrow one video channel without violating QoS requirement of video traffic in the original cell, then the maximum time that one video channel can be cross-borrowed, T_0, will be given by solving:

$$P_{N_V-1}(T_0,n_0) = P_B \tag{4}$$

Since call holding time for audio calls follows an exponential distribution, the probability that none of the active audio calls in the original cells will be completed within T_0 is:

$$P_A = Exp[-(N_A+1)\mu_A T_0] \tag{5}$$

By defining a reasonable threshold for P_A as P_T, the request for cross-borrowing will be granted only if:

$$P_A \le P_T \tag{6}$$

As the bandwidth required for carrying a video call is multiple times larger than bandwidth necessary for an audio call, we can further extend the cross-borrowing concept by permitting one video channel to be simultaneously cross-borrowed by multiple audio calls, if QoS criterion for video traffic servicing is satisfied. Suppose a video channel is cross-borrowed by an audio call request arrived at time origin t_1 for duration of T_0 (derived from Eq. 4) and second audio call requests arrives at time t_2 before completion of the first audio call carried by the once cross-borrowed video channel. Keeping in mind that the call holding duration is exponentially distributed and therefore is without memory, the probability that time duration $T_0 + t_1 - t_2$ is sufficient for both audio calls to be completed can be considered as a criterion for granting second cross-borrowing request. In other words, to maintain QoS requirement of video traffic, the second cross-borrowing will be allowed, if the probability of at least one audio call is not finished during $T_0 + t_1 - t_2$ is equal to or less than P_T. This statement can be expressed as:

$$e^{-\mu_A(T_0+t_1-t_2)(N_A+2)} + e^{-\mu_A(T_0+t_1-t_2)(N_A+1)} \\ \cdot(1-e^{-\mu_A(T_0+t_1-t_2)(N_A+2)}) \le P_T \tag{7}$$

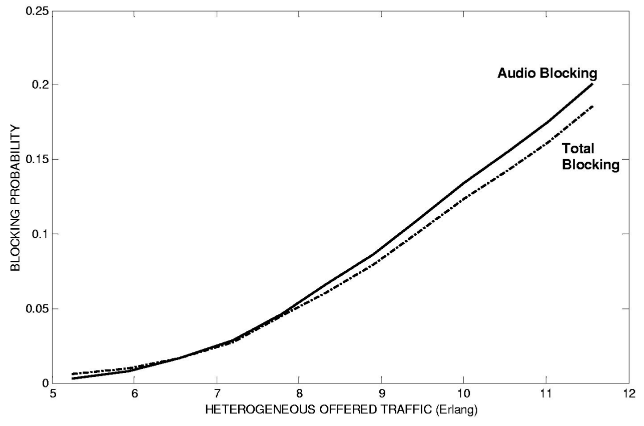
The afore-said criterion can be generalized for simultaneous cross-borrowing of one video channel by n audio calls in a straightforward manner.

4 Teletraffic Performance Evaluation of HCBA-UCB

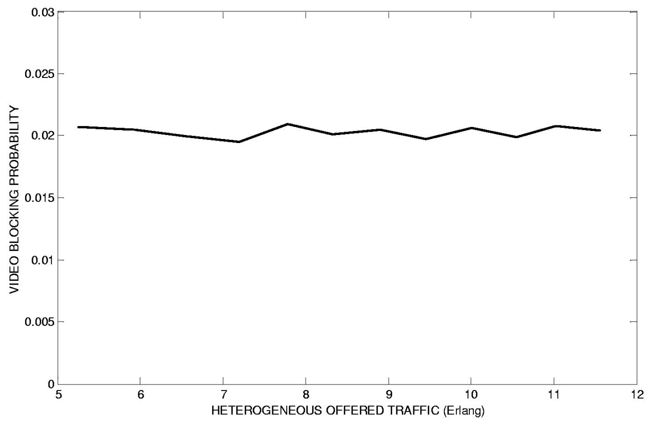
To simulate the performance of HCBA-UCB as realistically as possible, a novel discrete-event cellular network simulator was developed, which uses a wraparound topology. Simulation environment consists of 49 hexagonal cells arranged in a 7*7 array. In initial assignment phase 10 audio channels and 5 video channels were pre-assigned to each cell according to a 7-cell reuse pattern uniform FCA strategy. In other words, total allocated bandwidth to the cellular system consists of 70 audio and 35 video channels. In the presented research it has been assumed that bandwidth of a video channel is 5 times larger than bandwidth of an audio channel. Servicing processes of audio and video calls are Poisson with mean service rates of 1/98 and 1/189 calls per second, respectively.

The mechanism for advancing simulation time and guaranteeing that all events occur in correct chronological order is based on the Future Event List (FEL). FEL contains all event notices for events that have been scheduled to occur at a future time. Scheduling a future event means that at the instant an activity begins, its duration is somehow derived (usually according to a given pdf) and the end-activity event, together with its event time, is placed on the FEL. After the system snapshot at simulation time t_1 has been updated, the simulation clock is advanced to simulation time t_2 and the event associated with this moment in the FEL will be executed. At the time t_2, new future events may or may not be generated (randomly, with probabilities p_i and $1 - p_i$), but if any are, they are scheduled by creating event notices and putting them in their proper position on the FEL. This process repeats until the simulation is over.

Fig. 1 depicts overall, audio and video call blocking probabilities versus average heterogeneous offered traffic (Erlangs). In extracting these results it has been assumed that the probability of arrival of an audio call is 9 times greater than the probability of arrival of a video call. Also channel assignment strategy is solely based on uniform FCA. As our final goal from using HCBA-UCB is to improve audio teletraffic performance of the system without imposing additional costs upon video QoS performance, by changing the ratio of video call generation probability to audio call generation probability (R) for different amounts of average heterogeneous offered



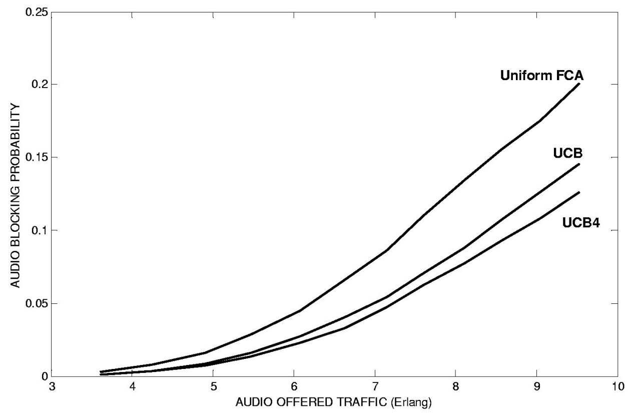
(a)



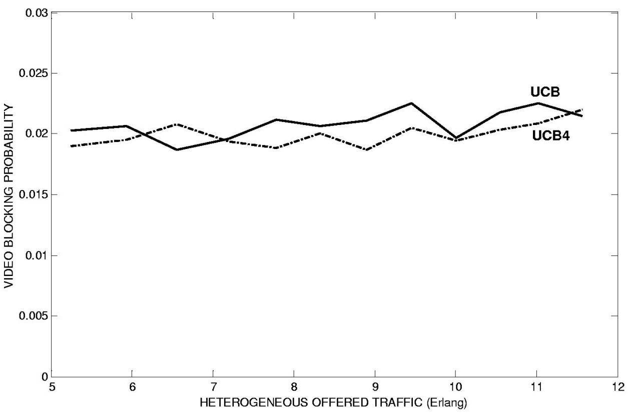
(b)

Fig. 3. (a) Audio, overall and (b) video average blocking probabilities for uniform FCA strategy corresponding to different values of R extracted from Fig. 2, versus heterogeneous offered traffic (Erlangs)

traffic in the case of uniform FCA strategy, the video call blocking probability has been kept fixed around 2% (with a tolerance of 0.002) while average heterogeneous offered traffic is increased. Fig. 2 shows different values of R obtained from this set of simulations and the corresponding overall, audio and video call blocking probabilities can be seen in Fig. 3.



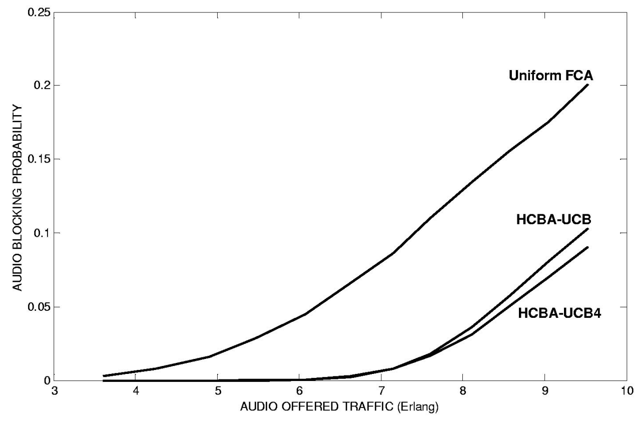
(a)



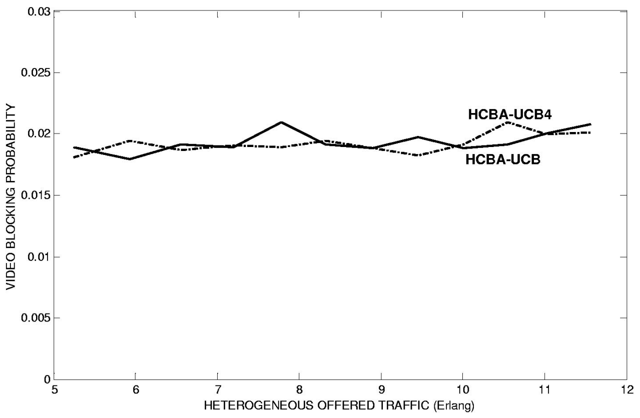
(b)

Fig. 4. (a) Audio average blocking probability versus audio offered traffic (Erlang) and (b) video average blocking probability versus heterogeneous offered traffic (Erlangs) for uniform FCA, UCB and UCB4 schemes, corresponding to different values of R extracted from Fig. 2

For assessing the mere effect of cross-borrowing on teletraffic performance of the cellular system, we applied this procedure to the uniform FCA strategy and the simulation results have been gathered in Fig. 4. By perceiving these results one can easily deduce that cross-borrowing concept has been successful in considerably improving audio teletraffic performance of the system without degrading video QoS performance. This deduction is even more perceptible when one cross-borrowed



(a)



(b)

Fig. 5. (a) Audio average blocking probability versus audio offered traffic (Erlang) and (b) video average blocking probability versus heterogeneous offered traffic (Erlangs) for uniform FCA, HCBA-UCB and HCBA-UCB4 schemes,corresponding to different values of R from Fig. 2

video resource is allowed to simultaneously carry up to 4 audio channels (UCB4). In blocking index of 2%, uniform FCA, UCB and UCB4 schemes can carry average offered audio traffics of 5.067, 5.65 and 5.95 Erlangs respectively, which can be interpreted as 12% and 17% increase in audio traffic carrying capacity for UCB and UCB4 over uniform FCA.

Finally, HCBA-UCB and HCBA-UCB4 (simultaneous cross-borrowing of a video resource by up to 4 audio call requests) schemes were subjected to a quantitative teletraffic performance evaluation. Fig. 5 depicts the obtained results in this regard.

As it was expected, combination of traditional resource borrowing and cross-borrowing concepts had an outstanding impact on audio teletraffic performance of the cellular system under simulation while negligibly affected its video teletraffic performance. In blocking index of 2%, HCBA-UCB and HCBA-UCB4 can carry 7.57 and 7.8 Erlangs of average offered audio traffic, which respectively represent 49% and 54% increase in audio traffic handling capacity comparing to uniform FCA.

5 Conclusion

In this paper we proposed HCBA-UCB as a resource allocation scheme specifically developed for cellular environments with heterogeneous offered traffic. In a cellular network with heterogeneous offered traffic consisting of audio and video calls, HCBA-UCB uses intra-cell borrowing and inter-cell unilateral cross-borrowing to enhance teletraffic performance of the system. Cross-borrowing of a video resource by one or multiple audio calls will be allowed only if QoS requirement for video calls is not violated. Performance evaluation of HCBA-UCB indicates that this allocation scheme is capable of improving audio teletraffic performance of the cellular network while insignificantly affecting video QoS performance.

References

1. J.Korhoner, Introduction to 3G Mobile Communications, MA, Artech House, 2001
2. B.Li et.al., "On the performance of channel assignment strategies in multi-service wireless cellular networks", 5th Int. Symp. on Wireless Personal Multimedia Comm., 2002, pp. 1379–1384
3. L.Jorguseski et al, "Radio resource allocation in third generation mobile communication systems", IEEE Communication Magazine, Feb. 2001, pp. 117–123
4. S.Dixity et al, "Resource management and quality of service in third generation wireless networks", IEEE Communication Magazine, Feb. 2001, pp. 125–133
5. M.Casoni et al, "Admission control in T/CDMA systems supporting voice and data applications", IEEE Trans. on Wireless Comm., Vol. 1, No. 3, July 2002, pp. 540–548
6. M.El-Kadi, S.Olariu & H.Abdel-Wahab, "A rate-based borrowing scheme for QoS provisioning in multimedia wireless networks", IEEE Trans. on Parallel & Distributed Sys., Vol. 13, No. 2, Feb. 2002, pp. 156–166
7. L.Ortigoza-Guerrero & A.H.Aghvami, Resource Allocation in Hierarchical Cellular Systems, MA, Artech House, 2000
8. M.Zhang & T.P.Yum, "The non-uniform compact pattern allocation algorithm for cellular mobile systems", IEEE Trans. on Veh. Technol., Vol. 40, May 1991, pp. 387–391
9. J.Choi & J.A.Silvester, "A fair-optimal channel borrowing scheme in multi-service cellular networks with reuse partitioning", IEEE Int. Conf. on Universal Personal Comm. (ICUPC'98), 1998, pp. 261–265
10. T.S.P.Yum & M.Chen, "Dynamic channel assignment in integrated-services cable networks", IEEE Trans. on Comm., Vol. 42, No. 2/3/4, 1994
11. D.Gross & C.M.Harris, Fundamentals of Queueing Theory, 2nd Ed., John Wiley & Sons, 1985

A New Bandwidth Guaranteed Routing Approach for Online Calculation of LSPs for MPLS Traffic Engineering

Karl Hendling, Brikena Statovci-Halimi, Gerald Franzl, and Artan Halimi

Vienna University of Technology, Institute of Communication Networks,
Favoritenstrasse 9/388, A-1040 Vienna, Austria
{karl.hendling, brikena.statovci, gerald.franzl,
artan.halimi}@tuwien.ac.at
http://www.ikn.tuwien.ac.at

Abstract. This paper presents a fast on-line routing algorithm for dynamic routing of label switched paths (LSPs) with bandwidth guarantees in MPLS networks, which handles requests that arrive one at a time without exploiting a priori knowledge of the traffic characteristics. Trying to avoid exacting calculations for each on-demand LSP request (e.g., maximum flow computation), we introduce a new link weight function for path selection. The link weights are calculated as a function of residual network and link capacity, hence we call the approach *Residual Network and Link Capacity* (RNLC) routing algorithm.
In terms of computer simulations we compare the performance of this new routing algorithm with four other on-line routing algorithms in two different network scenarios. Simulation results exhibit better performance of RNLC even if compared to more complex algorithms. We highlight that the new algorithm is fast and scalable due to its considerably low complexity.

1 Introduction

Traffic engineering is one of the main reasons for implementing multiprotocol label switching (MPLS) in IP backbone networks. This capability of MPLS is based on the fact that it efficiently enables explicitly routed paths, called label switched paths (LSPs), to be created between ingress and egress nodes. As a result, traffic flows can be controlled and engineered through the network. For an explicit LSP the route is determined at the ingress node. Once an explicit route is determined, a signaling protocol such as CR-LDP or RSVP-TE [1] is used to establish the LSP to the egress node.

The main goal of Internet traffic engineering is to efficiently optimize the performance of operational networks [1,2,3,4] in order to avoid the well-known shortcomings of the typical destination-based IP routing. Traffic engineering attempts to reduce or even avoid congestion hot spots and to improve the resource utilization across the backbone IP network. This may primarily be done by evenly distributing the incoming traffic over the available links in order to obtain balanced traffic scenarios. One core concept of traffic engineering, which is actually often identified as traffic engineering itself, is route optimization. Especially in networks with rather unbalanced traffic distributions, it can be applied to enhance the overall network quality. Another advantage is the support of specific quality of service (QoS) levels agreed with Service Level Agreements (SLAs) for services that need certain QoS requirements (e.g., a specific packet loss ratio, delay/jitter).

Current routing algorithms are mostly based on shortest path schemes, thus leading to unbalanced load distribution inside the networks and mostly do not consider QoS requirements. An efficient path selection procedure should enable the selection of a *feasible path* while achieving efficient resource utilization. A *feasible path* is one that has sufficient residual resources to satisfy the QoS constraints of a connection [5]. While a feasible path can be selected by a shortest path algorithm, if constrained by one metric only, additional optimality constraints need to be imposed to achieve efficient resource utilization. Several path selection schemes have been proposed and evaluated in the literature including *widest-shortest path* [6], *shortest-widest path* [7,8,9], and *utilization-optimized* algorithms. The *minimum interference routing algorithm* (MIRA) introduced in [10,11] takes into consideration future demands for its routing decision. Another routing algorithm inspired by MIRA was proposed in [12], which we call WSC (Wang-Su-Chen). MIRA and WSC both are *non-greedy* on-line routing algorithms, independent of the actual traffic profile. They achieve a better resource utilization in the network, however introducing high computation time. This can be of great importance when considering backbone networks where the ingress nodes operate at high loads, and have limited computing power.

Trying to avoid the maximum flow [13,14] computation, which introduces additional computation time, we present a new link weight function, which combines the following three criteria: saving of residual link bandwidth, optimal usage of network capacity, and minimization of path lengths. We calculate the link weights as a function of residual network capacity, link capacity, and a constant. Therefore, we call the algorithm *Residual Network and Link Capacity* (RNLC) routing algorithm. Simulation results exhibit better performance than MIRA and WSC for the studied scenarios.

Furthermore, we study the relationship between the performance of a routing algorithm and the network scenario. We show that the network scenario has a great impact on the performance of an algorithm. For comparison purposes we evaluate four algorithms: *minimum-hop algorithm* (MHA), *shortest-widest path* (SWP) algorithm, *minimum interference routing algorithm* (MIRA), and *Wang-Su-Chen* routing algoirhtm (WSC). The evaluated performance is compared with the RNLC routing algorithm in terms of different performance parameters and network scenarios.

The paper is organized as follows. Section 2 presents related work, and Section 3 introduces the problem definition and statement we consider in this work. Section 4 provides a detailed introduction to the new RNLC routing algorithm, and Section 5 gives a detailed complexity analysis of all studied algorithms. Section 6 points out the limitations of studied algorithms and, Section 7 introduces the studied simulation scenarios. Section 8 discusses the performance results and, Section 9 delivers some concluding remarks.

2 Related Work

The most commonly used algorithm for routing LSPs is the *minimum-hop algorithm* (MHA), where a feasible path with the least number of hops (links) connecting an ingress-egress pair is chosen. MHA gives highest priority to minimize resource occupation, however this can create bottlenecks for future flows, consequently leading to an under-utilized network.

Another routing proposal is the *shortest-widest path* (SWP) algorithm [7,8, 9], which considers two criteria. The first one is to pick the path(s) with the maximum reservable bandwidth amongst all feasible paths. If more than one such path exists, the one with the minimum-hop count is chosen. SWP gives highest priority to balance the network load across all links, however due to preferring detours less LSPs can be established.

In [10,11], the *minimum interference routing algorithm* (MIRA) has been proposed. Bandwidth guaranteed LSPs are explicitly routed such that minimum interference occurs between all possible connections (i.e., defined ingress-egress pairs) in order to be able to accommodate future LSP set-up requests, as well as LSP re-routing requests caused by link failures. Most importantly, this heuristic algorithm exploits information in terms of pre-defined ingress-egress communication pairs unlike other proposed *greedy* schemes. As mentioned above, MIRA performs its routing decision based on the interference level according to demands from other ingress-egress pairs. Interference is quantified by the notion of *critical links.* Critical links are links with the property that whenever an LSP is routed over these links the maximum flow values of one or more ingress-egress pairs decrease. The LSP should avoid the critical links as far as possible, which is achieved by generating a weighted graph where the weights assigned to the links are proportional to their criticality. The authors propose three different weighting schemes. For performance studies we apply the scheme where weight portions are inversely proportional to the maximum values, i.e., $\alpha_{sd} = 1/\theta_{sd}$, where θ_{sd} is the maximum flow value for the ingress-egress pair (s, d). This weighting implies that the critical arcs for the ingress-egress pairs with lower maximum flow values will be weighted heavier than the ones for which the maximum flow value is higher. MIRA gives its priority to minimize the criticality of LSPs and thereby minimizes the number of rejected future requests. The main complexity of MIRA per LSP request is given by $O((p-1)n^2\sqrt{m})$ ($p-1$ times maximum flow computation [13,14]) and $O(m^2)$ (critical links calculation), where p denotes the number of ingress-egress pairs, n denotes the number of nodes, and m denotes the number of links. However, MIRA focuses exclusively on the interference effect on single ingress-egress pairs, and is not able to estimate the bottleneck created on links that are critical for *clusters* of nodes.

Inspired by MIRA, Wang et al. [12] have proposed and studied another routing algorithm for bandwidth guaranteed LSPs. Similar to MIRA, WSC is an on-line algorithm, and is independent of traffic profiles. WSC is able to overcome some of MIRA's drawbacks (pointed out in [12,15,16]) by taking into account the overall bandwidth blocking effects of routing an LSP request. The main complexity of WSC per LSP request is given by $O((p-1)n^2\sqrt{m})$ ($p-1$ times maximum flow computation [13,14]).

Another approach called *Profile Based Routing* (PBR) is presented in [15, 16]. PBR is different from MIRA and WSC and assumes both known ingress-egress pairs and a traffic profile between them. A traffic profile is derived from measurements or service level agreements (SLAs) as a rough predictor for future traffic distribution. PBR uses the traffic profile in the pre-processing step (one multi-commodity flow computation), to determine certain bandwidth allocations on the links of the network. The on-line phase of the routing algorithm then routes LSPs using a shortest path like algorithm exploiting the additional information from the pre-processing phase, i.e., occupying resources according to the pre-allocated bandwidth.

3 Problem Definition and Statement

We model the network as a graph $G = (V, E)$, where V ($|V| = n$) denotes the set of nodes (routers), and E ($|E| = m$) denotes the set of links. A subset of nodes is assumed to be ingress-egress nodes, between which LSPs can be set-up. However, it is not necessary that there is a potential LSP between every ingress-egress pair. We assume that all ingress-egress pairs are known in advance and denoted by a set P ($|P| = p$). Each LSP set-up request arrives at an ingress node, which in turn determines an explicit bandwidth satisfying route. To determine the route, each ingress node needs to know the entire topology of the network and the current link states. The residual link capacity of link l is denoted as $R(l)$ and the residual network capacity as $N_c = \sum_{\forall\, l \in E} R(l)$, i.e., the sum over all $R(l)$ (with a complexity $O(m)$). Therefore, we assume that the entire topology is either known administratively or that a link state routing protocol is operational, and that its link state database is accessible. The routing protocol database keeps track of all residual link capacities, and we assume that all initial link capacities are known and thereby the initial network capacity also. Failures of LSPs due to link faults are detected from signaling protocol (e.g., CR-LDP or RSVP-TE) information by the edge nodes. The link state database is updated by the routing protocols, and edge nodes can then request a re-routing of the LSPs.

A request for an LSP set-up r_i is defined by a triple (s_i, d_i, b_i), where $(s_i, d_i) \in P$, s_i is the ingress node, d_i is the egress node, and b_i represents the amount of bandwidth required by the LSP. All QoS requirements for the flow have been folded into the bandwidth b_i. Furthermore, we assume that requests for LSPs arrive on-line, one at a time, and there is no knowledge of the characteristics of future demands. The objective is to find a path for LSP request r_i in the network from s_i to d_i along which each link has a residual capacity of at least b_i, otherwise the request r_i is rejected. In this work, we only focus on the establishment of bandwidth guaranteed paths.

4 Residual Network and Link Capacity Routing Algorithm

This section presents the new routing approach, based on *residual network and link capacity*, for short RNLC routing algorithm. RNLC provides a new link weight function which combines three criteria: saving of residual link bandwidth, optimal usage of network capacity, and minimization of path lengths.

The weight function is given by

$$w(l) = \frac{N_c}{R(l)} + C, \tag{1}$$

where N_c $(= \sum_{\forall\, l \in E} R(l))$ is the current residual network capacity and $R(l)$ is the current residual (i.e., unreserved) link capacity on link l at the arrival event of request r_i. The constant C determines the dynamic behavior of the generated link weights. Fig. 1 shows the dependence of this weight calculation scheme on $R(l)$ and N_c.

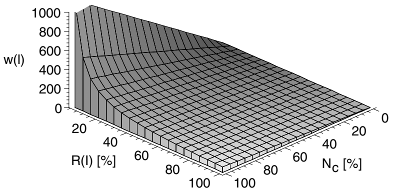


Fig. 1. Link weights as a function of residual network and link capacity.

If the residual link capacity approaches zero, the weight approaches infinite, thus eliminating all links with insufficient residual capacity. The dependence of the weight function on residual network capacity needs some detailed consideration: In case of low network load (i.e., high residual network capacity), links with less residual capacity (higher load) are assigned considerably higher weights than less loaded links. Consequently, paths over lightly loaded links are preferred and heavily loaded links are avoided. This keeps as many links as possible available for future requests, i.e., intends to avoid congestion. In case of high network load (i.e., low residual network capacity), all link weights are approximately the same as long as there is sufficient residual capacity on the links. Therefore, the minimum-hop path is preferred, and routing is performed subject to minimum resource occupation, leaving a maximum of resources available for additional requests. Reflecting, if the same weighting as the one used for low loads is applied, detours would be preferred, which save some residual link capacity (bandwidth) on individual links, but due to their higher number of hops, these paths would occupy more network capacity, and consequently reduce the available resources for future requests.

With the additive constant C the dynamic of the scheme can be controlled. If C is chosen very big, the scheme behaves like minimum-hop routing, still having the advantage of eliminating links with vanishing residual capacity. The smaller C is set, the stronger the link weights reflect the distribution of the load on the links, i.e., smaller C increases the variance of the link weights. This constant C should be chosen according to topology, meshing degree and traffic distribution — an evaluation on C is presented in Section 8.

This weight calculation scheme with its complexity $O(m)$ is fast, and routing can be done according to the Dijkstra or Bellman-Ford algorithm, both well known and common. The only drawback compared to shortest path and widest bottleneck bandwidth scheme is that the link weights can not be calculated autonomously by adjacent nodes, because the residual network capacity N_c is required. As previously mentioned, N_c is calculated as the sum over all $R(l)$ and therefore a network-wide distribution (or view) of $R(l)$ is required. Compared to MIRA and WSC this scheme is by far less complex, as no maximum flow computation is required, and both rely on accurate knowledge of link states as well.

Studying the fairness of the weighting scheme, we consider the mean link weight $\overline{w}(l)$ and find it being constant, i.e., equal to the number of links plus C $(m + C)$,

$$\overline{w}(l) = \frac{N_c}{\overline{R}(l)} + C = m + C, \tag{2}$$

where $\overline{R}(l)$ $(= \frac{N_c}{m})$ is the average residual link capacity. This overall stability yields fairness, and the dynamic can be controlled efficiently via the constant C. A similar scheme might be defined by using percentages instead of absolute figures for residual capacities. In that case C needs to be chosen accordingly smaller, e.g., $C = 1$ would quite severely leverage the dynamic. To achieve the same behavior as with the above shown scheme, C needs to be scaled by $\frac{1}{m}$.

High level view of RNLC routing algorithm:

Input: Graph $G = (V, E)$, the set of residual link capacities on all links, and the request r_i (s_i, d_i, b_i).
Output: A path from s to d, such that for each link l along this path $R(l) \geq b$.

1. Compute the weight $w(l)$ for each link l in the graph $G = (V, E)$ according to equation 1.
2. Eliminate all links l, which have residual bandwidth less than b leading to a reduced graph.
3. Compute shortest path in the reduced graph by means of Dijkstra's algorithm using the corresponding $w(l)$ on the links.
4. Route the demand b from s to d along this shortest path and update the residual link capacities.

5 Complexity Analysis

In this section we analyze the complexity of the studied routing algorithms. MHA needs no special requirements and the overall complexity is the shortest path selection (e.g., Dijkstra with $O(n^2)$ or Bellman-Ford with $O(n^3)$). Similar to MHA, SWP also needs no special requirements. The difference to MHA is that SWP applies a modified Dijkstra algorithm to select the shortest path with an overall complexity $O(n^2)$. RNLC provides a slightly higher complexity than MHA and SWP, additionally to the shortest path selection, $w(l)$ and N_c are required. Each of these values can be calculated with a complexity $O(m)$.

MIRA needs to perform $p-1$ (number of competing ingress-egress pairs) maximum flow computations. Each of these maximum flow computations takes $O(n^2\sqrt{m})$ time. Further, MIRA needs to enumerate the links belonging to minimum cuts with a complexity $O(m^2)$. Afterwards the link weight $w(l)$ for all links is calculated with an overall complexity $O(m)$. Finally the shortest path selection with $O(n^2)$ is performed. WSC performs the same steps as MIRA with the exception that the critical link calculation is skipped.

A detail record is given in Tab. 1, which shows the overall complexity of each routing algorithm when performing the path selection.

Table 1. Complexity of each routing algorithm per LSP request.

routing algorithm	preparation phase	$w(l)$ calculation	shortest path selection Dijkstra
MHA	—	—	$O(n^2)$
SWP	—	—	$O(n^2)$
RNLC	$O(m)$	$O(m)$	$O(n^2)$
MIRA	$O((p-1)n^2\sqrt{m}) + O(m^2)$	$O(m)$	$O(n^2)$
WSC	$O((p-1)n^2\sqrt{m})$	$O(m)$	$O(n^2)$

6 Example Scenario: Limitations of Studied Algorithms

Fig. 2 shows a network scenario termed as *collector-distributor* with its properties. This scenario exhibits a case where MIRA and WSC do not perform as well as RNLC and SWP.

Let us suppose the on-line sequence of 8 LSP requests arrives in the order $(S_3, D_3, 1)$, $(S_1, D_1, 1)$, $(S_3, D_3, 1)$, $(S_4, D_4, 1)$, $(S_3, D_3, 1)$, $(S_2, D_2, 1)$, $(S_3, D_3, 1)$, and $(S_4, D_4, 1)$. MHA always prefers the link $(7, 8)$ when traffic from (S_3, D_3) arrives, thus causing that the LSP request (S_2, D_2) is rejected. According to MIRA and WSC, traffic from (S_3, D_3) always prefers the links $(2, 3)$ and $(3, 4)$, as the link $(3, 4)$ only interferes with (S_4, D_4) traffic, while the link $(7, 8)$ interferes with traffic (S_1, D_1) and (S_2, D_2). They route 3 of the 4 demands from (S_3, D_3) over the links $(2, 3)$ and $(3, 4)$ and 1 demand over the link $(7, 8)$. In this case, the last demand from (S_4, D_4) is rejected, while in the other part of the network, the link $(7, 8)$ is under-utilized. An optimal algorithm routes 2 of the 4 demands from (S_3, D_3) over the link $(7, 8)$ and 2 demands over the links $(2, 3)$ and $(3, 4)$. For this scenario, both RNLC and SWP choose the LSPs like an optimal routing algorithm, and therefore all 8 LSP requests for all ingress-egress pairs can be served successfully.

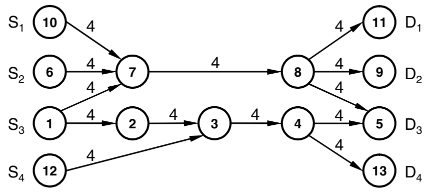


Fig. 2. The collector-distributor scenario.

7 Simulation Scenario

Without real network topologies and large amounts of traffic data, it is difficult to perform meaningful and conclusive experiments. Therefore, we follow the tradition set by other authors, and perform experiments on two handcrafted topologies depicted in Fig. 3 (taken from [16], here called the KL2+ scenario) and Fig. 4 (the more realistic ISP topology, widely used for studies on QoS routing). Fig 3 presents the network topology that has been used in [10,11] to propose MIRA, however, with two additional ingress-egress pairs (S_5, D_5) and (S_6, D_6). Due to these changes of the ingress-egress structure, the interferences change as well.

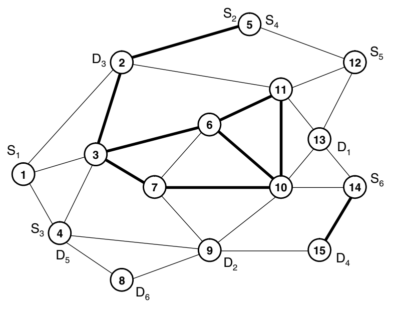


Fig. 3. The KL2+ network scenario.

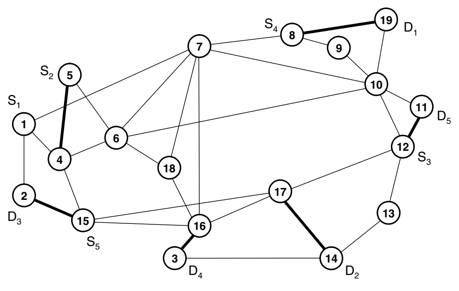


Fig. 4. The ISP network scenario.

The bandwidth of each light link and each bold link is 1,200 units and 4,800 units, respectively. These values are taken to model the capacity ratio of OC-12 and OC-48 links. Each link is bidirectional (i.e., acts like two unidirectional links of that capacity). The ingress-egress pair arrangements for LSP set-up requests are given in Fig. 3 and Fig. 4. All LSP set-up requests have been uniformly distributed among the ingress-egress pairs of the corresponding network scenario. Furthermore, LSP bandwidth demands are taken to be uniformly distributed between 1 and 4 units (only integer values are used).

8 Performance Studies

A study on the influence of the constant C is shown in Fig. 5. We can see that the achievable throughput in the KL2+ scenario reaches a maximum for $C \leq 1$. In contrast, for the ISP scenario, the influence of the constant C is marginal. The achievable throughput is almost constant within the shown scope, however, for $C = 1$ the throughput is negligibly higher. Generally, the higher C is chosen the more shorter paths become preferred, thus the more is the resource occupation minimized. The smaller C is selected, the more is load balancing preferred, thus congestion hot-spots better avoided. Based on this, we set $C = 1$ for both simulation scenarios.

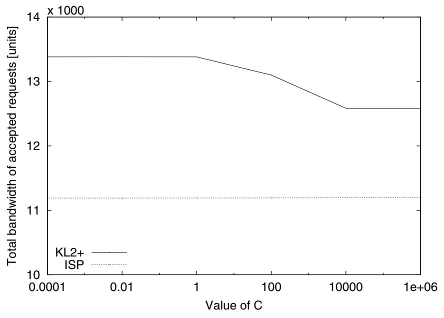


Fig. 5. The influence of C at the two network scenarios.

In the first set of experiments we load the two network scenarios with 8,000 LSP requests. We assume that all 8,000 LSPs are long-lived, i.e., once an LSP is routed it will not be terminated. We perform 20 trials, where for each trial newly randomly chosen long-lived LSPs are generated. Afterwards, we calculate the mean values for all evaluated parameters. Fig. 6 and Fig. 7 show the sum of the remaining maximum flow (allocatable bandwidth) of all ingress-egress pairs, after routing an LSP request with the studied algorithms and updating the link capacities. For each algorithm, the maximum flow (allocatable bandwidth) decreases with the number of accepted requests until a saturation point is reached at which no more requests can be accommodated.

From Fig. 6, we can see that the allocatable bandwidth for MIRA is a little bit higher than RNLC's within the range of $[1,910;4,550]$ requests. The insert exhibits that at higher loads ($> 4,550$ requests) RNLC provides more allocatable

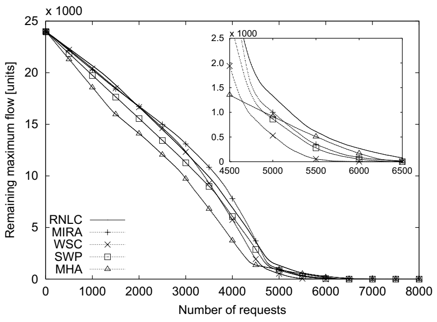


Fig. 6. Sum of maximum flow over all ingress-egress pairs in the KL2+ scenario.

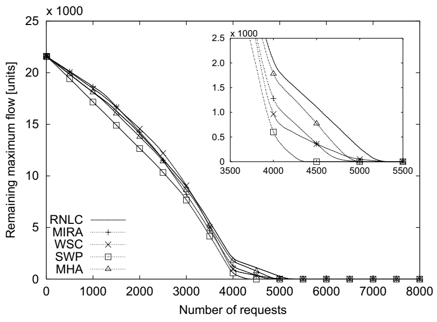


Fig. 7. Sum of maximum flow over all ingress-egress pairs in the ISP scenario.

bandwidth than any other studied algorithm, while MIRA drops behind MHA in a small interval. Fig. 7 exhibits a similar behavior as depicted in Fig. 6. At the beginning, MIRA and WSC provide more allocatable bandwidth than any other studied algorithm, but the decreasing rate (allocatable bandwidth) for MIRA and WSC is larger than RNLC's. Consequentially, RNLC provides more allocatable bandwidth at higher loads ($> 3,160$ requests) than any other studied algorithm.

Fig. 8 and Fig. 9 show the number of blocked requests (rejects) versus the total number of requests. RNLC shows in both figures (Fig. 8–9) the best performance, yielding the fewest rejects. Fig. 8 (the KL2+ scenario) exhibits that WSC causes more rejects than RNLC, MIRA and SWP. Fig. 9 (the ISP scenario) exhibits a better performance of MHA compared to the non-greedy algorithms MIRA and WSC.

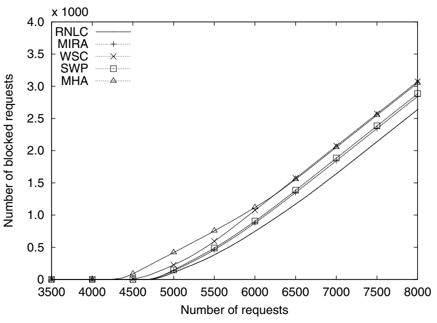


Fig. 8. Blocked requests in the KL2+ scenario.

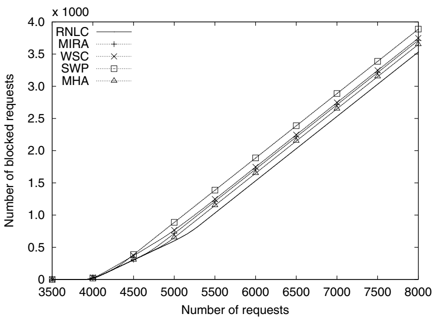


Fig. 9. Blocked requests in the ISP scenario.

Fig. 10 and Fig. 11 show the total bandwidth of accepted requests (throughput). For each studied algorithm, the bandwidth increases with the number of accepted requests until a saturation point is reached. In both figures, RNLC clearly shows the best performance in terms of maximum throughput. Due to the fact of the highest blocking, WSC exhibits the worst performance in Fig. 10 (the KL2+ scenario) above 6,240 LSP requests, thus yields the lowest maximum throughput.

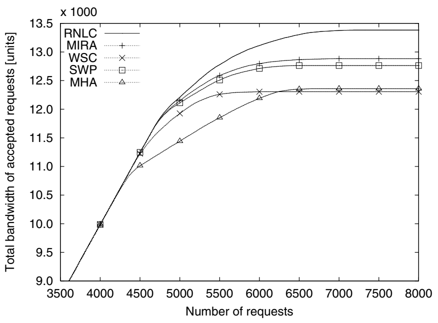


Fig. 10. Throughput of accepted requests in the KL2+ scenario.

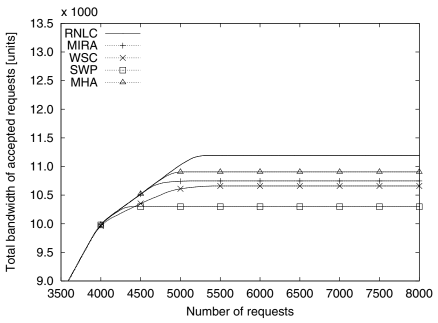


Fig. 11. Throughput of accepted requests in the ISP scenario.

In the second set of experiments, we determine the dynamic behavior of the routing algorithms. We use the same assumptions concerning the ingress-egress pair arrangements, uniform distribution of requests and bandwidth demands among all ingress-egress pairs. Each network scenario is first loaded with 4,000 randomly chosen long-lived LSPs and from this point on, all arriving LSP requests are assumed to have exponential holding time. The exponential holding time is chosen so that an average rate of 2,000 short-lived LSPs has to be accommodated additionally to the 4,000 long-lived LSPs. We run the experiment for 8,000 LSP requests (including the initial 4,000 LSP set-up requests), and perform 20 trials.

Fig. 12 and Fig. 13 show the number of blocked requests of each trial. As these two figures exhibit, RNLC causes smaller number of blocked requests than any other studied routing algorithm. This leads to improved performance and hence provides better overall network resource utilization. The differences in performance of MHA and SWP depicted in these figures show how scenario-dependent the performance of routing algorithms can be. Moreover, MIRA and WSC show instable performance (strong oscillation) while other algorithms exhibit more stable behavior with the same simulation scenarios.

9 Conclusion

In this paper, we have proposed a fast on-line routing algorithm for dynamic routing of bandwidth guaranteed LSPs. The proposed routing algorithm is different from the two commonly used *minimum-hop algorithm* (MHA) and *widest-shortest path* (WSP). This new algorithm avoids using residual capacity of congested links by means of routing traffic demands away from hot spots, even if

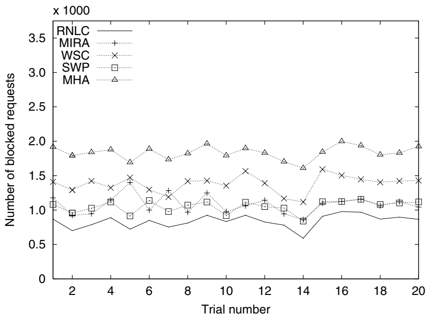


Fig. 12. Blocked requests during 20 independent trials in the KL2+ scenario.

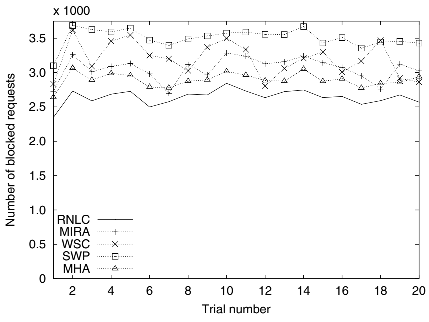


Fig. 13. Blocked requests during 20 independent trials in the ISP scenario.

the traffic then uses slightly longer paths. This increases the acceptance rate for future demands, and consequently reduces the rejection rate, which is especially important when re-routing due to a link failure is performed.

The *Residual Network and Link Capacity* (RNLC) routing algorithm is substantially faster due to much lower computational complexity than the recent proposals, *minimum interference routing algorithm* (MIRA) and the routing approach proposed by Wang-Su-Chen (WSC). This is a crucial point, because a high computation delay is a limitation for on-line routing algorithms. For the studied scenarios, RNLC shows better overall performance.

Moreover, we found that the chosen network scenario has a considerable influence on the performance of the routing algorithms. Especially, WSC shows an unfavorable performance when used with the KL2+ scenario, even though it overcomes some of MIRA's drawbacks by taking into account the overall bandwidth blocking effects of establishing an LSP request.

In practice, network operators may have to find a combination of off-line and on-line algorithms in order to operate their networks more efficiently. Off-line algorithms are used to re-optimize routing occasionally, while on-line algorithms (e.g., RNLC, MHA) allow for rapid response to new traffic demands or even new network conditions, for which off-line algorithms can be prohibitively slow. The scalability and fast response of the proposed routing algorithm make it applicable for large, even huge networks, if the link state broadcast is well implemented.

An aspect to extend our work is to alter C dynamically depending on the traffic demands (e.g., traffic type and size), another to consider off-line optimized $C(l)$.

References

1. Ghanwani, A., Jamoussi, B., Fedyk, D., Ashwood-Smith, P., Li, L., Feldman, N.: Traffic Engineering Standards in IP Networks Using MPLS. IEEE Communications Magazine **37** (1999) 49–53
2. Awduche, D.: MPLS and Traffic Engineering in IP Networks. IEEE Communications Magazine **37** (1999) 42–47

3. Wang, Z.: Internet QoS: Architectures and Mechanisms for Quality of Service. The Morgan Kaufmann Series in Networking. Morgan Kaufmann Publishers (2001)
4. Spraggs, S.: Traffic engineering. BT Technology Journal **18** (2000) 137–150
5. Chen, S., Nahrstedt, K.: An Overview of Quality-of-Service Routing for the Next Generation High-Speed Networks: Problems and Solutions. IEEE Network, Special Issue on Transmission and Distribution of Digital Video **12** (1998) 64–79
6. Guérin, R.A., Orda, A., Williams, D.: QoS Routing Mechanisms and OSPF Extensions. Proceedings of IEEE Global Communications Conference 1997 (GLOBECOM'97), **3** (1997), 1903–1908
7. Wang, Z., Crowcroft, J.: Quality-of-Service Routing for Supporting Multimedia Applications. IEEE Journal on Selected Areas in Communications **14** (1996) 1228–1234
8. Ma, Q., Steenkiste, P.: On Path Selection for Traffic with Bandwidth Guarantees. Proceedings of IEEE International Conference on Network Protocols 1997 (ICNP'97) (1997) 191–202
9. Kamei, S., Kimura, T.: Evaluation of Routing Algorithms and Network Topologies for MPLS Traffic Engineering. Proceedings of IEEE Global Communications Conference 2001 (GLOBECOM'01) **1** (2001) 25–29
10. Kodialam, M., Lakshman, T.V.: Minimum Interference Routing with Applications to MPLS Traffic Engineering. Proceedings of IEEE Computer and Communications Societies 2000 (INFOCOM'00) **2** (2000) 884–893
11. Kar, K., Kodialam, M., Lakshman, T.V.: Minimum Interference Routing of Bandwidth Guaranteed Tunnels with MPLS Traffic Engineering Applications. IEEE Journal on Selected Areas in Communications **18** (2000) 2566–2579
12. Wang, B., Su, X., Chen, C.L.P.: A New Bandwidth Guaranteed Routing Algorithm for MPLS Traffic Engineering. Proceedings of IEEE International Conference on Communications 2002 (ICC'02) **2** (2002) 1001–1005
13. Goldberg, A.V., Tarjan, R.E.: A New Approach to the Maximum-Flow Problem. Journal of the Association for Computing Machinery **35** (1988) 921–940
14. Ahuja, R.K., Magnanti, T.L., Orlin, J.B.: Networks Flows: Theory, Algorithms, and Applications. Prentice-Hall, Inc., Upper Saddle River, New Jersey 07458 (1993)
15. Suri, S., Waldvogel, M., Warkhede, P.R. In: Profile-Based Routing: A New Framework for MPLS Traffic Engineering. Volume 2156 of Lecture Notes in Computer Science. Springer Verlag, Berlin (2001) 138–157
16. Suri, S., Waldvogel, M., Bauer, D., Warkhede, P.R. In: Profile-Based Routing and Traffic Engineering. Volume 26 of Computer Communications. Elsevier Science B.V., Amsterdam (2003) 351–365

New Management Methods for Feature and Preference Interactions

Zohair Chentouf, Soumaya Cherkaoui, and Ahmed Khoumsi

Department of Electrical and Computer Engineering
Université de Sherbrooke
2500, boulevard Université
Sherbrooke (Québec) J1K 2R1 CANADA
zohair.chentouf@hermes.usherb.ca
{soumaya.cherkaoui, ahmed.khoumsi}@usherbrooke.ca

Abstract. This paper deals with a complicated *Feature Interactions* (FI) problem that results from new emerging telecommunications architectures, IP telephony for example. That is because these architectures introduce new complications since service providers and end users have the possibility to express their *preferences* in more sophisticated ways and non telephony applications are combined with telephony services. The article provides examples of new FI that have not been reported before. The main contribution of this work is that it proposes new FI management methods that are required to solve the new FI complications. Old management methods are also considered and some of them are enhanced in order to adapt to the new problem context. Intelligence requirements necessary to implement the proposed management model are finally underlined.

1 Introduction

The Feature Interactions (FI) problem can be defined as the situation in which two feature instances undesirably affect one another when running together [1, 2]. In the literature, this definition or similarly formulated ones are adopted [3]. They have in common to associate undesirability with altered feature behavior or user confusion with that behavior. A scholastic example usually evoked is the FI that occurs between Call Forwarding (CF) and Originating Call Screening (OCS) services. Suppose user A has OCS with which he has specified C as forbidden destination address. Thus, calls initiated from A's device to C are blocked by OCS. Suppose user B has CF with which she has specified C as forwarding-to address. This means that every call incoming to B's address is automatically redirected to C. Assume that user A calls user B. A's OCS will not block the call because the destination address is not C, which is the forbidden one. But B's CFU will redirect the call to C. Consequently, user A will find himself talking to C despite he has OCS that blocks calls to C.

In order to manage FI, three main operations may be used: *avoidance*, *detection* and *resolution*. Avoidance means to intervene at service, protocol and/or architecture design stages in order to avoid interactions whose causes are situated at some of these levels, respectively. Detection aims at detecting interactions using suitable test methods, performed on service models which are usually obtained using a suitable service modeling language. Resolution means to find and apply a solution to already detected FI. This article interest goes to FI detection and resolution. So, when the term "management" is used, it will mean these two operations.

FI management can be performed at two stages:

– off-line: before services are deployed in the network, or
– on-line: either when services are not yet publicly available (test phase) or after the services are available to public. [3]

We should also distinguish between two situations into the phase where services are available:

– management is done at runtime, i.e. when services are running, or
– it is done before runtime, when services are not running.

New emergent telecommunications architectures such as IP telephony, give more possibilities for end users to express their preferences and personalize services. They also bring the possibility to have services running in end user devices and third party provider servers. As user preference modeling possibilities grow, the interactions between preferences and between preferences and services also become more and more frequent and complex.

In [5], we have specified an architecture for runtime FI detection in IP telephony based on SIP protocol [6]. In the present article, the aim is at completing that specification by adapting it to the FI problem new constraints and management requirements without being restricted to IP telephony. The reached generality level permits to apply the model to the other emergent architectures such as JAIN [7] and 3GPP [8]. We propose new runtime and before-runtime management methods that should be added to old ones. Such a management, which includes runtime and before-runtime methods, is called *hybrid*. The new methods and old ones are consistently gathered into a single model. We here present a general model that encompasses many implementation options. This will be explained in the following sections.

In [4], we propose to formulate user preferences using a *preference rule* formalism. A preference rule is a condition–action rule, where condition part encompasses the events that, if they occur, the user preference, represented by the action part, is executed. Preference rules should be written in a suitable user preference language. At runtime, the detection procedure execution environment should be able to extract information from the exchanged call messages. For example, the caller and callee addresses, the communication subject, the used media type, etc. Such information is necessary for FI detection and should be modeled using the same language as services and preference rules.

It is worthy noting that, depending on the telecommunications architecture, the preference language expressive power and the deployed services, one can have preference rules that behave like services, service behaviors that can not be modeled as preferences and preference rules behaviors that are not available as services. For example, in a SIP architecture where preferences are modeled using CPL [9], forwarding all incoming calls to a given address can be modeled as a preference rule. If the Call Forwarding service is available, the same behavior may be available as a service provided by the service vendor and as preference rules programmed by end users. Otherwise, the only way to have this behavior is to model it as a preference rule. Call Intrusion, which means the possibility to interrupt a call between two users in order to talk to one of them, is not possible to model as a preference rule because CPL does not permit to model such a behavior.

The paper is organized as follows. Section 2 provides a taxonomy for the FI problem causes and presents a list of examples of new FI that are engendered by the characteristics of emergent telecommunications architectures. Section 3 describes the architecture model we adopt and the proposed associated FI management model. It shows, case by case, how FI presented in Section 2 should be managed by this model. The paper ends with some concluding remarks in Section 4.

2 Taxonomy of FI Causes

The main characteristics of new emerging telecommunications architectures are the following:

- Non telephony applications, such as email, can be involved with telephony ones.
- Some services, preference rules and non telephony applications are roomed by end user devices.
- End users specify preference rules, but they do not expect to have to detect and solve FI.

These characteristics translate in a heterogeneous architecture that encompasses nodes with different technologies. For the sake of generality, let us suppose the following hypothetical architecture model, which has those characteristics:

- End users can have preference rules on their devices and in a network device as well.
- They also can have services in their devices and in the network.
- They also can have non telephony applications that are combined to services and preference rules, in their devices and in the network.

In such an architecture, that contains characteristics of emerging telecommunications ones, the following FI causes may arise:

- Violated assumption

Services and preferences are modeled supposing a set of assumptions about the system behavior. The latter encompasses basic architectural policies about addressing (one-line-per-user or many-lines-per-user, 911 is the emergency address), under which conditions signaling events are issued ("busy" is issued if the callee line is in use), billing, etc. In some cases, a service or preference rule behavior alters an assumption or hides some of its results from other services or preference rules. Thus, the latter can not react as expected despite the call processing context is really the one they should react to.

Example 2.1: user B has Call Waiting service and user A a preference rule that sends an email to the callee if the latter is busy. Suppose B's Call Waiting implementation does not send a "busy" response because, for example, the caller is connected to a music on hold server that sends call acceptance response. If user A hangs on because he does not want to wait, the preference rule envisioned by user A will not be run despite user B is busy.

– Incompatible actions

There are four interaction causes that belong to this category:

– One action forbids the other one

Example 2.2: user A has a preference rule that forbids forwarding calls initiated by A and user B has Call Forwarding service. If A calls B, there will be an interaction between two actions: the preference rule action that forbids forwarding calls and the Call Forwarding service action that forwards calls.

- The two actions cause user confusion

Example 2.3: user B has a preference rule that forwards to her mobile all incoming calls from persons with whom she had a long distance call since last Monday. On Wednesday, B specifies A as a forbidden caller in Terminating Call Screening service. Suppose A calls B. Depending on the service and preference rule implementation and network conditions, they may be run together. This would cause a confusion to user A since he will be forwarded to B's mobile and receive a message like "Sorry, call later" from the screening service.

- It is impossible to run the two actions together

Example 2.4: in the precedent example, depending on the implementation and network conditions, it may be impossible to run the preference rule and the service together and one among them may be excluded. This may result in violating B's intention.

- The attempt to run the two actions together causes a system ambiguity

Example 2.5: in the precedent example, if the service and preference rule are located on the same node, it may be ambiguous for that node to run one or the other. This depends on their implementation and the rooming node policies. It was not reported for these service and preference rule to engender an ambiguity, but the situation is known to occur in the Intelligent Network when, for example, Call Waiting and Call Forwarding services are triggered by the same event [1, 10].

– Forbidden communication content

The communication content relates to the conversation subject or the media type and/or content that is conveyed by the exchanged messages.

Example 2.6: suppose user A has a preference rule that forbids to receive web pages that contain advertisement of religious sects. User B, being enthusiastic member of a sectarian organization, she has a preference rule that sends to all callers the organization's web page. This situation results in a preference conflict if A calls B.

– Ambiguous event content

Some signaling events may be associated a media in order to produce a user notification effect, such as busy tone. The media is the event content. The interaction occurs when the user associates the same content to two distinct events.

Example 2.7: the same tone or melody associated to incoming call and email reception events.

– Action priority order conflict

Depending on the architecture, a negotiation mechanism may be available such that services or preference rules of the caller and callee exchange a set of action proposals and counterproposals in order to negotiate a common solution which results in a single action to perform. The proposed actions should be weighted in order to express the preferable priority order. The other side issues a counterproposal by suggesting its own weighting order for those actions. If the proposal and counterproposal contain different weighting orders, there is an action priority order conflict. The negotiation mechanism should permit the two sides to negotiate a common solution by relaxing their preferences.

Example 2.8: user B has a preference rule that is triggered on call request reception. This rule specifies a list of forwarding destinations in the following preferable order: secretary (high), voicemail (medium), mobile (low). This proposal is communicated to the caller, say user A. The latter, however, has specified the following preferable order: secretary (medium), voicemail (low), mobile (high). This list is sent to user B as a counterproposal. Since it is different from B's proposal, there is an action priority order conflict. The two devices should then negotiate a common action to perform. Notice that the negotiation protocol may be different from the one adopted in this example. Weights may be expressed differently.

– Timing

Timing means at what time a particular event occurs or for how long an event lasts. Events may be delayed by external sources, such as the user, and then may have different effects. It is known that in POTS, for example, how long the switchhook is pushed can determine whether it signifies a flashhook or a disconnection [1]. The following example is a new FI that has not been reported before and that may occur in new telecommunications architectures.

Example 2.9: user B has a user device-located Forward-On-No-Answer service for which she has specified a preference rule that triggers this service after 6 rings, when a call request is received. User A has a user device-located Email-On-No-Answer service for which he has specified a preference rule that triggers it after 4 rings. So, this service sends an email to the user called by A if this user does not answer the call after 4 rings. Suppose A calls B when she is not available. After 4 rings an email is sent to B and A will no more hear ringing tone and then will hang on. Here is an interaction between the two preference rules because intentions of both users are not respected: A's one because his device concludes too early that B is not available, and B's one because her device does not actualize her intention at right time (to forward the caller). If user A specified a number of rings equal or greater than the number specified by user B, he would be forwarded to another person and this could be preferred for both users.

– Too generalized action condition

When specifying preferences, end users could neglect to consider special cases to be treated conveniently. In such a case, they associate a too generalized condition to a given action.

Example 2.10: user B has Do-Not-Disturb service programmed to reject all incoming calls on a specified interval of time, say between 10.30 and 13.00. Suppose user A calls user B on that interval of time, specifying a high call priority order and a precise subject. The call request will be rejected by B's Do-Not-Disturb service. There is an interaction between A's preference that translates in assigning a high call priority order and B's Do-Not-Disturb service.

– Preference formulation incompleteness

When specifying preferences, end users could neglect to consider all the possible cases. Thus, the preference formulation results in neglecting some cases that should be reacted to by the same action or an equivalent one.

Example 2.11: user A does not want to communicate with one of his old clients, say user B, because the latter became a competitor. A specifies B's address as a forbidden caller address in Originating Call Screening service and a forbidden callee address in Terminating Call Screening service. Therefore, he is expecting no call will be originated towards B and no call from B will be answered. Suppose user A's secretary calls B for some reason. The call attempt does not succeed because it is intercepted by Originating Call Screening, but an email is sent to B because A has a preference rule that sends an email to potential clients, on every call attempt, may they answer or not, with an updated new product list. Consequently, there is an interaction between Originating Call Screening service and the preference rule, which results in violating user A's intention. The latter being not to communicate with B.

Notice that the following FI causes are known in the FI problem [1]: violated assumption, incompatible actions, ambiguous event content, and timing. The following are new FI causes: forbidden communication content, action priority order conflict, too generalized action condition, and preference formulation incompleteness. Examples 2.3–2.7 and 2.9–2.11 are new interactions. They involve user preferences. Example 2.1 involves user preference but a similar one, that involves services only, is presented in [1]. Example 2.2 is quoted in [2, 11].

3 FI Management Methods

3.1 Related Work

Detection

Most FI detection work reported in the literature uses formal methods. Services are modeled using a formal language. Then, formal techniques are used in order to detect possible interactions. The detection is done off-line or on-line but before runtime. The commonly used formal techniques are temporal logic [12], theorem proving [13], Petri nets [14], extended finite state automata (SDL language for example) [15], processes algebra (LOTOS language for example) [16], etc. Informal methods are also used to detect FI off-line. For example, in [17] natural language processing is used to identify interactions between service logic requirements modeled by textual descriptions.

Resolution

FI resolution uses two main methods: *restriction* and *negotiation*. Restriction results in specifying a *precedence* or *exclusion* rule to apply in order to avoid to two services to interact. Precedence means to run one service before another and exclusion means to exclude one service and run only the other one.

The solution proposed in [18] uses software static and mobile agents to apply restriction rules specified during an off-line detection and resolution phase. Other examples of restriction can be found in [19-21].

There are mainly two models that use negotiation: *negotiating agents* and *polite service*.

– Negotiating agent model

The negotiation method to solve FI was inaugurated by the work due to Griffeth and Velthuijsen [11]. These authors proposed and prototyped a negotiation architecture in which end users have the possibility to specify their preferences. The negotiating agents represent users, network operator and network devices. FI are detected and solved at runtime. The resolution is performed in respect of user preferences. The key idea of the model is that resolution should not favor one feature over another, using restriction. When an agent intends to perform an action, it proposes that action to other agents with whom it is involved in the current call. The latter agents may refuse the proposed action because it violates their preferences. In such a case, each agent proposes another action or set of sub-actions to be substituted to the first one.

In the architecture proposed by Amer *et al* [2], network and user devices are controlled by device agents. User preferences are contained in user agents. A device agent may contain multiple feature agents that implement features. Users specify their preferences in terms of fuzzy values they assign to preference parameters. Those values express the percentage of preference the user assigns to each parameter.

– Polite service model

In this model, management also operates at runtime but does not use a detection phase before resolution one. It is proposed to operate in an IP telephony architecture based on SIP protocol [6]. Agents represent user devices. The key idea underlying [22] is that when an agent intends to perform an action that may affect another agent involved in the same call, it should ask that agent if it agrees or not. For example, Call Forwarding should not be executed before asking the authorization of the caller's agent. If the latter refuses to be forwarded, the callee agent should propose another action, voicemail service for example. The other work that has to be classified under polite service model is [23]. SIP is extended in order to convey negotiation communication. The latter, as in [22], aims at avoiding actions that may engender interactions. The call request conveys a list of action alternatives the caller agent proposes to be performed by the callee agent. The callee agent either accepts one action or refuses the whole list. This translates in accepting or rejecting the call, respectively.

3.2 New Management Methods

In order to manage FI, the architecture model that has been adopted in the beginning of Section 2 needs to satisfy the following requirement: *end user devices, non telephony servers and any telephony node that may be involved in FI should contribute together in order to manage FI.*

For this requirement to be satisfied, we suppose the architecture enhanced by FI management (FIM) agents located in all the devices and components that should cooperate in managing FI. Lets us call them like follows:

- UFIMA: User device Feature Interaction Management Agent, located on user devices.
- NFIMA: Network component Feature Interaction Management Agent, located in the telephony and non telephony nodes.
- CFIMA: Central Feature Interaction Management Agent, located in the network.

We suppose UFIMA and CFIMA capable of detecting FI. They also should contain a FIM knowledge that permits them solving the FI they detect. All the agents should be able to communicate with each other. Services and preference rules are coded in a common language. We suppose UFIMA has models of services and preference rules that are in the corresponding user device and in the network as well. CFIMA is supposed to be periodically provided by UFIMA and NFIMA with models of services and preference rules located in user and network devices. The FIM designer should determine which FI to be detected by UFIMA and which by CFIMA, in order to avoid redundancy in management effort. For example, UFIMA may be designed to deal with preference incompleteness FI only. We propose that FI detection should be done before runtime for the services and preference rules that belong to the same user. We propose also that it should be done before runtime, for services and preference rules that belong to users which frequently call each other. At runtime, detection is done for users which are not frequent callers to each other and for services and preference rules that contain information that is specified at runtime only, like in Example 2.10 as explained below.

Some FI causes that arise in the architectural new context we have adopted need new management methods:

– Preference formulation completion

There are two kinds of preference completion:

- *Preference generalization:*

The user preferences are formulated in a way that covers only a part of the situations they should deal with and neglect others. The preference formulation should then be generalized to the remaining situations. This is done before runtime by UFIMA or CFIMA, depending on whether the preference rule is in the user device or in the network, respectively. Preference generalization method deals with FI whose cause is preference formulation incompleteness but may also solve FI whose cause is violated assumption. The preference generalization FI resolution method is a new one that has not been reported in the literature.

Example 2.1 FI belongs to violated assumption type and may be managed in the following manner. Before runtime, user A UFIMA detects that "busy" event that is specified in the preference rule as triggering condition may be deduced from other events too, like "put on hold". This is a preference formulation incompleteness. So, depending on its FIM knowledge, UFIMA either:

– warns user A giving him a list of other situations that may be considered as implying "busy" event, and/or
– generalizes user A's preference formulation by adding those situations in the triggering condition. Depending on UFIMA knowledge, the update of the preference rule triggering condition may add as another condition, the fact that the user hangs on, in order to avoid having the situation where the user waits for the call to be answered and sends an email that means he will not talk.

In Example 2.11, the FI belongs to another cause: preference formulation incompleteness. It occurs because user A's screening preference expressed by the screening services logic is incomplete. Screening should be done on calls but also generalized to emails. For this kind of FI to be solved, preference generalization should be applied before runtime. UFIMA detects the preference incompleteness and then, depending on the FIM knowledge, either:

  - warns the end user and may help him expressing more complete preference rule, or
  - directly performs the necessary completion without help from the end user.

In both example cases, whether UFIMA asks the user or acts directly, is to be decided by the FIM designer when implementing the preference generalization method. The designer has also to determine when to perform the detection, for example, each time a service or a preference rule is updated.

– *Exception*

The user preferences are formulated in a way that does not take into consideration some special situation cases, where the user preferences should not be applied or should be modified. The exception resolution method prevents user preferences to be applied to those special cases, may be after asking the user. It may also modify user preferences in order to avoid FI. Whether asking the user or not, is an implementation option that may depend on the nature of involved services or preference rules. Notice that exception is not a new management method but it needs to be adapted to FI. In fact, the negotiation performed in [2, 11] aims at performing an exception in order to relax a too generalized preference.

In Example 2.9, the FI is caused by a timing situation. It may be solved by an exception. The example formulation supposes the two services are located on user devices. Assuming users A and B frequently call each other, CFIMA detects before runtime the FI and informs corresponding UFIMAs about it. The latter should ask users in order to perform an exception by changing the specified number of rings. If users A and B are not potential callers to each other, CFIMA detects the FI at runtime. If UFIMA are designed to negotiate such an interaction type, CFIMA inform corresponding UFIMA in order to negotiate. If they can not negotiate, CFIMA may be designed to let processing the call and then inform UFIMA about the FI. UFIMAs then behave like in before-runtime situation.

The FI caused by the too generalized action in Example 2.10 can be solved by performing an exception at runtime. It is not possible to perform the resolution before runtime because the call subject and emergency level are specified at runtime. UFIMA should be able to check at runtime if the priority order specified in caller request (user A in this example) would interest user B. UFIMA should be able to learn and maintain a keyword repository of the subjects preferred by the end user. When receiving a call with pretended high emergency, UFIMA then uses that repository to decide if it should warn the end user despite he has blocked incoming calls.

– Device cooperation

In order to avoid the FI of Example 2.6, cooperation is needed from web servers. So, the FIM becomes a responsibility shared, not only by telephony devices, but also by all other components which room applications that are combined with telephony services. This responsibility is carried by NFIMAs. A web content screening feature should be available on the involved web server. Before runtime, user A UFIMA informs the suitable web server NFIMA and the latter instructs the screening feature to screen incoming sectarian web content. In the same manner, each involved non telephony device should room a NFIMA in order to manage FI. In order not to be a burden for the device, NFIMA should operate autonomously in order to cooperate with other FIM agents.

– Negotiation
 Action priority order conflict needs a negotiation protocol that is different from the ones reported in the related work for FI problem (see Section 3.1). The resolution of Example 2.8 FI should be done at runtime in the following manner. When user B's preference rule is triggered, the list of weighted actions is communicated to user A UFIMA. The latter issues a counterproposal in which it specifies other weight values for those actions. Then, user B UFIMA proposes other values in an attempt to adapt to user A UFIMA counterproposal. The process may run over many steps until reaching a common solution that is to agree on a unique preferred action to perform. If the negotiation does not result in a common solution in a reasonable time, CFIMA should intervene to apply a solution it chooses from the previously rejected proposals or to apply a restriction. This depends on the FIM knowledge it possesses.

3.3 Applicability of Old Management Methods

The management models presented in Section 3.1 do not cater with all the constraints considered in this article and which better reflect the new telecommunications reality characterized by network heterogeneity and end user programmability (preference formulation advanced possibilities). The proof is that the management model we propose introduced two new FI management methods: preference generalization and device cooperation; and reformulated two old ones: exception and negotiation.

Formal methods do generally not convene to the architecture model we adopted because end users would be constrained to interpret formal results. They have not the required technical skills to do that. Formal methods also can not be used in the situation where management should be done at runtime. The other management models have the common shortcoming to load services before detecting if they will cause interactions, except [18]. If the resolution needs to exclude a service, this will not be possible since it already started to run, and if a negotiation should be performed, it will be constrained by the actions already performed by the loaded services. Another work [18], presents a different approach from the one adopted in the present work since it partially relies on code mobility. The model of Griffeth and Velthuijsen supposes a hierarchy of the possible actions to perform by services such that, if an action causes an interaction, it may be replaced by another action or set of actions. This assumption does not apply to all the possible actions however. Polite service model represents a partial solution. In fact, it deals only with services and preferences that are triggered by the action to issue or receive a call request but not other SIP messages and events. It also considers only services that involve multiple users and not conflicting services that belong to a single user. Finally, the interaction causes considered in negotiating agents and polite service models do not cover the list presented in this article, particularly the new FI causes. The FI management model presented in this article avoided these shortcomings in order to satisfy the new FI requirements.

Nevertheless, old FI management methods are still applicable to the examples that have not been treated in Section 3.2.

– Example 2.2: at runtime, when A calls B, CFIMA detects an incompatibility between B's Call Forwarding service and A's preference rule that forbids forwarding calls initiated by A. CFIMA has no solution other than applying a restriction that excludes B's Call Forwarding and rejects the call request.

- Example 2.3: if user B's preference rule and Terminating Call Screening service are located on her device, UFIMA performs a FI detection and detects two incompatible actions. Depending on UFIMA FIM knowledge, it either:
  - warns user B about the incompatibility or
  - decides to apply a restriction at runtime that privileges screening and excludes forwarding, for example. If this is done, it should be sure that it does not violate B's intentions. This is expected to be guarantied by the fact that UFIMA policies do not violate the user intention.

If both or one among the preference rule and the service are located in the network, and depending on the implementation management solution policies, CFIMA may do that detection and then behave like UFIMA does in the previous case.

- Example 2.7: The two events are given the same melody by the user device. Before runtime, UFIMA performs a FI detection and detects a content ambiguity. It then warns the user about the ambiguity.

3.4 New Intelligence Requirements

The FI management examples presented in Section 3.2 have shown that more intelligence is required to manage FI, especially when this is done by the new methods introduced by the present article. In fact, preference completion (generalization and exception) need from UFIMA to communicate with the user, explain him the FI situation and assist him in completing his preference formulation (examples 2.1 and 2.11). These are new capabilities needed to cater with the growing complexity of the new FI problem. Preference completion (generalization and exception) also requires from UFIMA to be *pro-active*. Pro-activity means the capacity of getting decisions on the right moment without control from another entity [24]. Preference completion also requires, in some cases, UFIMA to be capable of learning user preferences (Example 2.10). Some FI resolution cases require CFIMA to inform UFIMA about the FI in order to solve it (Example 2.9). This *delegation* operation is a special device *cooperation* way. In order to guarantee cooperation from non telephony nodes, they should have a NFIMA. The latter should act *autonomously* and should be able to control some operations of the hosting node, for example to screen web pages (Example 2.6). The *negotiation* protocol proposed in the present article is different from the one used in the negotiating agents and polite service models (see Section 3.1). In fact, this protocol is a generalization of those of these two models. Recall that the protocol used in the negotiating agents model proposed by Amer *et al* consists in one proposed action to which is opposed one action counterproposal. The one used in polite service model permits to choose one action from a proposed list. These two protocols are special cases of the one we proposed for Example 2.8. The protocol, however, needs more intelligence in the negotiating UFIMAs. UFIMA should have suitable knowledge and policies to relax preferences depending on the caller preferences. Notice also that, at our knowledge, negotiation before runtime our model introduced, is a new management method since, in the literature, negotiation is done at runtime.

4 Conclusion

This article presented the complicated FI problem that arises from considering the interactions that involve user preferences and non telephony applications with services. The presented examples of new interactions prove this concept. The new formulation of the problem implies new management methods that, as it has been shown with examples, solve the new complications brought by the growing user preference formulation possibilities and the combination possibilities of services with non telephony applications. The new management methods, however, require more intelligence capabilities in user and network devices. It is the authors feeling that explanation and learning, required from UFIMA, imply using corresponding Artificial Intelligence techniques. Explanation, learning, pro-activity, autonomy, cooperation, delegation and negotiation, lead to implement the architecture model we have proposed in this article as a multi intelligent agent architecture.

The management model this article proposed is a general one because of the generality of the adopted architecture model. We presented the possible implementation options that should be decided, when implementing the solution, taking into consideration the available services, the user device capabilities and operator policies. In future work, we will describe an ongoing implementation of this model in a SIP framework.

References

1. Cameron, E.J. *et al*: A feature interaction benchmark for IN and beyond. Feature Interactions in Telecommunications Systems. IOS Press, Amsterdam (1994) 1–23
2. Amer, M. *et al*: Feature interactions resolution using fuzzy policies. Feature Interactions in Telecommunications Systems. IOS Press, Amsterdam (2000) 94–112
3. Keck, D.O., Kuehn, P.J.: The feature and service interaction problem in telecommunications systems: a survey. IEEE Transactions on Software Engineering, Vol. 24. New York (1998) 779–796
4. Chentouf, Z., Khoumsi, A., Cherkaoui, S.: Conceptual foundations of user preference modeling. International Conference on Network Control and Engineering (NET-CON’2003). Kluwer Academic Publishers, New York (2003)
5. Chentouf, Z., Cherkaoui, S., Khoumsi, A.: Implementing online Feature Interaction detection in SIP environment: early results. 10th International Conference on Telecommunications (ICT'2003), Tahiti (2003)
6. Rosenberg, J. *et al* : SIP: Session Initiation Protocol. RFC 3261. IETF (2002)
7. JAIN. http://java.sun.com/products/jain
8. 3GPP. http://www.3gpp.org
9. Lennox, J., Schulzrinne, H.: CPL: a language for user control of Internet telephony services. IETF Draft (2000)
10. Chentouf, Z., Khoumsi, A., Cherkaoui, S.: Détection hors-ligne d’interactions de services dans les réseaux hétérogènes de télécommunications. Colloque Francophone sur l’ingénierie des Protocoles (CFIP’2002). Hermes, Paris (2002)
11. Griffeth, N.D., Velthuijsen, H.: The negotiating agents approach to runtime interaction resolution. Feature Interactions in Telecommunications Systems. IOS Press, Amsterdam (1994) 217–235
12. Blom, J. *et al*: Automatic detection of feature interactions in temporal logic. Feature Interactions in Telecommunications Systems. IOS Press, Amsterdam (1995) 1–19

13. Gammelgaard, A., Kristensen, J.E.: Interaction detection, a logical approach. Feature Interactions in Telecommunications Systems. IOS Press, Amsterdam (1994) 178–196
14. Nakumara, M. *et al*: Petri-net based detection method for non-deterministic feature interactions and its experimental evaluation. Feature Interactions in Telecommunications Systems. IOS Press, Amsterdam (1997) 138–152
15. Gibson, P., Mery, D.: Telephone feature verification: translating SDL to TLA+. Eighth SDL Forum (SDL'1997), Evry, France (1997)
16. Amyot, D. *et al*: Feature description and Feature Interaction analysis with Use Case Maps and LOTOS. Feature Interactions in Telecommunications Systems. IOS Press, Amsterdam (2000)
17. Charnois, T.: A natural language processing approach for avoidance of feature interactions. Feature Interactions in Telecommunications Systems. IOS Press, Amsterdam (1997) 347–363
18. Cherkaoui, S., Khoumsi, A.: Mobile and static agents for service interactions resolution in telecommunication environments. 9th IEEE International Conference on Telecommunications (ICT'2002), Beijing (2002)
19. Khoumsi, A.: Detection and resolution of interactions between services of telephone networks. Feature Interactions in Telecommunications Systems. IOS Press, Amsterdam (1997) 78–92
20. Blom, J., Johnsson, B., Kempe, L.: Using temporal logic for modular specification of telephone services. Feature Interactions in Telecommunications Systems. IOS Press, Amsterdam (1994) 197–216
21. Tsang, S., Magill, E.H.: Behavior based run-time feature interaction detection and resolution approaches for intelligent networks. Feature Interactions in Telecommunications Systems. IOS Press, Amsterdam (1997) 254–270
22. Rizzo, M., Garyfalos, A.: Using SIP to negotiate over user requirements in personalized Internet Telephony services. SIP'2000, Paris (2000)
23. Kolberg, M., Magill, E.H.: Handling incompatibilties between services deployed on IP-based networks. IEEE Intelligent Networks (IN'2001). IEEE Press, Boston (2001)
24. Jennings, N., Sycara, N., Wooldridge, M.: A roadmap of agent research and development. International Journal of Autonomous Agents and Multi-Agent Systems, Vol. 1 (1998) 7–38

Minimum Interference Routing: The Precomputation Perspective

Gábor Rétvari

High Speed Networks Laboratory, QoS IT Laboratory
Department of Telecommunications and Telematics
Budapest University of Technology and Economics
H-1117, Magyar Tudósok körútja 2., Budapest, HUNGARY
retvari@ttt-atm.ttt.bme.hu

Abstract. This paper focuses on the selection of bandwidth-guaranteed channels for communication sessions that require it. The basic idea comes from Minimum Interference Routing: select a feasible path that puts the least possible restriction on the transmission capacity offered by the network for other communicating parties. This is achieved by circumventing some critical bottleneck links. The main contribution of the paper is a method to assess the degree of link criticality facilitating efficient route precomputation even in the case, when up to date resource availability information is not immediately available.

Keywords: QoS routing, Traffic engineering, QoS management

1 Introduction

The deliberate, effective and adaptive selection of dynamic bandwidth-guaranteed paths in modern data networks has gained substantial research interest recently. Such assured-quality communication channels are essential for the introduction of various upcoming service classes. Obviously, any service that requires the transmission of loss and/or delay sensitive data (e.g., audio, video, etc.) over a packet switched public network infrastructure obligates the assurance of transmission quality guarantees on some prioritized traffic. This paper deals with the problem of finding a data path in a network for a traffic instance, which is both *feasible* and *efficient*. A path is feasible if it provides enough dedicated resources to satisfy pre-declared bandwidth demands. We say that a path is efficient, if it manifests some efficiency criteria of the network operator. For example, an efficient path may use as few resources as possible, or it would be such that the overall network resource utilization is maximized.

In the broader context the task of selecting feasible and efficient bandwidth guaranteed paths turns out to be the fundamental problem faced by modern QoS routing. In view of the serious scalability concerns raised by the Integrated Services approach, where bandwidth-guaranteed routes must be picked one by one for individual micro-flows, today's prevailing QoS management architecture, Differentiated Services, focuses on the assurance of aggregate QoS to a whole bunch of micro-flows. In accordance with the wider scope of routing decisions, the emerging *Traffic Engineering* discipline puts more emphasis on the criterion of network-global efficiency. In any case, the fundamental problem is to compute bandwidth driven paths for route requests arriving one at a time. Connections are fully characterized by their respective bandwidth demand. The set of potential traffic origin-sink pairs (sessions) is assumed to be known in advance, though, we do not presume any knowledge on the volume of traffic requested for a particular source-destination pair.

Nowadays, Internet routing is based on the Shortest-Path-First (SPF) discipline: some administrative costs are associated with network links and the path with the least aggregate cost is selected (if all costs equal to 1, we get to minimum-hop routing). The SPF algorithm proved to be easily implementable and deployable, thus it gained extreme popularity in the Internet community throughout the years. However, paths selected by the SPF algorithm are usually neither feasible nor efficient, thus SPF is deemed to be a major origin of congestion and unreliability in current Internet.

Several proposals exist to eliminate the shortcomings of the SPF algorithm. It is relatively easy to ensure path feasibility. The approach that protocol designers usually take is to augment a link state routing protocol to distribute resource availability information and use this dynamic data set to produce feasible paths. The widest-shortest-path (WSP) routing algorithm selects the path from the set of shortest paths, which offers the largest bottleneck bandwidth. The QoSPF routing protocol is a good example of the practical usefulness of the WSP algorithm [1]. The advantage of the WSP algorithm (and shortest-widest-path) is the ability to find a feasible path as long as such a path exists in the network. On the other hand, the signaling bandwidth consumed by frequent link state updates may substantially enlarge the protocol overhead. Therefore, it is a widely accepted phenomenon in QoS routing to limit the frequency of link state updates by means of some link state update triggering policy and precompute QoS routes no sooner than up to date routing information becomes available [2], [3]. As of the WSP algorithm, the selection rule that decides, which particular path from the set of feasible paths is to be picked is rather simplified. Hence, the WSP algorithm often causes various sorts of *interference* phenomena, such as alternate path blocking (an extensively utilized alternate path of a session starves another one). Minimum Interference Routing is the upcoming routing technology that aims to vigorously eliminate the disadvantages of traditional QoS routing protocols [4].

The basic idea of minimum interference routing is to select paths as to ensure that the chosen path blocks future requests to the least possible extent. This is achieved by maximizing some objective function, which describes the transmission capacity offered by the network for every single session. The full-fledged minimum interference routing problem is known to be *NP hard* [4]. However, there exists an algorithm, the so called Minimum Interference Routing Algorithm (MIRA), which is aimed to give a good approximate solution to the original problem. The key point of MIRA is the notion of *critical links.* A critical link fulfills the criterion that the admission of some traffic onto the link causes the reduction of the available capacity of one or more sessions. Hence, a critical link is subject to interference. On the other hand, if a link is not critical, then further traffic can be placed to it without adversely influencing any sessions. The rationale behind minimum interference routing lies in the ability to distinguish links based on a well-established criticality criterion and circumvent strongly critical links in the course the path selection in an attempt to minimize interference. Several recent research results have pointed out the potential of MIRA to improve network utilization while efficiently preserving resources for future requests in the same time. MIRA and some of its derivatives can be found in [4], [5]. [6] and [7].

One inherent limitation of MIRA comes from the fact that in an operational network environment MIRA's original intention to rely on the on-demand path selection model implies several drawbacks. On-demand path selection may result unbearable initial communication setup delay and the implied computational stress on routing hardware may be intolerable. In addition, the underlying routing protocol engine introduces significant inaccuracy in the routing state, because it may not re-synchronize resource availability information between the arrival of subsequent routing requests. This inaccuracy invalidates the need to calculate a unique route for each and every route request. In [4] the authors study MIRA performance in the case, when link criticality is only calculated once for every n connection request to limit the extent of calculations needed to perform in on-demand fashion. They conclude that MIRA does not suffer significant performance degradation due to *criticality precomputation*. Nevertheless, we shall show that without some clever modifications MIRA performance falls well under that of the WSP algorithm in the presence of precomputation.

This paper generalizes the notion of MIRA in a new way. The key point of MIRA is the decision, whether or not a particular network link is critical. We shall show that by the cost of some additional computational complexity introduced it is even possible to assess the extent of link criticality and use that particular information for routing purposes. Our new approach immediately gives a good algorithm to solve the problem of *Δ-criticality* left open in [4]. We shall present a new routing algorithm: the least-critical-path-first (LCPF) routing algorithm, and by means of extensive simulations we show that the proposed algorithm performs route precomputation much more efficiently, than MIRA or WSP.

The rest of this paper is structured as follows. Section 2 describes the Minimum Interference Routing Algorithm. In Section 3 we show, how to obtain a good quantity on link criticality and in Section 4 we define the LCPF algorithm that makes heavy use of this quantity. Simulation studies are presented in Section 5 and finally, Section 6 concludes our work.

2 The MIRA Algorithm

Experience suggests that in order to achieve efficient routing it is not enough to simply pick a path from the set of feasible paths. One has to find a sufficient policy on how to deliberately select a feasible path that manifests some network-global Traffic Engineering goal and define a good algorithm that implements the policy. The rationale behind MIRA comes from the recognition that if a network link acts as bottleneck for a communication session, then admitting traffic of another session to that critical link will cause interference. The more traffic flows through critical links, the more interference it will cause leading to inefficient routing in the long term. Thus, it is a plausible network-wide Traffic Engineering goal to minimize the interference along the selected path.

In order to better capture the notion of interference one has to invoke the elaborated toolset of network flow theory [8] [9]. Let $G(V, E, R)$ be a digraph. Let V be the set of nodes, E the set of edges and R the set of edge capacities.[1] We assume that G is such that there is only one (u, v) edge connecting any pair of nodes $u, v \in V$ (if there are more parallel edges, these can always be aggregated into one edge). We examine flow problems for source-destination pairs (s, d), all of which are members of a known set P. Such (s, d) pairs are called *sessions* for short. Let f be a *maximum flow* in G for some session (s, d). Then, $F_G(s, d)$ notes the value of the maximum flow $|f|$, $f(u, v)$ notes the flow traversing a particular edge $(u, v) \in E$, G_f notes the flow residual graph induced by f and C_{sd} is a set of edges, which belong to one or more *minimum cuts*.

A useful feature of maxflow theory is that the maxflow of a session defines an upper limit on the available transmission capacity offered to that session. By linear programming duality, associated with each maxflow there is a minimum cut. Edges belonging to the union set of the minimum cuts (C_{sd}) share the property that decreasing the capacity of the edge reduces the maxflow. Hence, we shall say that an (u, v) edge is *critical* for a session (s, d), if the edge is included in the minimum cut set for the session, i.e., $(u, v) \in C_{sd}$. Recall that any edge in C_{sd} is subject to interference. Therefore, for every link MIRA assigns an additive link weight, which is proportional to the number of sessions the link is critical for, and computes the minimum cost path to minimize overall interference.

Hence, the path selection for a traffic instance in session $(a, b) \in P$ of demand D involves the following basic steps in MIRA:

1. **Critical link identification:** compute the maxflow and the critical link set C_{sd} for each $(s, d) \in P \setminus (a, b)$. This yields cost contribution factors κ (κ describes the contribution of session (s, d) to the cost of link (i, j)) in the form:

$$\kappa_{sd}^{(i,j)} = \frac{\partial MaxFlow(s, d)}{\partial R(i, j)} = \begin{cases} 1 \text{ if } (i, j) \in C_{sd} \\ 0 \text{ otherwise} \end{cases} \tag{1}$$

2. **Cost assignment:** for each $(i, j) \in E$ compute the link cost:

$$w(i, j) = \sum_{(s,d):(i,j)\in C_{sd}} \alpha_{sd} = \sum_{(s,d)\in P\setminus(a,b)} \alpha_{sd}\kappa_{sd}^{(i,j)} \ , \tag{2}$$

[1] For the sake of simplicity we assume that all edge capacities are integral (either integer valued or can be expressed as an integer multiple of some bandwidth quantum). The actual implementation of the LCPF algorithm does not rely on this assumption.

where α_{sd} represents the relative importance of the session (s, d). In order to force path feasibility, a link with inappropriate resources ($R < D$) is either omitted at the SPF computation, or its cost is set to infinity.
3. **Path selection:** compute the shortest weighted path over the cost set defined by w.

The computational complexity of MIRA is dominated by the $p - 1$ maxflow computations needed for critical link identification. Without Step 1, MIRA boils down to a simple SPF calculation, which raises the idea of criticality precomputation: CPU intensive critical link identification is performed offline, once for every n route request. Setting n means to trade-off between routing accuracy and implied computational complexity.

In a situation, where one decision on link criticality may impact the routing of several subsequent requests, it would be of extreme usefulness to somehow detect if there is a chance that a link turns to critical in the near future. In this case, the routing algorithm could prevent routing further traffic onto that link to avoid interference until criticality precomputation is performed again. This brings us to the definition of Δ-criticality: a Δ-critical link for a session is a link such that if the capacity of the link is decreased by Δ, the maxflow value for the session decreases (by any value). [4] concludes that an algorithm that effectively detects Δ-criticality has the potential to increase routing performance, though, the authors do not supply an exact algorithm, neither they give an advice how to set Δ for maximum sensitivity.

3 The Criticality Threshold

In the previous section we concluded that the identification of critical links is the most important and CPU intensive process of MIRA. In this step, MIRA solves the following decision question: given an edge and a session, is it the case that the edge is included in one or more minimum cuts for the session? Throughout this paper, we use simplified form of the link criticality conditions proposed in [4]. The purpose of the alternative formulation is not really a practical one, but rather to provide further insight into the nature of link criticality.

Theorem 1 (Formulation of link criticality). *For an edge (i, j) and a session (s, d): (i, j) is critical (i.e., $(i, j) \in C_{sd}$) if all the following conditions hold:*

i) $f(i, j) > 0$ *and*
ii) $F_{G_f}(i, j) = 0$

Herein we do not present the proof of the theorem. In words, Theorem 1 states that a link (i, j) is critical for a session, if it carries nonzero flow from the session's maxflow and the endpoints of the link happen to reside in different cuts of the graph (i.e., there is no path from i to j in the residual graph G_f). The main contribution of this paper is the observation that there exists a well-defined threshold on the capacity of any link, the so called *criticality threshold*, such that if the capacity of the link falls beyond this threshold then the link turns to critical.

Definition 1. *Given a non-negative number K, a session (s,d) and an edge (i,j), K is criticality threshold, if:*

$$\textit{Criticality: } 0 < R(i,j) \leq K \Rightarrow (i,j) \in C_{sd} \ ,$$

$$\textit{Non-criticality: } R(i,j) > K \Rightarrow (i,j) \notin C_{sd} \ .$$

The significance of the criticality threshold is twofold. First, given a bandwidth demand of size Δ the criticality threshold tells whether admitting the demand to any (i,j) edge would render the edge critical. This is simply done by checking $R(i,j) - \Delta \leq K$. Observe that this gives an exact algorithm to detect Δ-criticality. Furthermore, the criticality threshold supplies a unique way to assess the criticality (or non-criticality) of any link. For example, a good measure would be $K/R(i,j)$ or $K - R(i,j)$.

The approach that we take to find the criticality threshold is to investigate the maxflow problem while changing the capacity of a particular link. For a graph $G(V,E,R)$, we define the *unconstrained graph w.r.t. link* (i,j) as a graph $G^U(V,E,R^U)$, in which we choose the capacity of (i,j) to infinity. The maxflow in G^U is called *unconstrained maxflow.* On the contrary, in the *constrained graph w.r.t. link* (i,j) we set $R^C(i,j) = 0$. The maxflow in G^C is called *constrained maxflow.* One can characterize the constrained, the original and the unconstrained maxflows as $F_{G^C} \leq F_G \leq F_{G^U}$.

From Theorem 1 we know that a link, which carries nonzero flow is critical if both its residual capacity is zero and all parallel augmenting paths are saturated in the flow residual graph. This sheds light on the rationale behind the definition of the unconstrained graph: instantiating some maxflow f in the unconstrained graph for which $f(i,j) > 0$ and setting $K = f(i,j)$ will generate a candidate for the criticality threshold. Such unconstrained maxflows constitute the so called feasible maxflow set $\mathcal{F}_G^{sd}$ w.r.t. link (i,j):

$$\mathcal{F}_G^{sd} = \{f : f \text{ is a } (s,d) \text{ maxflow in } G^U\} \ . \tag{3}$$

Recall that there are no $i \to j$ augmenting paths in the flow residual graph when (i,j) is critical. In other words, from a critical link no flow can be relocated to other flow paths, which would decrease the flow on (i,j). This implies that the flow generating criticality sends a *minimum* flow onto (i,j) in some sense. This idea is captured in the following definition.

Definition 2 (Committed flow). *The committed flow for a session (s,d) and a link (i,j) is defined as:*

$$\psi^{sd}(i,j) = \inf_{f \in \mathcal{F}_G^{sd}} f(i,j) \ . \tag{4}$$

The notion of committed flow insists that there are various ways to accommodate the maxflow in a graph, however, there is a certain amount of flow *committed* to a particular link in all cases. The following theorem proves that the choice $K = \psi^{sd}(i,j)$ indeed yields the criticality threshold:

Theorem 2. *For an edge (i, j) and a session (s, d) in graph G:*

$$0 < R(i, j) \leq \psi^{sd}(i, j) \Rightarrow (i, j) \in C_{sd} \ . \tag{5}$$

And conversely:

$$R(i, j) > \psi^{sd}(i, j) \Rightarrow (i, j) \notin C_{sd} \ . \tag{6}$$

Proof. First we prove that in case of $R(i, j) = \psi^{sd}(i, j) > 0$, all conditions in Theorem 1 hold, therefore (i, j) is critical. Let $G'(V, E, R')$ be a graph, such that $R'(i, j) = \psi^{sd}(i, j)$. Then, $F_{G^U}(s, d) = F_{G'}(s, d)$. Let f be a maxflow in G', such that $f(i, j) = \psi^{sd}(i, j)$, i.e., the flow on edge (i, j) is the least possible.

i) $f(i, j) = \psi^{sd}(i, j) > 0$
ii) We need to see that (a) $R_f(i, j) = 0$ and (b) there are no $i \to j$ feasible paths in the flow residual graph. (a) holds, because $R_f(i, j) = R(i, j) - f(i, j) = \psi^{sd}(i, j) - \psi^{sd}(i, j) = 0$. (b) is also true, because if a $i \to j$ alternative path happens to exist, then some flow can be shifted from (i, j) to the alternative path. This yields a flow f', for which $f'(i, j) < f(i, j)$. Observe that f' is a maxflow for the unconstrained graph too, therefore $f'(i, j) < f(i, j) = \psi^{sd}(i, j) = \inf_{f \in \mathcal{F}_G^{sd}} f(i, j)$ is a contradiction.

It is also true that reducing the capacity of a critical edge both reduces the maxflow and leaves the edge as critical. This proves (5). On the other hand, for any $G'(V, E, R') : R(i, j) > \psi^{sd}(i, j)$, the (i, j) link is not critical. This is because any flow that is a maxflow in the unconstrained graph G^U is also a maxflow in G' and there exists a maxflow f such that $f(i, j) = \psi^{sd}(i, j)$, which leaves non-zero residual capacity on (i, j). This proves (6). □

A naive way to find the criticality threshold would be to search through all feasible maxflows and find the one that commits the minimum flow to the selected edge. Though, this method would not be too effective. The following fundamental flow theory result gives an easy way to compute the criticality threshold.

Theorem 3. *For a graph G, an edge (i, j) and a session (s, d), the committed flow is the difference of the unconstrained and the constrained maxflow for (i, j):*

$$\psi^{sd}(i, j) = F_{G^U}(s, d) - F_{G^C}(s, d) \ . \tag{7}$$

Proof. We need to show that $F_{G^U}(s, d) - F_{G^C}(s, d) = \inf_{f \in \mathcal{F}_G} f(i, j)$. It is easy to show that there exists a flow f in G^U, such that $f(i, j) = F_{G^U}(s, d) - F_{G^C}(s, d)$. Now, we need to show that this is indeed the infimum. Suppose that there exists a maxflow f' in G^U, such that $f'(i, j) < f(i, j)$. Invoke flow decomposition to eliminate all flow from the graph that traverses the (i, j) edge, furthermore, remove the (i, j) edge from the graph. Observe that the resultant graph is identical to the constrained graph with respect to (i, j), and such, the resultant f'' flow is a maxflow in G^C. Hence, $F_{G^C}(s, d) = |f''| = F_{G^U}(s, d) - f'(i, j) > F_{G^U}(s, d) - f(i, j) = F_{G^C}(s, d)$, which is a contradiction. This proves that $F_{G^U}(s, d) - F_{G^C}(s, d)$ is indeed the infimum. □

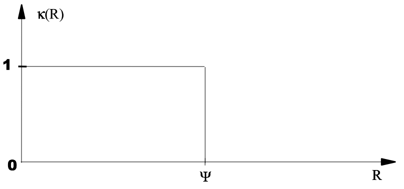


Fig. 1. Cost contribution profile in standard MIRA

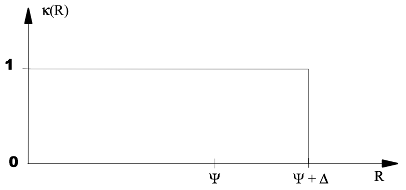


Fig. 2. Cost contribution profile for Δ-criticality

4 The Least-Critical-Path-First Routing Algorithm

So far we have seen that the cost of a link is composed of $\alpha_{sd}\kappa_{sd}^{(i,j)}$ per-session contributions. The notion of criticality threshold provides a comprehensible way to represent the cost contribution as the function of the link capacity. Such a $\kappa(R)$ graph is called cost contribution profile. The cost contribution profile for standard MIRA is depicted in Figure 1. The cost contribution profile used to detect Δ-criticality is shown in Figure 2. Note that MIRA does not compute the criticality threshold, instead, MIRA solves the decision question using an alternative of Theorem 1.

Nevertheless, the notion of criticality threshold elaborated in the previous section provides an impressing mean to not only solve the decision question, but to even calculate a quantity on the actual extent of link criticality. The basic idea of the least-critical-path-first (LCPF) routing algorithm is to determine the criticality threshold and use that sophisticated piece of information to compute bandwidth-guaranteed dynamic paths. In order to manifest the criticality of link (i, j) for session (s, d), we use the following intuitive cost contribution profile:

$$\kappa(R) = \frac{\psi_{sd}(i,j) + D}{R(i,j)} \ , \tag{8}$$

where the requested bandwidth is D units. Recall that the cost contribution profile should implement both path feasibility and efficiency. We chose the unit contribution at the point, where sending D flow to the link would practically drive it right at the edge of criticality. This choice is called to represent effectiveness: any link with contribution less than 1 is able to tolerate a request of D size without the risk of turning to critical. On the other hand, as the link capacity converges to zero, the contribution increases dramatically to force path feasibility. Without the restriction of generality, we can assume that $\sum_{(s,d)\in P\setminus(a,b)} \alpha_{sd} = 1$, thus, for the link cost we get:

$$w(i,j) = \sum_{(s,d)\in P\setminus(a,b)} \alpha_{sd} \frac{\psi_{sd}(i,j) + D}{R(i,j)} = \frac{\Psi(i,j) + D}{R(i,j)} \ , \tag{9}$$

where $\Psi(i, j)$ is the so called *committed load.* In its simplest form the committed load is the average of the committed flows over the set of sessions ($\alpha_{sd} = \frac{1}{p-1}$). Otherwise, it represents the relative importance of sessions as well.

Hence, (9) yields the Least-Critical-Path-First-Routing Algorithm:

Least-Critical-Path-First Routing Algorithm (LCPF):

INPUT: A graph $G(V, E, R)$ with the set of link capacities R, a set of potential sessions P, an ingress node a and an egress node b between which a flow of D units have to be routed.

OUTPUT: The least critical feasible path between a and b.

ALGORITHM:

```
1. For all $(s,d) \in P$ and for all $(i,j) \in E$ compute the constrained maxflow $F_{G^C}(s,d)$ and the unconstrained maxflow $F_{G^U}(s,d)$. This yields the criticality threshold in the form $\psi_{sd}(i,j) = F_{G^U}(s,d) - F_{G^C}(s,d)$.
2. For all $(i,j) \in E$ compute the committed load $\Psi(i,j)$ and execute the cost assignment $w(i,j) = \frac{\Psi(i,j)+D}{R(i,j)}$
3. Compute the shortest weighted path over the link weight set defined by $w$ and route the demand of $D$ units along that path
```

Remarks:

Flow priorization: the algorithm facilitates for the network operator to represent his or her precedence of sessions by setting α factors accordingly. If the α factor is set to a high value for some important session, then the LCPF algorithm will do its best to circumvent the critical links of that session.

Complexity: according to [4] the computational complexity of MIRA is dominated by the set of $p-1$ maxflow computations. LCPF involves $2P|E|$ maxflow computations (2 for every session and every link), thus, its worst-case complexity is $|E|$ times as high as that of MIRA. For the first glance, the huge number of necessary maxflow computations is disappointing. However, it must be mentioned that in its current form, LCPF lends itself to various sorts of optimizations, which have not been exploited herein (for example, $\psi_{sd}(i,j) = 0$ for any link, such that $j = s$, or $i = d$, etc.). Therefore, we developed an optimized algorithm that provides an average running time comparable to that of standard MIRA, though, the detailed discussion of this algorithm is beyond the scope of this document. For now, it suffices to claim that the worst case complexity of LCPF is no worse than $|E|$ times of MIRA's complexity

Criticality precomputation: criticality precomputation implies that the CPU intensive criticality calculations are performed only after every nth connection request. As we shall see in the next section, simulation results obtained on large graph sets demonstrate that owing to the sophisticated "criticality-detection" LCPF significantly outperforms MIRA in the presence of criticality precomputation. Hence, reducing LCPF to a simplistic shortest path computation in the average case and doing criticality computations offline makes the LCPF algorithm an overly effective and lightweight solution.

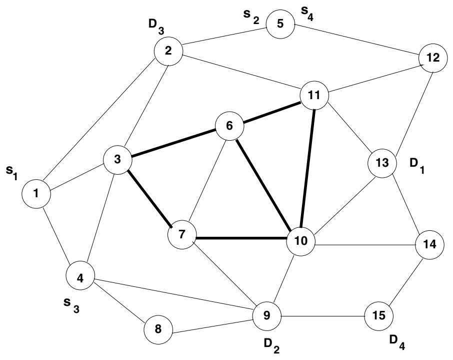


Fig. 3. The KL graph

Feasibility: one may argue that while MIRA returns a feasible path as long as such a path exists in the graph, LCPF does not. MIRA omits all links with inappropriate resources ($R < D$) before the path selection takes place. On the contrary, LCPF represents path feasibility in the contribution profile – LCPF orders high contribution to infeasible links, which probably yields a path consisting of feasible edges exclusively. We can say that LCPF applies "soft feasibility" on path selection. In the presence of criticality precomputation the accuracy of link state information is doubtful, so MIRA might blindly omit feasible links based on outdated link state information. Therefore soft feasibility transforms into better call acceptance in such cases.

We say that LCPF is a generalization of MIRA, as MIRA boils down to a special case of LCPF when used with the cost contribution profile depicted in Figure 1.

5 Simulation Results

Several research results [3], [10] demonstrate that in a realistic QoS routing environment some sort of precomputation is inevitable due to the inherent nature of the underlying protocol architecture. Therefore, in the course of the simulation studies presented in this section our main goal was to compare the routing performance of the LCPF algorithm to that of MIRA, WSP and SPF in the presence of route precomputation (for details on precomputed WSP, please refer to the Appendices of [1]).

In the scientific literature related to MIRA the so called KL graph (Figure 3) has slowly become the de facto simulation topology [7], [5], [4]. In the illustrative example, the capacity of the light links is 12K units and the dark links is 48K units and each link is bidirectional. In all the simulation experiments described in this paper, requests are uniformly distributed over all sessions and arrive randomly at the same average rate.

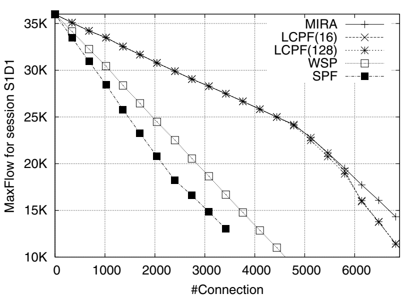


Fig. 4. MaxFlow for session S1-D1 after every connection is routed

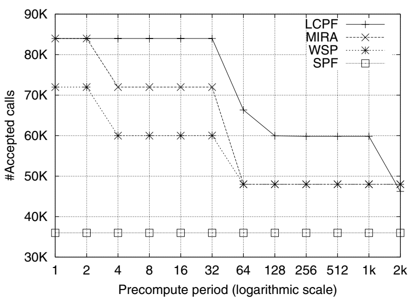


Fig. 5. Successfully routed connections as the function of the precomputation period

Preserving the available transmission capacity offered for communicating sessions is the key objective towards minimum interference routing. Figure 4 shows that LCPF is indeed able to bring this policy into effect even in the presence of criticality precomputation. In the figure, the maxflow for session S1-D1 is depicted after setting up long-lived connections one by one in the KL graph. All requests are of equal size (10 units). Precomputation was not applied to MIRA and WSP, however, LCPF is run at precomputation period $n = 16$ and $n = 128$. The diagram implies that LCPF is, almost irrespective of the precomputation period, able to preserve the transmission potential of the S1-D1 session, thus it manifests minimum interference almost as effectively, as MIRA in this case.

Now we show that in the presence of criticality precomputation the deliberate criticality detection of LCPF transforms into better call acceptance. First, we experienced how many unit-sized, long-lived connections a particular algorithm is able to accommodate in the KL graph one after another until all sessions are blocked. Figure 5 depicts the result of this experiment as the function of the precomputation period. Both MIRA and LCPF are able to fill up the network to the theoretical maximum (84000 units of traffic) when precomputation is not applied, however, as the precomputation period increases MIRA (and WSP) suffers significant performance degradation. Nonetheless, regardless of the degree of applied precomputation, each algorithm outperforms SPF (which is, by nature, not sensitive to the precomputation period). The graph also shows that both LCPF and MIRA are superior to WSP in the case of on-demand routing ($n = 1$), however, LCPF is able to retain the precedence as long as the precomputation period remains reasonable (take note of the logarithmic scale on the x axis).

Similar behavior can be deduced from Figure 6, which shows the average call blocking ratio (CBR) as the function of the Poisson request arrival intensity at a precomputation period of $n = 20$. The request size was uniformly distributed between 10 and 30 units. MIRA performs reasonably better than WSP for $n = 1$ [4]. However, when $n = 20$ MIRA produces non-zero call blocking even at a request arrival intensity, where LCPF or WSP does not. LCPF performs better than any other algorithms at all request arrival rates.

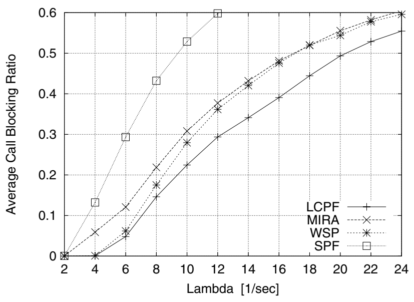


Fig. 6. Average CBR as the function of request arrival intensity

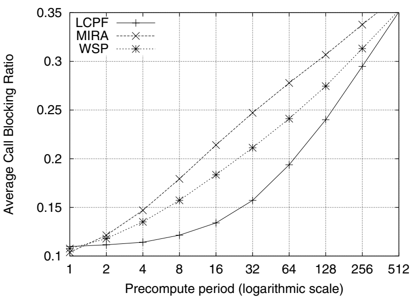


Fig. 7. Average CBR in 30 pseudo-random graphs as the function of n

One may argue that the performance benefits implied by LCPF (and MIRA) may be limited to the scope of the KL graph, and in other scenarios minimum interference routing may not prove so prosperous. Therefore, we carried out simulations on a set of 150 random graphs, which were obtained by relocating links in the KL graph. In all random graphs it was assured that there are at least two link-disjoint paths between any particular session source and destination. A random number of sessions ($3 \leq p \leq 7$) was located randomly in the graphs. Per-session Poisson request generation intensity (λ_i), exponential holding time (μ_i) and mean request size (B_i) is set as to assure that the average load is kept at a constant rate ($\sum_{i=1}^{p} B_i \frac{\lambda_i}{\mu_i} = const$). Figure 7 depicts the average call blocking ratio (precomputation period is again represented in a logarithmic scale). First, in the absence of criticality precomputation, MIRA and LCPF produces similar outstanding routing performance. In fact, MIRA performs slightly better, which seems to be a consequence of the "soft feasibility" approach of LCPF. However, as the interval between subsequent link state updates increases (as it would be the case in a realistic traffic engineering environment), and hence the effects of precomputation efficiency become dominant, MIRA loses precedence over WSP and the revenue of sophisticated criticality detection and soft feasibility of LCPF emerges (SPF, irrespective of n produces 0.48 average call blocking ratio). For $n > 2$, LCPF outperforms every other algorithms and preserves the same good efficiency at modest precomputation periods. Only at the extreme case of $n = 1024$, LCPF performance falls into the range of WSP and MIRA.

6 Conclusions

In this paper we investigated the impacts of precomputation on several QoS routing algorithms. By means of extensive simulation studies we have shown that the overly effective MIRA algorithm is not sufficient for route precomputation, though, precomputation is an inevitable and straight consequence of the underlying protocol architecture. Therefore, we elaborated the notion of committed flow and criticality threshold and showed that these are fundamental properties in network flow theory. We exploited the potential of the criticality threshold to implement sophisticated proactive criticality detection in order to design the least-critical-path-first routing algorithm. We have shown that the LCPF algorithm manifests the minimum interference policy more deliberately than MIRA even in the case of large precomputation periods. Nevertheless, we concluded that the new routing algorithm is far more expensive in terms of offline computational requirements than MIRA, though, further work is necessary in this field.

Acknowledgement. This work has been done within the research cooperation framework between HSNLab at DTT-BUTE and Ericsson Research. The author is grateful to Miklós Boda (Ericsson), Tamás Henk and Tibor Cinkler (HSNLab) for their support.

References

1. R. Guerin, A. Orda, and D. Williams, "QoS routing mechanisms and OSPF extensions." IETF RFC 2676, 1999.
2. A. Orda and A. Sprintson, "QoS routing: the precomputation perspective," in *INFOCOM (1)*, pp. 128–136, 2000.
3. G. Apostolopoulos, R. Guerin, and S. Kamat, "Implementation and performance measurements of QoS routing extensions to OSPF," in *INFOCOM (2)*, pp. 680–688, 1999.
4. K. Kar, M. Kodialam, and T. Lakshman, "Minimum interference routing of bandwidth guaranteed tunnels with MPLS traffic engineering applications," *IEEE Journal on Selected Areas in Communications*, vol. 18, December 2000.
5. K. Kar, M. Kodialam, and T. V. Lakshman, "MPLS traffic engineering using enhanced minimum interference routing: An approach based on lexicographic max-flow." Proceedings of Eighth International Workshop on Quality of Service (IWQoS), Pittsburgh, USA, June 2000.
6. I. Iliadis and D. Bauer, "A new class of online minimum-interference routing algorithms," in *Networking 2002, Proceedings of Second the International IFIP-TC6 Networking Conference*, p. 959 ff., May 19-24 2002.
7. S. Suri, M. Waldvogel, D. Bauer, and P. R. Warkhede, "Profile-based routing and traffic engineering," *Computer Communications*, vol. 26, pp. 351–365, 2003.
8. R. K. A. snd T. L. Magnanti and J. B. Orlin, *Network Flows: Theory, Algorithms, and Applications.* Prentice-Hall, Englewood Cliffs, NJ, 1993.
9. M. S. Bazaraa, J. J. Jarvis, and H. D. Sherali, *Linear Programming and Network Flows.* John Wiley & Sons, January 1990.
10. G. Apostolopoulos, R. Guerin, S. Kamat, and S. K. Tripathi, "Quality of service based routing: A performance perspective," in *SIGCOMM*, pp. 17–28, 1998.

A Virtual Signalling Protocol for Transparently Embedding Advanced Traffic Control and Resource Management Functionality in ATM Core Networks

K. Katzourakis[1, 2], G. Kormentzas[1, 3], K. Kontovasilis[1], and C. Efstathiou[2]

[1] National Center for Scientific Research "DEMOKRITOS",
Institute for Informatics & Telecommunications,
GR-15310 AG. PARASKEVI, POB 60228, GREECE
{gkorm; kkont}@iit.demokritos.gr
[2] Technological Education Institute of Athens,
Dept. of Informatics,
GR-12210, Egaleo, GREECE
ckatzoura@cs.teiath.gr; cefsta@teiath.gr
[3] University of the Aegean,
Dept. of Information and Communication Systems Engineering,
GR-83200, Karlovassi, GREECE
gkorm@aegean.gr

Abstract. Following recent proposals for open and programmable network infrastructures, the paper discusses an out of band Virtual Signalling Protocol (VSP), which bypasses the conventional control information channels and supports externally defined traffic control and resource management functionality. The examination of VSP messages that run through the L and U interfaces (defined according to the IEEE P1520 reference model) constitutes one of the major objectives of the paper. The functionality of VSP is tested by means of an implemented prototype platform, which enables the portable deployment of advanced traffic control and resource management algorithms in heterogeneous ATM networks.

1 Introduction

Traffic control and resource management are important mechanisms for any Quality of Service (QoS) providing network environment and this importance will continue to grow towards broadband networking environments of the future, as more complex services and stringent and more diverse QoS demands will emerge. In this setting, signalling systems supporting advanced traffic control and resource management functionality have to play a key role towards the assurance of requested QoS and the simultaneous maximization of network availability and minimization of network operational costs.

The native ATM signalling protocols (e.g., ATM Forum UNI 3.1 [1], ATM Forum UNI 4.1 [2], ITU-T Q.2931 [3], ATM Forum PNNI 1.1 [4], etc.) support only internally defined traffic control and resource management functionality through inband signalling channels. This fact contradicts the current trend of open signalling, which enables external functionality through out-of-band generic/abstract control channels. The main idea of OpenSig (Open Signalling) [5,6] concerns the use of programmable interfaces, which provide open control and management access to various network devices (e.g., ATM switches, IP and MPLS routers, etc.). According to OpenSig, the control/management and data paths are strictly separated and the defined open programmable interfaces can be seen as channels for flexible and transparent deployment of new advanced control and management algorithms.

Based on the OpenSig concept, various related works appear in the literature. Among them, we can discriminate: An attempt for standardisation of open programmable interfaces for control and management of various types of networks (e.g., ATM, SS7, IP, etc.) by the IEEE P1520 project [7], a virtual out-of-band signalling protocol by the Xbind project [8], and the Tempest open programmable framework [9]. For the implementation of such out-of-band open signalling protocols, a variety of architectures (e.g., FIPA [10], OMG MASIF [11], etc.), communication technologies (e.g., CORBA [12], DCOM [13], Java [14], etc.) and agent platforms (e.g., Grasshopper [15], Voyager [16], etc.) can be used.

Major issues for the selection of the appropriate open signalling protocol are the availability and the simplicity of the employed programmable interfaces, the maturity of the architectural framework that is going to be followed and the performance overheads, which enforces in the underlying network and the required implementation time [5]. Satisfying in an adequate level these issues, the paper presents a Virtual Signalling Protocol (VSP), which bypasses the conventional control information channels and supports externally defined traffic control and resource management functionality. The presented VSP builds upon earlier work [17]; in introducing constructs allowing the interaction of traffic control with resource management, towards increasing the network's potential for acceptance of calls in a dynamic way.

Besides the signalling protocol itself, the paper outlines a generic framework for the employment of several IEEE P1520 constructs, such as L and U interfaces, as well as NGSL (Network Generic Services Layer), in a standardised and systematic way for the purposes of device independent traffic control and resource management. The functionality of the VSP is tested through an implemented Java-based prototype.

The organization of the rest of the paper is as follows: Section 2 gives the overall architectural framework, which incorporates the proposed virtual signalling protocol. Section 3 presents the protocol, while Section 4 outlines the prototype implementation and comments on the protocol's performance. Finally, Section 6 concludes the paper.

2 Software Switch Extensions Enabling Virtual Signalling

The incorporation of an open signalling protocol supporting externally defined traffic control and resource management functionality into a heterogeneous networking platform requires out-of-band control paths/channels, which can bypass the ones established by fixed standards signalling protocols. Addressing this issue, Figure 1 presents a generic architectural framework, which enables enhanced traffic control and resource management functionality to be communicated between the nodes of a heterogeneous network. A detailed description of the framework may be found in [17]. The key-idea is that the network nodes export appropriate well defined programmable interfaces to distributed software entities, thus providing a virtual environment for the deployment of traffic control and resource management functionality on top of the nodes. The deployed functionality is then exchanged between the software entities through an out-of-band signalling channel.

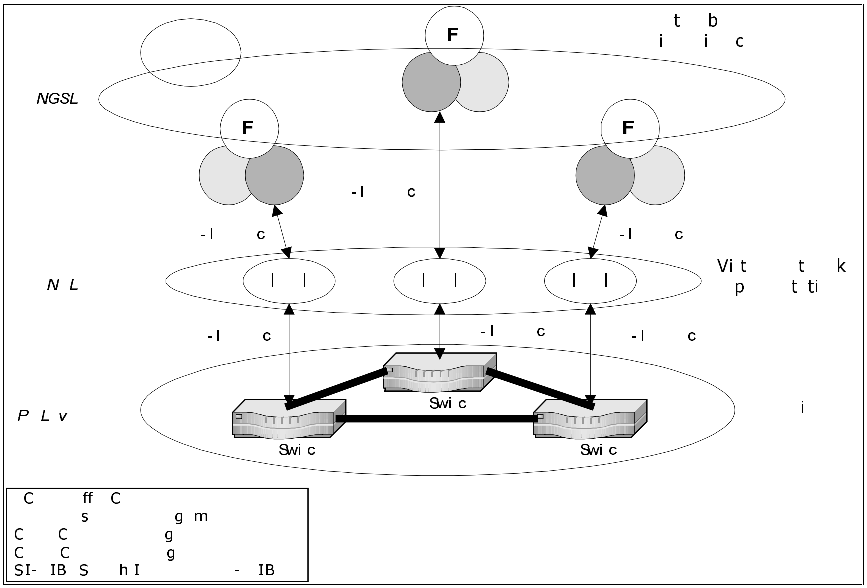


Fig. 1. Software switch extensions enabling virtual signalling

In terms of P1520, each node in Figure 1 exports a generic and switch-independent L interface, enabling the information flow between traffic control and resource management algorithmic components and the so-called Virtual Network Device Layers (VNDLs). A VNDL constitutes a virtual (software) representation of a node, containing all information relevant to the control and management algorithms operating on top of the node. In the presented architecture, the VNDL (i.e., the collection of appropriate managed/controlled objects) is built around a switch-independent MIB, called SI-MIB, which provides a portable virtual representation of the resources and traffic load conditions within the corresponding network node. For an in-depth description of the SI-MIB see [18]. The communication between SI-MIB and its corresponding underlying network node is based on a switch dependent CCM-interface.

The distributed software entities implementing the control and management algorithms (i.e., TCMs and RMMs) are uncoupled from equipment details by referring to (and manipulating) only switch-independent information contained in the SI-MIB. These entities belong to NGSL; their communication coordinated by CFMs through an out-of-band signalling channel. CFMs take instructions by a centralised (per domain-level) routing module, which is responsible for intra-domain routing in a heterogeneous ATM core network.

The following section presents an appropriate out-of-band Virtual Signalling Protocol (VSP) to support the control information exchange of just described architectural framework.

3 The Virtual Signalling System

As indicated in Figure 1, the exchange of control information between switches occurs at the virtual level out-of-band, i.e., outside the control paths/channels established between the inter-connected switches, bypassing ATM signalling. This section describes the exchange of VSP messages for handling a call request/call release.

The VSP's traffic control functionality concerns CAC, while the corresponding resource management functionality refers to bandwidth redistribution between Virtual Multiplexing Units (VMUs). A VMU corresponds to a group of Virtual Paths (VPs) sharing common resources (part of the output port's buffer space and link capacity) and serving a traffic mix that consists of connections with the same QoS requirements. Heterogeneous QoS may be supported by appropriately distributing traffic into different VMUs. In this context, the generic distributed software entities TCM and RMM of architectural framework of Figure 1 shift to CAC modules and Bandwidth Redistributor (BR) modules correspondingly.

3.1 Call Set-up

In the process of handling a new call request, one of the following two things may happen:

1. The CAC modules of all nodes across a path between the source and destination terminal accept the call request.
2. One of the involved CAC modules rejects the new call request. In that case, the currently checked route is generally rejected and an alternative (if it exists) is examined. However if the route under examination is the last possible one, instead of directly rejected, the offending CAC module triggers its corresponding BR to redistribute the bandwidth between VMUs to make room for placing the new call. If the redistribution is successful in all offending CAC modules, the call request is accepted, otherwise it is rejected.

(Note that for both cases, we consider whenever it is applicable that destination accepts the call request.)

For the first case (i.e., an accepted call request without bandwidth redistribution), the interactions between the end terminals (Source and Destination respectively) and the routing module, CAC modules, call forwarding modules and SI-MIBs that are placed on top of the switches are depicted in Figure 2. (The number associated with each message in Figure 2 indicates the message's relevant position in a chronological sequence.)

As shown in Figure 2, the source terminal sends a call request message (`call_req`) to the routing module. Parameters of `call_req` are the source and destination terminal identification strings (parameters `source_id` and `dest_id`

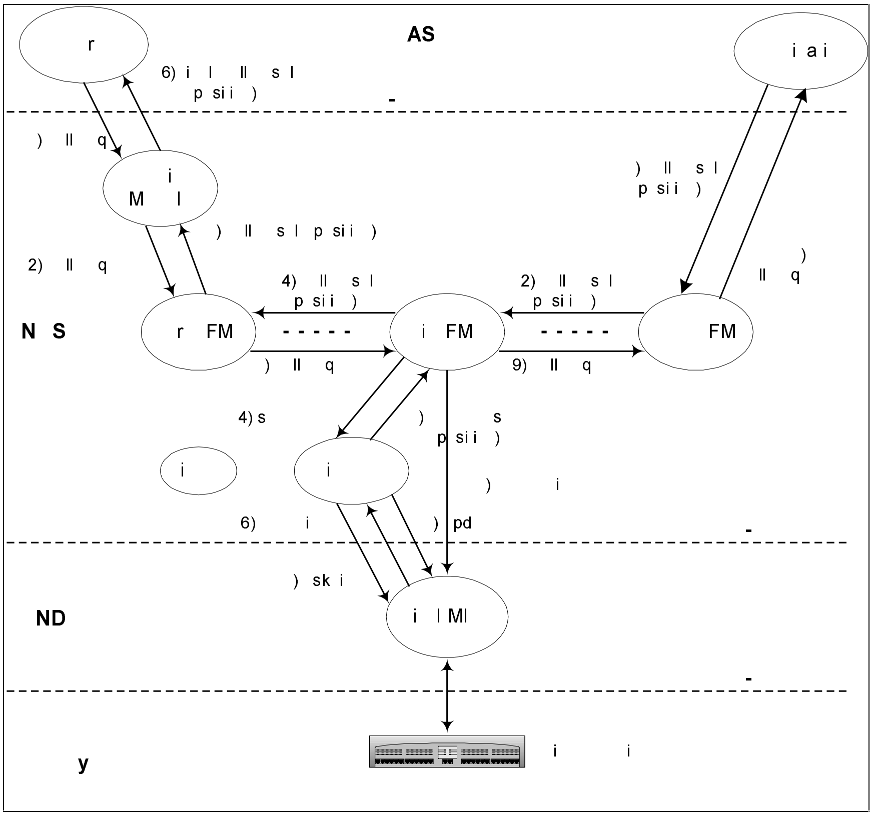


Fig. 2. Call acceptance without bandwidth redistribution

respectively), the requested level of service quality (parameter `QoS`), the expected traffic profile of the requested call (parameter `TrProf`) and a unique identification number (parameter `RefNum`), used to distinguish the call.

Receiving `call_req`, the routing module defines the possible routes that connect the source terminal to the destination one and selects to examine one of them. The routing vector of the selected route (object `RVector`) is wrapped together with the information of message `call_req` into a new message `call_req_cont`, which is forwarded to the call forwarding module of the first node participating in the chosen route (Source Call Forwarding Module - SrcCFM). The object `Rvector` contains the pairs of input port – input VPI (parameters `inputPortID` and `input-VPI` respectively) and output port – output VPI (parameters `outPortID` and `out-VPI respectively`) for every hop contained in the selected route.

Moving now in the i-node of the routing path between source and destination, i-CFM sends to its corresponding CAC module (i-CAC) a message `start_CAC` in order to commence the CAC process. The message `start_CAC` contains the parameters `outPortID`, `QoS`, `TrProf` and `RefNum`. i-CAC uses the first two of these parameters in a message ask_info in order to retrieve from the i-SI-MIB (i-SI-MIB), through a message `CAC_info`, appropriate information for the application of a CAC scheme. This information consists in available resources (parameter `res`) and the load conditions (parameter `BackgroundTraffic`) within the (specified by the parameters `outPortID` and `QoS`) Virtual Multiplexing Unit (VMU) of the source switch (SrcSwitch).

Using the information of the parameters `QoS`, `TrProf`, `res` and `BackgroundTraffic`, i-CAC performs a CAC scheme in order to accept or reject the new call request. Given that i-CAC accepts the call (see Figure 2), it proceeds to compute the new value `updBackgroundTraffic` of the parameter `BackgroundTraffic`. Then, it firstly issues a message `update_VMU` to i-SI-MIB in order to update the load conditions of the examined VMU (preventive bandwidth reservation) and secondly it informs i-CFM, through a message `CAC_answer`, for the success of the CAC process. `CAC_answer` includes a boolean parameter `answer`, which actually contains the positive CAC answer for the examined scenario.

Subsequently, i-CFM forwards the message `call_req_cont` to the next node along the route being examined and the aforementioned process is repeated on all nodes along the selected route until the destination switch (DestSwitch). After the receipt of `CAC_answer` by the destination CF module (DestCFM), the message `call_req_cont` is forwarded to the destination terminal. The destination terminal (according to the presented scenario) accepts the call request and returns a positive message `call_result` to DestCFM (the boolean parameter `result` of the message has the value `true`). The latter transfers this message to the previous call forwarding module, which passes it to its own previous call forwarding module, etc. Eventually, the message `call_result` reaches at SrcCFM and from there at the routing module, which creates the message `final_call_result` and passes it to the source terminal informing it about the call acceptance.

Besides passing a message `call_result` to the previous call forwarding module, each call forwarding module sends a message `connection` to its SI-MIB (with parameters `inPortID, inVPI, inVCI, outPortID, outVPI, outVCI`) directing the SI-MIB to create the appropriate VC cross-connections into the underlying ATM switch. At the final stage, the successful set-up of a new call is the result of the application of messages `connection` from all the involved to the new call ATM switches.

If i-CAC does not accept the call request and provided that the examined route is not the last one, it delivers a negative message `CAC_answer` (the parameter `answer` takes the value false) to i-CFM. Subsequently, i-CFM sends to CFM of the preceding node along the current route a negative message `call_result` (the boolean parameter `result` of the message has the value `false`). In addition, each CFM informs its corresponding SI-MIB to release the pre-reserved resources through a message `reverse_VMU`. Eventually, the negative message `call_result` reaches at SrcCFM and from there at the routing module. Subsequently, the routing module starts to check an alternative routing path, which connects the source and destination terminals.

The second examined case of call set-up concerns an accepted call request after bandwidth redistribution. This scenario (see Figure 3) starts at the point where the call request has been rejected across all the possible routes except the last one, which is under check.

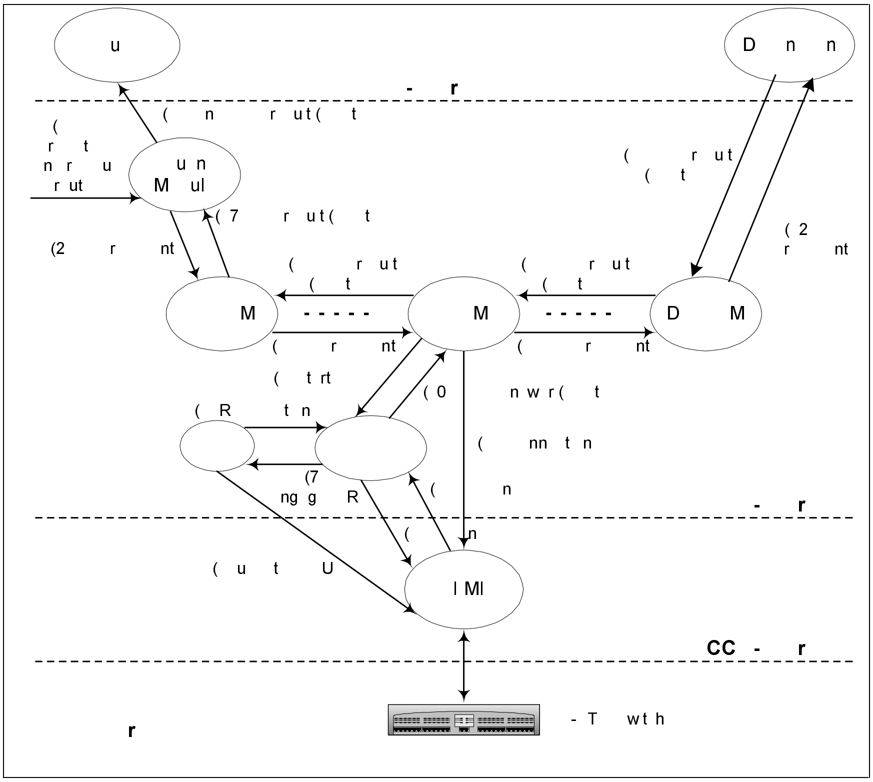


Fig. 3. Call acceptance after bandwidth redistribution

According to Figure 3, the signalling path for this case is similar with the path of the previous case until the point where i-CAC performing a CAC scheme does not accept the requested call due to the fact that the examined VMU does not have adequate resources to serve the request. Then, i-CAC sends a message `engage_BR` to the respective Bandwidth Redistributor (i-BR) in order to initiate the bandwidth redistribution process. The latter scans the port (indicated by the parameter `outPortID` of the message `engage_BR`) in which the examined congested VMU is attached and finds another VMU with bandwidth surplus. The amount of bandwidth needed by the incoming call is released from the VMU with unused residual bandwidth and appended to the congested VMU, thus allowing it to serve the new call request. i-BR notifies i-CAC about the successful bandwidth redistribution through the message `redistribution_ok`. Receiving this message, i-CAC continues its operation as in the case of accepted call request without bandwidth redistribution. The difference from the previous case is that it is the BR module, which updates the load conditions of examined VMU (adding the new call request) and not the CAC one.

The last case in the call set-up area, which is depicted in Figure 4, concerns the call rejection by the network.

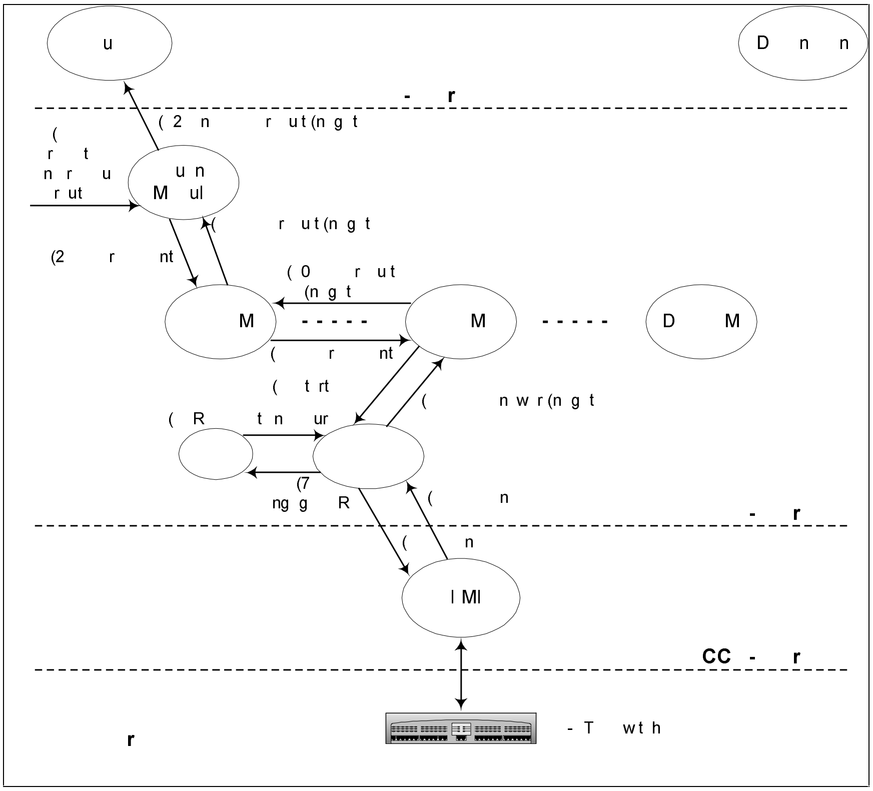


Fig. 4. Call rejection by the network

According to this case, the call request has been rejected in its all possible routes except the last one (as in Figure 3) where a i-CAC rejects the call request and its corresponding i-BR fails to perform bandwidth redistribution. i-BR announces to i-CAC its failure though the message `redistribution_failure`. Subsequently, i-CAC sends to i-CFM a message `CAC_answer` (where the boolean parameter `answer` of the message has the value `false`). i-CFM creates a negative message `call_result` (the parameter `result` has the value `false`) and forwards it to the previous call forwarding module along the route, which passes it to its own previous call forwarding module, etc. Eventually, the negative message `call_result` reaches at SrcCFM and from there at the routing module, which creates the negative message `final_call_result` (the parameter `result` has the value `false`) and passes it to the source terminal informing it about the call rejection.

Besides passing a negative message `call_result` to the previous call forwarding module, each call forwarding module (having performed bandwidth pre-reservation) sends a message `reverse_VMU` to its SI-MIB (with parameters `outPortID`, `QoS` and `BackgroundTraffic`) directing the SI-MIB to recall to the corresponding involved VMU the original load conditions.

3.2 Call Release

According to the call release scenario depicted in Figure 5, the source terminal initiates the call release function by sending a message `call_release` to the routing module. The message `call_release` contains a parameter `RefNum`, which is used by the routing module to locate the first node of the routing path under release. Then the routing module forwards the message `call_release` to the call forwarding module of the first node (SrcCFM), which passes it both to its corresponding CAC module (SrcCAC) and to the next call forwarding module along the route under release.

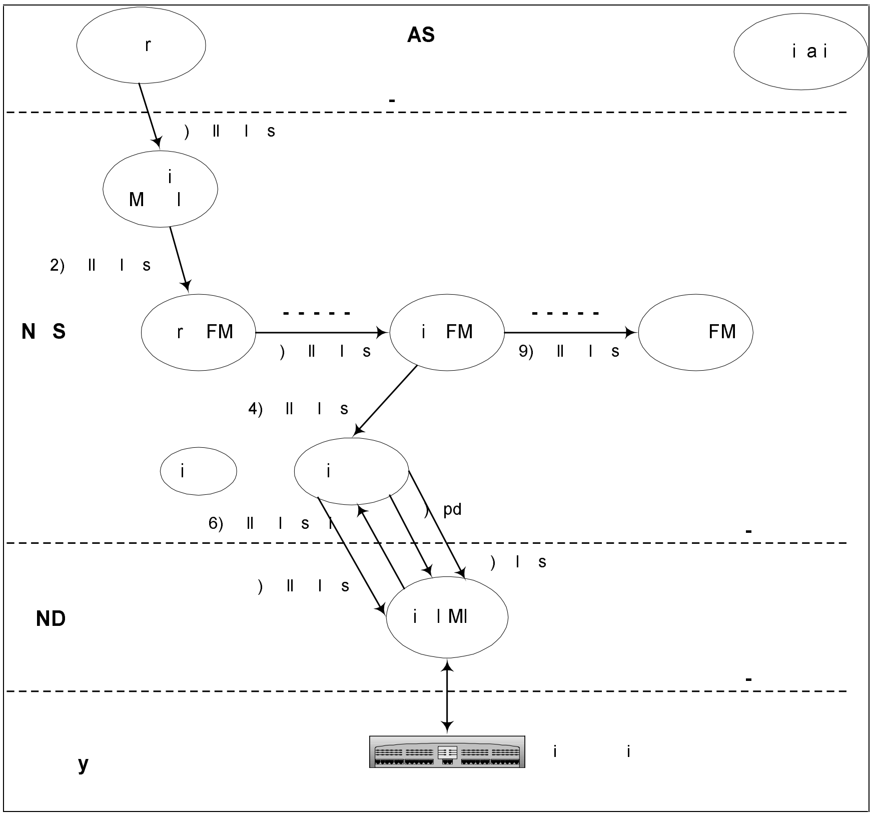


Fig. 5. Call release

Considering the i-node of the routing path between source and destination, i-CAC forwards the message to the i-SI-MIB and gets a reply in the form of a message `call_release_info`. The message `call_release_info` contains the parameters `QoS`, `TrProf`, `res` and `BackgroundTraffic`. These parameters are used for the calculation of the updated (the final amount after the call release) background traffic on the specified VMU (parameter `updBackgroundTraffic`). The VMU is updated with the new load conditions through the message `update_VMU`, sent by i-CAC to i-SI-MIB. Besides the message `update_VMU`, i-CAC sends a message `release_conn` (with parameters `inPortID, inVPI, inVCI, outPortID, outVPI, outVCI`) to the i-SI-MIB, directing it to remove the VC cross-connections created to serve the released call. The aforementioned procedure is performed on every node along the route, leading, as requested by the source terminal, to a complete call release from the network.

As an overall observation on Figures 2,3,4 and 5, it is apparent that some of the messages run through the L and U interfaces (defined according to the IEEE P1520 reference model), while some others are internal messages within NGSL.

4 A VSP Prototype Implementation

A prototype implementation of VSP runs over a software platform, which can be installed over heterogeneous ATM network islands for providing advanced traffic control and resource management functionality. The platform includes various distributed software entities created by extending an intelligent agent platform, which was built using the BAT object library [19]. VSP is implemented through facilities based on the Java Remote Method Invocation (JRMI) programming and the powerful mechanism of signature matching. The invocation of methods belonging to the distributed software entities of the platform is based on the agent ontology specified by FIPA 10. The directory services of BAT are employed to locate and invoke the agents in the distributed environment. Every agent incorporates at least one worker module, which is usually implemented in a thread form, to carry out the agent's transactions.

Figure 6 outlines a portion of the platform, operating on top of a small ATM network consisting of one domain with three switches. As shown in this figure, the system consists of a number of distributed software entities that communicate through VSP. Five different types of entities may be identified:

1. Terminal Agents – TAs: Software agents representing the terminal devices. They are responsible for the message `call request` creation providing the user interface for the definition of the corresponding parameters (`source_id`, `dest_id`, `QoS`, and `TrProf`).
2. Routing Agent – RA: Single central agent responsible for intra-domain routing in a core ATM network. RA also has coordinating responsibilities.
3. CAC Agents – CAs: Distributed agents responsible for the performance of the CAC function on the ATM switches. CAs are also responsible for the forwarding of messages `call request` and `call result`. Each CA groups with its corresponding SwWA agent, leading to a "one-to-one" formation dedicated to the respective underlying ATM switch.
4. Switch Wrapper Agents - SwWAs: Delegated agents that wrap the ATM switches by abstracting their hardware resources into the SI-MIB. Switch wrapper agents sit on top of the ATM switches and provide the virtual environment for the deployment of traffic control and resource management operations on the switches.
5. Bandwidth Redistributors – BRs: The main purpose of a BR is the congestion relief of VMUs. It achieves this goal by transferring to a congested VMU a part of the surplus bandwidth of a VMU attached to the same port.

The bandwidth reallocation takes place when an incoming call cannot be served by a fully utilized VMU along its route.

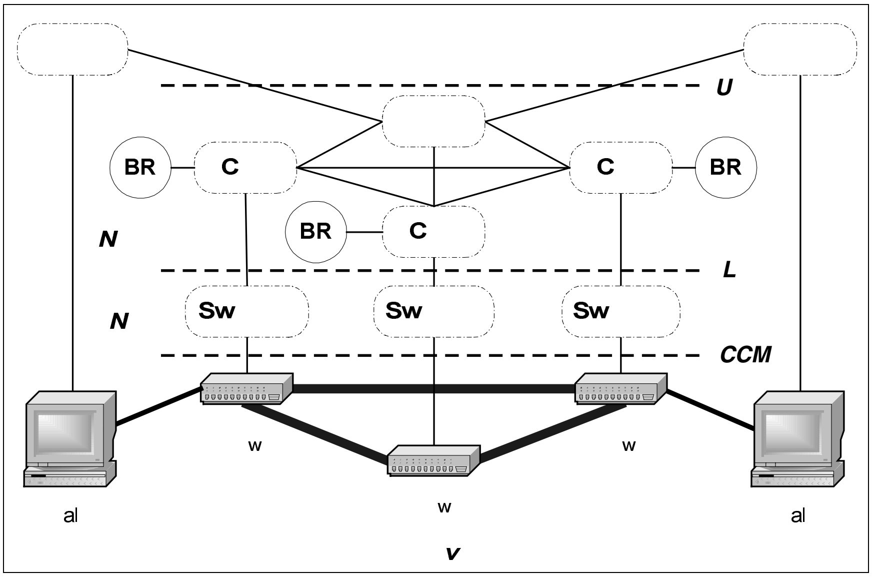


Fig. 6. A VSP prototype implementation

For details on the structure, functionality and implementation of the above entities, the reader is referred to [20]. Using the implemented prototype, the VSP call performance was benchmarked. Figure 7 depicts the VSP call set-up (without BR involvement) and call release delays. The involvement of BR in a call set-up process adds 5ms per hop. As Figure 7 shows, the average call set-up delay on a single Fore ASX-200BX ATM switch was measured at 76ms, increasing almost linearly with the number of nodes/hops per route. The corresponding average call release delay was measured at 33ms, increasing also linearly with the number of nodes/hops per route. The almost linear increment of VSP call set-up and release time constitutes a significant proof of protocol's robustness and scalability. Furthermore, concerning the fairness of VSP, it should be noted that all calls are treated equally.

An actual picture of VSP call performance is given through the comparison of VSP call set-up and release time with the native UNI 3.0 ones. According to [21], for a Fore ASX-200BX ATM switch (this type of switches are used in the implemented prototype) the call set-up and call release delays are 27ms and 6ms respectively. The variation of VSP's delays to UNI 3.0 ones can be both justified and accepted. It can be justified by the out-of-band nature of VSP and performance limitations of Java. It can be accepted considering that the goal of VSP is not to give call set-up/release times faster than the conventional standardized signaling protocols achieve, but to provide externally defined traffic control and resource management functionality.

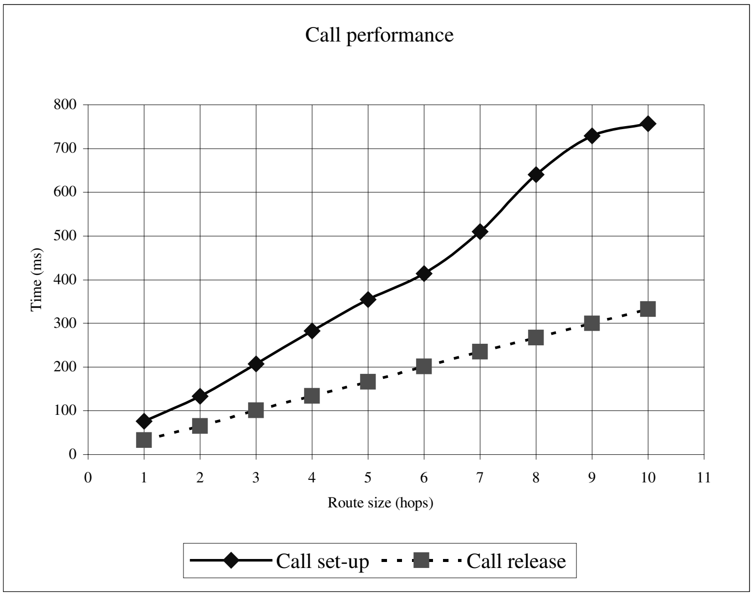


Fig. 7. VSP call performance

5 Conclusion

Following recent proposals for programmable network infrastructures, the paper presents a Virtual Signalling Protocol (VSP), which bypasses the conventional control information channels and supports externally defined traffic control and resource management functionality. Furthermore, the paper gives a generic framework for the employment of the IEEE P1520 L and U interfaces, as well as of NGSL in a standardized and systematic way for the purposes of device hardware independent traffic control and resource management. It should be noted that, although the protocol presented in the paper is tailored to ATM networking equipment, it is fairly general and can be extended for covering network nodes of a different technology (provided that this technology supports the potential for QoS concept and at least some notion of "connection" or, more generally, "traffic flow").

References

1. ATM User Network Interface (UNI) Specification V3.1, Electronically available at http://www-mo.atmforum.com/ftp/atm/approved-specs/af-uni-0010.002/.
2. ATM User Network Interface (UNI) Signalling Specification version 4.1, Electronically available at ftp://ftp.atmforum.com/pub/approved-specs/af-sig-0061.002.pdf.

3. Recommendation Q.2931 – User-Network Interface layer 3 specification for basic call/connection control. Electronically available at http://www.itu.int/rec/recommendation.asp?type=items&lang=e&parent=T-REC-Q.2931-199502-I.
4. Private Network-Network Interface Specification V1.1, Electronically available at ftp://ftp.atmforum.com/pub/approved-specs/af-pnni-0055.001.pdf.
5. A.T. Campbell, H.G. De Meer, M.E. Kounavis, K. Miki, J.B. Vicente and D. Villela, "A survey of programmable networks," *ACM SIGCOMM Computer Communication Review*, vol. 29, no. 2, pp. 7–23, April 1999.
6. Open Signalling Working Group, http://comet.columbia.edu/opensig/
7. J. Biswas, "The IEEE P1520 Standards Initiative for Programmable Network Interfaces", *IEEE Communications Magazine, Special Issue on Programmable Networks*, October 1998.
8. M. Chan, J. Huard, A. Lazaar, K. Lim, "On realizing a Broadband Kernel for Multimedia Networks", 3rd COST 237 Workshop on Multimedia Telecommunications and Application, Barcelona, Spain, November 25–27, 1996.
9. J. Van der Merwe, S. Rooney, I. Leslie, S. Crosby, "The Tempest – A Practical Framework for Network Programmability", *IEEE Network*, November 1997.
10. Foundation for Intelligent Physical Agents, *FIPA00023 Agent Management Specification,* 2002, Electronically available from http://www.fipa.org/specs/fipa00023/SC00023J.html
11. Mobile Agent System Interoperability Specification, Electronically available at http://www.omg.org/cgi-bin/doc?orbos/97-10-05.pdf
12. S. Vinosky, "CORBA: Integrating Diverse Applications Within Distributed Heterogeneous Environment", *IEEE Communications Magazine*, Vol. 14, No. 2, February 1997.
13. Distributed Component Object Model http://www.microsoft.com/com/tech/DCOM.asp
14. Java Platform http://java.sun.com/
15. Grasshopper 2 Agent Platform http://www.grasshopper.de/
16. Voyager Agent Platform http://www.recursionsw.com/products/voyager/voyager.asp
17. G. Kormentzas, K. Kontovasilis, "An Open Architecture for Transparently Embedding Advanced Traffic Control Functionality in ATM switches", *Journal of Communications and Networks, Special Issue on Programmable Switches and Routers*, Vol. 3, No 1, March 2001.
18. G. Kormentzas, J. Soldatos, E. Vayias, K. Kontovasilis, and N. Mitrou, "An ATM Switch-Independent MIB for Portable Deployment of Traffic Control Algorithms", In Proc. *7th COMCON Conference*, July 1999, Athens, Greece.
19. J. Bigham, L.G. Cuthbert, A.L.G. Hayzelden, Z. Luo and H. Almiladi, "Agent Interaction for Network Resource Management," In Proc. *Intelligence in Services and Networks '99 (IS&N99) Conference*, Barcelona, April 1999.
20. K. Katzourakis, G. Kormentzas, K. Kontovasilis, C. Efstathiou, "A Software System for Providing Traffic Control and Resource Management Functionality in ATM Networks", Submitted to *INFOCOM 2004 Conference*, March 2004, Hong Kong.
21. R Pillai, KL Su, J Biswas and CK Tham, "Call Performance Studies on ATM Forum UNI Signalling Implementations", *Computer Communications,* Vol. 22, Issue 5, pp. 463–469, April 1999.

Providing Enhanced Differentiated Services for Real-Time Traffic in the Internet

Tamrat Bayle[1], Reiji Aibara[2], and Kouji Nishimura[2]

[1] Department of Information Engineering,
Graduate School of Engineering, Hiroshima University,
1-4-1 Kagamiyama, Higashi-Hiroshima, 739-8527, Japan
tamrat@hiroshima-u.ac.jp

[2] Information Media Center, Hiroshima University,
1-4-2 Kagamiyama, Higashi-Hiroshima, 739-8511, Japan
{ray, kouji}@hiroshima-u.ac.jp

Abstract. The Differentiated Services (DiffServ) architecture offers a scalable alternative to provide Quality of Service (QoS) guarantees for performance-sensitive applications in the Internet. Within the DiffServ framework, efficient traffic scheduling mechanism is a key component to ensure such QoS guarantees. In this paper, scheduling algorithm called Enhanced Weighted Fair Queueing (EWFQ) is proposed that enables fair bandwidth sharing while supporting tight bounds on end-to-end delay for real-time traffic such as voice over IP (VoIP) in DiffServ networks. EWFQ allows to create service classes and assign proportional weights to such classes efficiently according to their resource requirements. The results from the simulation studies show that the mechanism is able to ensure both the required end-to-end delay bounds and bandwidth fairness based on the specified service weights. Besides, our scheme has lower implementation complexity, along with scalability to accommodate the growing traffic flows in the Internet backbone.

Keywords: Internet QoS, DiffServ, Scheduling Algorithm, VoIP

1 Introduction

The current Internet architecture provides only best effort service. Such best effort service is adequate for traditional Internet applications like e-mail, web browsing or file transfers. However, the new emerging real-time applications, such as voice over IP (VoIP), and multimedia conferencing, are sensitive to delay and delay variation, and require bandwidth guarantees. Consequently, the need to equip the Internet infrastructure with mechanisms to enable Quality of Service (QoS) is critical.

The research efforts by the Internet Engineering Task Force (IETF) to enable end-to-end QoS over IP networks have led to the design of two different architectures: the *Integrated Services* (IntServ) architecture [1] and more recently, the *Differentiated Services* (DiffServ) architecture [2], which although different, support services that go beyond the best effort service. Furthermore, due to the scalability limitations of the IntServ model for deployment in network backbones, the DiffServ architecture defines a scalable framework for providing QoS in the Internet. The DiffServ approach addresses the scaling concerns by reducing all traffic flows only into a small number of traffic aggregations, each with a different set of QoS requirements. [3] describes a DiffServ *per-hop behavior* (PHB) called *expedited forwarding* (EF) intended for use in building a scalable, low loss, low latency, low jitter, assured bandwidth, end-to-end service that appears to the endpoints like an unshared, point-to-point connection. Typically real-time, and mission-critical applications require this service. On the other hand, *assured forwarding* (AF) PHB [4] is suggested for applications that require a better reliability than the best-effort service. However, as a QoS control technology, DiffServ involves different traffic management mechanisms to provide the required multiple levels of services [5]. In this context, therefore, the packet scheduler is one of the key components of DiffServ networks, which plays important roles in service isolation because it actually gives different services to different traffic classes.

We propose a simple and efficient scheduling mechanism for DiffServ based Internet that enables fair bandwidth sharing while supporting better bounds on end-to-end delay for QoS-sensitive applications such as VoIP. It is based on our previous work [6] and incorporates the best characteristics of some existing approaches, notably the rate-based packet fair queueing algorithms [7,13, 14]. However, our scheme considers only serving a greatly reduced number of service classes rather than potentially a huge number of flows, and so lowers significantly its implementation complexity, while maintaining the robustness properties, required for end-to-end delay bounds and bandwidth fairness, of previously proposed alternatives. We call this algorithm *Enhanced WFQ* (EWFQ), as it depicts the capacity to adapt efficiently to Differentiated services environment.

The rest of the paper is organized as follows. Section 2 briefly reviews existing alternatives, and explains both their strengths and limitations. Section 3 discusses a new scheduling mechanism that lowers the implementation complexity, and ensures tight delay bounds for real-time packets within EF service class and bandwidth fairness among all traffic classes. Section 4 analyzes the performance of the proposed mechanism using simulations. The same Section, first describes the network topology and traffic models considered and some of the assumptions made for evaluation of the mechanism, and then presents the simulation results and discusses their implications. Finally, Section 5 concludes the paper with a brief summary and outlines directions for future work.

2 Background and Related Work

Scheduling disciplines are the key to fairly share a limited amount of network resources and provide QoS for performance-sensitive applications. A queue scheduling discipline allows to manage access to such a fixed amount of output link bandwidth by selecting the next packet that is transmitted on output port.

So, when there is congestion, then scheduling is the mechanism used to differentiate traffic and provide the required QoS. But the key challenge is to find an appropriate scheduling algorithm, especially with desirable properties of *low end-to-end delay bound*, *efficiency (low complexity of implementation)*, *fairness* and *scalability.*

There have been many research work done regarding this issue, each attempting to find the right balance among complexity, control, scalability, and fairness. For instance, priority queuing (PQ) [8] is the basis for a class of queue scheduling algorithms that are designed to provide a relatively simple method of supporting differentiated service classes. There is, however, a potential undesirable side effect with this scheme. Traffic with low-priority can become starved if there are a large number of high-priority classes. Moreover, PQ provides just a delay guarantee but no bandwidth guarantee. On the other hand, deficit round-robin (DRR) scheduling algorithm [10] addresses the limitations of the PQ mechanism by accurately supporting the weighted fair distribution of bandwidth when servicing queues that contain variable-length packets. DRR, though, does not provide the required good end-to-end delay bounds as other queue scheduling disciplines, which are described below. DRR provides a bandwidth guarantee but no delay guarantee.

WFQ [7] also known as Packetized Generalized Processor Sharing (PGPS) is more appropriate queue scheduling approach for applications with variable-size packets. WFQ supports bounded delay and fair bandwidth distribution for variable-length packets by approximating the ideal generalized processor sharing (GPS) system [9]. It does so by time-stamping each arriving packet with a *finish time*, the expected completion time of the packet if it were scheduled under the ideal GPS scheduler. However, WFQ implements a complex algorithm that requires the maintenance of a significant amount of per-packet state and iterative scans of state on each packet arrival and departure. Such computational complexity impacts the scalability of WFQ when attempting to support a very large number of flows on high-speed networks, which simply turns out not to be practical to implement it. As a result, many variants of WFQ [11,12,13,14] have been proposed with different trade-offs. Among the well-known variants, the worst case fair weighted fair queueing (WF^{2}Q) [13], and worst-case fair weighted fair queueing plus (WF^{2}Q+) [14] achieve tight delay bounds and worst-case fairness properties. While WFQ uses only *finish times* of packets in the GPS system, WF^{2}Q uses both the *start* and *finish times* of packets to achieve a more accurate emulation of a GPS system to provide improved worst-case fairness. However, WF^{2}Q has the same computational complexity as WFQ. On the other hand, WF^{2}Q+ is an enhancement to WF^{2}Q, and implements a new virtual time function that results in lower complexity. Even though both schemes are fair in the *worst-case* sense and tend to have low delay, they were not designed to provide service differentiation among classes in the context of DiffServ networks.

Non of the scheduling disciplines described above, however, avoid the maintenance of either per-packet or per-flow state somewhere in the network or oth-

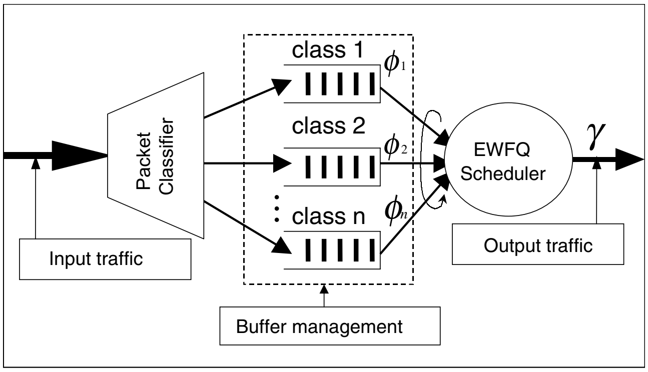


Fig. 1. Scheduler based on EWFQ

erwise dealing only with flows instead of with differentiated service classes. They significantly affect the scalability of the Internet backbone.

Therefore, by exploiting the DiffServ architecture, and use of the *start* and *finish* times for a packet only at the head of each of the active service class queues, the approach used in this paper goes a step further to reduce the size of huge traffic flows into very few service classes, as well as reducing the computational complexity, with respect to system virtual time computation, required for its implementation in high-speed backbone routers when compared with the above approaches.

3 Proposed Scheduler: EWFQ

In this section, we propose a scheduling mechanism called EWFQ for real-time IP traffic in DiffServ Networks. EWFQ is an improved mechanism in such a way that not only controlled bandwidth sharing and tight delay bounds are supported but also simplifications of implementation complexity are offered in the context of DiffServ environment.

EWFQ algorithm requires the maintenance of only aggregate state of very few traffic classes in the DiffServ networks, instead of a huge number of flows. In our approach, flows are aggregated and such aggregated flows are mapped into separate queues with different weights corresponding to service classes. Then the service time for a packet only at the head of the queue of active service class is calculated. Also, the sorting to transmit the next packet is done only among the head of very few active QoS classes. This implies that the complexities associated with EWFQ scheduler both for computing the system virtual time (for tagging with virtual finish time) and maintaining the set of eligible classes sorted by virtual finish times depend only on the number of supported service classes, which typically are much smaller than the number of sessions or flows. This simplification with our approach greatly reduces the computational complexity that is attached inherently with other approaches.

```
EWFQ Algorithm

/* a new packet $P$ of class $i$ arrives */
FOR $i \leftarrow$ index of class $i$'s queue that will hold new $P$
DO
BEGIN Enqueue($i, P$)
    if ($queue[i]$ == $empty$) {
        enqueue($P$, $queue[i]$);
        /* compute starting/finishing times for */
        /* packet at the head of class $i$ queue */
        $S[i] \leftarrow max(V, F[i])$;
        $F[i] \leftarrow S[i] + L_i^P / \phi_i$;
        /* update system virtual time */
        $V \leftarrow max(min(S[j])_{j \in B}, V)$;
    } else
        /* append packet to end of class $i$ queue */
        enqueue($P$, $queue[i]$);
END Enqueue

BEGIN Dequeue($i$)
        /* dequeue and transmit (tx) the head packet */
        $P_{tx} \leftarrow$ dequeue($queue[i]$);
        send($P_{tx}$);
        /* get the next head packet from the same class $i$ */
        /* queue and compute starting and finishing times */
        $P_{next} \leftarrow$ getfromhead($queue[i], head$);
        if ($P_{next}$) {
            $S[i] \leftarrow F[i]$;
            $F[i] \leftarrow S[i] + L_i^{P_{next}} / \phi_i$;
        } else
        /* update system virtual time */
        $V \leftarrow max(min(S[j])_{j \in B}, (V + L_i^{P_{tx}} / \sum_{j \in B} \phi_j))$;
END Dequeue
```

Fig. 2. Enqueue and Dequeue Pseudocode for EWFQ

Consider n traffic classes, each class i, $i = 1, ..., n$, assigned to a *separate queue*, is associated with weight ϕ_i, and the link capacity is shared among all active classes in direct proportion to their weights, such that the sum of the weights of all classes is no larger than a predefined value γ. That means, if we consider Fig. 1, in which there are n service classes, then

$$\boxed{\phi_1 + \phi_2 + ... + \phi_n \leq \gamma \; , i = 1, ..., N \; .} \tag{1}$$

Here, the weight of the class specifies a relative share of how much of the capacity of the output link the class is entitled to receive. Furthermore, each class i is associated with two variables $S[i]$ and $F[i]$ that represent, respectively, the starting and the finishing times corresponding to the packet at the head of the queue for a particular class i. Finally, a global variable V, called system virtual time, is associated to the system. The EWFQ can be then described briefly as follows: First initialize variables $S[i]$, and $F[i]$ for all classes i, and V to *0*, and other variables, such as ϕ_i and n to their respective values. Then proceed with *Enqueue*, and *Dequeue* operations as described below.

Enqueue: The *Enqueue* operation is called whenever a new packet of class i arrives. According to Fig. 2, the scheduler visits each queue and when a packet P of length L for class i arrives at its queue i, it executes the first part of the pseudocode. The function first checks whether class i just becomes backlogged[1] from the idle state. If the case is a transition from the idle state, it first places the newly arriving packet into the head of its corresponding class queue. Then it involves computing the starting $S[i]$ and finishing $F[i]$ times, as well as updating the system virtual time V for packet at the head of queue of class i, in this case, which is the newly arriving packet. Otherwise, if class i queue were previously backlogged, i.e., non-empty, it only appends the newly arriving packet to the end of queue of class i. In updating the system virtual time, $min(S[j])_{j \in B}$ represents the minimum of starting times among all backlogged classes.

Dequeue: The Dequeue function is the core of the algorithm that schedules packets from the queues corresponding to different service classes. Consider the set of all backlogged traffic classes B, such that their *starting times* are no larger than the system virtual time V, i.e., $S[j] \leq V$, for any class j in B, and form a set of *eligible* classes E for service. The algorithm then selects class i among the set of classes in E that has the *smallest finishing time.* Accordingly, the algorithm dequeues the *head packet* from this class queue and denotes it by P_{tx} for transmission. Finally, it proceeds to execute the remaining second part of the pseudocode in Fig. 2. The $getfromhead()$ function gets the next available packet P_{next} from the head of the same class i's queue whose head is passed as a parameter to the function. Then $S[i]$ and $F[i]$ are computed for the class. Otherwise, if $P_{next} \equiv NULL$, the system virtual time V will be updated.

In Eq. (1), if $\gamma = C$, where C represents the link capacity, then the weight of a class represents the minimum bandwidth that the class is guaranteed to receive. When the traffic class i is constrained by a token bucket (ρ_i, σ_i), where ρ_i is the average token generation rate, and σ_i is the token bucket depth, then class i is guaranteed to obtain a minimum fair service rate shown in Eq. (2).

$$r_i = \begin{cases} \left[\frac{\phi_i}{\sum_{j \in B} \phi_j}\right] \gamma \ \forall i \in B \\ 0 \qquad\qquad\qquad \text{otherwise.} \end{cases} \tag{2}$$

[1] A class is called backlogged if it has at least one packet in the queue waiting for transmission on the output link, or idle otherwise.

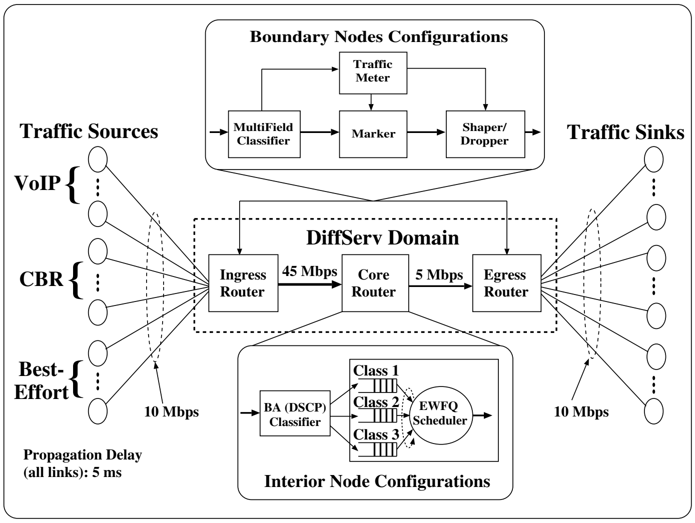


Fig. 3. DiffServ Network Model for Simulations

4 Performance Evaluation

The EWFQ algorithm is evaluated using a network simulation test-bed. We implemented the algorithm in the most popular ns-2 network simulator [16], and then validate its functionality and effectiveness with several simulation results. The key aspect of the experiment is to evaluate EWFQ scheme on its assurance of bounds on delay and jitter, as well as the minimum guaranteed bandwidth for the class that was given high priority, while equally observing its fair allocation of link bandwidth to other low priority service classes. These performance characteristics enable to determine whether the suggested scheme is fair and efficient, and can support VoIP applications in DiffServ networks in order to achieve an acceptable voice quality.

Note that the implementation of different PHBs is achieved through a combination of a variety of packet classification, packet marking, traffic conditioning, and buffer management mechanisms. These are on top of our packet scheduling mechanism, and thus, we make use of all the DiffServ components in our modified ns-2 to implement both EF and AF PHBs. Also note that the scheduling mechanism analyzed in this section assumes that the input traffic for both EF and AF is rate regulated [2] at the ingress hop using traffic policing technique. Thus, the proposed scheduler does not provide performance guarantees if the input traffic is not conditioned according to some service level agreements (SLAs). Another assumption is that the effect on high priority traffic should be minimal or negligible as the medium (AF) and low priority traffic classes are varied.

[2] This rate-limiting functionality is achieved using a token bucket scheme that regulates class i traffic with (σ_i, ρ_i) model in the DiffServ network we consider.

Table 1. Classifying traffic packets into their service Classes (1^{st} scenario)

Traffic Types	**Mapped Classes**	**No. Flows per Class**	**Service (PHBs)**	**Relative Weights**
VoIP	Class 1	100	EF	$\phi_1 = 60\%$
CBR (data)	Class 2	5	AF	$\phi_2 = 30\%$
FTP	Class 3	5	BE	$\phi_3 = 10\%$

4.1 Simulation Network Model

To evaluate the effectiveness of the algorithm through simulations, we use the simple network model as depicted in Fig. 3. In this model, we consider a network connection between six computers over a single-domain route in which the network connection passes through three DiffServ routers.

4.2 Traffic Model

In order to perform more realistic simulations with a variety of traffic behaviors, we consider three types of traffic mixes as follows: voice traffic, constant-bit-rate (CBR) data over UDP, and FTP data over TCP. The voice traffic is assumed to be an exponential distributed on-off traffic of two states, speaking and silence [15], and consists of 100 flows; each characterized by a packet size of 84 bytes, burst time 350 ms, idle time 650 ms, and peak rate of 64 kbps during on period. Therefore, the average sending rate for each voice source is 22.4 kbps, which comprises the total voice traffic rate about 2.24 Mbps. Our aim is to emulate a voice traffic over differentiated IP networks so that we can use this as VoIP traffic model to be treated with DiffServ EF PHB. In fact, the characterization depends on the coding system utilized, but for our simulations purpose we use such a simple voice traffic model to simulate VoIP. To describe both the second, and third traffic classes, we consider two different scenarios based on different changes in the traffic types and their sending rate behaviors.

Scenario 1. In the first scenario, we imagine the CBR traffic is sending at a rate more than its subscription rate (oversubscription case), and the background traffic class comprises 5 FTP sessions. And thus, the second class of traffic consists of 5 CBR sessions, which can be treated with AF PHB, and the third class, which represents best effort (BE) background traffic, consists of a set of 5 FTP connections. For FTP traffic transportation, we consider TCP Reno [17], which is widely used in today's Internet. Here, the main objective is to observe the ability of EWFQ mechanism how it can satisfy the delay, and average bandwidth requirements of the traffic in the first class (voice traffic) by protecting it from other misbehaving traffic classes. In this case, the CBR source is sending traffic at an average rate of 4 Mbps to be served with AF treatment. Imagine the bandwidth of the bottleneck link is 5 Mbps.

Furthermore, packets from these classes of traffic are scheduled using EWFQ policy with relative weights of $\phi_1 = 60\%$, $\phi_2 = 30\%$, and $\phi_3 = 10\%$, to represent

Table 2. Classifying traffic packets into their service Classes (2^{nd} scenario)

Traffic Types	**Mapped Classes**	**No. Flows per Class**	**Service (PHBs)**	**Relative Weights**
VoIP	Class 1	100	EF	$\phi_1 = 60\%$
CBR (data)	Class 2	5	AF	$\phi_2 = 30\%$
Self-Similar	Class 3	30	BE	$\phi_3 = 10\%$

DiffServ PHBs of EF, AF, and BE, respectively. Table 1 shows the summary of the traffic mapping into their respective classes, and the corresponding PHBs treatments for the first scenario. EWFQ was setup at the hot-spot link between the Core and Egress routers, according to Fig. 3. That means, the three service classes share the same bottleneck link of capacity 5 Mbps. To isolate the three types of service classes, each router uses three separate physical output queues of each size of 100 packets scheduled with EWFQ policy. Then, with this scenario, we carried out several tests, and measured the results of throughput, latency, jitter and packet loss for all traffic classes to ensure that the oversubscription by medium priority (AF) class should not affect the high priority class but only the low priority class.

Scenario 2. Table 2 shows the summary of the traffic mapping into their respective classes, and PHBs treatments for the second case. In this scenario, as a background best-effort traffic, we use 30 Pareto On/Off sources, each with an average rate of 100 kbps to generate an aggregate traffic rate of 3 Mbps. Traffic characterization according to a Pareto ON/OFF distribution has been proved to be self-similar in nature [18]. Such traffic modelling is widely considered to describe well the selfsimilarity (burstiness) nature of the Internet traffic. Moreover, the total CBR traffic rate was reduced from 4 Mbps into 1.5 Mbps (still 5 CBR sessions). Also, in order to reduce the delay experienced by all packets because of the queue depth, the three queue sizes are set to be 50 packets.

4.3 Simulation Results and Discussion

In this section, we discuss the results of the simulation described above by showing the benefits of the proposed scheduler in DiffServ network environments. We record the packet loss, throughput, latency and jitter for each class, as these are vital performance metrics for supporting real-time voice traffic over IP networks.

Table 3 depicts the summary of packets statistics for each DSCP $\rightarrow$ Class match for the first simulation scenario. Here, the packets for each class are identified by two differentiated services code points (DSCPs), signifying in profile, and out of profile packets, according to the predefined SLAs. So, this enables, first to classify marked packets into their corresponding classes, and second, within each service class, to identify the drop precedences for the buffer management mechanisms.

Table 3. Summary of Packets Statistics for each DSCP → Class match (1st scenario)

DSCP Mapping	Total Pkts Received	Total Pkts Sent	Total Pkts Drops	Pkts Loss Rate (%)
00 → BE	5415	5101	314	6
01 → BE	86	65	21	24
10 → EF	197300	197300	0	0
11 → EF	0	0	0	0
20 → AF	29493	15493	14000	48
21 → AF	0	0	0	0

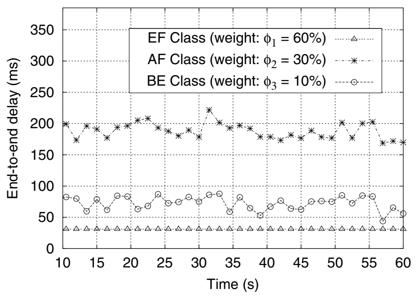


Fig. 4. Average end-to-end delay (1st scenario)

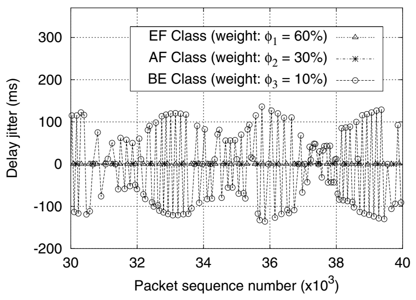


Fig. 5. Inter-packet delay jitter (1st scenario)

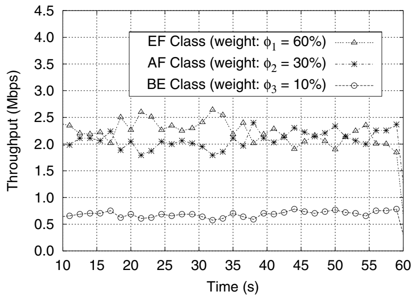


Fig. 6. Bottleneck link bandwidth (1st scenario)

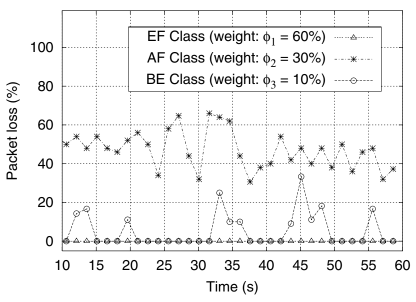


Fig. 7. Packet loss at the bottleneck (1st scenario)

For the first scenario, the results for different performance metrics are shown in Fig. 4 through Fig. 7. Figure 4 and Fig. 5 present the end-to-end packet delay and jitter of each class, respectively. From these simulation results, we observe that voice traffic, which is classified as high priority class (Class 1) for EF PHB treatment with EWFQ, receives much tight delay bound, while its jitter is almost insignificantly small. Its request for bandwidth is also fully satisfied compared with medium (Class 2) and low priority (Class 3) traffic classes. This

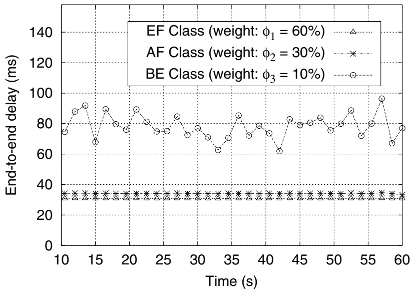


Fig. 8. Average end-to-end delay (2^{nd} scenario)

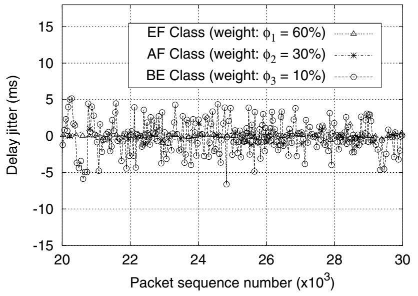


Fig. 9. Inter-packet delay jitter (2^{nd} scenario)

is true even when the rates of the background best effort traffic is increased. For example, if the weight ϕ_1 is set to be more bigger, and at the same time the voice traffic class has many more sessions to establish, then the performance of other classes is degraded significantly due to the delay in their respective queues. This is because the scheduler visits much often the queue for high priority class for it has a big weight assigned by EWFQ.

As one can see clearly from Fig. 4 and Fig. 7, the delay and packet loss for AF class are rather high compared to the best-effort class. There are two obvious factors that can be causing such behavior. First, the background traffic consists of a set of TCP flows, and thus react to congestion by shrinking their flow-control windows and sending less packets to reduce the packet loss. In fact, as it can been seen from Fig. 6, and Fig. 4, this behavior of TCP traffic results in lower bandwidth and poor response times, respectively. Second, and more important is that the traffic destined to AF class is sending with a rate much higher than the subscribed rate, 4 Mbps. As a result, packets from this class suffer much from the combination of both packet loss and longer delay in the queue.

On the other hand, the real-time packets marked for EF treatment are forwarded with very low end-to-end delay (about 30 ms), minimal jitter and no packet loss compared with packets marked for other service classes. This holds true as long as it complies with the SLA of the network. We also observe that it is difficult to create a wide range of influence on the targeted service classes when the background best-effort traffic consists of a set of TCP flows. This is, as described above, partly because of the TCP behavior itself, and partly there are several other parameters needed to be tuned, especially with RIO (RED with In and Out) buffer management mechanism that makes it extremely difficult to generate the desired effect.

For the second scenario, Fig. 8 through Fig. 11 present the end-to-end delay, jitter, throughput, and packet loss results, respectively, using EWFQ scheme. These results are supplemented with the packet loss results in Table 4. Note that we use self-similar traffic as a background best-effort traffic. Even in this case, the performance of voice traffic is not affected by the self-similarity nature

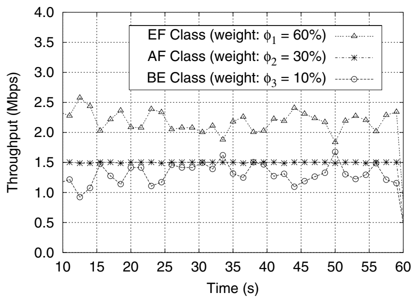


Fig. 10. Bottleneck link bandwidth (2nd scenario)

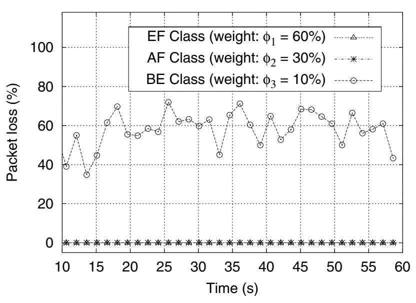


Fig. 11. Packet loss at the bottleneck (2nd scenario)

Table 4. Summary of Packets Statistics for each DSCP → Class match (2nd scenario)

DSCP Mapping	Total Pkts Received	Total Pkts Sent	Total Pkts Drops	Pkts Loss Rate (%)
00 → BE	59486	29666	29820	50
01 → BE	32877	8373	24504	75
10 → EF	197429	197429	0	0
11 → EF	23	23	0	0
20 → AF	11154	11154	0	0
21 → AF	0	0	0	0

of the traffic. This time, what one can see from Fig. 8, and Fig. 9 is that for both EF and AF classes the delay is minimal, and almost no jitter, respectively. And from Fig. 11, and Table 4, we also observe no packet loss for the two classes except best-effort class, which is expected. This is mainly because that our scheduler also bounds the delay and packet loss for AF class, as long as the traffic is within its allowed rate.

As for the bandwidth, both EF and AF traffic are also guaranteed with their requirements. This can be seen clearly in Fig. 10. And since the voice traffic consists of on/off traffic sources, during the on period it is ensured all its QoS requirement, but during its off time, the bandwidth which is not used by it is consumed by the best effort traffic. So, whenever excess bandwidth is available, EWFQ distributes this extra bandwidth among all the classes proportionally, i.e., according to their relative weights.

With EWFQ, all results in the above Figures show clearly how all voice packets are served with the highest assurance of delay bounds, minimal jitter, bandwidth guarantees, and packet loss regardless of the rates of medium and low priority classes. For other classes, the delay and delay variations are correlated largely with the increment of their average queue sizes during congestion, and their corresponding weights, as well. This is mainly due to the 60% of the service time of the scheduler is spent for serving the voice class. As a result, the more the queue is in congestion, the more the delay for packets to reach the destination is increased. The delay and jitter problems will be more aggravated if the queue size is getting large. On the other hand, the queue size should be set large enough to avoid packet loss. So, one has to face a tradeoff between packet loss and delay for medium and low classes during times of high congestion.

5 Conclusion and Future Work

In this paper, we study whether it is feasible to achieve fair and scalable bandwidth sharing while supporting better bounds on end-to-end delay for real-time IP traffic classes within the DiffServ framework. To this end, we propose a scheduling mechanism called EWFQ. The proposed scheme is effective in providing efficient and enhanced services for real-time multimedia traffic class, such as VoIP, over other non-real-time traffic classes while providing the additional benefits of fair bandwidth sharing among all the classes proportionally. With EWFQ, it is possible to maintain tight delay bound for high priority class, and distribute the bandwidth according to the predetermined weights for all service classes. The results demonstrate that the mechanism has the capability for providing service isolation needed among different traffic classes in the network by protecting high priority class from other misbehaving traffic classes, which is vital in order to support real-time multimedia applications in the Internet. Another important aspect of EWFQ, in the same context, is that the scheme lowers significantly its implementation complexity, while maintaining the required end-to-end delay bounds and bandwidth fairness, along with scalability.

Further work on testing the algorithm in the realistic Internet environments is needed to see the impacts of extreme events. Other future work to consider, which would also most likely be important in this regard is to provide mathematical analysis on the end-to-end delay bound of EWFQ algorithm to show further its strengths and limitations in comparison with other approaches.

Acknowledgement. This work was supported in part by Telecommunication Advancement Organization Grant for Application Research and Development on Japan Gigabit Network (JGN-P341005).

References

1. R. Braden, D. Clark, S. Shenker, "Integrated Services in the Internet Architecture: an Overview," RFC 1633, June 1994.
2. S. Black, *et al.*, "An Architecture for Differentiated Services," RFC2475, Dec. 1998.
3. V. Jacobson, K. Nichols, K. Poduri, "An Expedited Forwarding PHB," RFC2598, June 1999.
4. J. Heinanen, *et al.*, "Assured Forwarding PHB Group," RFC2597, June 1999.
5. Zheng Wang, "Internet QoS: Architectures and Mechanisms for Quality of Service," Morgan Kufmann, March 2001. (ISBN: 1558606084)
6. T. Bayle, R. Aibara and K. Nishimura, "Scheduling IP Traffic for Enhanced Services in DiffServ Networks," Proc. of APCC2002, September 2002.

7. A. Demers, S. Keshav, and S. Shenker, “Analysis and Simulation of a Fair-Queueing Algorithm,” Proc. ACM SIGCOMM ’89, September 1989.
8. H. Zhang and D. Ferrari, “Rate Controlled Static Priority Queueing,” Proceedings of IEEE INFOCOMM 1993.
9. A. K. Parekh and R. G. Gallager, “A Generalized Processor Sharing Approach to Flow Control in Integrated Services Networks: The Single-Node Case,” IEEE/ACM Transactions on Networking, Vol. 1, No. 3, June 1993.
10. M. Shreedhar and G. Varghese, “Efficient Fair Queueing using Deficit Round Robin,” Proc. ACM SIGCOMM ’95, Vol. 25, No. 4, October 1995.
11. S. J. Golestani, “A Self-Clocked Fair Queueing Scheme for Broadband Applications,” Proc. INFOCOM ’94, Apr. 1994. April 1994.
12. P. Goyal, Harrick M. Vin, H. Cheng, “Start-time Fair Queueing: A Scheduling Algorithm for Integrated Services Packet Switching Networks,” IEEE/ACM Transactions on Networking, Vol. 5, No. 5, October 1997.
13. Bennett, J. and Zhang, H. “WF^{2}Q: Worst-case Fair Weighted Fair Queueing,” Proceedings of IEEE INFOCOM ’96, March 1996.
14. Bennett, J. and Zhang, H. “Hierarchical Packet Fair Queueing Algorithms,” Proc. ACM SIGCOMM ’96, August 1996.
15. J. G. Gruber, “A Comparison of Measured and Calculated Speech Temporal Parameters Relevant to Speech Activity Detection”, *IEEE Trans. Communs.*, Vol. COM–30, No. 4, pp 728–738, 1982.
16. “The Network Simulator, NS-2,” http://www.isi.edu/nsnam/ns/.
17. J. Padhye *et al.*, “Modeling TCP Reno performance: a simple model and its empirical validation,” IEEE/ACM Transactions on Networking 8, April 2000.
18. W. E. Leland, *et al.*, “On the Self-Similar Nature of Ethernet Traffic,” Proc. SIGCOM93, 1993, San Francisco, California, pp. 183–193.

Automatic Management of the QoS within an Architecture Integrating New Transport and IP Services in a DiffServ Internet

C. Chassot, G. Auriol, and M. Diaz

Laboratoire d'Analyse et d'Architecture des Systèmes du CNRS (LAAS-CNRS)
7, avenue du Colonel Roche, 31077 Toulouse cedex 04. France
{chassot, gauriol, diaz}@laas.fr

Abstract. Lots of research works have been performed for about ten years around the QoS problem in the Internet, both but *separately* at the Transport and at the IP levels. Taking into account the emerging traffic engineering-based QoS solutions (Diffserv-oriented), this paper targets the integration of new Transport services and protocols together with these solutions. Starting from performance measurements performed over a national DiffServ plat-form, contributions exposed here deal with the proposition and the implementation of a session level protocol allowing the application programmers to be masked with the complexity of choosing the underlying new Transport and IP services, still being provided with a per flow QoS.

1 Introduction

Technical evolutions in telecommunications and computer science have led to the development of new types of distributed applications such as interactive videoconferencing systems or distributed interactive simulation. These applications present challenging characteristics to network designers, such as the need for bounded delays together with small loss rates, guaranteed throughputs, etc. In order to tackle these new needs, several works have been performed both but *separately* at the Transport level then at the IP level.
As far as the Transport level is concerned, an important work has been performed in order to define new Transport protocols and architectures, both well suited with regard to the features of multimedia applications and independent of the underlying network layer technology. Particularly, starting from the fact that existing protocols only provide two kinds of service, "reliable and ordered" one (TCP) or "unreliable and unordered" one (UDP), whereas multimedia applications have partial reliability (PR) / partial order (PO)[1] constraints, the need became obvious to provide applications with PR / PO Transport services [1] [2]. Two major benefits result from the use of these new services within multimedia Transport architecture: [3] proved that PR/PO protocols allow to optimize both network resources (end-to-end storage buffers and bandwidth) and the transit delay of the application data units, still respecting reliability and logical synchronization constraints inherent to the distribution of a multimedia document.

[1] I.e. logical synchronization between, for instance, audio and video.

As far as the IP level is concerned, several research projects have been initiated to target the QoS problem through traffic engineering in a DiffServ context [4] integrating (often) the MPLS technology. In the European context, let us cite in particular the TF-TANT activity and the TEQUILA, AQUILA and CADENUS IST projects[2], proposing and/or implementing architectural frameworks for providing QoS in a multi-domain environment. In TEQUILA for instance, the focus has been made on a *policy-based* QoS management system, a policy being defined as a "way to guide the behavior of the network through high-level declarative directives" [5] [6] [7].

From the previous considerations, it results the need of an end-to-end communication architecture: (1) *integrating the QoS mechanisms and protocols developed at the IP and Transport levels,* and (2) able to provide the Application level with *guaranteed end-to-end QoS*, including order/reliability parameters together with temporal ones (transit delay).
Work presented in this paper targets this problem, our final goal being to develop an end-to-end architecture for a multi-domain DiffServ environment providing a guaranteed QoS (including reliability and transit delay) for each Application level data flow, and accessible through a generic API[3], allowing its user: (1) to express the required QoS with generic (i.e. non ad hoc) parameters; (2) to be masked with the complexity of choosing and using the underlying new Transport and IP services, by mean of a specific management protocol. Contributions exposed in this paper are the following:
– in section 2, we propose to formalize the conceptual link between the application needs and the communication system by defining how to map the application requirements together with deterministic or non deterministic underlying QoS services (such as AF-based ones at the IP level). This formalization is included in an actual end-to-end communication architecture (the @IRS[4] architecture) whose principles, implementation and performance measurements are also exposed;
– in section 3, we first study how to quantify the end-to-end QoS resulting from the coupling between new PR/PO Transport services together with EF or AF-based IP QoS in a mono domain DiffServ environment. Then we propose an algorithm allowing to select the adequate Transport and IP services from an Application level QoS request. Experimental tests allowing to observe the behavior of the end-to-end communication system for a videoconference application are exposed and analyzed;
– conclusions and future work are finally exposed in section 4.

2 The @IRS End-to-End Architecture

The basic principle that supports the @IRS architecture is one of many dedicated to the transfer of multimedia documents [3] [8] [9]. The idea is that the exchanged traffic can be decomposed into several data flows (one per media for a multimedia application), each one

---

[2] http://www.dante.net/tf-tant, http://www.ist-tequila.org/, http://www.cadenus.org/, http://www-st.inf.tu-dresden.de/aquila

[3] API: Application Programming Interface

[4] @IRS (*Integrated Networks and Services Architecture)* is the name of a project of the French *National Network of the Telecommunication Research*. Finished since the end of 2001, it is currently pursued in a second phase (@IRS++).

requiring a specific QoS in terms of reliability, transit delay, etc. In the @IRS architecture, the application software is then allowed, through a specific API (see Fig. 1), to establish *sessions* containing one or many *end-to-end communication channels*, each one being: (1) unicast and unidirectional, (2) dedicated to the transfer of a single applicative data flow, and (3) able to offer a specific QoS, including order, reliability and transit delay as explicit parameters, together with new *semantics of guarantee* detailed here after.

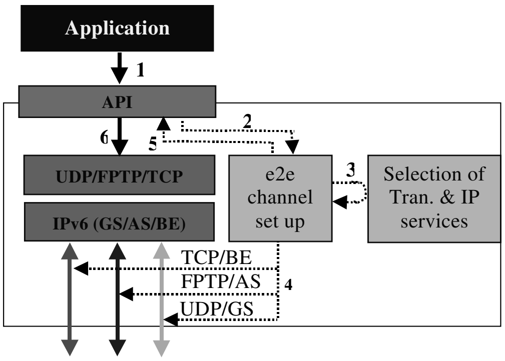


Fig. 1. @IRS end-to-end architecture

To perform the required QoS, two levels of protocols have been considered in the @IRS architecture: the Transport and the IP levels, both being able to provide three kinds of services (described in sections (2.2) and (2.3)). In the first version of the architecture, the Application level had to select for each of its flows both the Transport service and the IP service it wanted to be applied on each end-to-end channel. Such a choice is not obvious for two major reasons: the first one is that application programmers are not supposed to be expert in network protocols ! The second reason comes from a knowledge even network experts do not have yet: what is the QoS resulting from the coupling of new IP services (particularly AF[5]-based services) together with new Transport services ?
The @IRS architecture we are talking about in this paper extends the initial version in that it includes an *automatic* selection of the Transport and IP services able to perform (if possible) the QoS required for a flow (see section 3). Before addressing this part of our work, we first expose the main components of the @IRS architecture and then the networking platform over which it has been implemented, tested and evaluated.

2.1 API : QoS Parameters and Semantics of Guarantee

Several definitions of QoS parameters have been proposed in the past that often depend on the tested applications. The @IRS API is a "generic" API, i.e. parameters have been defined so as to take into account all possible kinds of applications needs.

[5] AF: *Assured Forwarding* per hop behavior [10].

QoS parameters. The @IRS QoS parameters are:
– an intra and inter flow partial order, expressing logical synchronization constraints either within or between flows;
– a partial reliability and a maximum end-to-end transit delay defined by:

- the percentage, noted τ_r, of sent packets that the application wishes to receive;
- the percentage, noted (τ_d, [a, b]), of sent packets that the application wishes to receive in the time interval [a, b];
- the maximal number of consecutive lost packets.

Guarantee Semantics
TCP and UDP respectively provide total or null reliability/order guarantee. Analogously, the semantics introduced here allow to establish a link between the applications needs and the communication system as far as the delay and reliability are concerned. More precisely, τ_r et τ_d are associated with the following semantics:
– a *absolute guarantee*, noted *A* meaning that the parameter value will be exactly obtained, for example by using a Transport level retransmission mechanism;
– an *average guarantee*, noted *M* meaning that the parameter value will be obtained by a statistical characterization of delay and reliability. Here, there is an incertitude on the parameter value, even if the communication system accepts the request.
– an *average guarantee with notification*, noted *N*, that extends the M guarantee by a notification to the application when the parameter value is not respected.

2.2 Transport Layer

Three services have been defined at the Transport layer:
– the first one is implemented by TCP and provides total order and total reliability guarantees on the data transfer;
– the second one is implemented by the PR/PO FPTP[6] protocol [11] that provides programmable partial order and partial reliability guarantees on the data transfer. In the following, we consider that this protocol may also be configured so as to limit the number of its retransmissions (when a loss occurs) to a given number *n*;
– the third service is implemented by UDP and provides neither order nor reliability.

2.3 IP Layer

Three services have been defined and implemented at the IP level:
– GS (*Guaranteed Service* - analogous to the *Premium Service*) provides its user with an almost fixed transit delay and a total reliability concerning routers congestions (however, destination host congestions or bit errors may occur);
– AS (*Assured Service*) provides *in-profile* traffics with an almost total reliability (concerning routers congestions) and an “acceptable” transit delay variation when

[6] FPTP (*Fully Programmable Transport Protocol)* is a PR/PO protocol implemented and tested over an international IPv6 platform in the IST GCAP project http://www.laas.fr/GCAP/

congestions occur in the network. Part of the traffic exceeding the characterization profile is conveyed in AS as far as no congestion occurs on the path used by the flow;
– BE (*Best Effort*) provides no QoS guarantee.
Implementation of these services (which follows the specification given in [10] and [12]) is based on several mechanisms described hereafter. Implementation details at routers input and output interfaces may be found in [13].

Control path QoS mechanisms. The main mechanism involved in the control path is the *admission control* which takes care of the acceptance of new AS or GS flows, a flow being identified (in @IRS) by the *flow id* and the source IP address of the IPv6 packets:
– for AS, the control is applied at the edge of the network only; it is based on the amount of AS traffic already authorized to enter the network. This guaranties that the amount of *in-profile* packets (i.e. respecting the traffic contact) in the network will be at most the sum of the AS authorized at each edge router;
– for GS, as a delay guarantee is needed, the admission control is supposed to involve all the routers on the data path. All our experiments have been done under the hypothesis that such an admission control was done.

Note that in the two cases, the path between the considered hosts is supposed to be fixed.

Data path QoS mechanisms. QoS functions involved in the data path are policing, scheduling and congestion control.
Policing and congestion control. Policing deals with the actions to be taken when *out-of-profile* traffic arrives at the edge of the network. For AS, the action is to mark the out-of-profile packets. When congestion occur, packets marked “OUT” are dropped first by means of a congestion control mechanism called *Partial Buffer Sharing* (PBS). For GS, the chosen policing is to shape the traffic at the edge router and to drop out of profile GS packets.
Scheduling. Scheduling is different for AS and GS packets: GS scheduling is implemented by a *Priority Queuing* (PQ) mechanism ; the remaining bandwidth is shared by a *Weighted Fair Queuing* (WFQ) between AS and BE traffic.

2.4 @IRS Platform Configurati on and Performance Measurement Results

The @IRS architecture has been implemented over a national IPv6 platform (Fig. 2).

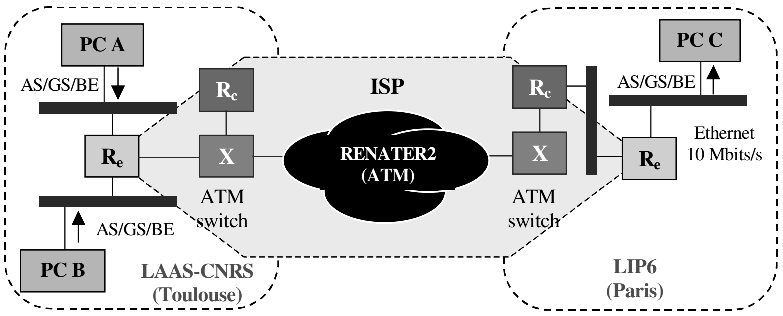


Fig. 2. Platform configuration (2 sites on 7)

Platform configuration. Seven local platforms[7] have been connected by an edge router (R_e) to an *Internet Service Provider* (ISP) represented by the national ATM RENATER2 platform. By means of its edge router, each site was then provided with an access point to the ISP, characterized by a traffic contract (equivalent to the *Service Level Agreement* of [BLA98]). It was the edge router's responsibility to implement the SLA as it introduced flows within the ISP. Bandwidth of the link connecting sites to the ISP (via a CBR ATM Virtual Path) was such that the maximal throughput provided at the UDP level is 107 Kbytes/s for 1024 bytes length packets.

Edge and core routers were configured with the following hypothesis:
– the max amount of GS (resp. AS) traffic that could be introduced by the edge router was 20 Kbytes/s (resp. 40 Kbytes/s), i.e. 20% (resp. 40%) of the link bandwidth;
– the rate control applied by the core router was 100 Kbytes/s;
– the weights associated to the AS and BE scheduling (WFQ) were resp. 0.5 and 0.5.

Performance measurements: results and analysis. The goal of the performance measurements exposed hereafter was to answer the following question: given that a DiffServ-oriented solution was implemented on the tested platform[8], would it be possible to observe a *reproducible* per flow QoS (for AS and GS flows)?
Measurements have been realized in order to evaluate the QoS provided to several UDP flows served in AS or GS whose number and load were varying. For experiment sessions (about 300 seconds), measured parameters were (1) the loss rate and (2) the minimal, maximal and average values of the transit delay together with the transit delay distribution[9]. A complete description of the three scenarios may be found in [14].
All the measurements have been done between two sending hosts (BSD PCs) located at Toulouse, noted A and B in Fig. 2, and one receiving host located at Paris, noted C. Three scenarios have been defined. Each time, in order to test the "worst" case, a BE traffic used the totality of the link bandwidth:
– the first scenario aimed at validating the impact of the number of AS flows on the AS QoS, when the network was overloaded; no GS flow was generated;
– the second scenario aimed at validating the impact of the number of GS flows on the GS QoS, when the network was overloaded; no AS flow was generated;
– the third scenario aimed at validating the impact of the number of AS (resp. GS) flows on the GS (resp. AS) QoS, when the network was overloaded; AS and GS flows were generated together.
Results (that are partially exposed in Fig. 3) have allowed one to conclude that:
– the impact of the number of GS flows on the AS or GS QoS was weak;
– the impact of the number of AS flows was similar but it might be discussed a little more. Indeed, if AS QoS was almost unchanged for about 90% of the traffic, 10% of the packets had a delay slightly increased. Whereas no solid explanation has been

[7] Only two local platforms are represented in Fig.2 (LAAS and LIP6 platforms).
[8] Let us keep in mind that a DiffServ-oriented QoS approach in the core network is applied on IP packets coming from several flows.
[9] The study has been done for a given and unchanged configuration of the platform (size of routers queues, WFQ weights, etc.). Indeed, our goal was neither to evaluate the impact of the platform configuration, nor its topology.

given, this result is acceptable with regard to the AS QoS specification; moreover, it is particularly important for the characterization of an AS-like service on a DiffServ platform like the @IRS one: indeed, a strong impact would have been made difficult such a characterization (described hereafter in section 3).
Let us precise that these results are not representative of an exhaustive study. Such a study has been performed in simulation (based on *ns*-2), our goal being also to evaluate the impact of the network topology. This work has not yet been published.

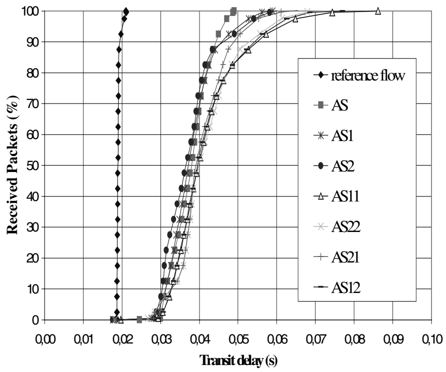


Fig. 3(a). Results of scenario 1

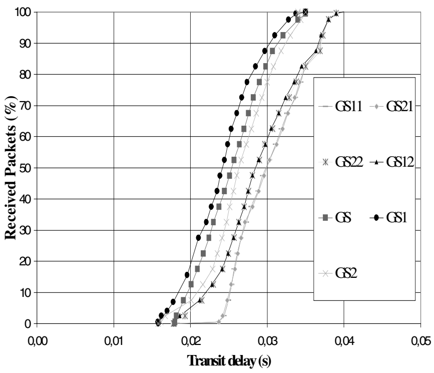


Fig. 3(b). Results of scenario 2

3 Towards an Architecture Integrating Transport and IP QoS

3.1 Characterization of the QoS Resulting from Transport and IP Services

Starting from the previous results, the goal of this section is to propose a characterization of the QoS resulting from the coupling between a DiffServ IP service (such as GS or AS) and a Transport protocol allowing n retransmissions ($FPTP_n$, $n \geq 0$). Two specific QoS parameters are targeted: the loss rate and the transit delay between two points (e.g., two edges routers), and for a given state of the network (e.g., a state of congestion).

Characterization of the QoS for a GS-based service. From the previous performance measurements results, it comes that the GS service may be characterized (per flow) by a total reliability and an end-to-end maximal transit delay. As this service has to provide a total reliability, the interest of its coupling with another Transport service than the ones provided by UDP or $FPTP_0$ is not obvious (except when error bits occur). In the following, we'll do the hypothesis that GS is always associated with UDP or $FPTP_0$.

Characterization of the QoS for an AS-based service. The characterization of the loss rate and the transit delay resulting from the coupling of an AS service together with a Transport service allowing n retransmissions has to be discussed a little more.

Loss rate characterization. *Let:*
– ε be the estimated percentage of AS packets which are lost by routers congestions;

– $\mathbf{r_n}$ be the estimated percentage of Application data units transferred between the two considered points), *n* designing the number of Transport level retransmission(s).
For n = 0, i.e. without Transport level retransmission: $r_0 = 1 - \varepsilon$
For n = 1, i.e. after one retransmission: $r_1 = (1 - \varepsilon) + \varepsilon \,.\, (1 - \varepsilon) = 1 - \varepsilon^2$
For n = 2, i.e. after two retransmissions: $r_2 = (1 - \varepsilon) + \varepsilon \,.\, (1 - \varepsilon) + \varepsilon \,.\, [\, \varepsilon.(1 - \varepsilon)\,] = 1 - \varepsilon^3$
Finally, r_n is defined by: $\mathbf{r_n = 1 - \varepsilon^{n+1}}$

End-to-end delay transit characterization. Let **f(t)** represent the percentage of AS packets which are received without retransmission with an end-to-end transit delay less than or equal to *t*. This function directly depends on:
– the DiffServ domain configuration, i.e. queue sizes, weights of WFQ, etc.
– the path between the sending and receiving hosts;
– the load of the network on the considered path.
However, it comes from the previous section (2.4) that f(t) is sufficient enough to statistically characterize the AS service between two points on a given path and for a certain state of the network[10] (in our measurements, a state of congestion).
Let $\mathbf{f_{n,T}(t)}$ be the function representing the delay distribution after *n* Transport level retransmissions. For n = 1, the relation between $f_{1,T}(t)$ and f(t) is: $f_{1,T}(t) = f(t) + \varepsilon \,.\, f(t - T)$
- T being the value of the TPDU[11] retransmission timer (supposed to be constant).
For n = 2, $f_{2,T}(t)$ is defined by: $f_{2,T}(t) = f(t) + \varepsilon \,.\, f(t - T) + \varepsilon\, [\, \varepsilon \,.\, f(t - 2.T)\,]$
Finally, $\mathbf{f_{n,T}(t)}$ is defined by: $f_{n,T}(t) = f(t) + \varepsilon \,.\, f(t - T) + \ldots + \varepsilon^n \,.\, f(t - n.T)$

i.e. $$\mathbf{f_{n,T}(t)} = \sum_{i=0}^{n} \boldsymbol{\varepsilon^i \,.\, f(t - i.T)}$$

Let us now have a look at the use of this characterization in order to select a couple (Transport service, IP service) able to match a QoS request expressed for an end-to-end channel by means of the QoS parameters and semantics of the @IRS API.

3.2 Service Selection Algorithm

In order to simplify the following explanations, we'll consider that the QoS request is expressed by means of only two QoS parameters: (τ_d, [a = 0, b]) and τ_r and that the semantics are reduced to the absolute one (A) and the average one (M).
Let an application be requiring a given QoS for one of its flows between two sites noted S_A and S_B. Let us also suppose that:
– UDP, $FPTP_n$ and TCP are the available protocols at the Transport level;
– GS, AS, BE are the available services at the IP level;
– when the network is in state of congestion (supposed to be the "worst" case)
  ▪ GS is characterized between S_A and S_B by a max delay = t_0 and a loss rate = 0;
  ▪ AS is characterized between S_A and S_B by $f_n(t)$ and r_n.

[10] Let us recall here that this assertion (hypothesis on which is based the following of this work) is currently studied in simulation (*ns*-2), our goal being to identify its limits, particularly with regard to the network topology.
[11] TPDU: Transport Protocol Data Unit.

Algorithm description. To know if the QoS request may be satisfied, the following algorithm must be applied.

) Verify the request *coherency* and then:
- in case of incoherency, reject the request;
- in case of coherency:
) Choose a couple (Transport / IP)
    - if the IP service = AS, evaluate if it allows to satisfy the QoS request
        - if the evaluation fails, choose another couple (that necessarily exists);

) if the IP service = GS or AS then perform an admission control
    - if the control fails, return to the previous point and choose another couple;
    - in case of failure for all couples, reject the request.

Request coherency verification. The request is *coherent* when:
– the packet rate (τ_r) the application wishes to receive is higher than the packet rate the application wishes to receive with a specific transit delay (τ_d), i.e: $\mathbf{0}\quad {}_{\mathbf{d}}\quad {}_{\mathbf{r}}\quad \mathbf{1}$
– the maximal transit delay *b* is greater than or equal to the GS end-to-end transit delay, i.e: $\mathbf{b \geq t_0}$.

Choice of a Transport / IP couple. Once done this coherency verification, the choice has to be done of an IP and a Transport services are chosen allowing for the satisfaction of the QoS parameters together with their semantics of guarantee.
This choice is based on the consultation of a knowledge base (see Fig. 4) that provides, for all possible configurations of (τ_r, σ_r) and (τ_d, σ_d), one or many couples (Transport / IP) allowing (a priori) for the satisfaction of the request. The way this base is initialized is described at the end of the section.

	$\tau_r = 0$	$0 < \tau_r < 1$			$\tau_r = 1$
$\tau_d = 0$	UDP/BE(*)	FPTP$_n$/BE(*)			TCP/BE(*)
$0 < \tau_d < 1$	∅	σ_d \ σ_r	A	M	UDP/GS(*)
		A	UDP/GS	UDP/GS	
		M	UDP/GS	C	
$\tau_d = 1$	∅	∅			UDP/GS(*)

Fig. 4. Knowledge base ("∅": incoherent request - (*): whatever the semantic)

When several couples are possible (case C in the table of Fig. 4), then the "cheapest" one is selected, the "cost" of a couple being defined as it follows:
– at the IP level: GS cost > AS cost > BE cost
– at the Transport level: TCP cost > FPTP$_n$ cost > UDP cost
– the choice is done so as to minimize first the IP cost (e.g.: FPTP/AS < UDP/GS).
For each choice, there are three possibilities depending on the IP service:
– if the IP service is BE, the choice has no more to be discussed; it is retained;
– if the IP service is GS, the choice is retained if the admission control is positive;

– if the IP service is AS, the choice has to be studied a little more so as to verify if it matches the QoS required for the parameters τ_d and τ_r.

Evaluation of the choice (Transport / IP) when the IP service is AS.

Let $\mathbf{n_r}$ be the minimal retransmissions number allowing for the satisfaction of τ_r

=> n_r must verify: $\tau_r = 1 - \varepsilon^{nr+1}$

It then results that: $\mathbf{n_r = Ent\ [\ (\ ln\ (1 -\ \ {}_r) / ln(\varepsilon)\ ) - 1\ )\ ]}$

- with: ln is the logarithm function and Ent is the *entire part* function (e.g. Ent(0.5)=0).

Let now $\mathbf{n_d}$ be the maximal retransmissions number allowed by the *b* parameter

=> n_d must verify: $n_d \,.\, T + t_0 = b$

It then results that: $\mathbf{n_d = Ent\ [\ (b - t_0) / T\ ]}$

- with: T is the value of the TPDU retransmission timer (supposed here to be constant).

In order to satisfy τ_d, the question is: $\exists?\ n \in [0, n_d] / f_n(b) \geq \tau_d$

If *n* does not exist, the couple (* / AS)[12] has to be rejected (because τ_d cannot be satisfied)

else | if $n_d < n_r$ then the couple (* / AS) has to be rejected (τ_r cannot be satisfied)
| else the choice is retained and the number of retransmissions is $p = \max [n, n_r]$.
| In other words, the choice is AS/FPTP$_p$ with $p = \max [n, n_r]$[13].

Initialization of the Knowledge Base

The initialization of the knowledge base depends on:

– the available services, both at the Transport and at the IP levels;

– the set of definition of parameters and semantics, e.g.:
  - for τ_r and τ_d: the interval [0,1] divided into three subsets : 0,]0,1[and 1;
  - for σ_r et σ_d: A and M.

For each (τ_r, σ_r) / (τ_d, b, σ_d) possible combination, all the solutions allowing for the satisfaction of each couple separately (i.e. (τ_r, σ_r) on one side, (τ_d, σ_d) on the other side) have been identified, but only the solutions allowing for the satisfaction of the two couples have been kept in the table. Let us illustrate this by way of a simple example. Consider the following combination: ($0 < \tau_r < 1$, $\sigma_r = A$) / ($0 < \tau_d < 1$, [a, b], $\sigma_d = M$) :

– two solutions are possible for the request related to the couple (τ_r, σ_r):
  - * / GS, where * designates all possible Transport services ;
  - FPTP$_\infty$ / *, where * designates all possible IP services ("∞" designating that there is no limit on the retransmission number);

– three solutions are possible for the request related to the couple (τ_d, b, σ_d):
  - UDP / AS if $\tau_d \leq f(b)$
  - FPTP$_n$ / AS if $\exists n / \tau_d \leq f_{n,T}(b)$
  - UDP / GS if $b \geq t_0$ (condition verified in the coherency test)

The single solution satisfying the two couples is then UDP/GS.

[12] * designating either UDP or FPTP$_n$.

[13] If $n = n_r = 0$, the choice between UDP/AS and FPTP$_0$/AS depends on the "partial order" QoS parameter.

Let us now look at the implementation and the test the algorithm for an actual multimedia application (a videoconferencing system).

3.3 Implementation and Experimental Tests

In addition to the service selection algorithm, two QoS management protocols have been specified and implemented (see Fig. 5). They allow the channel set up and the acceptation or reject of a QoS requested for the considered channel (including the service selection algorithm).

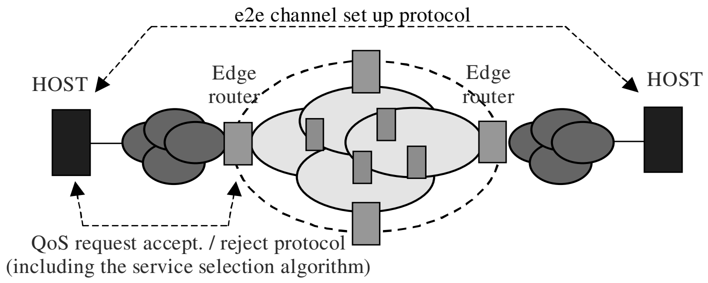

e2e channel set up protocol
HOST
Edge router
Edge router
HOST
QoS request accept. / reject protocol
(including the service selection algorithm)

Fig. 5. QoS management protocols (before the data transfer phase)

Due to space limits, these protocols are not described here and the focus is only done on the service selection algorithm.

Implementation of the service selection algorithm. In order to minimize the storage buffer of the points representing the distribution of the transit delay of the AS packets (i.e. f(t)), a model based on a trigonometric function has been adopted:

$g(t) = A_3+A_1.htan(t.A_2-A_4)$ where *htan* designates the hyperbolic tangent function

Calculation of the (A_1, A_2, A_3, A_4) parameters is performed by a Matlab program. From a discrete set of points, this program allows to deduce the best approximation of f(t) for the chosen model.

Parameters given in the example illustrated in Fig. 6 provide the model g(t) of the function f(t) between Toulouse and Paris. It has been deduced from the measures performed on the @IRS platform (whose results are given in section 2.4).

Test specification. The networking platform over which different QoS requests of the application have been tested has been simplified; it only includes two hosts directly connected to each other by an Ethernet link (100Mbits/s):

– the service selection mechanism is implemented on the sending host;
– the admission control is supposed to be positive for each request;
– the service characterization file (on the sending host) contains the following information. Between the sending host and 140.93.200.33, for GS, t_0=18ms; for AS: g(t) parameters are: A_1=48.77; A_2=162.12; A_3=51.54; A_4=6.37 and ε=0.

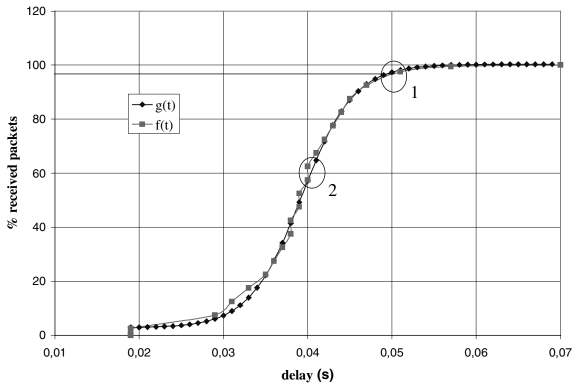


Fig. 6. Distribution of the AS packets transit delay: f(t) vs. g(t)

Three different QoS requests have been tested :
– in first case, the user wants a perfect quality both on the audio and video channels; for both channels (audio and video), this quality may be expressed by: τ_r=1, σ_r=A and τ_d=1, b=40 ms, σ_d=A;
– in the second case, no QoS is required from the user; for both channel, this quality may be expressed by: τ_r=0 and τ_d=0 (no guarantee semantic is specified)
– in the third case, the user may tolerate some possible degradations on the video but wants a sufficient enough quality on the audio so as to understand what is spoken about; this quality may be expressed as it follows:

- for the video channel: τ_r=60%, σ_r=M and τ_d=60%, b=50ms, σ_d=M
- for the audio channel: τ_r=90%, : σ_r=M and τ_d=90%, b=40 ms, σ_d=M

Results and analysis. Here are the Transport and IP services resulting from the automatic services selection for each of the three previous requests (Fig. 7).

	1st case	2d case	3d case
Audio channel	UDP/GS	UDP/BE	UDP/GS
Video channel	UDP/GS	UDP/BE	UDP/AS

Fig. 7. Results of the automatic services selection

Let us verify if this these results are correct:

In the first case, the maximal value required on the transit delay (b = 40 ms) is greater than t_0. As the required guaranty is the absolute one (A), the communication system must select UDP/GS for both channels;

In the second case, as no constraint has been expressed, the communication system must select UDP/BE for both channels (cheapest cost);

In the third case, for both video and audio channels, the question is (see section 3.2.3):

$$\exists?\ n \geq 0 \ /\ f_n(b) \geq \tau_d \text{ with } 0 \leq n \leq n_d = \text{Ent}\ [(b - t_0) / T]$$

T being the retransmission timer value (greater that the minimal RTT, i.e. $2.t_0$)

– for the video channel: b and t_0 values implicate that n=0 (no retransmission); in parallel, the point noted 1 in Fig. 6 indicates that about f(0.050)=98% (more than τ_d=60%) of the packets are estimated to be received with a transit delay less than 50ms. The system must then select UDP/AS ;
– for the audio channel: b and t_0 values implicate that n=0 again; in parallel, the point noted 2 in Fig. 6 indicates that only f(0.040)=60% (less than τ_d=90%) of the packets are estimated to be received with a transit delay less than 40ms. The system cannot select UDP/AS and must the select UDP/GS.

4 Conclusion and Future Work

From the recent evolution of the Internet QoS-oriented communication services and protocols, it comes an important need in a clear definition of a generic architecture integrating all the new solutions both at the IP level and at the Transport level. The work exposed in this paper consists in a particular instantiation of such an architectural framework. Taking into account the emerging traffic engineering-based QoS solutions, the targeted problem concerns the integration of a PO/PR Transport architecture together with these solutions. Starting from performance measurements performed over a DiffServ national platform, contributions exposed here deal with the proposition and the implementation of an end-to-end communication architecture providing guaranteed end-to-end QoS, and allowing the application programmers to be masked with the complexity of the underlying protocols and mechanisms. Our efforts have been made on (1) the definition of the services provided to the application layer (including QoS parameters and semantics of guarantee), and (2) the conception, the implementation and the test of a mechanism allowing the application programmers to be masked with the choice of the underlying new Transport and IP services when using the communication system. The major perspective currently under development tackles a larger problem related to the interconnection of several DiffServ domains (i.e. a multi domain context).

References

1. P. Amer, C. Chassot, T. Connolly et al, “Partial Order Transport Service for multimedia and other application”, IEEE ACM Transaction on Networking, vol.2, n°5, 1994.
2. C. Chassot, A Lozes, M. Diaz. “From the partial order concept to partial order connection”, Journal of High Speed Network, vol5, n°2, 1996.
3. P. Owezarski, M. Diaz, C. Chassot, “A Time Efficient Architecture for Multimedia Applications”. IEEE JSAC, April 1998, vol.16, n°3, pp 383–396.
4. S. Blake, D. Black, M. Carlson, “An Archi. for Differentiated Services”, RFC 2475.
5. P. Trimintzios, et al. "A management and control architecture for providing IP differentiated services in MPLS-based networks". IEEE Communication Magazine, Vol. 39, n°5, May 01.
6. P. Trimintzios, P.Flegkas, G. Pavlou et al. "Policy-based network dimensioning for IP differentiated services networks". IPOM’02, Dallas, USA, Nov 2002.
7. G. Cristallo, C. Jacquenet. "An approach to inter-domain traffic engineering". XVIII World Telecom. Congress (WTC 2002), Paris, France, Sept. 2002.

8. A. Campbell, G. Coulson, D. Hutchinson, “A QoS architecture”, ACM Computer Communication Review, 1994.
9. K. Nahrstedt, J. Smith. “Design, Implementation and experiences of the OMEGA end-point architecture”, IEEE JSAC, vol.14, 1996.
10. J. Heinanhen, F. Baker et al. “An Assured forwarding PHB”, RFC 2597.
11. P. Sénac, E. Exposito, M. Diaz. “Towards a new generation of generic transport protocols”, IWDC 2001., Taormina, Italie, Sept. 2001.
12. V. Jacobson, K. Nichols et al. “An Expedited Forwarding PHB”, RFC 2598.
13. F. Garcia, et al. “Conception, implementation and evaluation of a QoS based architecture for an IP environment supporting differentiated services”, IDMS, Lancaster, UK, Sept. 2001.
14. C. Chassot, F. Garcia, G. Auriol et al. “Performance Analysis for an IP Differentiated Services Network”, ICC02, New York, USA, Mai 2002.

Achieving Relative Differentiated Services Using Proportional Probabilistic Priority Scheduling on Network Processor

Chee-Wei Tan and Chen-Khong Tham

Department of Electrical and Computer Engineering,
National University of Singapore, Singapore 119260
cheewei@alumni.nus.edu.sg, eletck@nus.edu.sg

Abstract. This paper studies the design and performance of the Probabilistic Priority (PP) packet scheduling algorithm to schedule packets. Unlike an earlier design that uses fractional arithmetic and prohibits large number of classes, we present an integer PP algorithm and show that PP is a special scheme of applying lottery scheduling to bandwidth allocation in a strict priority sense. We then propose a Multi-winner PP (MPP) scheduler using multi-winner lottery scheduling to improve the throughput and response time accuracy and a flexible ticket transfer algorithm to improve the deadline violation probability in probabilistic scheduling. Finally, we investigate the issue of parameter assignment for an MPP scheduler and use our techniques to implement a prototype Assured Forwarding (AF) mechanism in a network processor.

1 Introduction

In the Differentiated Service (DiffServ) architecture, individual flows with similar Quality-of-Service (QoS) requirements are aggregated, and given the same treatment as described by a Per-Hop-Behavior (PHB) in terms of QoS metrics such as average packet delay, packet loss and jitter. The routers do not keep per-flow states and there are no complex resource signaling mechanism involved [1]. The Assured Forwarding (AF) PHB guarantees only that the assured traffic is delivered with a higher probability than the best-effort traffic; in the case of severe network congestion, the assured traffic can still experience severe losses and high delay. This paper analyzes the Probabilistic Priority (PP) scheduling discipline within the framework of relative service differentiation. PP adopts a probabilistic relative service model. At every service round, each class takes a bid. Since higher priority classes have higher probabilities associated with them, in the long run, they will be served more often than lower priority classes. As compared to Strict Priority (SP), this increases fairness among classes and prevent the starvation of lower priority classes. We first show that PP is a cross application of lottery scheduling in a strict priority sense to provide proportional bandwidth sharing among classes. This in turn allows us to benefit from numerous techniques presented in [7,8] to control PP. The lottery and stride scheduling algorithms are very well-known scheduler for statistical allocation of CPU resources [7,8]. Lottery scheduling randomizes resource allocation among clients whose shares of resources are represented by tickets using policies such as ticket inflation and deflation. An allocation is performed by holding a lottery, and the resource is granted to the client with the winning ticket. Multi-winner lottery scheduling is a variant of lottery scheduling that produces better throughput accuracy for many workloads. Based on this multi-winner concept, we formulate a multi-winner PP algorithm to improve the response-time variability of PP. As lottery scheduling is effectively stateless, a great deal of complexity is removed in comparison to other proportional schedulers. The feasibility of using lottery scheduling in packet forwarding has been analyzed in [2,4,9] but no work has been done to address its weaknesses at the packet level due to its probabilistic nature. The probabilistic relative service model is only suitable for applications that are able to tolerate deadline violations of a few packets. We propose a technique that is analogous to the idea of dynamically-controlled ticket transfer which has been applied to graphics rendering and Monte-Carlo tasks [8] to address this problem.

The rest of the paper is organized as follows. We propose an efficient integer PP algorithm in section 2 and show that PP is indeed a cross application of lottery scheduling. We use the multi-winner concept to generalize PP to improve its throughput accuracy and reduce its response-time variation. We present a technique based on flexible ticket transfer to reduce the deadline violation probability in times of congestion. In Section 3, we investigate parameter assignment and propose a framework to implement Assured Forwarding. Our implementation is described in Section 4. A performance study on a network processor-based router is presented in Section 5. We conclude in Section 6.

2 Probabilistic Priority Scheduler

2.1 Basic PP Integer Algorithm

The work conserving PP Scheduler is based on the SP scheduler with each queue being assigned a probability p_i [4]. By appropriate setting of a parameter $p_i \in [0, 1]$, $i = 1, \ldots N-1$ and $p_N = 1$ in a multi-class system, a class is selected with a probability corresponding to equation (1) for service at every cycle. A class parameter of $p_i = 1$ means that the class i definitely gets served when polled if all higher priority classes are empty or not selected during the cycle. Hence PP reduces to SP when $p_i = 1.0$, $i = 1, \ldots N$. Here, we derive an integer algorithm and show that it is indeed a cross application of lottery scheduling in the strict priority sense.

First, consider a multi-class system of N priority levels with the highest priority level denoted by 1. Let us define the weight of class i to share the server [4] as

$$r_i = p_i \prod_{j=1}^{i-1} (1 - p_j) \qquad (1)$$

Without loss of generality, assume that all classes in the system are busy so that the normalized weight of class i among all classes is

$$\hat{r}_{i\in\Omega} = \frac{r_i}{\sum_{j\in\Omega} r_j} \tag{2}$$

where Ω consists of all queues, i.e. $\{1,\ldots,N\}$. After rearranging all r_i such that they share a common denominator of lowest common multiple, we have

$$\hat{r}_{i\in\Omega} = \frac{x_i}{\sum_{j\in\Omega} x_j} \tag{3}$$

where x_j is the numerator of the normalized relative weight $\hat{r}_i$. It is easy to see that this will also be true for all network conditions:

$$\hat{r}_{i\in BQ} = \frac{x_i}{\sum_{j\in BQ} x_j} \quad , BQ \in \Omega \tag{4}$$

where BQ is the set of non-empty queues in Ω. For example, $BQ = \{1,2\}$ which denotes non-empty queue 1 and 2 can be found in Ω. The total number of possible network conditions, i.e. the permutation of non-empty queues in Ω, is equal to $2^N - 1$ but the most interesting set would be the total number of possible network conditions with more than one non-empty queue which is equal to $M = \sum_{i=1}^{N-1}\sum_{j=1}^{N-i}\binom{N-i}{j} = 2^N - N - 1$. We now have numerator x_i to calculate $\hat{r_i}$ without having to store in advance $\hat{r_i}$ for all possible combinations of empty and non-empty queues with each combination corresponding to a particular instance of Ω. This effectively removes both the need for fractional arithmetic in recalculation of network states whenever p_i changes dynamically and the restriction for a small set of all possible network states. The integer algorithm of PP works *without* the need for *a priori* network state computation. PP is analogous to having sets of different numbers of tickets that are present in a service round with each set corresponding to one of the network conditions in M.

2.2 Multi-winner PP (MPP) Integer Algorithm

Multi-winner lottery scheduling is a generalization of the basic lottery scheduling technique that produces better throughput accuracy and smaller response-time variation [8]. Instead of selecting a winner per round, N_w winners are selected with only the first winner being randomly selected and each winner is guaranteed the use of the resource for one quantum. The set of N_w consecutive quanta allocated by a single multi-winner lottery is referred to as a super-quantum. Due to the probabilistic nature of PP, the highest priority class can exhibit substantial variability over small time scales which can cause its HOL packet to miss its deadline if sufficient numbers of service round are given to its lower priority classes instead. At worst, this may cause buffer overflow and incoming high priority packets to be dropped. This necessitates incorporating a deterministic mechanism in PP to achieve predictable behavior at small time scales. We use the multi-winner concept to extend the original PP integer algorithm. In this paper, we use a fixed value of $N_w = 20$. The ordering of the winners in MPP is based on a fixed permutation that goes in a round robin fashion, starting from the first winner and followed by its immediate lower priority class. This integer algorithm requires a total of $2^N - 1$ uniform distributions of integer random numbers for N classes. This is analogous to the total number of tickets differing in every service round of lottery scheduling. Waldspurger *et al.* [7] provides a multiplicative linear congruential Park-Miller pseudo random number generator in MIPS assembly language code but we use a generic algorithm *U-map* described in section 4 to scale uniform distributions without using multiplication assembly language instructions. In our algorithm, each super-quantum is reset back to 0 when the network condition changes which would happen very often if the system is highly loaded. This implies that MPP is able to reduce the throughput error and response-time variability. Through extensive simulations under heavy load conditions, we observe that the super-quantum is reset on an average of about 85% of the total time. Hence N_w does not have a significant impact on the reduction rate of throughput error. The advantage of MPP over PP appears to be small for 8 classes but by keeping the number of classes small, we can increase the number of winners to provide stricter throughput guarantees within a class.

2.3 Flexible Ticket Transfer Algorithm

In the previous section, we described an extension of PP to achieve throughput guarantee. In this section, we aim to reduce the time given up to the lower priority classes by the higher priority classes ("slack" in probabilistic scheduling) by setting a rate of approaching strict prioritization using the relationship between delays of different classes. In particular, we use the following propositions of average delay of class i, $\overline{W_i}$ proven in [6] to affect p_i.
(1) As $p_j \uparrow [0 \rightarrow 1]$[1] for $j < i$, $\overline{W_i}$ is continuously and monotonically increasing.
(2) As $p_i \uparrow [0 \rightarrow 1]$, $\overline{W_i}$ is continuously and monotonically decreasing.
(3) As $p_j \uparrow [0 \rightarrow 1]$ for $j > i$, $\overline{W_i}$ is nearly constant under congested network conditions.
Let us define the initial parameter r_i for class i that satisfies the relationship $r_1 \geq r_2 \cdots \geq r_i \geq \cdots \geq r_N$ for the multi-class system where Class 1 is the highest priority class. Such assignment means that the probability of higher priority class is larger. This algorithm consists of the following two steps. The first step is to reduce the probability of a lower priority class after it has been served by transferring some probability to its immediate higher priority class. Note that the transfer of tickets from the class served to its immediate higher priority class will create a snowball effect that will cause the highest priority class to be eventually served while still using probabilistic scheduling. The second step is to preserve as much as possible the priority allocation that is defined at the start of

[1] Following [6], the notation "$x \uparrow [0 \rightarrow 1]$" means "$x$ increases from 0 to 1".

the algorithm by transferring probability starting from the lowest priority class even though it has not been served to the immediate higher priority class of the class being served if the first step persists. Eventually the class that continuously gets served will lose its bid after the probabilities of all lower priority classes have been depleted.

Table 1. Outline of ticket transfer algorithm

At each service round, suppose classes 1 to L, corresponding to a particular network condition $BQ \in M = 2^N - N - 1$ where N is the total number of classes, are busy,

1. If class i, $1 < i \leq L$, gets served, then $r_i^{'} = max\,(r_i - \triangle_i, r_{i+1})$, and $r_{i-1}^{'} = min\,(r_{i-1} + \triangle_i, 1.0)$, such that $r_i^{'} \geq r_{i+1}$, i.e. transfer $\triangle_i$ of probability being served to the immediate next higher priority level with $r_i \neq 0$.

2. If $r_i = r_{i+1}$, then $r_k^{'} = (r_k - \triangle_k)^+$, $i < k \leq L$ where k is the lowest priority class in BQ that satisfies $r_k \neq 0$, and $r_{i-1}^{'} = min\,(r_{i-1} + \triangle_k, 1.0)$, i.e. transfer $\triangle_k$ of probability being served to the immediate next higher priority class $i - 1$.

3. If the highest priority class is served or the network condition BQ changes, $r_i^{'} = r_i$, i.e. reset all class parameters back to their original r_i.

From the algorithm shown in Table 1 and equation (1), we can make the following propositions:
(a) If $p_{i+1} < \frac{p_i}{1-p_i} \leq 1$ and $\triangle_i$ of probability to be served is transferred from class i to class $i-1$, $\hat{p}_i$ decreases, $\hat{p}_{i-1}$ increases, and $\hat{p}_j$, $j \neq i, i-1$ remains constant.
(b) If $p_{i+1} = \frac{p_i}{1-p_i} \leq 1$, and $\triangle_k$ of probability to be served is transferred from class k, $i < k \leq L$ to class $i-1$, $\hat{p_j} \uparrow \left[p_j^{orig} \rightarrow 1\right]$, $j \leq i$ where p_j^{orig} is the original PP parameter of class j.
Proposition (a) states that only the probabilities of the class served and its immediate higher priority class will change while the other classes will maintain the original PP configurations at the initial stages after the algorithm begins while proposition (b) states that higher class priority will approach the configuration of SP, i.e. $\hat{p}_j \rightarrow 1$, $\hat{p}_j \neq 0$, $1 < j \leq i$ if the situation where the highest priority class HOL packet is not served while class i is constantly being served persists. Therefore, from proposition (1) and (2), the average delays of classes with higher priorities than class i will decrease monotonically over time while those classes with lower priorities than class i will increase monotonically over time. We introduce an additional parameter $\triangle_i$ to provide a dynamic feed-forward mechanism based on the current workload or the slack of the corresponding high priority HOL packet. This user-tunable class parameter $\triangle_i$ can be a function of the class's burstiness or the higher priority classes' backlog. It provides a way for the original PP to approach SP in a configurable length of time so that the HOL packet of higher priority classes will not exceed its deadline unnecessarily.

2.4 Simulation Studies

In this section, we consider scenarios with high traffic loads and tight deadlines for each class. For each class, we use Long range dependent (LRD) traffic modeled as Pareto On-off processes with shape parameter 1.3 since aggregated traffic in real DiffServ networks is LRD in nature. The mean service time is taken to be the unit of time and the service times of packets in each class follow the same exponential distribution with mean 1.0 units. Results are averaged over 10^6 time unit simulation windows unless otherwise indicated. Throughout this paper, we use λ_i and ρ_i to denote the arrival rate and traffic intensity of class i respectively. In Table 2, the arrival rates for all classes are the same, i.e. $\rho_i = 0.125$ so the system is not overloaded, i.e. $\rho = 1.0$. Each class has the same parameter i.e. $p_i = 0.6$, $i \neq N$. To compare the performance between the various schemes, we use deadline violation probability in Table 2 as a performance metric. The deadlines for class 1 to N, where $N = 8$, are arbitrary selected as 11, 16.5, 22, 27.5, 33, 38.5, 44, and 49.5 time units respectively. The probability transfer quantum is the same for all classes, i.e. $\triangle_i = min\,(0.15, r_i)$.

Table 2. Comparison of (a)deadline violation probabilities (%) and (b)average delay (time units) under full utilization condition

	PP/Lottery	MPP	MPP w/ ticket xfer	SP
Class 1	0.114	0.069	0.036	0.000
Class 2	0.172	0.082	0.068	0.010
Class 3	4.297	1.680	0.646	0.410
Class 4	23.513	17.883	11.451	9.647
Class 5	34.696	27.678	21.410	14.609
Class 6	57.679	45.020	35.453	27.650
Class 7	91.627	88.020	64.952	58.243
Class 8	94.831	90.671	67.316	100.000

	PP/Lottery	MPP	MPP w/ ticket xfer
Class 1	1.350	1.170	1.201
Class 2	1.990	1.460	1.450
Class 3	4.920	3.550	3.471
Class 4	55.290	45.640	37.400
Class 5	198.490	214.620	120.810
Class 6	555.440	402.180	240.080
Class 7	6719.060	3726.320	1973.990
Class 8	22243.940	19847.430	7240.370

Results in Table 2 indicates that ticket transfer algorithm does not have an adverse effect on low priority class though it discriminates against them by allowing high priority classes to be selected as fast as possible. In addition, it also suggests that this mechanism improves deadline violation probability of low priority classes as opposed to intuition which we investigate next.

We now consider the ticket transfer algorithm used in a 4-class system to evaluate its effectiveness. Each class has parameter $p_1 = 0.5, p_2 = 0.55, p_3 = 0.6$ and $p_4 = 1.0$. Note this parameter assignment provides lower priority classes with higher probabilities of being serviced than in previous simulations. Fig. 1 shows the probabilities of all possible network conditions occurring in the system for SP, PP and MPP with ticket transfer schedulers at both short (10^3 time units) and long timescales (10^6 time units) with respect to packet service times under different loads. Each network condition is binary-coded as follows: bit 0 corresponds to the highest priority class, class 1 hence 0101B implies that only class 1 and 3 are present. Note that the network condition is a function of offered loads and scheduling mechanism. We also compare the Pareto on-off traffic model with the token bucket-constrained traffic model with a bucket depth of 17 time units which exhibits short bursts.

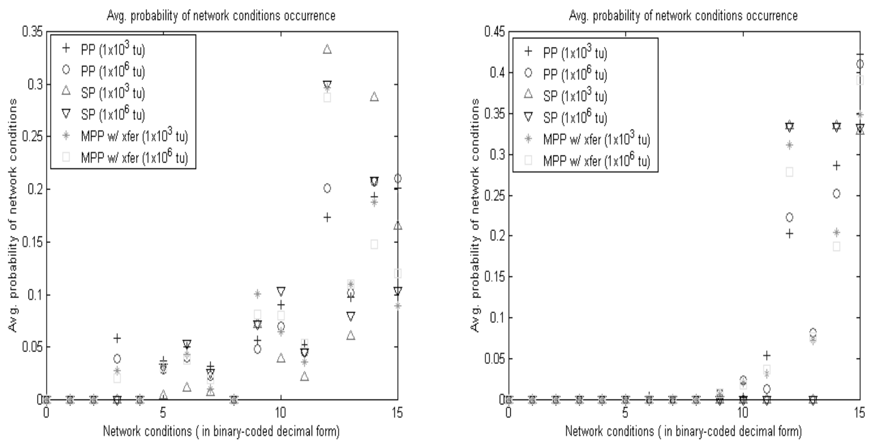


Fig. 1. (a) Comparison of network condition probabilities between PP, SP and MPP with ticket transfer scheme under light load (b) Comparison of network condition probabilities between PP, SP and MPP with ticket transfer scheme under heavy load

Note that, in contrast to intuition, the deadline violation probability of the lowest priority class is improved significantly when the ticket transfer algorithm is used because higher priority classes are assured to get transmitted within short timescale and this implies that the probability of network conditions containing these high priority classes occurring within a longer time frame will be smaller than that in comparison to normal PP scheduling. From Fig. 1 and Fig. 2, we can make the following observations:

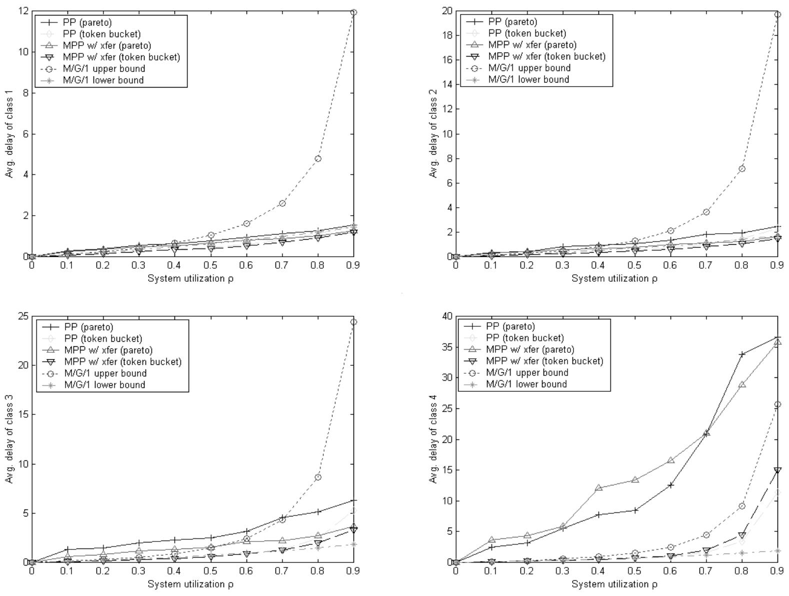


Fig. 2. Average queueing delay under different traffic loads using Pareto on-off and token bucket filter constrained traffic

– We found that MPP with ticket transfer can always achieve smaller average delay and deadline violation probability than PP and MPP scheme for most classes. Its deadline violation probability of the lowest priority class can be better than SP.

– Generally the delay of token bucket-constrained traffic lies in between the $M/G/1$ delay bounds derived in [6]. But the heavy-tailedness of Pareto on-off, for eg. with a shape parameter of 1.3, and burst rate 0.25 can cause the delay to exceed the $M/G/1$ delay bound.

– The ticket transfer algorithm has an evident impact on reducing the mean delay of all classes except the lowest priority class. This is due to: (a) the probability of the network condition 12 (1100B) that contains only the two lowest priority classes becomes higher, and (b) the probability of the network condition 15 (1111B) that contains all classes becomes smaller, and in both cases, they approach that of SP. Both (a) and (b) increase the probability of the lower classes being serviced. Since the algorithm differentiates that higher priority classes are served as fast as possible when network conditions containing them appear, the mean delays of higher priority classes will therefore be much smaller than PP.

3 Achieving Assured Forwarding Using MPP

We consider 8 QoS classes and we configure a MPP scheduler to have 2 segregation groups AF_1 and AF_2. Each group has the last parameter $p_4^{AF_1} = p_4^{AF_2} = 1$. In each group, the AF classes are assigned parameters $p_1^{AF_i} < p_2^{AF_i} < \cdots < p_4^{AF_i}$, $i = 1, 2$. The following theorem ensures that this parameter assignment guarantees AF classes to obtain better statistical relative delay service differentiation than its immediate lower priority class. The group segregation property states that in PP, the service discipline among segregation groups is exactly the Strict Priority discipline hence the first AF group is guaranteed to have better service than the second group in terms of delay [4]. By means of segregation, this framework (a) provides more isolation among high priority classes that demand low delay and deadline violation probability, and low priority classes that require at least best effort service, and (b) reduces the number of classes within a group as this means a smaller number of network conditions within each group therefore we can configure more number of winners within each super-quantum, i.e. smaller spacing between consecutive winners to improve the response time variability in multi-winner scheduling. Since each group is based on MPP scheduling, there is fairness in the resource allocation within each group by means of fair distribution to excess capacity [4]. The ticket transfer algorithm is used in the first segregation group to provide improved deadline violation probability and average delay. Since we do not consider admission control, we expect some form of policing to limit the burst size and amount of bandwidth admitted to each class to prevent starvation if a non-conforming flow enters the node.

Theorem 1. *An assignment of average probability parameter for each class where* $0 < p_1 < p_2 \cdots < p_N = 1$ *satisfies the priority hierarchy for relative service differentiation.*

Proof. Define r_i as in equation (1) and q_i as the average queue length of class i. At steady state, we want higher priority classes to have shorter backlogs. Hence using the relationship that $\frac{r_i}{r_1} \propto \frac{q_i}{q_1}$, we can use Little's theorem [5] to show that $p_i = \frac{\lambda_i}{\prod_{k=1}^{i-1}(1-p_k)\sum_{j=1}^{N}\lambda_j}$, $i = 1, \ldots, N$. Since $p_{i-1} = \frac{\lambda_{i-1}}{\prod_{k=1}^{i-2}(1-p_k)\sum_{j=1}^{N}\lambda_j}$ therefore $\frac{p_{i-1}}{p_i} = \frac{\lambda_{i-1}}{\lambda_i}(1 - p_{i-1})$ which can be further simplified to $p_{i-1}\lambda_i\left(\frac{1}{p_i} + \frac{\lambda_{i-1}}{\lambda_i}\right) = \lambda_{i-1}$. Since the highest priority class gets served with the highest probability, its average departure rate must be the greatest among all classes, i.e. $\lambda_1 > \lambda_2 \cdots > \lambda_N > 0$. Thus we have $p_{i-1}\left(\frac{1}{p_i} + \frac{\lambda_{i-1}}{\lambda_i}\right) > 1$. Rearranging the term leads to $\frac{p_i p_{i-1}}{p_i - p_{i-1}} > \frac{\lambda_i}{\lambda_{i-1}}$ hence $p_{i-1} < p_i$. Thus the theorem is implied. Since this assignment is independent of the number of classes in the system, $p_1 > \frac{1}{2}$.

4 Efficient Implementation of MPP in IXP1200

We implemented our proportional bandwidth guaranteed probabilistic priority multi-class framework proposed in the previous section on Intel IXP1200 network processor [3][2]. To convert the class parameter p_i to tickets in lottery scheduling, all the assigned parameters p_i within a group are normalized to their least common multiple. For a large number of classes, we use Euclid's Greatest Common Divisor algorithm to speed up computation in the StrongARM core before supplying the scheduler's parameters in a numerator vector string to the microengines. For an 8-class system at an egress port, all the class parameters are stored in only two SRAM memory words with each parameter p_i occupying 8 bits thus the smallest probability being addressable is $\frac{1}{256}$ which offers relatively high computational granularity. In comparison, earlier implementation [6] will require over 100 Bytes of parameters' storage and larger memory access overheads. Clearly, our approach reduces memory access overhead drastically and accommodates more classes in a multi-port setting.

4.1 Fast Algorithm for Scaling Uniform Distribution

The StrongARM is elected to run a periodic task of generating uniform pseudorandom numbers in the SRAM. When the microengines require a random number for computation, they simply do a table lookup. This table has to be updated often by StrongARM to prevent a microengine from reading the same entry twice. However, we note that too high a refreshing frequency will lead to a higher latency for a microengine's SRAM read operation to this shared table due to increased contention between StrongARM and microengines. Numerous techniques exist for scaling uniform random numbers. An exact scaling method would convert the random number from an integer to a floating-point number between 0 and 1, multiply it by X, and then convert the result back to the nearest integer [8]. Alternatively, 32-bit random numbers in a particular uniform distribution, $Uniform[0, X]$ can be obtained by dividing any random 32-bit wide number in the range 0 to $2^{32} - 1$ by X and keeping the remainder under the assumption that $X \ll 2^{32} - 1$ [8]. Due to the significant computation overhead of integer division (measured as 378 cycles and independent of the value size of X), this method is not scalable without a pseudo random number generation co-processor. From the observation that each bit in any 32-bit uniformly distributed random number has an equal chance of being a "1" or "0", we use a simple generic bit-wise algorithm to map this uniform random number into another equally uniform random number, effectively scaling $Uniform\left[0, 2^{32} - 1\right]$ to $Uniform\,[0, X]$. This algorithm shown in Table 3 first performs the AND operation, and then re-claims those bits lost in the AND operation, ignoring bits which are outside the desired range. It is noteworthy that the instruction cycle count for this algorithm depends on the value size of X, i.e. we can trade-off computational granularity with speed. For X less than 255, this algorithm takes 41 instruction cycle counts. In the worst case, mapping a full 32-bit value of X requires a maximum of 173 instruction cycle counts but the gain is already an exponential increase in computational granularity to approximately $2^{32} - 1$.

[2] We use Intel network processor IXM1200 c-PCI hardware based on IXP1240 chipset.

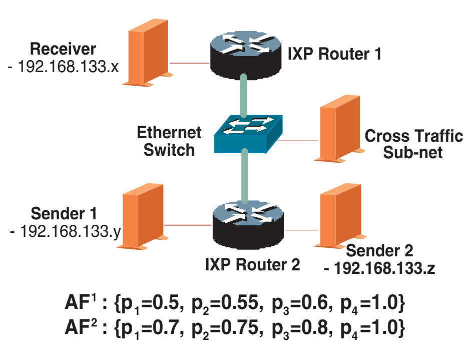
Table 3. *U-Map* scaling algorithm

```
    int result         =    0;
            int comparator    =     denominator & random_number;
if(comparator == 0) comparator = denominator;
while(comparator)
            {
                result      | =    (comparator & random_number);
                comparator  >>=   1;
            }
            if(result > denominator)    result =    denominator ^ result;
            return  result;
```

5 Performance Study and Results

In this section, we evaluate the performance of our implementation. In our experiments, we use token bucket metering to characterize the service and allocate a pre-calculated buffer space for each class. We present here the results in terms of mean delay and deadline violation probability. The topology of the experimental test-bed is shown in Fig. 3. All network links are full-duplex and have a capacity of 100 Mbps. We classify the traffic generated as Assured Forwarding (AF) and Best-Effort (BE). We implement 8 QoS classes with DiffServ Codepoints (DSCP) classification using our framework with two segregation groups. Each priority class in AF has marking 0x2e, 0x0a, 0x12, 0x1a, 0x22, 0x0c, 0x14, and 0 respectively. Class 1 and 2 traffic is sent from Sender 1 with the rest of the traffic in Class 3 to 8 from Sender 2. All flows are independent Poisson processes with exponentially distributed packet lengths and have the same mean sending rate and mean packet size. In order to simulate congestion, we use one IXP1200 (IXP Router 2) to generate high volume of traffic at the Gigabit output which is in turn forwarded to the Fast Ethernet output port on the other IXP1200 (IXP Router 1) which runs the MPP scheduler algorithm. Additional cross-traffic is also generated in the background to vary the congestion load pattern. All traffic terminates at Receiver.

The parameters for the framework are as shown in Fig. 3.The deadline violation probabilities of class 1 and class 2 are shown in Fig. 4(a). As expected, the deadline violation probabilities of class 1 and class 2 of MPP with ticket transfer scheme lie in between that of normal PP and SP. At low load, the deadline violation probability is very close to that of SP. Fig. 4(b) shows the average delay ratio between classes of the MPP with ticket transfer scheme measured within an interval of 1 hour. As the traffic load increases, the delay differences between classes become wider. Thus, with appropriating setting of the class parameter as described in section 4, higher priority classes get better delay differentiation at medium to high load. Due to space limitation, we do not present the packet loss statistics but we observe that the packet loss for each class in our experiments is strictly increasing as the priorities get lower. Note in Fig. 4(b) that the delay



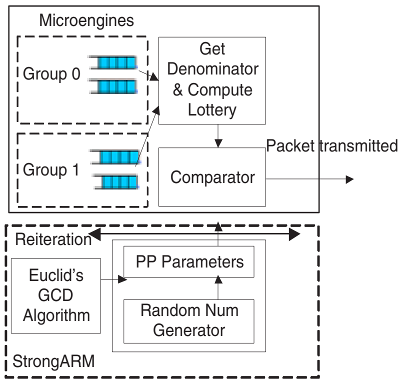
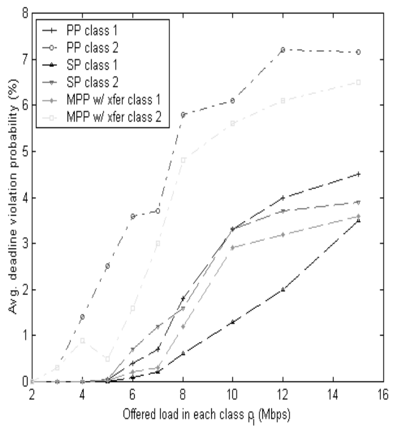


Fig. 3. (a)Relative DiffServ Test-bed and Assured Forwarding framework configuration (b) Block diagram of implementation on IXP1200 network processor

spacing between the last class in AF_1 and the first class in AF_2 is quite small. However, MPP scheduler observes the strict priority rule between segregation groups hence we can expect packet loss and deadline violation probability of the first class in AF_2 to be higher.



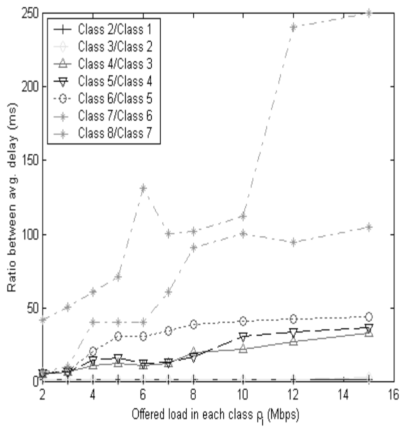


Fig. 4. (a)Deadline violation probabilities (b)Delay ratios between classes

In order to compare the impact of packet sizes on the performance of the MPP with ticket transfer scheme with PP scheme under congested conditions, we repeated the same experiments with different packet size distributions. The observed experimental results were largely similar to those obtained above but we note that for large packet size close to MTU, the benefit of the ticket transfer algorithm is not so obvious because, at high load, the time for a single packet transmission becomes longer thereby increasing the probability of the network condition where all classes' HOL packets are present as is in the case of SP. Nevertheless, the queuing delays in this case are still not as high as those for SP or the PP scheme. In summary, the MPP with ticket transfer scheme is good when the primary goal is to provide relative delay differentiation as in SP while ensuring that deadlines of higher priority classes are not unnecessarily violated, and also meeting specific timing requirements, for eg. small delay bounds for high priority classes as in absolute QoS.

6 Conclusion

This paper showed that PP is a special scheme of lottery scheduling in the strict priority sense and the PP algorithm was generalized to a Multi-winner PP algorithm which ensures that high priority classes get served within deterministic time quanta. A ticket transfer algorithm was proposed to overcome the problem of the highest priority class from missing its deadline at small time scales. Simulations showed that MPP with ticket transfer surpasses the original PP and SP using bursty traffic class aggregation. Our algorithm provides lower deadline violation probability and mean delay to most classes than original PP. Finally, we presented the performance of our algorithms on high-speed routers.

References

1. S. Blake, D. Black, M. Carlson, E. Davis, Z. Wang and W. Weiss, An architecture for differentiated services, IETF RFC 2475, Dec 1998.
2. J. Eggleston and S. Jamin, Differentiated services with lottery scheduling, *Proc. Int'l Workshop on Quality of Service (IWQoS'01)*, Jun 2001.
3. Intel IXP 1200 Network Processor: Microcode Programmer's Reference Manual Revision 11, Part No. 278304-011 March 2002.
4. Y. Jiang, C.K. Tham and C.C. Ko, A probabilistic priority scheduling discipline for multi-service networks, *Computer Communications 25 (13) 2002 pp. 1243–1254*, 2002.
5. L. Kleinrock, Queueing Systems: Vol.2, Computer applications, John Wiley & Sons, 1976.
6. C.-K. Tham, Q. Yao and Y. Jiang, Achieving differentiated services through multi-class probabilistic priority scheduling, *Computer Networks, 40, pp. 577–593*, 2002.
7. C. Waldspurger and W. Weihl, Lottery scheduling: Flexible proportional-share resource management, *Proc. the First USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, Nov 1994.
8. C. Waldspurger, Lottery and stride scheduling: Flexible proportional-share resource management, Ph.D. Thesis, Massachusetts Institute of Technology, Sep 1995.
9. M. Zhang, R. Wang, L. Peterson and A. Krishnamurthy, Probabilistic packet scheduling: Achieving proportional share bandwidth allocation for TCP flows, *Proc. IEEE INFOCOM 2002*, Jun 2002.

Realizing on Demand Networking

John Strassner

Intelliden Corporation
90 South Cascade Avenue, Colorado Springs, Colorado 80903
john.strassner@intelliden.com

Abstract. The promise of e-business is being increasingly adopted. However, applications are getting more sophisticated, and often require conflicting resources and/or conflicting network configurations. Meanwhile, the environment in which these applications operate is becoming more complex. This paper describes what is needed to build On Demand Networking capabilities. It will be shown that two related problems need to be overcome. First, the services that the network provides are in general not related to how the business operates. Second, network management must undergo a revolutionary change – one that enables it to become autonomic. The result of this approach is a new genre of management applications that ensure that the network delivers the services asked of it by the business, on demand.

1 Introduction

Dr. Irving Wladawsky-Berger defines on demand computing this way: "An on demand business is one that can respond with complete flexibility to changing market conditions, customer demand or external threat, in real time, as they are occurring, because all its business processes are thoroughly integrated" [1].
This is certainly a compelling vision. Imagine, for example, an autonomic network, one which can self-heal, self-configure, self-protect and self-optimize according to changing needs and the changing environment. Unfortunately, this vision looks extremely unlikely given today's management environment. It is common practice to use multiple management systems from the same vendor to manage different devices manufactured by that vendor. Using heterogeneous devices exacerbates this problem, since different languages and programming models are introduced. In addition, there is no connection between management software to control network devices and services and applications that provide revenue. Since Operational Support Systems (OSSs) from best-of-breed applications, there are usually multiple management applications of different categories.
Most importantly, however, there is no standard way to tie business rules that describe how the organization operates to the network. Thus, there is no means to ensure that the network delivers the services that the business needs at any given time. Until this is solved, then it is impossible to have the network change the services that it provides to accommodate changing business needs, because it is unaware of business needs in the first place.

This paper presents a new approach towards network management. This approach is based on two observations. First, current network management approaches are not enabling Service Providers or Enterprises to use business rules to develop and deploy device configuration changes. Solving this problem enables business rules to *drive* the configuration of the network. Second, any form of autonomic operation starts with a simple premise: the autonomic entity must *know itself*. The solution must leverage the knowledge of the system and its environment, and embed in the system the ability to react to changes in the environment. This paper proposes a variant of the OMG's Model Driven Architecture (MDA) [2] as one way to implement this "higher intelligence".

The organization of this paper is as follows. Section 2 defines terms used in this paper. Section 3 describes the current problems with network management and anticipates new problems that will arise, based on the ever-growing complexity of applications and user tasks. Section 4 provides a brief overview of the DEN-ng information model (Directory Enabled Networks new generation) which is being developed in the TeleManagement Forum (TMF). Section 5 describes the roles of policy and process management, and how the DEN-ng models support them. Section 6 examines the generic MDA approach. Section 7 describes specific enhancements made to MDA by this new On Demand Networking approach. Section 8 presents conclusions and future work. Finally, Section 9 lists references for this work.

2 Terminology

Definitions in this section are taken from [3].

2.1 DEN versus DEN-ng

DEN, or Directory Enabled Networks [4], is a *specification* of an object-oriented information model describing the entities in a managed environment, and how they are related to each other. It also specifies a *model mapping* to an (L)DAP implementation. The original DEN specification is *not* solely a mapping to an LDAP model, nor does it have anything to do with WBEM, as [5] states.

DEN new generation is the next version of the DEN specification. It is tightly bound to the TMF NGOSS architecture [6], and contains business, system, and implementation entity definitions. Its main purpose is to define the functionality and synchronize the relationships between Customers, Processes, Products, and Network Services and Capabilities. It is the model that is being used in the approach described in this paper.

2.2 Policy

A *Policy* is a set of rules that are used to manage and control the changing and/or maintaining of the state of one or more managed objects. (Note in particular the use of *state* in this definition, which is new – see [3] for more detail.)

2.3 Information Model versus Data Model

An *information model* defines the properties, operations, constraints, and relationships of managed objects. It is independent of any specific repository, software usage, platform, or access protocol.
In contrast, a *data model* is a concrete implementation of an information model that is bound to a particular platform. It includes data structures, operations, and rules that define how the data is stored, accessed and manipulated.

2.4 Model Mapping

A *model mapping* is a translation from one type of model to another type of model. Model mapping changes the representation and/or level of abstraction used to another representation and/or level of abstraction.

2.5 Business Driven Device Management (BDDM)

BDDM is defined as the ability to use business rules to manage not just the construction of configuration files and commands for a device, but also to enforce how the configuration of a device is created, verified, approved, and deployed. With business-driven device management, configuration management becomes the foundation for managing the entire network (see [7] for more detail).

3 Problems in Current Network Management Approaches

Current network management solutions, and especially OSSs, suffer from the inability to use business rules to translate business needs and processes into device configuration changes. This is being exacerbated by the ever-increasing complexity in applications and the desire to build one network to serve the needs of multiple people. This section will examine some of the problems in building an OSS.

3.1 Barriers to Operational Efficiency and Data Exchange

The three largest barriers to operational efficiency are: (1) fragmented business processes, (2) fragmented data, and (3) incompatible systems with weak levels of integration.
There are always business rules that govern how devices should be managed. Unfortunately, these business rules tend to be ignored, because of the *lack of correlation between business rules and network configuration*. For example, people shouldn't simply telnet into a router and start typing CLI to fix a problem. Yet they do, because business processes either aren't specified or are ignored. Unless the network is treated like any other business asset, it will always be handled differently, and it will always be "OK" to ignore business process. We need a paradigm shift –

one which *encourages* the user to “play by the rules”. This means that we need to *automate* how business rules govern network configuration, because no matter how pretty and functional the GUI, people won’t use it if they can find an alternative means to do the same job faster.

This paradigm shift requires a common information model to represent not just different managed objects, but also the interaction between these managed objects. (The question of why another information model should be built is answered in the next subsection.) Currently, data exists as isolated information islands, because there is no common information model that is used in building an OSS. This is because of the proliferation of stovepipe management applications, and is shown in Figure 1 below (which is taken from a real OSS in the industry). Note that the figure is supposed to look complicated.

Current systems, like that shown in Figure 1, are designed using stovepipe applications because stovepipe applications provide best-of-breed functionality. Unfortunately, each stovepipe application is designed with itself as the “center of the universe”. This creates an interoperability nightmare, because objects and information are continually renamed and redefined. It leads, inexorably, to systems with an amazing amount of complexity, as shown in Figure 1. Such systems have a large number of *brittle* interconnections, which means changing any component in this OSS is a challenging task. Realizing that vendors won’t discard existing applications just because a new information model exists, the approach is rather to build a common mediation layer between these disparate management applications. This in turn means that such mediation must be inherently extensible and automated to keep up with the changing business demands.

3.2 Why Another Information Model?

Before the advent of DEN-ng and the TMF’s SID (Shared Information and Data) model, there were three main policy models in the industry: (1) the DEN model [8]), (2) the IETF model (as defined in [9], [10], [11], and [12], and (3) the DMTF model ([13]). Of these, DEN was used as the basis for both the IETF and the DMTF models. There are in addition other efforts, such as Ponder ([14]), that imply additions and/or changes to one or more of these models.

Policy models are critical to the approach proposed in this paper. However, they need to be integrated with models of other objects (e.g., users, services, and so forth) so that policy can be used to control these other objects.

With the exception of the DEN-ng model, the problem with all of them to date is fourfold. First, they all focus on modeling the current state of an object. While this is important, this does not describe the life cycle aspects of the managed object. Therefore, models have only been used in the design process, and not in the actual management process.

Second, current models can be categorized as either business or system models, but not both. This makes it impossible to implement BDDM, and ensures that the business and networking worlds will remain separate.

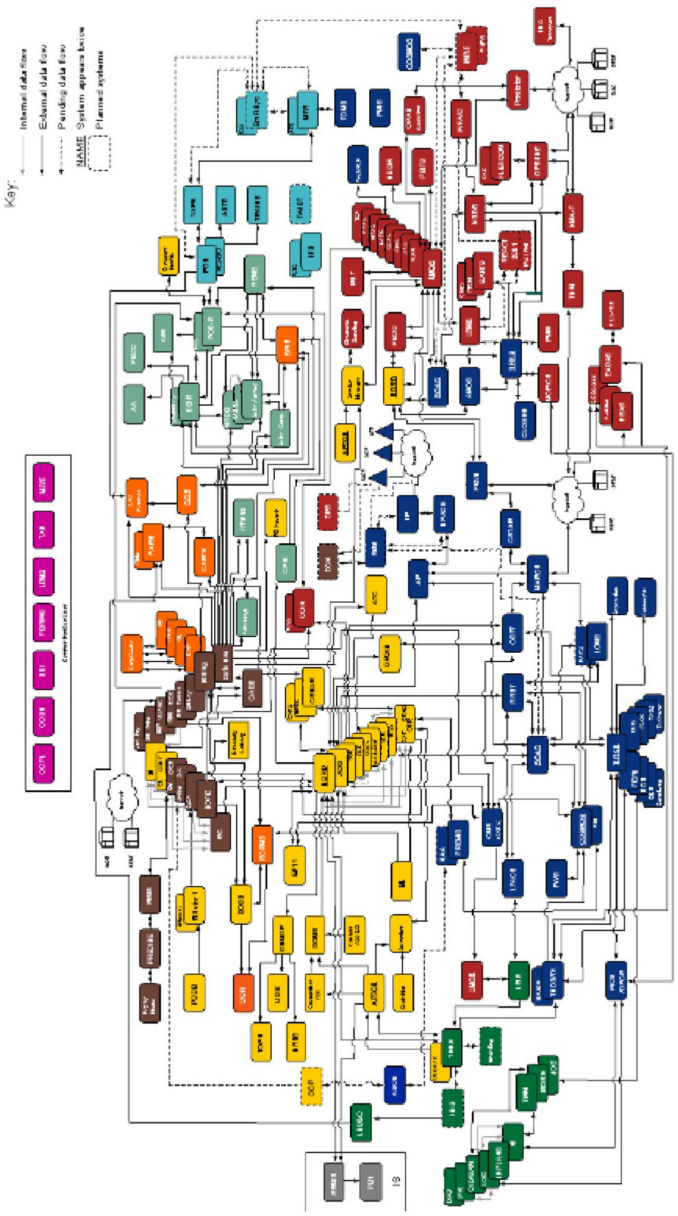


Fig. 1. A Traditional OSS

Third, previous models focus on policy as a domain that is only loosely integrated to other domains. In contrast, the main use case for the DEN-ng policy model was to define a policy model that was closely integrated with the rest of the managed

environment. This would enable the policy information model to define how policy interacted with the rest of the managed environment.
Finally, previous information models have been built without considering how to derive data models from them. Management data has too many diverse characteristics to enable all types of management data to be stored in a single repository. Model mappings provide formal methods to translate between models, and enable different repositories to be federated.

3.3 Sharing and Reusing Data

Stovepipe applications, such as those shown in Figure 1, cause many integration issues. A GUI shows its own application-centric view of the piece of the network that it is managing, not of the entire system. However, an application is simply a realization and manipulation of its underlying object model. Object models prescribe their own view of the world upon the applications and systems that use them. Unfortunately, this means that the same data will be needlessly redefined, or restructured, or worse, conflict with other management data, instead of being *reused*. This in turn implies that applications that could have used information from other applications must instead build a duplicate process to obtain that data. For example, a provisioning application requires up-to-date topological information as well as knowledge of which hardware is running which release of which operating system.

3.4 Using Business Rules to Drive the Configuration of the Network

There are two conflicting desires for network design. First, people want to use the same network to do more things, rather than have multiple networks doing specialized tasks. Second, more people are using networks with more sophisticated applications. The problem is that applications having vastly different resource utilization requirements need to peacefully coexist and run concurrently on the same network. Clearly, business rules must be used that decide which application gets priority usage of shared resources. In order to meet these requirements, the current focus of network management products must change to enable business rules to drive how the network is configured.
DEN-ng uses the concept of viewpoints (as defined in RM-ODP, in [15]). A viewpoint is an abstraction obtained by using a selected set of architectural concepts in order to focus on a set of particular concerns within a system. One can think of the DEN-ng models as organized along two dimensions – knowledge domain and viewpoint. For each knowledge domain (e.g., policy, resource, service, and so forth), three viewpoints (business, system, and implementation) abstract the entities and relationships in that viewpoint. This makes DEN-ng unique, as other information models don’t have the concept of viewpoints. This makes it possible to connect the business, system, and implementation viewpoints, which enables business needs to be translated to implementation software and hardware. This is a prerequisite to meeting the needs of On Demand Networking.
Figure 2 shows a simplified UML diagram of how DEN-ng solves this problem.

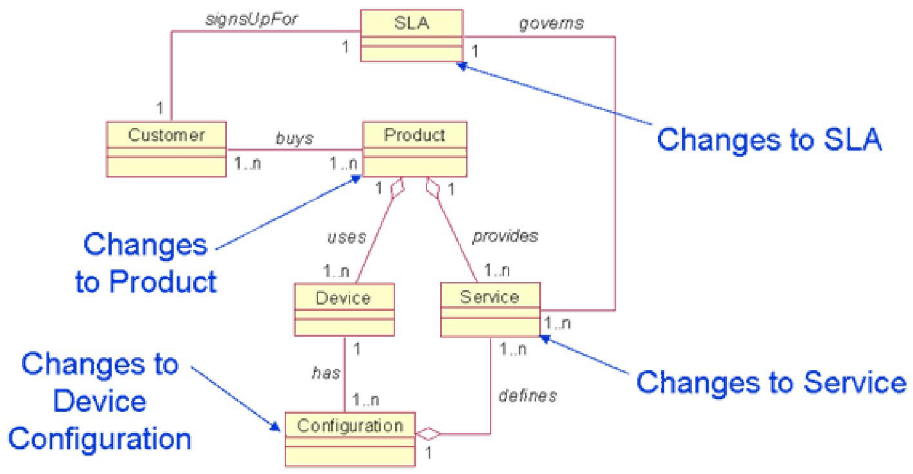


Fig. 2. Business Changes Must Drive Network Configuration

Previous solutions have not integrated multiple viewpoints. In contrast, DEN-ng links them firmly together by defining the (business) notion of a Product as containing Resources and Services. Figure 2 is a simple UML drawing that shows that a Customer buys one or more Products. (Note that the DEN-ng models are much more complex.) Each Product uses one or more Devices, and provides one or more Services. Note that Configurations exist for the Device and the Services that the Device provides. Changes to a Product, a business definition of a Service, or an SLA all affect the Devices and/or network definitions of Services in the Product, and hence are reflected in Device Configuration changes. Finally, the network is driven by business changes!

4 Overview of DEN-ng

The original goal of DEN was to define a set of network services that met the needs of a set of applications according to the policies that were applicable at the current moment. In essence, it proposed a *lingua franca* that enabled applications to express their needs in terms that the network could understand, and vice-versa.
The DEN-ng work is being done in the TeleManagement Forum (TMF – see www.tmforum.org) because the TMF had already started work on its NGOSS architecture, and because the TMF realized the value of having a set of models that met the requirements of the previous section.
DEN-ng, like DEN, is built around using a finite state machine model to describe and control managed entities (another key point that other information models do not have). DEN-ng defines three fundamental types of classes to support a finite state machine: (1) classes to model the current state of a managed entity, (2) classes to

model the changing of state of a managed entity, and (3) classes to control when the state of a managed entity is changed.

Conceptually, DEN-ng is built as a framework of frameworks, as shown in Figure 3 below.

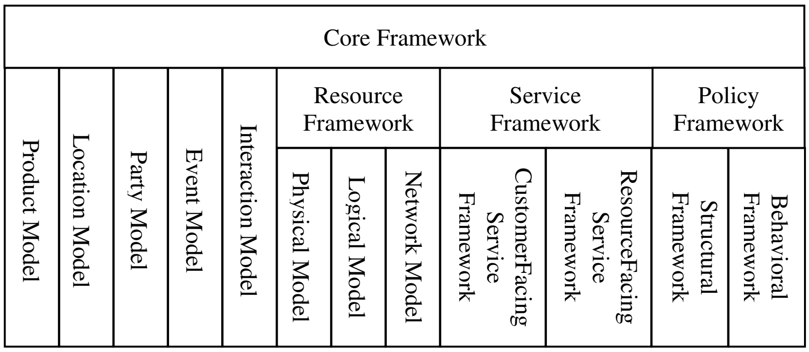
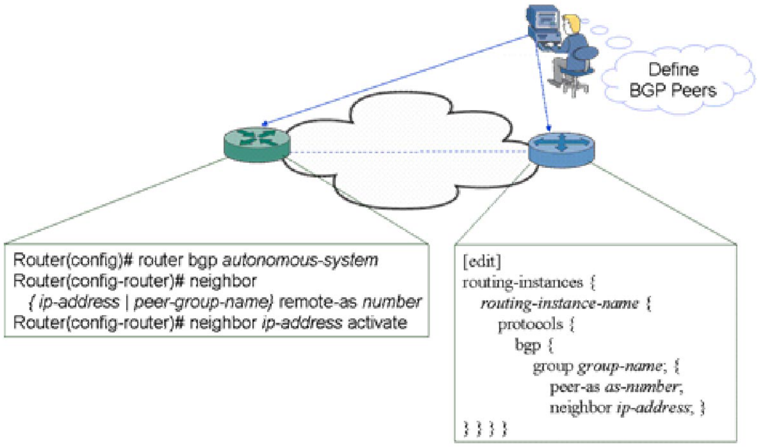


Fig. 3. Simplified View of DEN-ng

The Core Framework contains high-level entities and relationships that enable more specific domain models (even from other standards bodies and fora) to be integrated into a single cohesive whole. The Product model provides an abstraction for containing Resources and Services, and links both to other business entities, like Customer. The Location model enables semantics to be attached to locations. The Party model is an abstraction for individual and groups of users, along with organizations. The event model is based on the UML metamodel definition of event, and has been refined to include RM-ODP concepts of announcement and interrogation [16]. The Interaction model is based on several UML metamodel concepts, such as Collaborations, and is used to define the semantics of how entities interact with each other. The Resource Framework provides additional abstractions for representing physical, logical and network resources (each of which has their own detailed domain model). The Service Framework abstracts Services into two types – Services that a Customer is explicitly aware of, and Services that a Customer is not explicitly aware of. For example, a VPN is a service that a Customer can buy, so it is a CustomerFacingService. That VPN may use BGP to advertise routes, but (hopefully!) the Customer is blissfully unaware of BGP. Thus, BGP is a ResourceFacingService. Finally, the Policy model is divided into two domains. The Structural Framework contains a set of entities that can represent Policy across the Policy Continuum [3]. The Policy Behavioral Framework connects the representation of Policy to the other models in DEN-ng.

Note that the TMF has defined a Shared Information Model (SID) in [17] that is beginning to incorporate some of the DEN-ng models. Similarly, DEN-ng is heavily influenced by the SID. The SID is currently chartered just to work on the business model, whereas DEN-ng is focused on building out the business, system, and implementation viewpoints of the Resource, Service, and Policy domains. DEN-ng uses the SID work and then extends that work to the system and implementation viewpoints. These extensions are being proposed back into the formal SID work.

Unlike the other information models cited in this paper, DEN-ng and the SID use modern software techniques (such as Patterns [18], [19]) and various types of abstraction techniques, such as roles [20]. Two critical abstractions introduced by DEN-ng are capabilities and constraints. Capabilities are a means of abstracting the functionality of a device into a set of interoperable building blocks.



```
Router(config)# router bgp autonomous-system
Router(config-router)# neighbor
  { ip-address | peer-group-name} remote-as number
Router(config-router)# neighbor ip-address activate
```

```
[edit]
routing-instances {
  routing-instance-name {
    protoools {
      bgp {
        group group-name; {
          peer-as as-number;
          neighbor ip-address; }
} } } }
```

Fig. 4. Normalizing Different CLIs

Figure 4 shows an operator trying to accomplish the same task on two different routers. Even though the task is the same, the CLI is completely different. More importantly, the device on the left has different configuration *modes*, which are absent in the device on the right. Thus, the *programming model* for these devices is different. This makes it hard to build a single end-to-end service involving these two devices. Capabilities *normalize* the different functions of heterogeneous devices, so that a single control plane can be used to manage each device. This enables generic functions to be described independently of the programming model and type of device, which simplifies using heterogeneous devices.

The DEN-ng model also differs from other models in the level of granularity of information that it models. For example, in addition to high-level concepts like Product and Device, it contains very low-level concepts, such as a DeviceInterface, or the Configuration of a Device. This is because one of the important goals of the DEN-ng model is to be able to model configuration changes, as well as to show how policy can be used to control the deployment of these changes. Experience has shown that this mix of levels of abstraction is required to implement support for On Demand Networking, because the On Demand Network serves multiple users operating at multiple levels of abstraction.

Constraints are used to model restrictions on using certain device functions by the current environment. For example, one application may prohibit the use of a command, or restricting configuration changes to a particular time.

This combination of capabilities and constraints is essential for realizing the most fundamental of all requirements for building an autonomic network: *self-knowledge*.

The world will never adopt a single command language, or use a single programming model, or agree on a single platform. The only way to define knowledge is through a set of abstractions that can accommodate the different functionality and behavior of the elements of a system. The DEN-ng model is the first step in that direction.

5 The Holistic Combination of Policy and Process Management

Business process automation can be characterized by the separation of the expression and execution of business processes and services from the software that implements these business processes. This separation enables the application of business management techniques to the business processes that are implemented. There are two major efforts for standardizing business processes – the eTOM of the TMF [21] and ITIL [22]. The eTOM team in the TMF is currently producing a mapping between the eTOM and the relevant portions of ITIL.

Policy management is defined as the usage of policy rules to mange the configuration and behavior of one or more entities. The DEN-ng policy model [3] is designed as a class hierarchy that is used to represent the structure of policy rules as well as their semantics. This combination enables the DEN-ng policy model to be used as *reusable objects that control the state transitions of the managed objects of the environment*. This enables models to be built to represent the entire life cycle of the managed system.

Policies have traditionally been used primarily as a means to prioritize the allocation and access to resources by different applications. Commercial implementations, as well as the literature, have not defined *how* this prioritization occurs – it is just assumed that it has occurred, and is simplified into administrators listing policy rules in an *ad-hoc* fashion. The answer to this question lies in realizing that Policy implies the use of a methodology. Finite state machines have been chosen in DEN-ng because they are standardized in UML and they are a simple, well-known, and effective concept.

Currently, Process Automation techniques and Policy Management are separated. Instead, the approach defined in this paper suggests that the holistic combination of policy management and process management is what is needed. A simple, yet elegant, model ensues from this combination: policies are used to define goals to be achieved, which result in the selection of one or more processes. The execution of these processes are monitored, and their collective results analyzed to adjust (if needed) the set of policies that are active at any given time. Thus, a closed loop system is achieved. This is shown in Figure 5.

The above process starts with the important concept of separating the act of constructing a configuration change from the act of deploying that change. This realizes the basic fact that no two types of configuration changes are different. Sadly, most current approaches do not make this separation.

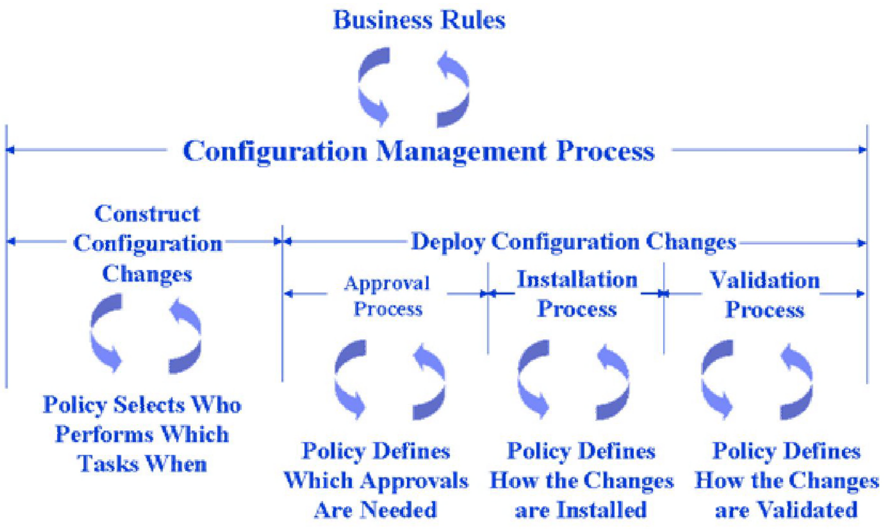

Business Rules
Configuration Management Process
Construct Configuration Changes
Deploy Configuration Changes
Approval Process
Installation Process
Validation Process
Policy Selects Who Performs Which Tasks When
Policy Defines Which Approvals Are Needed
Policy Defines How the Changes are Installed
Policy Defines How the Changes are Validated

Fig. 5. Using Policy and Process Management for Resource Configuration

Consider two different configuration changes – a simple change of the SMTP address of a device, versus changing BGP peering relationships. These are very different in nature – the former is a simple and straightforward change, whereas the latter can be quite complicated, and can affect many different parts of the organization. However, even though the first change is simple, most organizations will define a time period for implementing this change. This is, of course, a *business policy*; the question is, how is it enforced in the network?

The latter change is more complicated to construct. In addition, this change could affect other devices and other services. Other (business) organizations will probably need to look at this change, since it could have significant adverse financial consequences if it is implemented incorrectly. Thus, this change should require multiple levels of technical and business approval. Unless all of these different reviews are related to each other, it will be impossible to relate device and service changes to external events, which could adversely affect the liability of the business. Again, how is this enforced in the network?

This paper proposes that policy and process management work together to control how a resource is (re)configured. The output of the business processes is monitored; the results of monitoring the business processes are used to adjust the set of policies that can be used at any given time. This closed loop system ensures a stable, repeatable means for adjusting the network to meet the demands of the organization.

6 The Generic Model Driven Architecture (MDA) Approach

MDA is defined by the OMG in [23]. The heart of the MDA effort is based on the use of open standards defined in the OMG – Unified Modeling Language (UML) [24]; Meta-Object Facility (MOF) [25]; XML Meta-Data Interchange (XMI) [26]; and Common Warehouse Meta-model (CWM) [27]. MDA uses these four standards to help separate business and application logic from their implementation. This results in better application reuse and increases the overall portability of an application that uses MDA. More importantly, it helps an organization implement their intellectual property in a modular fashion. A notable feature of MDA is that it addresses the complete life cycle of designing, deploying, integrating, and managing applications and their data.

UML is the key enabling technology for the MDA, as it enables every application to be based on a normative, platform-independent UML model. In the OMG approach, a platform-independent model (PIM) is mapped onto one or more platform-specific models (PSMs).

7 The New on Demand Networking MDA Approach

The MDA is a good approach. Nascent efforts in implementing the mappings are also good approaches. However, the scope of this paper is slightly different. It is not trying to solve the *general* PIM-PSM problem, nor is it trying to build *general* code generators. Instead, it is trying to use MDA techniques to build a new type of network management system – one that is driven by business requirements.

In order to connect the business world with the networking world, we need a way of representing information from the business and system viewpoints, and a method to tie them together. In order to implement this in a product, we need to tie the implementation viewpoint with the previous two viewpoints. Thus, these three viewpoints *jointly* affect the design of the models. The fact that there are three viewpoints means that abstraction must be used to encourage subject-matter experts in the business, system, and implementation domains to use the approach. The business entities are designed *in response to* standard process descriptions from the eTOM. The system view is the abstract specification of how these entities work together. The implementation view is the preparatory step towards implementing the system, and ensures that fundamental characteristics, such as the support of namespaces, are used. Experience has proven that building separate models to support these viewpoints cannot work, because it is impossible to keep the models synchronized. Rather, the analogy of a “database view” is used – information is hidden or shown as appropriate to support the current viewpoint.

Models are important to the On Demand Networking architecture, because they represent reusable abstractions that capture, in a formal way, the components used in the business, and how those components relate to the overall infrastructure as well as the managed environment. Thus, it is the model that drives the development of the application.

The key to using models for On Demand Networking is to use UML's inherent extensibility features to capture the unique semantics of networking. A *UML Profile* is a collection of model elements that have been customized for a specific domain or purpose by extending the metamodel using stereotypes, tagged definitions, and constraints. A *stereotype* is a model element that defines additional values (based on tag definitions), additional constraints, and optionally a new graphical representation. *Tag definitions* specify new kinds of properties that may be attached to model elements. The actual properties of individual model elements are specified using *Tagged Values*. *Constraints* express (typically) invariant conditions that must hold true for the set of entities that are being modeled. Note that constraints by definition do not have side effects, and therefore cannot alter the state of the system.
Figure 6 shows the code generation process used for DEN-ng.

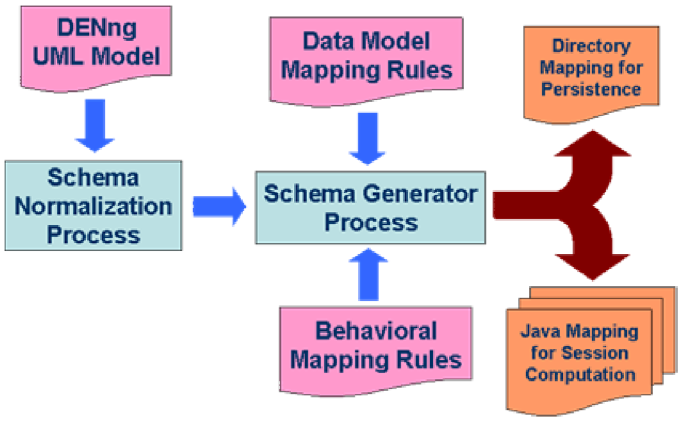

DENng UML Model
Data Model Mapping Rules
Directory Mapping for Persistence
Schema Normalization Process
Schema Generator Process
Behavioral Mapping Rules
Java Mapping for Session Computation

Fig. 6. Generating Code for On Demand Networking

Since DEN-ng contains business, system and implementation information, this process is independent of the type of model for which code is being generated. The UML source files are parsed and put into a canonical format as preparation for being mapped into a set of data models. This process uses one or more model mapping rule files to generate schemata for the appropriate target data models, along with rules that define precise semantics for the behavioral components of the UML models. We have to date successfully generated schemata for directories as well as for Java environments. More importantly, data coherency is maintained even though the two data models are significantly different, because they are both derived from the same information model.
It is important to understand why both of these data models are generated. Directories are for searching, and should only be used for storing entities that change much slower than the replication frequency of the directory. The Java entities are used for session computation, and are better suited to implement business rules and constraints specified in the model. They are used for per-session computation, with the result being persisted in an appropriate repository.

8 Conclusions and Future Work

This paper has presented a new approach to building support for On Demand Networking. This support is based on four key principles. First, the DEN-ng information model is used as a single means to represent management information. This choice is made because of its following unique characteristics: it is based on UML, it is inherently extensible through its use of patterns, it supports multiple levels of abstraction and multiple viewpoints, it uses a finite state machine, and it contains low-level information modeling (e.g., a "device interface") that other information models do not have. Second, the DEN-ng information model supports multiple mappings to different data models. This is mandatory to support the diverse types of management information that On Demand Networking requires. Third, the holistic combination of policy management and process management are used to mange entities in the On Demand Network. Conceptually, policies control state transitions of a managed entity, and processes are used to implement the goal of the policy. Finally, a variant of the OMG's MDA approach was defined. This variant was specifically tailored to meet the needs of On Demand Networking.
Future work will explore three different areas of research. First, more complicated network support will be built, to prove that this approach does indeed work generically. Second, the introduction of other autonomic elements, such as network devices or host systems, will be explored. Third, the concept of knowledge that spans multiple system elements for On Demand Systems will be examined.

References

[1] Please see: http://www7b.software.ibm.com/dmdd/library/techarticle/0302iwb/0302iwb.html
[2] Please see: www.omg.org/mda
[3] J. Strassner, "Policy-Based Network Management", Morgan Kaufman Publishers, ISBN 1-55860-859-1, to be published 2003
[4] J. Strassner, *Directory Enabled Networks,* Macmillan Technical Publishing, 1999, ISBN 1-57870-140-6
[5] The DMTF has redefined DEN to fit better to its CIM and WBEM efforts. The author does not agree with this redefinition. The DMTF definition is found here: http://www.dmtf.org/standards/standard_den.php
[6] TM Forum, NGOSS Architecture, TMF 053
[7] J. Strassner, "A New Paradigm for Network Management: Business Driven Device Management", SSGRRs Conference, August, 2002
[8] J. Strassner, "*Directory Enabled Networks*", Chapter 10, Macmillan Technical Publishing, ISBN 1-57870-140-6
[9] B. Moore, E. Ellesson, J. Strassner, A.Westerinen, "*Policy Core Information Model – Version 1 Specification*", RFC 3060, February 2001
[10] B. Moore, L. Rafalow, Y. Ramberg, Y. Snir, A. Westerinen, R. Chadha, M. Brunner, R. Cohen, J. Strassner, "*Policy Core Information Model Extensions*", draft-ietf-policy-pcim-ext-06.txt, November 2001
[11] Y. Snir, Y. Ramberg, J. Strassner, R. Cohen, B. Moore, "*Policy QoS Information Model*", draft-ietf-policy-qos-info-model-04.txt, November 2001

[12] S. Gai, J. Strassner, D. Durham, S. Herzog, H. Mahon, F. Reichmeyer: "QoS Policy Framework Architecture", February 1999
[13] Current work is in the DMTF Members only site, under: http://www.dmtf.org/apps/org/workgroup/policy/. The current release of CIM's policy model as of this writing is version 2.7.1, and is located at: http://www.dmtf.org/standards/standard_cim.php.
[14] Please see the following web page for detailed information about Ponder: http://www-dse.doc.ic.ac.uk/Research/policies/ponder.shtml
[15] Open Distributed Processing Reference Model – Foundations, ISO/IEC 10746-2, 1996
[16] Open Distributed Processing Reference Model – Overview, ISO/IEC 10746-1, 1998
[17] TMF, "Shared Information/Data (SID) Model", GB922 and its Addenda (one for each domain, 10 so far)
[18] E. Gamma, R. Helm, R. Johnson, and J. Vlissides, "Design Patterns: Elements of Reusable Object-Oriented Software", ISBN 0-201-63361-2, October 1994
[19] The Patterns Homepage contains a variety of useful links, and is at: http://hillside.net/patterns/
[20] D. Bäumer, D. Riehle, W. Siberski, M. Wulf, "The Role Object Pattern", can be downloaded from: http://jerry.cs.uiuc.edu/~plop/plop97/Proceedings/riehle.pdf
[21] F, "eTOM, the Business Process Framework", GB921, version 3.5, 2003
[22] The home page of ITIL is: http://www.itil-itsm-world.com/
[23] The home page of the MDA effort is: www.omg.org/mda
[24] The home page for the UML effort is: http://www.omg.org/technology/documents/formal/uml.htm
[25] The home page for the MOF effort is: http://www.omg.org/technology/documents/formal/mof.htm
[26] The home page for the XMI effort is: http://www.omg.org/technology/documents/formal/xmi.htm
[27] The home page for the CWM effort is: http://www.omg.org/technology/documents/formal/cwm.htm

An Architecture for Access Network Management with Policies (AN-PBM)

Olivier Corre[1,2], Idir Fodil[1,2], Vladimir Ksinant[1], and Guy Pujolle[2]

[1] 6WIND,
Immeuble Central Gare, Bat. C, 1 place Charles De Gaulle,
78180 Montigny-le-Bretonneux, France
[2] University of Paris 6, LIP6 Lab. Network and Performance,
8 rue du Capitaine Scott,
75015 Paris, France

Abstract. The actions led so far, especially by the IETF, have resulted in the definition of a network management architecture based on policies. Some information models, as well as transport and policy representation protocols, have been designed. The content of policies, a topic a bit neglected until now, has to answer ISPs and their customers' requirements for the definition of new value-added services. The proposal made by the IETF consists in configuring individually some network interfaces, rather than in globally managing the services. This approach has contributed to mark the difference between the service definition agreement and the network equipment configuration parameters. In this paper, we propose a policy-based service management architecture. This architecture eases the ISP implementation and management of the services it offers. Quality of Service is an area of experimentation for this architecture. Besides, this architecture gives ISP customers the possibility to carefully control their network quality of service.

Keywords. Architecture, Network Management, Policy, QoS, Service Provider, Customer

1 Introduction

For an ISP (or a SP - Service Provider), to manage a network consists in guaranteeing the service levels sold and rapidly solving the problems that may occur in the network. Network management is based on the following principles:

- Planning: customer-ISP service level specification,
- Checking: to ensure that the service level sold is really guaranteed,
- Detection: to notify that a service level is not suited,
- Reactivity: to adapt the network behaviour while the transport quality may affect the service level delivery.

Nowadays, the ISP network management is not much automated. Each router is configured with CLI commands and monitored in using SNMP directly on the machine or remotely. These tasks are most of the time repetitive, and need skilled staff able to work in an heterogeneous network. An accurate knowledge of the monitoring and configuration software is required for each equipment, and this is not an easy task taking into account the diversity of the devices. Moreover, the amount of data travelling the network makes the traffic heterogeneous and data come from applications which have different features. In addition, the amount of devices connected to the network is continuously increasing. It then becomes very difficult to manage the network efficiently. Network control requires a skilled manpower and some regular bandwidth extensions. The management cost will continue to grow in the near future, since service providers will offer more and more complex services, as those that exist in the Telecom world. For that reason, networks must be managed differently. In this paper, we will introduce an advanced management architecture for Service Providers. This architecture will then be experimented in the scope of Quality of Service. The IETF Policy-Based Management architecture, and our proposal, will be overviewed in the first part of the article. Quality of Service mechanisms will be detailed in a second section. Then, our network management approach will be described for QoS. The policies defined and their implementation in the 6WINDGate equipment will be detailed. Finally, a Voice over IP with Admission Control scenario will be depicted.

2 Policy Based Network Management

2.1 State of the Art

The opportunity to manage a network thanks to policies became a significant research topic from 1997. The first promising initiative has been submitted by Cisco and Microsoft in May 1997 with the Directory Enabled Network (DEN[DMT98]) information model, what objective was to use directories to give efficiency to network device management. A Working Group (DEN Ad Hoc Working Group) has been set up in order to empower most vendors to participate to the DEN specifications. By this way, the interoperability of distributed management tools was made easier [Kos01]. The Working Group success led to the creation of a consulting committee with more than 500 industrial partners, and the specifications of DEN were accepted by the DMTF in 1998. The integration of DEN has been based on the Common Information Model (CIM[DMT99][DMT]) already approved by the DMTF. CIM is an information model and its objective is to ease the management of users and machines in a company environment.

In 1999, the Policy Framework (PF) Working Group has been founded at the IETF in order to bind the IETF and the DMTF, to integrate the CIM and DEN models in a network environment, and to map DEN into LDAP. The PF Working Group has designed an extension of CIM, called Policy Core Information Model (PCIM[MESW01][MRR$^+$01]), and some extensions of PCIM for QoS [SRS$^+$01], and for network security [JRV01].

The IETF snmpconf Working Group has also purposed to facilitate configuration management in defining a Management Information Base (MIB) that describes the network capabilities to be used for managing network devices through policies [WSH03].

Concurrently, the Resource Allocation Protocol (RAP) WG has submitted a policy-based network management architecture (PBNM) [YPG00]. In this approach, a central decision server, called Policy Decision Point (PDP), provides the underlying equipments, called Policy Enforcement Points (PEP), with configuration directives to be followed with policies. The policies are a set of conditions and actions. The evaluation of the conditions entails the execution of the actions. The policy exchange between the PDP and its PEPs may be achieved by using SNMP, Diameter, LDAP, or a proprietary communication protocol. However, the RAP Working Group has created the Common Open Policy Service (COPS[DBC$^+$00]) protocol for that purpose.
The IETF architecture, as known as PEP/PDP architecture, has been declared Informational RFC during the 54th IETF meeting in July 2002. At the same time, the COPS transport protocol became an Informational Draft.

2.2 The Blanks in the Current IETF Architecture

The IETF architecture has been accepted by academic and industrial researchers as the basic architecture for implementing advanced network management solutions. Nevertheless, the major drawback of this architecture and the policy representation model purposals is the lack of transition models between the SP-client SLA and the network devices. For example, to set up a videoconference - contractually defined with a delay, a loss rate, a jitter and a bandwidth rate - the SP network administrator must know the scheduling and queuing algorithms (among others) to be used, in order to specify policies for well configuring routers and ensuring a good service delivery. Today's network equipment diversity makes it difficult to achieve correctly, due to the lack of equipment independent transition mechanism between the videoconference parameters and the network equipment configurations.

2.3 How to Fill the Gap?

The likely approach to fill the gap introduced in the previous section should be to dissociate the network infrastructure from the offered services. Adding a further abstraction level is necessary. This abstraction level provides the administrator with the possibility to deploy services without having to know which device parameters to configure. This abstraction level allows the service level agreement translation in an equipment-independent configuration. In the previous example, the SP administrator must only specify the QoS (rate, delay, jitter) and security parameters that are written in the agreement.
Our innovative management architecture purpose, Access Network - Policy-Based Management (AN-PBM), is based on the IETF PEP/PDP architecture.

The objective of the AN-PBM architecture is to provide ISPs and their customers with tools enabling them to control their network by managing their services rather than configuring their network devices.

2.4 AN-PBM Architecture

The AN-PBM architecture allows a service provider to manage its traded services and give its customers the possibility to personalize the services they buy. For this purpose, two policy classes have been designed in the AN-PBM architecture: **SP Policies** (SP) and **Customer Policies** (CPP).
According to the agreement binding the SP and its customer, the SP provides the customer's access router with first class policies. The SP administrator is the only person able to make a change in the policy content. Nevertheless, the customer can define his own policies on his access router in order to personalize the services he subscribed to. For instance, while establishing a VPN, the customer negotiates with the SP the security and QoS parameters for this VPN. With the AN-PBM architecture, the client can customize the VPN service and authorise and deny flows the way he wants without having to notify the SP.
However, conflicts between SP and Customer policies may occur. A solution consists in giving a higher priority to the SP policies and to add a referee control mechanism on the access router (CPE).
The AN-PBM architecture differentiates the SP and customer rights, using two main components: the Management Center and the customer access router (CPE).
Thanks to the Management Center, the Service Provider can offer services, translate these services into policies according to the customer requirements and deploy these services in the customer equipments.
The CPE receives the policies sent by the Management Center and installs them locally. The customer can then adapt his QoS access network in generating and installing Customer policies in the CPE.

3 AN-PBM for QoS

In the previous parts, DiffServ and the IETF Policy-Based Network Management architecture have been introduced. QoS appears as a main problem in network management, and its use is a necessity for ISP whishing to offer innovative services based on the differentiation between users and/or applications [Jud01]. However, the IETF PEP/PDP architecture is incomplete even through it is a worthy foundation, since ISP and customer needs have not been translated into suitable policies. That is the reason why we have decided to adapt the IETF architecture with respect to the ISPs and their customers requirements.

3.1 QoS Objectives

The goal of the AN-PBM architecture is to guarantee the quality of service for SPs and customers.

Service Providers. SPs needs are summarized as follows:

– Offer of differentiated and guaranteed services: bandwidth, delay, jitter and different service levels.
– Service management rather equipment configuration: dynamic and global view of the network, auto-adaptation to solve the problems, and fast and easy configuration of network equipment.

Customers. Customer needs are different from SP needs. Customer needs are focused on the Internet access, for what they have to prioritise critical applications and "privileged" users. Therefore, it is necessary to write dynamic policies granting different QoS levels according to applications and users.

3.2 Architecture

The AN-PBM architecture has two main components (figure 1):

– The Management Center takes on the SP sold SLA guarantee,
– The PE and the CPE, known as "PE/CPE" generic bloc. The CPE is the access router linking the customer to the SP core network. The PE binds a set of CPEs.

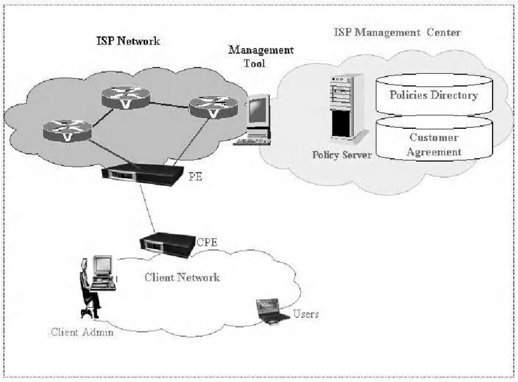


Fig. 1. AN-PBM Architecture (2)

Management Center. The Management Center is the component of the architecture related to the SP. The Management Center is responsible for the SLA negotiation, the generation of relevant policies and the application of these policies in the network devices (PE/CPE). The Management Center also allows to monitor and deploy SP services. The Management Center is a set of five modules:

- The Service Portal (SPo): the business interface between the service provider and the customers. The Service Portal provides the customers with a graphical SLA negotiation interface and a graphical service trade interface.
- The Customer Agreement Database (CAD): the database where are stored the SP-customer SLAs. The Customer Agreement Database security must be paid careful attention.
- The Policy Server (PS): the core of the Management Center. This module is the equivalent of the PDP in the IETF PEP/PDP architecture. The Policy Server is responsible for:
  - generating the set of SP policies that will be enforced in the network equipment to ensure the enforcement of the negotiated SLAs, the service levels, and to monitor the SP network.
  - enforcing the policies in equipment. If any problem occurs, the PS must take a decision to solve it.
  - monitoring the network by the way of reports periodically sent to the PS. The PS deduces a global view of the network and takes decisions when a problem occurs.
- Policy Database (PDB): the storage of the PS policies.
- Management Tool (MNT): used by the network administrator to manage the PS, the databases and the Service Portal.

PE/CPE. In the AN-PBM architecture, the PE/CPE is the equivalent of the PEP in the IETF architecture. However, the PE/CPE is more "intelligent" than a simple PEP since it gives possibility for the operator to customize policies and for the customer to insert its own policies for its own needs. The PE/CPE component plays the following roles:

- enforcement of the policies sent by the PS,
- translation of high-level policies in proprietary configurations,
- auto-adaptation according to the network state, reconfiguration or new PS policies solicitations,
- periodic delivery of monitoring information up to the PS.

Management Center $\rightleftharpoons$ PE/CPE communication. The Policy Server is the link between the Management Center and the PE/CPE. The communication between the Policy Server and the PE/CPE is achieved via 4 exchange types: provisioning (from PS to PE/CPE through a secured protocol), policy enforcement reports, monitoring information reports (periodically sent from PE/CPE to PS and stating what happens in the access network), and policy sollicitation (when an unknown behaviour occurs, the PE/CPE sends a request to the PS. The PS deals with the problem, takes the appropriate decisions and sends the relevant policies to the PE/CPE).

3.3 Policy Identification

Policies from the AN-PBM architecture are grouped in 2 families: Service Provider policies and Customer policies.

Service Provider policies. Service Provider policies constitute the set of SP requirements for guaranteeing the traded service levels. There are 3 subtypes of Service Provider policies:

- **Class Of Service policies (COS)**: COS policies point to the DiffServ class of service configuration parameters related to the SP-customer SLA. Among these parameters, the service type (EF, AF, BE), the marking (of the DSCP field of this class of service packets), the rate, the queuing algorithms (RED, TailDrop), the traffic in excess management algorithms (dropping, shaping, remarking), the queue sizes, and the allowed packet burst sizes.

- **Monitoring policies (MON)**: MON policies give a global view of the SP network and enable to make an automatic adaptation in the equipment configurations according to the network state. For instance, a SP may specify that when congestion occurs in an AF queue of one POP, the queue sizes for this POP should be increased about 20%.

- **Service policies (SRV)**: SRV policies materialize the added value a SP may offer to his customers. SRV policies allow service deployments, like a VoIP with admission control, a videoconference, or a dynamic VPN establishment. For example, for a VoIP with admission control service, the admission condition is chosen by the customer among a set of choices supplied by the provider, whereas the treatment to be done when the admission control condition is not reached is personalized by the customer.

Customer policies. Customer policies define a set of rules chosen by the customer administrator in order to carefully control the outgoing access of its network. Two subtypes of policies exist:

- **Flow policies (FLW)**: FLW policies group the classification rules for differentiating the applications and the users. For instance:

  -
    ```
    if application==http then
        if user==CEO then marking AF1.1
                     else marking BE
    ```
  - web traffic is prohibited from twelve o'clock to 2pm

- **Customer Monitoring policies (CMP)**: CMP policies enable the modification of a CPE classification rule according to the network state. For example, the network administrator can specify that when congestion occurs in AF class of the customer access router, the ftp traffic has to go in the BE class.

3.4 Policy Implementation in 6WIND Equipment

The deployment and the storage of a policy differ depending on the policy family. SP policies are stored in the Management Center, performed by the PS and sent to the PE/CPEs where they are enforced. Customer policies are stored in the CPE and translated into local classification rules.

Service Provider policies. The implementation of SP policies are detailed in this section.

– **COS policies**: the official document binding a customer and a service provider is a SLA, negotiated via the Portal Service where the offered service classes and their parameters are defined. From this SLA, the Policy Server obtains a set of COS policies that are enforced in the customer CPEs. As many COS policies are needed as there are service classes subscribed.

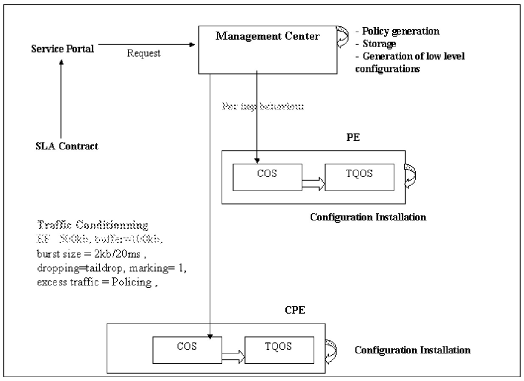


Fig. 2. COS Policies

Configurations are different between PE and CPE. A CPE is responsible for the classification and conditioning of the traffic (shaping, remarking). A CPE is then supplied with the detailed configurations of service classes, that is, with complex mechanisms like the shaping algorithm configuration parameters. A PE, connecting the customer CPE to the SP network, receives policies which contain the Per Hop Behaviour related to the customer service classes. The Policy Server generates 2 policies: one for the PE and one for the CPE. For a 6WINDGate CPE, a component called "COS" receives the policies and translates them into configuration parameters (figure 2). The Security (Setkey) and QoS (TQOS) APIs of the 6WINDGate services are then called.

– **MON policies**: When a problem occurs, the Policy Server generates some policies to modify the underlying CPE configurations. This mechanism is called network auto-adaptation. A component of the 6WINDGate, called "MON", receives these policies and translates them into low-level configurations (figure 3).

– **SRV policies**: SP value-added services representation is done thanks to these policies. In order to make the services independent of the customers, a service is a generic program written in Java. This program is stored in the Management Center, and may be completed for being in concordance with customer choices. The Java program is downloaded in the customer CPE. Each 6WINDGate owns a subscribed services database and an Execution Environment where the service program is performed when the use if the service by the customer is detected.

Customer policies. Thanks to the embedded configuration interface on the CPE, the customer network administrator is able to specify FLW and CMP policies at any time. These policies are not sent to the Policy Server since the Policy Server does not need to know the traffic travelling in the sold service classes, only in case a problem occurs when installing a policy.

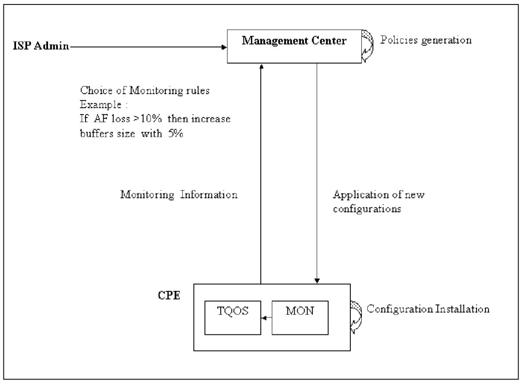

ISP Admin
Management Center
Policies generation
Choice of Monitoring rules
Example :
If AF loss >10% then increase buffers size with 5%
Monitoring Information
Application of new configurations
CPE
TQOS
MON
Configuration Installation

Fig. 3. MON Policies

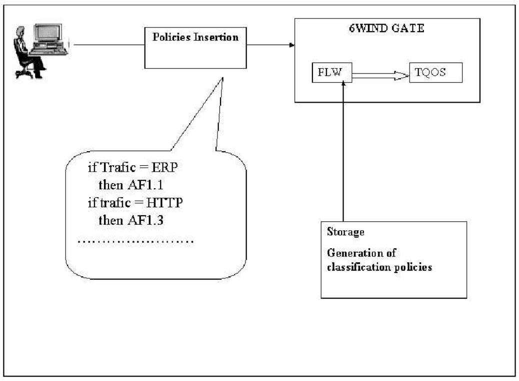

Policies Insertion
6WIND GATE
FLW
TQOS
if Trafic = ERP then AF1.1
if trafic = HTTP then AF1.3
....
Storage
Generation of classification policies

Fig. 4. FLW Policies

– **FLW policies**: the customer network administrator can make a differentiation between users and applications inside his network, in specifying the different QoS levels related to the flows. For example:

```
If User=Lambda and Traffic=beta
then mark flow with class X ..(1)
```

The administrator then insert in the rule the IP addresses and masks, as well as TCP and UDP ports used by the application. IP addresses may be specified statically or dynamically in alerting the CPE when the user connects to the network. A 6WINDGate component, called "FLW", is responsible for storing the FLW policies, generating the classification rules sent to the CPE QoS management module (figure 4).

– **CMP policies**: CMP policies allow to specify classification rules appropriate for network problems occurring during the lifetime of a service. The customer network administrator can give some privileges to critical applications and priorities to the users to provide different access qualities. A 6WINDGate component, called "CMP", is responsible for storing the CMP policies and generating the related new classification rules.

4 The VoIP with Admission Control Service

To illustrate the use of the AN-PBM architecture, we can use a service of VoIP with admission control. This service may be summarized as follows: if the access condition is true, then any new VoIP session is accepted. Else, any new session is rejected. The treatment performed when a session is accepted may be marking the relevant flow with the EF class DCSP value. The access condition is negotiated between the Service Provider and the customer when negotiating the service membership. For instance, an access condition could be that the EF class use rate must be less than 0.8, or could be that the session is accepted if it is initiated outside the working time. In the example, we consider that a new session is accepted if the number of ongoing sessions does not exceed 50. In the AN-PBM architecture, the deployment of such a service consists in the enforcement of a Service Policy (SRV) in the customer access router:

```
(VoIP Service)
If (AC Condition) then accept session
                  else reject session
```

Nevertheless, a customer may wish to control more carefully the service. Indeed, the CEO of the company will see his (her) VoIP session rejected if more than 50 employees are phoning. Such a scenario is unacceptable and compels the Service Provider to allow the company to define one or more further access conditions. The customer administrator can personalized the service in defining some Company Monitoring Policies (CMP).

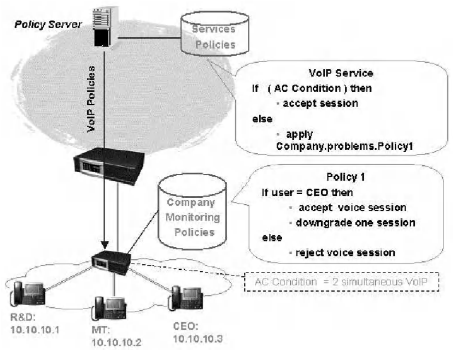


Fig. 5. Service Provisionning

```
(Company Monitoring Policy 1)
If user = CEO then accept session, downgrade one session
              else reject session
```

The Service Policy (SRV) deployed by the Service Provider becomes therefore:

```
(VoIP Service)
If (AC Condition ) then accept session
                   else company policy (Company Monitoring Policy 1)
```

Through this mechanism, the customer subscribes to a service on which he keeps a control level sufficient for solving his internal requirements. The Service Policy defined previously is sent by the Policy Server to the CPE (figure 5). This policy is then transformed into the CPE in a small program for generating suitable classification rules in the 6WINDGate Execution Environment.
Thus, any classification rule is added in the CPE classifier, but after a VoIP flow is detected, a Flow Policy (FLW) is automatically generated and the related

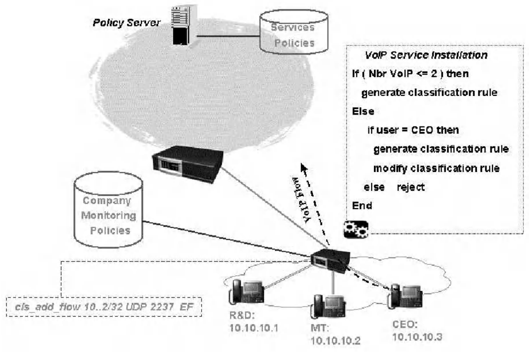

Policy Server
Services Policies
VoIP Service Installation
If (Nbr VoIP <= 2) then
generate classification rule
Else
if user = CEO then
generate classification rule
modify classification rule
else reject
End
VoIP flow
Company Monitoring Policies
cls_add_flow 10..2/32 UDP 2237 EF
R&D: 10.10.10.1
MT: 10.10.10.2
CEO: 10.10.10.3

Fig. 6. Service Installation

classification rule is dynamically added in the CPE classifier (figure 6).
The service deployment is dynamic. However, it is necessary to bring a tool or a module in the AN-PBM architecture to be able to recognize the nature of the flow travelling through the customer CPE.
The AN-PBM is the warranty of easy service deployments and a good work of these services. Moreover, any information that the customer owns(like IP addresses of machines or staff first names) remains confidential. These data are only used while the classification rules are written and stored in a database of local variables. This database enables for instance the SP to provide a customer person - the CEO for example - with an appropriate service. For that, a variable called CEO in the provisioned Service Policy is used, and the exact value of this variable - the CEO computer IP address - is stored in the CPE database of local variables. The IP address is then used when writing the CPE classification rule. The AN-PBM architecture comes up to the Service Provider and customer expectations and ensures a high security level.

5 Conclusions

In this paper, the Access Network - Policy-Based Networking architecture, based on the IETF PEP/PDP architecture, has been overviewed. The lack of transition mechanisms in the IETF architecture, from the provider-customer signed agreement to the network level equipment configurations, led us to propose this architecture. The AN-PBM architecture allows Service Providers to offer value-added services and customers that buy these services to personalize them and gain control over their access network QoS. The Service Provider and customer administration areas are separated, that is why the AN-PBM architecture keeps a high security level. The AN-PBM architecture is under implementation and the testbed scenario is the VoIP with admission control application. This scenario brings up a use of policies from the Service Provider to deploy the service, and from the client to customize the service. A study is already led in order to extend the AN-PBM architecture to the mobility and security problems.

References

[DBC+00] David Durham, Jim Boyle, Ron Cohen, Shai Herzog, Raju Rajan, and Arun Sastry. The COPS Protocol. RFC 2748, January 2000.

[DMT] DMTF. *CIM Standard Schema.* http://www.dmtf.org/standards/standard_cim.php.

[DMT98] DMTF. *DEN initiative webpage*, 1998. http://www.dmtf.org/standards/standard_den.php.

[DMT99] DMTF. *Common Information Model (CIM), Specification, Version 2.2*, June 1999. www.dmtf.org/spec/cim_spec_v22.

[JRV01] J Jason, L Rafalow, and E Vyncke. Internet Draft: IPsec Configuration Policy Model. draft-ietf-policy-pcim-ext-08.txt, November 2001.

[Jud01] Michael Jude. Policy-Based Management: beyond the hype. *Business Communication Review*, pages 51–56, March 2001.

[Kos01] David Kosiur. *Understanding Policy-Based Networking.* Wiley Computer Publishing, 2001.

[MESW01] Bob Moore, Ed Ellesson, John Strassner, and Andrea Westerinen. Policy Core Information Model – Version 1 Specification. RFC 3060, February 2001.

[MRR$^+$01] B Moore, L Rafalow, Y Ramberg, Y Snir, A Westerinen, R Chadha, M Brunner, R Cohen, and J Strassner. Internet Draft: Policy Core Information Model Extensions. draft-ietf-policy-pcim-ext-08.txt, November 2001.

[SRS$^+$01] Y Snir, Y Ramberg, J Strassner, R Cohen, and B Moore. Internet Draft: Policy QoS Information Model. draft-ietf-policy-qos-info-model-04.txt, November 2001.

[WSH03] S Waldbusser, J Saperia, and T Hongal. Internet Draft: Policy Based Management MIB. draft-ietf-snmpconf-pm-13.txt, March 2003.

[YPG00] Raj Yavatkar, Dimitrios Pendarakis, and Roch Guerin. A Framework for Policy-Based Admission Control. RFC 2753, Informational, January 2000.

Administrative Policies to Regulate Quality of Service Management in Distributed Multimedia Applications[⋆]

Michael Katchabaw, Hanan Lutfiyya, and Michael Bauer

Department of Computer Science
The University of Western Ontario
London, Ontario, Canada N6A 5B7

Abstract. Properly capturing and handing administrative requirements for Quality of Service (QoS) management is a challenging, infrequently studied, problem. In this paper, we formalize administrative requirements as administrative policies, and use policy-based management techniques for enforcing them. Doing so adds a great deal of flexibility and power to QoS management. We discuss our general policy-based approach to QoS management, provide several examples of administrative policies, and present highlights from the use of a prototype policy-based QoS management system that uses administrative policies in a variety of experiments with a distributed multimedia application.

Keywords: Administrative policies, QoS management

1 Introduction

An application's Quality of Service (QoS) refers to its non-functional, run-time requirements. An example QoS requirement for an application receiving a video stream is the following: "The number of video frames displayed per second must be 25, plus or minus 2 frames". In addition to performance requirements, an application's QoS may include availability and reliability requirements. A QoS requirement is *hard* for an application if the application is not functionally correct if the QoS requirement is not satisfied at run-time. Otherwise, the QoS requirement is said to be *soft*. Examples of hard QoS requirements can be found in flight control systems and patient monitoring systems. Most distributed multimedia applications, however, have *soft* QoS requirements. Application QoS requirements for multimedia applications are also often dynamic in that they may vary for different users of the same application, for the same user of the application at different times, or even during a single session of the application.

[⋆] An extended version of this paper can be found as Technical Report #596, Department of Computer Science, The University of Western Ontario.

The allocation and scheduling of computing resources is referred to as *QoS management*. QoS management techniques such as resource reservation and admission control can be used to guarantee QoS requirements, since resource reservations are based on worst-case scenarios. This is useful for applications that have hard QoS requirements, but often leads to inefficient resource utilisation in environments that primarily have applications with soft or dynamic QoS requirements. Such approaches also tend to encounter difficulties in environments in which all resources are not under the same administrative control.

We have developed a QoS management system that deals with soft and dynamic QoS requirements by providing management services that detect when an application's run-time behaviour does not satisfy the application's QoS requirements, determine the possible causes of the violation of requirements, and formulate corrective actions to resolve the situation. These actions depend not only on the cause of the violation, but also depend on the constraints imposed on how to achieve the QoS requirement. These constraints are referred to as *administrative* requirements. For example, administrators may want to prioritize applications so that if the system is overloaded, only high priority applications get the computing resources needed to ensure their QoS requirements are satisfied. Administrative requirements can also attempt to prevent QoS violations. For example, administrators may wish to limit access to the computing environment to protect previously admitted applications from QoS requirement violations.

Administrative requirements that are implicitly structured into the code of the QoS management system makes the system rigid, inflexible, and unable to cope when changes are needed to these requirements or when new administrative requirements are to be put in place. To address this problem, the QoS management system that we have developed is *policy-based*. A *policy* is defined [1,7] as a rule that describes the actions to occur when a specific condition occurs. Policies can be used to semi-formally express both QoS and administrative requirements. Using policies separates the process of formulating management decisions from the management system mechanisms carrying out these decisions, thus allowing for different management decisions to be executed by the same mechanisms.

In our earlier work in [6], we presented a simple policy-based QoS management system capable of supporting a single administrative policy for service differentiation. This work, however, was somewhat crude, limited, and ad hoc. Since then, we have demonstrated how the QoS management system [8] services can be mapped to the IETF architectural framework [7] for a more comprehensive solution. This work provided a needed upgrade to the policy distribution and handling mechanisms from our work in [6]. In our current work, discussed in this paper, we extend this previous work to handle a much richer, broader, and more powerful variety of administrative policies to meet the requirements of modern computing environments. Experimental results derived from our improved prototype system using these new administrative policies are also discussed.

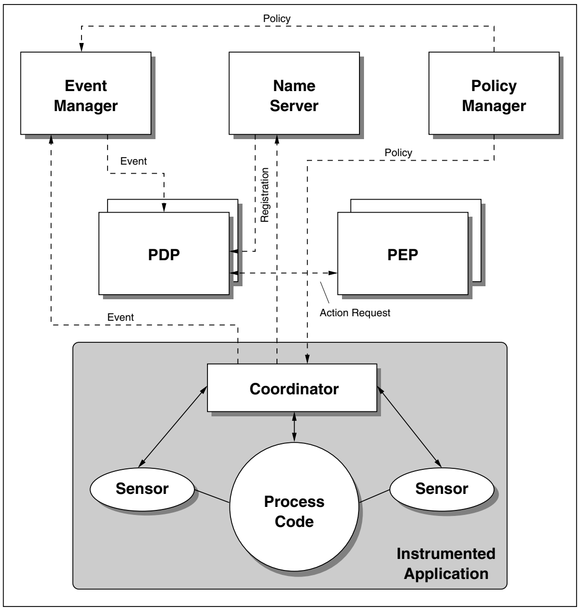


Fig. 1. Policy Architecture

2 Policy Architecture

We first describe the components of our policy architecture, depicted in Figure 1, and their interactions using the following informal QoS requirement.

Example 1 *A video client is to receive video at a frame rate of 25 frames per second, plus or minus 2 frames.*

Coordinator. The coordinator is the interface between a process and the management system. QoS requirements are defined in terms of application specific attributes. An attribute of an application can be monitored with a Sensor and thresholds can be associated with a sensor so that a sensor will only report to the coordinator when the monitored data's value is not within the specified threshold. For Example 1, the attribute to be monitored by a sensor is the video frame rate. If the frame rate is below 23 or above 27, then a report of this is sent to the Coordinator, which in turn forwards an event report to the Event Manager.
Event Manager. An Event Manager receives and aggregates event reports (from violations of QoS requirements or other monitored data) and allows other management entities to register an interest in an event. For Example 1, the Event Manager receives the report from the Coordinator and passes this information on to a Policy Decision Point that has registered interest in this kind of event to resolve the situation.
Name Server. The Name Server receives and maintains registration information in a repository. It assigns a unique instance identifier for each registered process,

thereby enabling unique identification of each registered component. The Name Server coordinates the interaction between the QoS management components and the Coordinator during the initialization process for an application by notifying the necessary Policy Decision Points of a newly registered process.

Policy Decision Points (PDPs). PDPs are used to make decisions on actions to be taken based on the receipt of an event which is generated from monitoring and relayed through an Event Manager. PDPs are responsible for diagnostics, by determining the cause of the QoS requirement violation and then determining corrective actions. For Example 1, the PDP that received the report from the Event Manager carries out diagnostics to determine how to resolve the situation and notifies the appropriate Policy Enforcement Point of the actions necessary. If the video client in this example has a frame rate below 23, the PDP may determine that the action to be taken is to increase the CPU resources of the client. PDPs are also used to authorize actions on behalf of Policy Enforcement Points. Furthermore, within an administrative domain, there is one PDP that communicates with a repository where QoS and administrative policies are stored, and information about users is kept. This PDP is referred to as a Policy Manager.

Policy Enforcement Points (PEPs). PEPs apply the actions determined by PDPs. A PEP is a management process that is responsible for monitoring the device that it is on and executing actions. A PEP receives requests from a PDP, verifies that the action to be executed is allowed (by consulting with another PDP and using administrative policies), and performs the requested action if permitted. For Example 1, when the PEP receives the instruction to increase CPU resources for the video client from the PDP, it will verify this action with an authorizing PDP, and then add CPU resources if the action is permitted.

3 Policy Specification

In our current work, policies are formally described using a notation called Ponder [1]. Our use of Ponder as a specification language is intended to make it simpler for administrators to specify policies. The Ponder language provides constructs to define policy types and instances of the policy types. In other words, a policy can be thought of as a class that can be instantiated with specific elements; for example, host machine names or process identifiers.

In this formalism, an *obligation* policy specifies the action that a subject must perform on a set of target objects when an event occurs. A *positive authorization* policy specifies permissible actions and a *negative authorization* policy specifies the forbidden actions. Actions specified are carried out by subjects on the specified targets. Targets and subjects are specified using domains, providing a mechanism for applying a policy to a collection of objects or for specifying the collection of objects that can carry out given actions. Administration requirements can be specified using a mixture of Ponder obligation and authorization policy constructs.

Example 2 This policy, a formalization of the requirement in Example 1, states that the PEP is allowed to increase the CPU priority for a video client process if the process belongs to GroupA and if there are enough CPU resources. This is calculated by CPUResourcesAvailable. The parameters to this action include the process identifier (ProcessId) and a normalised value (normvalue) representing the difference between the attribute's current value and the expected value.

```
type auth+ authCPUIncreaseT (subject s, target t) {
  action  CPUIncrease(ProcessId,normvalue)
  when belongs(GroupA, ProcessId) and CPUResourcesAvailable(normvalue); }
```

4 Policies Studied

This section presents several different administrative policies. Each policy is described informally, and also defined formally using Ponder.

4.1 Admission Control

Policies can be used to limit the number of violations of QoS requirements by limiting the number of applications executing in the environment. Admission control refers to the process of comparing projected application resource needs with available resources to determine if an application should be allowed to consume computing resources. Upon application registration, policies are used to determine if an application may continue. This can be application specific.

Example 3 In this policy, the PEP is authorized to admit a process ProcessId when at least 5% of the CPU and 100 pages of free physical memory are available.

```
type auth+ authAdmissionControl (subject s, target t) {
  action  admitProcess(ProcessId)
  when CPUResourcesAvailabibility() > 5 and FreeMemory > 100; }
```

4.2 Differentiated Services

It is not always possible to satisfy quality of service requirements for all applications at the same time. There are a number of possible ways to deal with this scenario, depending on the answers to a variety of questions: Should all applications receive equal service or should some receive preferential service to meet requirements? If the latter, how would the application to favour be selected? What kind of favouring would be provided? These decisions can be formulated as service differentiation policies. There are two types of such policies: uniform service policies and priority-based differentiated service policies.

An example of a priority-based differentiated service policy was given earlier in Example 2. Another policy is needed to handle the case when the CPU resources are not available. An example is given below.

Example 4 This policy specifies that the PEP requests that the video client process's resolution is to be changed if there are not enough CPU resources available. This should allow for the application to maintain the desired frame rate, but at the expense of a lower picture quality. The implementation of of this is done through the use of an actuator, which is an instrumentation component of an application [5].

```
type auth+  authChangeApplicationResolution (subject s, target t) {
  action ChangeResolution(ProcessId);
  when not(CPUResourcesAvailable(normalizedvalue)); }
```

Example 5 In addition to the policy stated in Example 2, one may have an additional policy statement that reduces the CPU priority of all processes that are in GroupB. This is only done when CPU availability is low.

```
type auth+ authCPUDecrease (subject s, target t) {
  action  CPUDecrease(ProcessId,normalizedvalue);
  when belongs(GroupB, ProcessId) and (CPUResourcesAvailability() < 5); }
```

Example 6 This uniform service policy basically states that any requesting application will receive an increased CPU priority if the resources are available.

```
type auth+ authCPUIncreaseT (subject s, target t) {
  action  CPUIncrease(ProcessId,normalizedvalue)
  when CPUResourcesAvailable(normalizedvalue); }
```

4.3 User Hints

So far, all the events we have been dealing with are QoS requirement violations. User hints are events that indicate that the user's interest or focus of activity has changed, as discussed in detail in [5]. Hints such such as minimizing and restoring windows, the covering and uncovering of windows, and the activation and deactivation of screen savers and locks are all excellent indicators of the relative interest users have in their applications. A user hint can be used to reduce the amount of host computing resources consumed by an application process that is no longer useful or of interest to the user by reducing its CPU priority, which in turn causes it to consume fewer CPU cycles.

Example 7 In this policy, the PEP is obligated to reduce the CPU priority of a process whose identifier is ProcessId when a user hint event, as described above, has occurred. This policy is assumed to be used by the PDP which has registered an interest in the event UserHint. The action is sent to the appropriate PEP, and is validated through an authorization policy used by the appropriate PDP.

```
type oblig user_hint {subject s, target t) {
    on UserHint(ProcessId)
    do CPUDecrease(ProcessId,normalizedvalue}; }
```

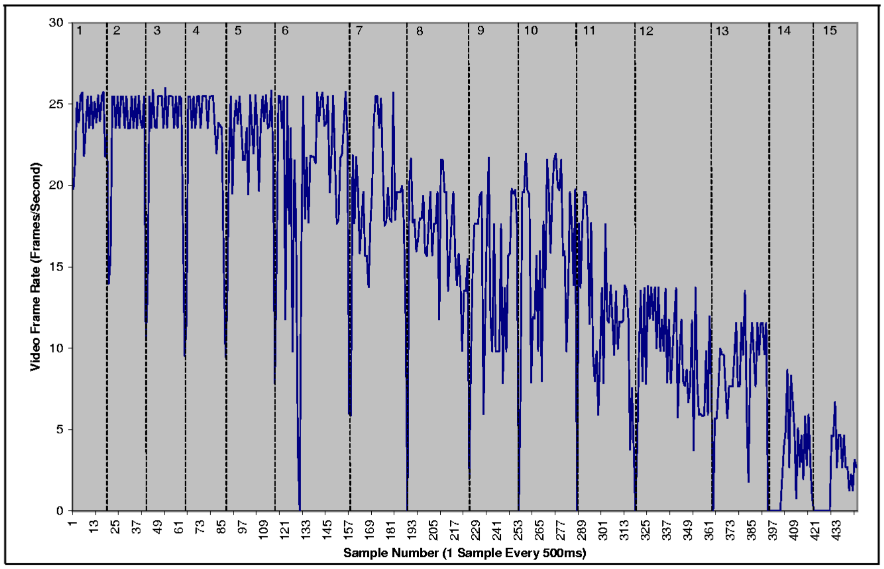


Fig. 2. Video-on-Demand without Admission Control

5 Experience

Using the policy based management techniques and sample policies from the previous sections, we have implemented a prototype policy-based QoS management system. In this section, we discuss our experience with this system to date.

5.1 Prototype

We have developed a prototype policy-based QoS management system for Solaris 2.6. This prototype was based on work discussed in [6] and extended in [5] and again in [8]. It is capable of managing and regulating access to multiple computing resources, including CPU cycles and memory, with integrated support for networking resources nearing completion. It also supports a wide variety of distributed multimedia applications, as well as more traditional applications.

The policy engine in our prototype, used for formulating and executing management decisions, is built on the Java Expert System Shell (JESS) [3], which is compatible with the C Language Integrated Production System (CLIPS) [4] used in our earlier work. Policies are specified using Ponder, and then mapped into JESS rules for use in our prototype system.

5.2 Experimental Results

For experimentation demonstrating the effectiveness of administrative policies, we chose to use a distributed client-server video-on-demand application. In response to a request from a client, the server streams a given MPEG video over

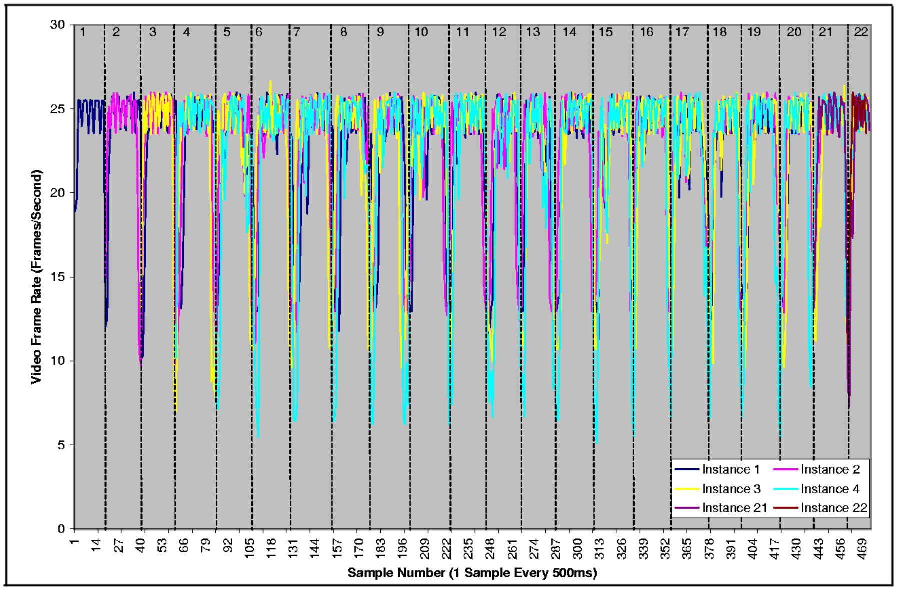


Fig. 3. Video-on-Demand with Admission Control

a TCP connection back to the client, which decodes the video and presents it to the user. Three Sun Microsystems UltraSparc workstations with Solaris 2.6 were used in experimentation: farquarson, strawberry, and vanilla.

Admission Control. For these experiments, the host farquarson was used to serve multiple video feeds to clients launched on strawberry. Every 10 seconds, a new instance of the video-on-demand application client was launched on this host, and instructed to play the same small 80×60 8-bit video from a file at a mean frame rate of 25 frames per second, plus or minus 2 frames (yielding an acceptable quality range of 23 to 27 frames per second). The video was small enough to be pre-loaded into memory once, and fed to clients without further disk access at run-time.

Figure 2 shows a representative run from the first instance of the video-on-demand client started during experimentation. As noted in the figure, additional instances of the application were started roughly 10 seconds apart (sometimes more, depending on the load on the system induced by other instances of the application). As can be seen from the figure, quality is quite good until the fifth instance of the player is added. Variation in quality deliver begins to increase at this point with each subsequent player added, and mean quality drops. Figure 3 shows a representative run of this experiment, with the admission control policy of Section 4.1 in place to limit access to the system to maintain quality of admitted applications. In this case, our prototype system admitted only four

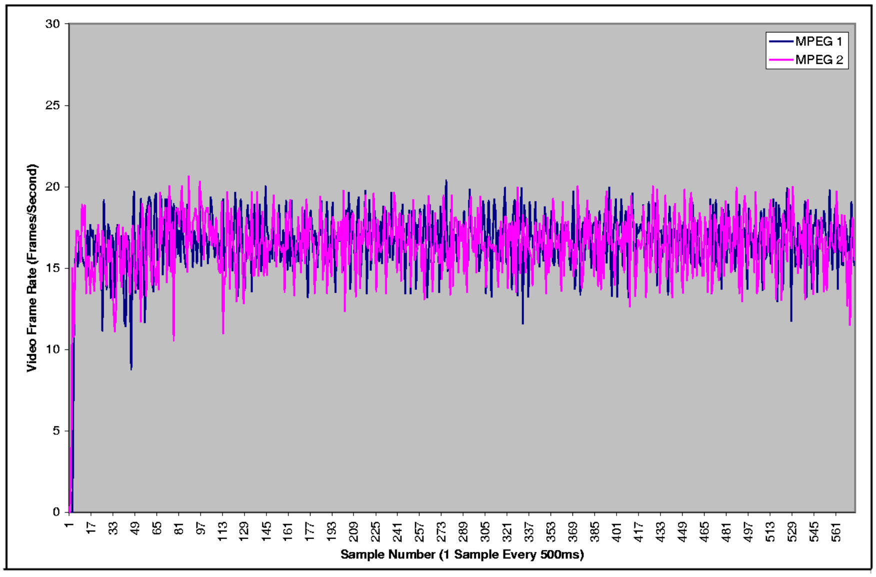


Fig. 4. Video-on-Demand with Uniform Services

instances of the application at a time, which allowed quality to be delivered in accordance with QoS requirements.

Differentiated Services. For experimentation with service differentiation, we chose to use the same application, with a different workload. The application workload assigned was a 10 minute 320×240 8-bit video of footage recorded by our network operations counter cameras stored in a local file. The expectation policy in place dictated that this workload be delivered at a mean frame rate of 24 frames per second, plus or minus 2 frames per second (yielding an acceptable quality range from 22 to 26 frames per second). In this case, the file was too large to cache in memory, so we used two different servers, on farquarson and on vanilla to stream the video to two clients on strawberry.

Figure 4 shows a representative run of this scenario using the uniform services policy discussed in Section 4.2, in which both instances of the video-on-demand application compete for resources on a first-come, first-serve basis. Neither instance of the application is able to meet its QoS requirement, as both instances evenly share available resources. Figure 5 shows a representative run in this scenario with the service differentiation policies from Example 2 and Example 5 discussed in Section 4.2 in place to favour one application instance over the other. In this case, service differentiation was enacted as soon as the supply of resources on the host was exhausted, yet demand for the resources was still high. Our prototype system scaled back resources allocated to the second instance of the application and reallocated them to the first. Consequently, the first instance

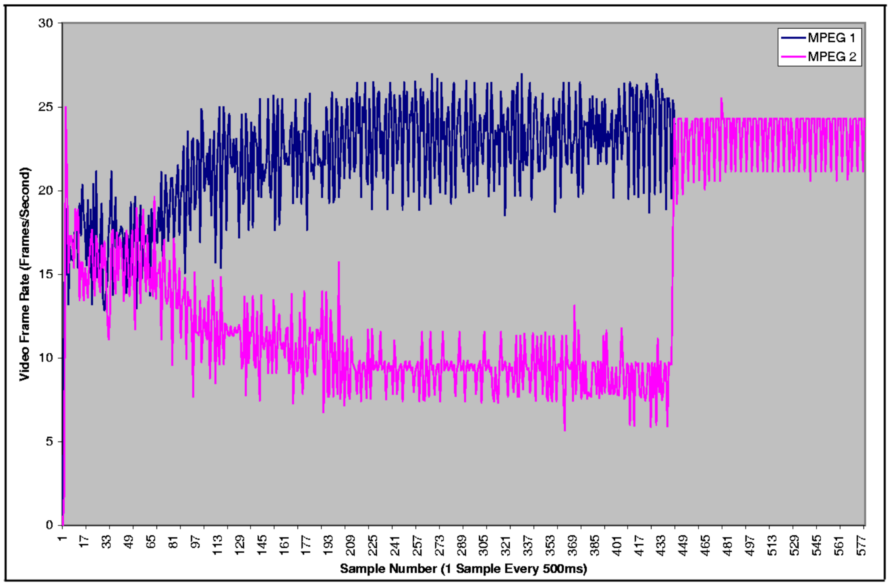


Fig. 5. Video-on-Demand with Differentiated Services

was able to converge and deliver stable quality meeting requirements, while the second was still able to deliver quality at 50% of its target rate.

User Hints. For experimentation with user hints, we used the same scenario used in the previous section for differentiated services experiments. This includes the same application, workload, and host configurations as before.

Figure 6 shows a representative run of this scenario with the user hints policy from Section 4.3 in place. In this case, any user hints indicating a change in user focus or attention would result in a corresponding shift of computing resources. When the application instances started, the video window of one was placed over the other, generating a user hint event, and adjusting resource allocations in its favour. At time A, the ordering of windows was reversed, shifting resources from one instance to the other. At time B, the ordering was reversed once again, resulting in a shift back to the original allocation.

Multiple Administrative Policies. The last scenario examined in experimentation involved the use of our admission control, differentiated services, and user hints policies all at the same time. In this experimentation, we used the same application, workload, and host configuration as in the previous two sections.

Figure 7 shows a representative run from this experimentation. Because sufficient resources were deemed available, both instances of the video-on-demand application were admitted. Windows from both applications were placed in non-overlapping positions at the top of the window stack. Both application instances

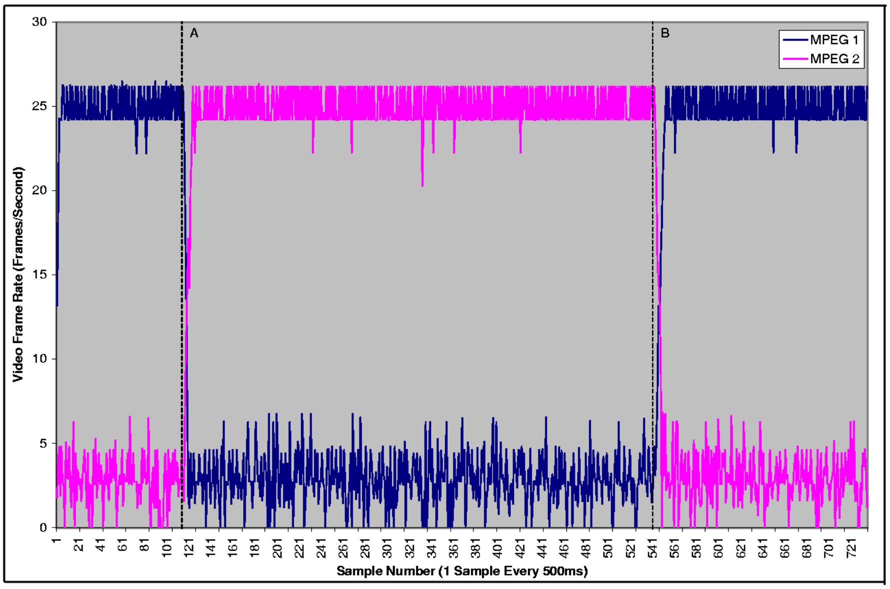


Fig. 6. Video-on-Demand with User Hints

competed for resources until the supply was exhausted, at which time service differentiation kicked in to favour one instance over the other. When windows were repositioned so that the favoured application was covered, the user hints policy overrode service differentiation, and reallocated resources to the uncovered application instance. When this instance terminated, the remaining instance was reallocated resources and completed its execution.

6 Related Work

The IETF is currently developing a set of standards (current drafts found in [7, 11,13]) that includes an information model to be used for specifying policies, a standard that extends the previous standard for specifying policies to specifying QoS policies, and a standard for mapping the information model to LDAP schemas. IETF policies are reasonable for specifying QoS requirements, but we have found that it was much more difficult to specify administrative and diagnostic policies. Generally, we found it easier to specify policies using Ponder and then translate to the IETF format as necessary.

Other language standards have been proposed by the IETF and others, as summarized in [12]. These languages primarily focus on policies applied to a network device; none of this work addresses the use of policies applied to applications. Other policy specification languages (for example, [9] and others) focus on security related policies. The Ponder Policy Specification language [1] has a

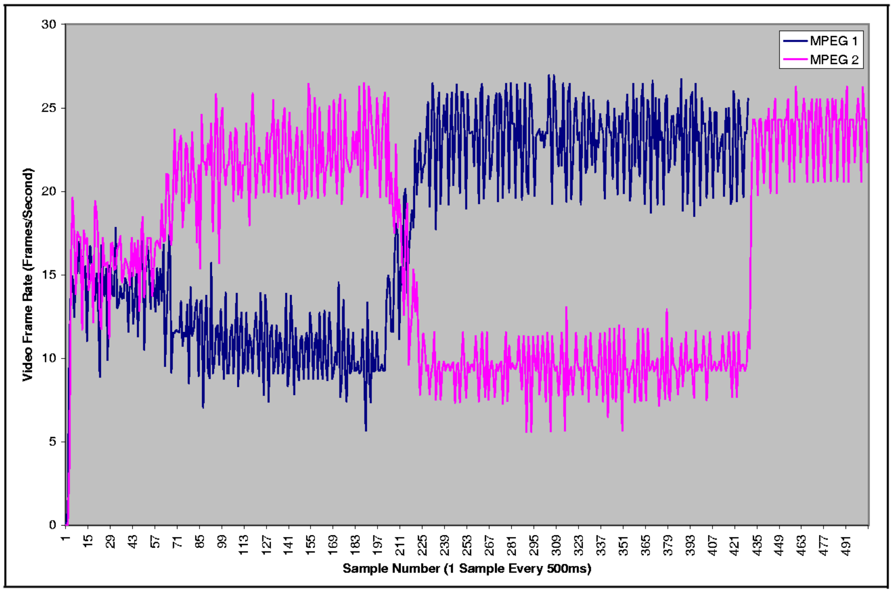


Fig. 7. Video-on-Demand with Usage Based Differentiated Services

broader scope than most of the other languages, in that it not only was designed with specifying security, but also with management policy in general.

There has been some recent work in service level management [10] that describes an architecture and policies for the management of a differentiated services network so that users receive good quality of service and fulfill the service level agreements. The policies are primarily applied to network devices and calculate the drop rate for an incoming line.

There has also been significant work (for example, [14] and others) that has looked at the translation of policies to network device configurations. In most of this work, policy distribution is initiated by an administrator and focusses on one device. The emphasis was on the framework and the effectiveness of the translation as opposed to the effectiveness of different policies.

A deployment model was developed in [2] for Ponder. Each policy type is compiled into a policy class by the Ponder compiler, and instantiated as policy objects. Each policy object has methods that allow a policy object to be loaded to an enforcement agent and unloaded from an agent. Agents register with the event service to receive relevant events (as specified by the policies) generated from the managed objects of the system. There is no discussion on the configuration of devices or applications. Furthermore, this approach has its drawbacks; we found that the use of JESS would make certain tasks easier and more efficient.

Generally, we found that little related work focusses on policies at the application level. We are addressing this oversight through our work in this area.

7 Concluding Remarks

The work we describe here is part of ongoing work on addressing issues in the quality of service management of distributed multimedia applications. This paper focusses on the use of administrative policies to provide flexible and dynamic control over the quality of service management process to balance the individual needs of users against the broader goals of administration. We developed a general policy architecture for facilitating this, and presented a variety of administrative policies that address several quality of service issues. Using a prototype system based on this approach, we have been able to carry out a number of experiments that show the positive effects administrative policies can have on the delivery of quality of service, and demonstrate the effectiveness of our approach.

We have identified several avenues for future work in this area. We need to investigate other administrative policies that are suitable for quality of service management. We need to examine and refine the administrative policies we have already defined to further improve their effectiveness and applicability. To ease adoption and simplify tasks for administrators, we need to investigate more user friendly approaches to policy specification, which can then be translated to Ponder or JESS directly. With an increasing number of administrative policies to activate at the same time, we need to also increase efforts into techniques for detecting and resolving conflicts between policies automatically. We also need to complete on-going work integrating network resource management into our management system, as well as facilities for I/O and disk resource management. We need to continue porting efforts to other platforms, including the Microsoft Windows family, other variants of the Unix environment, and Java.

References

1. N. Damianou, N. Dalay, E. Lupu, and M. Sloman. Ponder: A Language for Specifying Security and Management Policies for Distributed Systems: The Language Specification. *Imperial College Research Report DOC 2000/01, Imperial College of Science, Technology and Medicine*, April 2000.
2. N Dulay, E. Lupu, M. Sloman, and N. Damianou. A Policy Deployment Model for the Ponder Language. *Proceedings of the 7th IEEE/IFIP Symposium on Integrated Network Management (IM'01)*, Seattle USA, May 2001.
3. E. J. Friedman-Hill. *Jess, The Rule Engine for the Java Platform.* Sandia National Laboratories Report (SAND98-8206 (revised)), 2003.
4. J. Giarratano and G. Riley. *Expert Systems: Principles and Programming.* PWS Publishing Company, 1998.
5. M. J. Katchabaw. *Quality of Service Resource Management.* PhD thesis, The University of Western Ontario, June 2002.
6. G. P. Molenkamp, M. J. Katchabaw, H. L. Lutfiyya, and M. A. Bauer. Distributed Resource Management to Support Distributed Application-Specific Quality of Service. *Proceedings of the Fourth IFIP/IEEE International Conference on the Management of Multimedia Networks and Services*, Chicago, Illinois, October 2001.
7. B. Moore, J. Strassmer, and E. Elleson. Policy Core Information Model – Version 1 Specification. Technical report, IETF, May 2000.

8. N. Muruganantha and H. Lutfiyya. Issues in Policy Specification, Distribution and Architecture for Quality of Service Management. *Integrated Network Management, Volume VIII*, March 2003.
9. R. Ortalo. A Flexible Method for Information System Security Policy Specificatio". *Proceedings of 5th European Symposium on Research in Computer Security (ESORICS 98)*, Louvain-laNeuve, Belgium, Springer-Verlag, 1998.
10. P. Pereira, D. Dadok, and P. Pinto. Service Level Management of Differentiated Services Networks with Active Policies. *3rd Conferencia de Telecomunicacoes.*, Rio de Janeiro, Brazil, December 1999.
11. Y. Snir, Y. Ramberg, J. Strassner, and R. Cohen. Policy Framework QoS Information Model. Technical report, IETF, April 2000.
12. G. Stone, B. Lundy, and G. Xie. Network Policy Languages: A Survey and New Approaches. *IEEE Network*, 15(1):10–21, January 2001.
13. J. Strassner, E. Ellesson, B. Moore, and Ryan Moats. Policy Framework LDAP Core Schema. Technical report, IETF, November 1999.
14. P. Trimintzios, I. Andrikopoulos, G. Pavlou, and C. Cavalcanti. An Architectural Framework for Providing QoS in IP Differentiated Services Networks. *Proceedings of the 7th Symposium on Integrated Network Management*, Seattle USA, May 2001.

Predicting Violations of QoS Requirements in Distributed Systems

Sandra Taylor and Hanan Lutfiyya

Department of Computer Science
The University of Western Ontario London, Ontario, Canada N6A 5B7
{taylor,hanan}@csd.uwo.ca

Abstract. A Quality of Service (QoS) requirement refers to a non-functional requirement such as performance. A QoS management system allocates and schedules computing resources. A dynamic QoS management system is one that dynamically allocates resources to an application during its lifetime. This is usually done when the application has a QoS requirement that is not being satisfied at run-time. Ideally, the QoS management system is able to predict when the QoS requirement will be violated before it is violated. This paper describes an approach to prediction and hows shows how this was applied to a case study.

Keywords: Quality of Service, Fault Management, Prediction, Multimedia

1 Introduction

There has been an increase in distributed applications that involve the exchange of data that is time-sensitive. Applications include video-on-demand, distance education, tele-medicine, tele-conferencing and electronic commerce. Users of these applications expect them to perform at acceptable levels, that is, they expect a high level of quality of service (QoS). Quality of service in this context refers to non-functional, run-time (or operational) requirements, such as the application's performance or availability. A possible QoS requirement for an application receiving a video stream is the following: "The number of video frames per second displayed must be 25 plus or minus 2 frames". A QoS requirement is *soft* for an application, if the application is functionally correct even though the QoS requirement is not satisfied at run-time. Otherwise, the QoS requirement is said to be *hard* (e.g., flight control systems, patient monitoring systems). This work primarily focusses on soft QoS requirements.

The term *QoS management* is used to refer to the allocation and scheduling of computing resources. QoS management techniques, such as resource reservation and admission control techniques (e.g., [4]), can be used to guarantee QoS requirements, but these techniques usually base the resource reservation on worst-case scenarios which leads to inefficient resource utilisation. This is important for applications that have hard QoS requirements but is not usually needed for applications with soft QoS requirements since these applications are still considered functionally correct if the QoS requirement is not satisfied. Other QoS management techniques focus on either having applications adapt their behaviour based on reduced resource availability (e.g., change of video resolution) or make adjustments to resource usage in the system.

We developed a QoS management system [6] that deals with soft and dynamic QoS requirements by providing management services (implemented by a set of management processes and resource managers that support the following: (1) Detecting that an application's run-time behaviour does not satisfy the application's QoS requirements. This violation of QoS requirements (also called a *symptom*) is a manifestation of a fault in the system. (2) Determine (based on a set of symptoms) a set of hypothesis identifying possible causes of a location of the fault causing the violation of the QoS requirements. (3) Adaptation which may take either the form of resource allocation adjustments or application behaviour adjustments.

Currently, detecting that an application's QoS requirement is violated occurs after monitoring shows that the application's attributes used in the specification of the QoS requirement does not satisfy the constraints specified in the QoS requirement. In the example of an application receiving a video stream, the QoS requirement is detected to be violated when the frame rate is less than 23 or more than 27. If this QoS requirement is not being satisfied, it is most likely that the user has noticed that the application is not executing as expected.

Ideally, the QoS management system will be able to predict when the QoS requirement will be violated, i.e., determine that the QoS requirement will be violated before it is actually violated, and make the necessary adjustments. It may not always to be possible to make the necessary adjustments before the QoS requirement is violated, but at the very least the management system is more responsive.

This paper is organised as follows. Section 2 defines configuration models and describes how configuration models are related to detection. Section 3 analyses an application's QoS requirements, and shows how prediction rules may be generated. Section 4 describes an architecture and implementation that is able to do the prediction. Section 5 provides an analysis of the effectiveness of the methods described in Section 3. Section 6 provides a discussion. Section 7 describes related work and Section 8 provides conclusions.

2 Relationship between Configuration Models and Detection

This section describes the relationship between configuration models and diagnostics. This relationship can be used to derive rules for prediction. The relationship is illustrated by examining multimedia applications.

Defining Configuration Model

Fig. 1 illustrates the internal structure of the application and the underlying system from an end-to-end perspective.

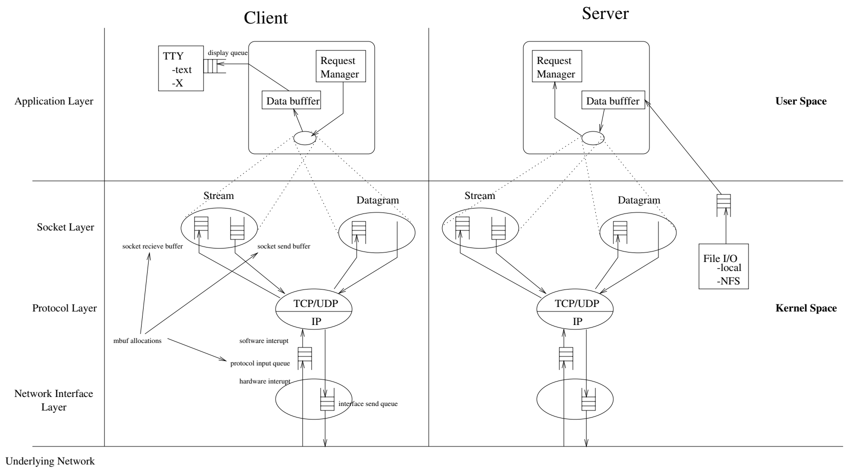


Fig. 1. Configuration Model of Multimedia Application

The data for the video image is retrieved from the disk into a buffer in user space. This gets copied into data buffers (*mbuf* for Unix System 4 kernels) that are used by the TCP/UDP protocol processes. This progresses through to the Protocol layer where the IP processing occurs. Each layer has data buffers as presented in Fig. 1.

The client side consists of a process (called the Request Manager) that has its own communication data buffer for transferring data. The client request manager sends the processed frames to the display process which is a text display or it can continue through an X server event queue to a graphical display system or to any other device.

Generally for any application the internal structure of the application and the underlying system can be represented as a graph $C = (V, E)$, where V represents the structural components of the application and system (in terms of the services provided) and for any $o_i, o_j \in V$ there exists $(o_i, o_j) \in E$ if o_i has a direct dependency on o_j i.e., o_i needs o_j in order to be functionally correct. This is refered to as a *configuration model.*

From the configuration model, *dependency chains* are identified. If an object o_i is dependent on o_{i+1} and o_{i+1} is dependent on o_{i+2} and so on until we have o_{i+j-1} is dependent on o_{i+j} (where $j \geq 1$) then o_i is said to be indirectly dependent on o_{i+j}. $o_i, o_{i+1}, ..., o_{i+j}$ is said to be *dependency chain* if for an l and m, where $i \leq l < m \leq i + j$, o_l is indirectly dependent o_m. A dependency chain does not have to have every single component possible in the chain. In the example presented in this section, a possible dependency chain is the following: client display process, client request manager and server request manager.

Identifying Measurements. At this point, we identify those attributes of the components identified in the previous step that characterise its behaviour and can be measured. These are referred to as *observable indicators*. The subset of these observable indicators that are directly used in the specification of QoS requirements are referred to as *QoS attributes*. Assuming that we have a graph $C = (V, E)$ then for each $o_i \in V$, we can associate a set $\{a_{i1}, a_{i2}, ...a_{in}\}$ where a_{ij} is an observable indicator associated with o_i. For our example, observable indicators include the size of each of the incoming buffers. QoS attributes (which have an impact on the user's perception of the satisfaction of QoS requirements) include frames per second. These are observable indicators of the client display process.

Determining Correlations

In this context, a *fault* refers to the lack of a computing resource needed by an application to satisfy its QoS requirements.

We now define the concept of a *chain of failures*. A fault may occur in object o_n. This may cause an error to manifest itself in object o_{n-1}, where $(o_{n-1}, o_n) \in E$. This in turn may manifest itself as an error in object o_{n-2}, where $(o_{n-2}, o_{n-1}) \in E$. This results in a sequence of objects $o_n, o_{n-1}, .., o_1$ where o_n is where the fault occurs and o_1 is assumed to include QoS attributes and is assumed to have no other objects dependent on it. This sequence is a chain of failures. A fault may cause more than one chain of failures.

As an example, consider the dependency chain identified earlier: client display process, client request manager and server request manager. A CPU overload on the client machine impacts the rate at which the client request manager is able to process frames which, in turn, impacts the rate at which the client display process is able to display the frames. Thus, an example of a chain of failures is the following: server request manager, client request manager and client display process.

For each observable indicator a_{kl} we refer to the t^{th} measurement of a value of a_{kl} as a^t_{kl}. This is referred to as a *sample*. Let $S_{a_{kl}}$ denote the set of samples of a_{kl} taken during a run. Let g_k be a function of $S_{a_{kl}}$ and let x be the variable that takes values x_1, x_2, ..., x_n which are computed by $g_k(1)$, $g_k(2)$ and $g_k(n)$ respectively. Let y be a variable that takes values y_1, y_2, ..., y_n where $y_i = a^i_{ij}$. We say that there is a strong dependency between a_{ij} and a_{kl} if o_i and o_k can be found in the same dependency chain, there is a strong statistical correlation and if a relationship can be defined between y and x. If a_{ij} is a QoS attribute then it may be possible to use values of a_{kl} to predict values of a_{ij} (and thus predict when a QoS requirement is going to be violated), if g_k has the property that the value of x_i is contributed by a subset of those a^t_{kl} that represent samples that were taken before a^i_{ij}. In this paper, a_{ij} will be the frame rate and a_{kl} will be the size of the incoming communications buffer of the client's request manager. We will see that $g_k(i)$ results in the sum of the four previous buffer sizes i.e., $x_i = g_k(i) = b_{i-1} + b_{i-2} + b_{i-3} + b_{i-4}$.

It should be noted that it may be possible to use more than one attribute to predict a_{ij}. However, this paper focusses on the use of defining the relationship between two attributes: frame rate and size of the client request manager's communication buffer.

The concept of a configuration model is similar to that of a causality model (for a good overview of causality models and how they are used in fault management see [8]). The primary difference is that this work associates with each component a set of attributes characterising its behaviour.

3 Analysing the RTVC Application

This section describes the experiments and methods of analysis that were used to determine possible correlations for the Real Time Video Conferencing (RTVC) tool which is a part of the Sun Multimedia kit. The configuration model is similar to that of Fig. 2. The example QoS requirement referred to in the rest of this paper is the following: "The number of video frames per second displayed must be 25 plus or minus 2 frames". The QoS attribute of interest is the frame rate. Observable indicators include the size of each of the incoming buffers. We chose to use the size of the client's communication buffer (incoming buffer to the client's request manager).

Experiments were performed on laboratory machines that were isolated from other machines. Loads were generated based on a Poisson distribution.

3.1 Measuring Frame Rate and Communication Buffer Size

The frames per second rate is calculated as follows. Assume that f_i represents the i^{th} occurrence that the frames per second rate is reported and that a report is to be made after every ten frames that have been displayed. Let t_{f_n} represent the system clock time when the n^{th} frame is displayed and let $t_{f_{n+10}}$ represent the system clock time when the $(n+10)^{th}$ frame is displayed. The frame rate is calculated as $(10/(t_{f_{n+10}}$ - $t_{f_n})$ The choice of determining the current frame rate after ten frames was received is based on experimentation. Anything less seemed to cause inaccurate predictions by reporting more false negatives. Anything after ten frames seemed to not be able to predict problems.

We measured the current size of the communication buffer (referred to as buffer throughout the rest of the paper), where b_i denotes the number of bytes occupying the buffer as measured in the ith sample, and the average number of bytes stored in the buffer. This value is calculated by adding 10 per cent of the current number of bytes in the buffer to 90 per cent of the average buffer value calculated in the previous sample. This was done every second.

3.2 Correlation Analysis

Graphing representative runs show that the the frames per second is negatively affected at or near the times when the buffer size spikes. There appears to be a good correlation between the buffer size and the frames displayed per second.

For an example representative run, the arithmetic mean of the sample buffer sizes is $\bar{b} = \sum_{i=1}^{599} b_i = 1996.633$ and the arithmetic mean of the sample frame rates is $\bar{f} = \sum_{i=1}^{599} f_i = 25.622$. The standard deviation for the buffer size, s_b is calculated as follows: $s_b = \sqrt{\frac{1}{n-1}\sum(b_i - \bar{b})^2} = 3319.864$. The standard deviation for the frame rate, s_f is calculated as follows: $s_f = \sqrt{\frac{1}{n-1}\sum(f_i - \bar{f})^2} = 5.919$. These are relatively high measurements of standard deviation.

This then leads to the analysis of the correlation [2] between the two sets of variables. The standard calculation for correlation measures the strength of the linear relationship between two variables. When the two sets of variables being used are represented by x_i and y_i, and the means and standard deviations of the two variables are represented by $\bar{x}$ and s_x for the x values, and $\bar{y}$ and s_y for the y values, the equation is as follows: $r = \frac{1}{n-1}\sum\left(\frac{x_i-\bar{x}}{s_x}\right)\left(\frac{y_i-\bar{y}}{s_y}\right)$.

The measure of the relationship is reflected in a numerical result which lies between -1 and 1. The closer to zero the result is, the weaker the relationship is between the two variables. The strength of the relationship increases as r approaches either 1 or -1. If r is positive, the relationship between the variables is positive, whereas if r is negative, the relationship is negative. As we attempted to determine the best relationship between our two variables, we performed the correlation analysis using $s_x = s_f$, $s_y = s_b$. Table 2 presents values of r for different values computed for x_i.

x_i	**r**
b_i	-0.70258
b_{i-1}	-0.75709
b_{i-2}	-0.72961
b_{i-3}	-0.71012
$\frac{b_{i-1}+b_{i-2}}{2}$	-0.80955
$\frac{b_{i-1}+b_{i-2}+b_{i-3}}{3}$	-0.82667
$\frac{b_{i-1}+b_{i-2}+b_{i-3}+b_{i-4}}{4}$	-0.83506
$\mid b_i - b_{i-1} \mid$	-0.42106

Fig. 2. Correlation Table

Correlation values were computed for $x_i = b_i$, $x_i = b_{i-1}$, $x_i = b_{i-2}$ and $x_i = b_{i-3}$ to get -.70258, -0.75709, -0.72961 and -0.71012 respectively.

These negative values reflect the fact that there is a negative relationship between the values of the buffer size and the frames per second samples; i.e., as the buffer size increases, the frames per second figure decreases, rather than a positive relationship where they would both go in the same direction.

There were several more relationships tested moving the time differential further way, but r continued to decrease in the same pattern as it did after peaking at b_{i-1}.

We then tried correlations where x_i was defined using the average of measurements of buffer size of two samples $(b_{i-1}+b_{i-2})$ which represents the average buffer size in the previous two seconds, three samples $(b_{i-1}+b_{i-2}+b_{i-3})$ which represents the average buffer size in the previous three seconds, and four samples $(b_{i-1} + b_{i-2} + b_{i-3} + b_{i-4})$ which represents the average of buffer sizes in the previous four seconds. This was examined since often there may be a spike in the buffer size that lasts briefly and not have a significant impact on frames per second. Each of these resulted in a higher correlation measurement than the initial set described. The results are summarised in Fig. 3.

There were further combinations of buffer values used in our attempts to come up with the best relationship between the buffer values and the frames per second values. For instance, we looked at the absolute difference in the buffer size values $(\mid b_i - b_{i-1} \mid - \bar{b})$ from one sample to the next to see what relationship existed between those values and the frames per second values. The resulting correlation value was -0.42106. This result when compared to the previous results showed that this was the wrong direction in which to proceed.

3.3 Regression Analysis

The correlation analysis shows that there is a linear relationship between the buffer size and the frames per second values that are reflected between three and six samples later. We used a linear regression model to describe this relationship. More formally, consider a first-order linear regression model of the following form: $y_i = \mu + \alpha x_i$ where $y_i = f_i$ and $x_i = (b_{i-1} + b_{i-2} + b_{i-3} + b_{i-4})/4$. Least-squares regression was used to determine the value of α and μ. The predicted value is denoted by $\hat{f}$. The equation corresponds to a trendline and is called the trendline equation. Least squares regression was used to determine the values of the parameters of the linear regression model. This resulted in the following: $\hat{f} = 28.9653 - (1.701309E^{-3})((b_{i-1} + b_{i-2} + b_{i-3} + b_{i-4})/4$

The value of α is a measure of the change in the fames per second for every increment of one in $(b_{i-1} + b_{i-2} + b_{i-3} + b_{i-4})/4$. That is, every time the buffer occupancy level increases by one byte, the frames per second rate will decrease by $1.701309E^{-3}$. If the model fits the measured data well, then α is a direct measure of the average of the last four sampled buffer sizes on the frame rate. Specifically, if α is statistically non-zero(at say the 95% confidence level), then we conclude that the frame rate depends on the average of the last four sampled buffer sizes with a strength of α. We used t-statistics and P-value to confirm this.

We used the equation $\hat{f} = 28.9653 - (1.701309E^{-3})((b_{i-1} + b_{i-2} + b_{i-3} + b_{i-4})/4)$ to not predict the frames per second, but whether the frames per second is going to fall below or fall above a specific threshold value as defined by the QoS requirements.

If the target frame rate is 25 then for $25 = 28.9653 - (1.701309E - 3)((b_{i-1} + b_{i-2} + b_{i-3} + b_{i-4})$ we have that $(b_{i-1} + b_{i-2} + b_{i-3} + b_{i-4})/4$ should be equal to 2315 (this is rounded). For a target frame rate of 23 the value of $(b_{i-1} + b_{i-2} + b_{i-3} + b_{i-4})/4$ should be 3525 and for a target frame rate of 27 the value of $(b_{i-1} + b_{i-2} + b_{i-3} + b_{i-4})/4$ should be 1191.

Thus, for the QoS requirement "The number of video frames per second displayed to the user must be at least 25 plus or minus 2 frames", if the buffer size is less than 1191 or greater than 3525 then this is an indicator that the QoS requirement will be violated.

4 Architecture

Monitoring of the application is needed to collect values of the attributes (e.g., current_fps). One monitoring mechanism is through instrumentation of the application which is briefly described here.

An attribute is associated with a sensor. A sensor is a class with variables for representing threshold and target values. Sensors are used to collect, maintain, and process a wide variety of attribute information within the instrumented processes. The sensor's methods (*probes)* are used to initialise sensors with threshold and target values and collect values of attributes. Probes are embedded into process code to facilitate interactions with sensors. The *coordinator* is the instrumentation component that is between the process and the management system. The coordinator is implemented as a thread of the application process.

There are two sensors used in this work: fps_sensor and socket_buffer_sensor. Fig. 3 illustrates the use of fps_sensor in example psuedo-code for a video playback application that has the QoS requirement "The number of video frames per second displayed to the user must be at least 25 plus or minus 2 frames". This QoS requirement suggests an upper threshold of 27 frames per second and a lower threshold of 23 frames per second. Sensor fps_sensor includes probes such as the following: (1) An initialisation probe (line 3 of Fig. 3) that takes as a parameter the default threshold target value, and the lower and upper bounds. When the coordinator is instantiated (line 2 of Fig. 3) it communicates with the QoS management system to get the application's QoS requirements in the form of a condition list which represents a condition by an attribute identifier, the identifier of a sensor that monitors that attribute, a comparison operator and value that the attribute is to be compared to using the comparison operator. Thus when the sensor requests this information, the coordinator will already have retrieved it. (2) A probe, probe_framerate(), that (i) determines when ten frames have been displayed. For every ten frames displayed, it calls a function that returns the system clock time. This and the system clock time returned ten frames ago is used to calculate the current frame rate and checks to see if this time falls within a particular range defined by the lower and upper acceptable thresholds. Unusual spikes are filtered out; and (ii) informs the coordinator through the report method if the frames per second fall below the lower threshold or is higher than the upper threshold. The report method is implemented as a thread.

The socket_buffer_sensor monitors the length of the communication buffer. This is done as follows. A socket provides for interprocess communication. In UNIX, a socket is a file descriptor. The kernel allocates an entry in a private table in the process area, called the user file descriptor table and notes the index of this entry. The index is the file descriptor that is returned to the process. The entry allocated is a pointer to the first *mbuf* structure. The memory associated with

```
Given: Video application v.
       QOS expectations e.
------------------------------------------------
1.   Perform initialization for v.
2.   Initialise coordinator c.
3.   Execute fps_sensor → init_probe(e)
4.   while (v not done) do:
5.       Retrieve next video frame f.
6.       Decode and display f.
7.       Execute fps_sensor → probe_framerate()
8.   endwhile
```

Fig. 3. Instrumentation Example

the *mbuf* structures of the process is the buffer. The sensor, socket_buffer_sensor, implements a function getBufferStats() that given a file descriptor for a socket returns buffer statistics including the length of the buffer.

Both the fps_sensor and the socket_buffer_sensor have read and report methods. When these sensors are instantiated, the report methods are initiated as threads through a function call so that they can communicate with the coordinator on an ongoing basis, separate from the application.

The report method of the socket_buffer_sensor calls at regular intervals of one second the read method which in turn calls getBufferStats() which returns the buffer statistics. In the i^{th} interval, the read method computes $(b_{i-1}+b_{i-2}+b_{i-3}+b_{i_4})/4$ and compares it to a threshold value. Based on the trendline equation computed in the previous section, it determines if the calculated value falls below 1191 or rises above 3525. If it does then a *warning* indication is generated. Otherwise, a *no warning* indication is generated. These are indicated by 0 and 1 respectively and are sent to the coordinator.

For the purposes of analysis, the fps_sensor was subclassed with the probe_framerate() rewritten so that it reports all calculated frame rate values and not just the frame rate values that violate the QoS requirements. The coordinator was subclassed so that it takes reports from the fps_sensor and the buffer_sensor and puts those reports into a logfile.

5 Analysis

To analyse the effectiveness of the approach we had to find a way to transpose the log file reports of frames per second into a second-by-second calculation that can be used in our analysis. This is denoted by fps_{t_i} where t_i represents the time at i seconds into the run.

We let fps_j represent the j^{th} occurrence (between time t_i and t_{i+1}) that the frames per second rate is reported. The equation used for averaging is the following: $fps_{t_n} = \frac{fps_1+fps_2+...+fps_j}{j}$

After extracting the warnings from the log report, a file of warnings would look something like this: 0,0,0,0,0,1,1,1,1,1,1,0,0,0,0,1,0,0,0,0,0,1,1,1,1,1,....

We correlated these warnings with the frames per second rate. We only needed to know when the warning was turned on and when it was turned off. Repetitions of a warning value were filtered.

This information was used in our analysis that focussed on determining the effectiveness of our approach to prediction. These results are now presented and are only with respect to frame rates being too low. We have the following categories of warnings:

- On-Time Warning – This warning is issued between one and five seconds in advance of the time frame where the frames per second rate drops below the minimum threshold.
- Late Warning – This is a warning which is issued, but is only issued less than one second in advance of the drop in frames per second rate below the minimum threshold.
- False Positive Warning – This is a warning that is issued where there is no instance of a frames per second reading which is below the minimum threshold.
- No Warning – This is an occurrence of the frames per second rate dropping below the minimum standard where no warning at all is received.

In Fig. 4, we show an example of the warnings that have been received in a specific run of our experiments. Series 1 represents the values of the frames per second as calculated above, and Series 2 represents the warnings given and filtered as previously discussed. We can readily see in this graph that the warnings received appear to arrive, in most cases, just at the beginning of a decrease in frames per second values.

The portion of correct early warnings is represented as P_p. The portion of false predictions is represented by P_f and the portion of late warnings or no warnings is represented by P_l (these were combined since late and no warnings have the same end result).

Warning percentage calcuations for three case studies of video each of which is approximately ten to fifteen minutes in length was calculated. The value of P_p for each of the case studies was 0.63, 0.62, and 0.59 respectively. The value of P_f for each of the case studies was 0.013, 0.027 and 0.068 respectively. The value of P_l for each of the case studies was 0.35, 0.37 and 0.36 respectively. The results are consistent and demonstrate that is possible to correctly predict in advance the violation of the QoS requirement approximately 60 per cent of the time. The excellent results though are reflected in the paucity of false positives that we received. This is an important feature since reacting to a false positive warning could be more costly if the QoS management system is forced to allocate increased system time and resources to the application that would go to waste.

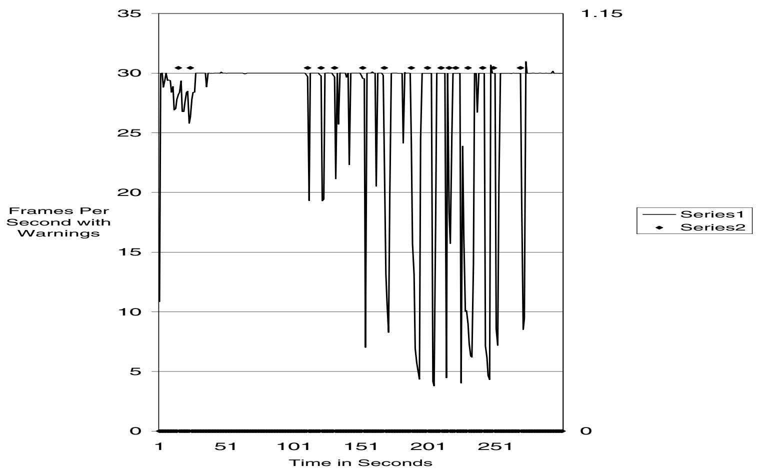


Fig. 4. Graphic Display of Warnings Relative to FPS Rates

6 Discussion

1. The sensors presented here can be used in any application that receives a multimedia stream. Only the parameters of the trendline equation and the threshold values would change. These could be read from a configuration file by the coordinator upon application start-up.
2. We initially tried to correlate the observable indicators of frame loss and jitter with the frame rate. The statistical correlation computed was so low that it was not deemed necessary to continue along these lines. We would like to improve the percentage of on-time warnings without increasing false positive warnings by examining the use of multiple attributes to predict that a QoS requirement will be violated.
3. In this work, the relationship between attributes of the different objects of dependency chains is represented using linear regression. More case studies are needed to determine other ways to relate information. There is the possibility that different parameters could be used in a vector relationship or a matrix of vectors or possibly in a weighted combination of parameter statistics.
4. This paper assumed that the fault will a heavy CPU load that causes the CPU to be a resource lacking by the application. However, it may be the case that fault is a lack of several computing resources. This should be taken into account.

7 Related Work

The earliest work found in predictive fault detection is found in [5]. These papers characterise normal network behaviour. They developed models that seem rather ad-hoc to estimate weekly patterns. This takes into account that the definition of normal changes over time. This is experimentally validated through a case study where the introduction of backups (this caused heavy network traffic) causes a change in normality. Changes in the statistics of traffic data such that they do not conform to the definition of normality can be used to detect anomalies. However, this work does not particularly focus on prediction.

The work in [9] describes how to detect changes in network behaviour that are closely correlated with service interruptions. Some approaches to fault detection have been developed based on Bayesian networks (e.g., [3]). In [7], the basic approach is to model the threshold metric at two levels: non-stationary behaviour as in workload forecasting and stationary behaviour with time-serial dependencies, in order to compute the probability of violations. This apparently worked well with two caveats: if the actual values of predictions were far enough from the threshold and if the prediction horizon was not too far in the future.

There is a good deal of work on intrusion detection that is somewhat relevant. A great deal of research in intrusion detection is based on the hypothesis that network attacks cause abnormal behaviour. Much of the work in this area models normal network behaviour so that behaviour that does not satisfy the constraints of normal network behaviour is assumed to be caused by intrusion. Examples of this work include [1].

Most of the work described focusses on characterising network traffic. The work in [10] focusses on characterising application behaviour. They use the number of requests processed per second as a measurement of web performance. This was measured every 15 minutes. We will refer to each measurement as an observation. The methodology used discarded all observations that were known to correspond with a fault. The rest of the observations were used to create distribution models for "normal" behaviour.

As can be seen, there is an area of research that is being done in proactive or predictive detection of failures, but most of the work is being done using network parameters and statistics. Work that is directed towards proactive failure detection at the application is not abundant.

8 Conclusions and Future Work

The work in this paper shows that an approach to prediction based on dependency chain can provide good results. Future work includes addressing the questions brought up in the discussion section. The long-term goal is to develop a set of tools that facilitate the development of trendline equations and sensors that make use of these equations.

References

1. M. Bykova, S. Ostermann, and B. Tjaden. Detecting network intrusions via a statistical analysis. In *Proceedings of the 33rd Southeastern Symposium on System Theory*, pages 309–314, 2001.
2. R. Carter Hill, William E. Griffiths, and George G. Judge. *Undergraduate Econometrics*. John Wiley and Sons, Inc., 2000.
3. C. Hood and C. Ji. Probabilistic network fault detection. In *Global telecommunications Conference, 1996: Communications: They key to Global Prosperity*, pages 1872–1876, 1996.
4. M. Jones, D. Roşu, and M. Roşu. CPU Reservations and Time Constraints: Efficient, Predictable Scheduling of Independent Activities. *Proceedings of the Sixteenth ACM Symposium on Operation Systems Principles*, St. Malo, France, October 1997.
5. R. Maxion and F. Feather. A case study of ethernet anomalies in a distributed computing environment. In *IEEE Transactions on Reliability 39(4)*, pages 433–443, October 1990.
6. G. Molenkamp, H. Lutfiyya, M. Katchabaw, and M. Bauer. Resource management to support application-specific quality of service. *IEEE/IFIP Management of Multimedia Networks and Services (MMNS2001)*, October 2001.
7. D. Shen and J. Hellerstein. Predictive models for proactive network management: Application to a production web server. In *Proceedings of the 2000 IEEE/IFIP Network Operations and Management Symposium*, pages 833–846, 2000.
8. M. Steinder and A. Sethi. The present and future of event correlation: A need for end-to-end service fault localization, 2001.
9. M. Thottan and C. Ji. Fault prediction at the network layer using intelligent agents. In *Proceedings of the 1999 IEEE/IFIP International Symposium on Integrated Network Management*, pages 745–759, 1999.
10. A. Ward, P. Glynn, and K. Richardson. Internet service performance failure detection. In *1998 Web Server Performance Workshop*, pages 38–43, June 1998.

Scheduling Time-Sensitive IP Traffic

Pedro Sousa, Paulo Carvalho, and Vasco Freitas

Universidade do Minho, Departamento de Informática,
4710-057 Braga, Portugal
{pns,paulo,vf}@uminho.pt

Abstract. This article presents an hybrid priority queuing model based scheduler for real-time traffic differentiation. The proposed scheduler is designed as a mechanism to provide queuing delay differentiation among real-time traffic classes. The novel characteristic of the presented scheduler is the capability to simultaneously try to achieve an upper time limit for queuing delays and, under heavy load conditions, deny class starvation by providing an expectable differentiation schema for excess queuing delays. The attractiveness of the proposed scheduler is its hybrid differentiation capabilities based on a simple queue selection procedure. Additionally, the enhanced differentiation behavior of the scheduler is also highlighted as three distinct configuration modes are possible.

1 Introduction

The advent of Quality of Service (QoS) in the Internet [1] is being fostered by an increasing need to provide adequate network services to a vast range of QoS-demanding applications. In the presence of distinct applications and traffic profiles, service providers need to differentiate customers so that an efficient and cost-effective network resource management can be achieved. In addition to admission control [2], reservation protocols [3], resource management solutions [4,5], or even when these are not present, acceptable QoS conditions can be obtained in the presence of an appropriated delay differentiation mechanism. In fact, delay differentiation can be extremely useful to integrate real-time applications and other delay sensitive applications. From the applications' perspective there are two crucial aspects for the integration of real-time applications in IP networks: the ability to satisfy end-to-end delay requirements and the capability to absorb excess delays. The former aspect is related to real-time applications, such as voice over IP and other interactive applications which are highly delay sensitive [6]. In this context, the deployment of scheduling mechanisms providing queuing delay bounds plays a crucial role in the integration of real-time traffic in IP networks. The latter aspect is related to mechanisms used by real-time applications in order to smooth excess delays [7,8] or to adapt to network conditions [9,10]. This means that in addition to end-to-end delay bounds it is useful to provide expectable differentiation mechanisms to handle excess queuing delays inside the network. In this context, the use of rigid admission control procedures and resource reservations protocols (e.g. in the IntServ architecture [11]) play a relevant role. These solutions, suffering from well known lack of scalability and flexibility, led to lighter and easier to deploy solutions (e.g. DiffServ [12]). As expected, relaxing QoS-guarantees in the network raises additional problems of integration of real-time applications mainly due to the absence of rigid admission control in the core routers. Even existing admission control at network edges the network core may cause feasibility problems to schedulers due to: *i)* the transient effect caused by dynamic flow aggregation; *ii)* traffic distortion and packet clustering caused by cascade queuing effects; *iii)* possible shaping inaccuracies at edge routers and *iv)* path changes caused by a route flip. In addition, delay-oriented scheduling mechanisms may show feasibility problems as regards their configuration parameters. Service starvation for low priority classes may also occur in some schedulers using strict priority schemes for class differentiation [13]. In conclusion, to contribute for a QoS-capable Internet it is important to develop scheduling mechanisms able to react to particular congestion situations in order to achieve expectable differentiation behaviors among real-time traffic classes. In this context, this work proposes a new hybrid scheduling mechanism with the following characteristics: *i) easy configuration-* it is possible to control the *expectable queuing delay* and *congestion queuing delay* of each traffic class through simple parameter configuration; *ii) simplicity-* the hybrid behavior of the differentiation mechanisms is obtained resorting to simple queue selection procedures, and not using additional node state information (e.g. arrival class packet counters, long-term average delay counters, etc.); *iii) enhanced differentiation capability-* the scheduler allows three configuration modes each one involving distinct differentiation semantics (mixed configuration modes are also possible); *iv) unfeasible scheduling regions control-* the differentiation mechanism also contributes to the control of unfeasible working regions of scheduling mechanisms by providing controllable and expectable differentiation schema for delay deviations. The proposed scheduler can be used in distinct operational scenarios independently of the network model. Nevertheless, and due to its delay-oriented nature, a *rate-oriented* mechanism might be used to allocate/differentiate bandwidth between elastic and real-time traffic, and within the latter, the proposed mechanism provides for delay differentiation.

The paper is organized as follows: Section 2 presents related work. Section 3 details the proposed queuing model. Section 4 presents simulation results of the scheduler behavior including: single-node differentiation, flow granularity, results verification and mixed configurations. Section 5 presents the conclusions of the work.

2 Related Work

Recently, there has been considerable research focusing on the use of priority queuing models for IP traffic differentiation. Some of these queuing models can be found in [14], where brief mathematical explanations are also given. The work presented in [15,16,17, 18] focuses on the use of Relative Differentiation, where a multiplicative time dependent model is used to achieve proportional differentiation behavior of a network node. Additionally [19,20] study some possible adaptive behaviors which can be applied to the previous models. In [21,22] an overview of different delay differentiation schemas including proportional, additive and an hybrid upper-time queuing model are presented. The latter model allows the coexistence of the proportional model along with an unique upper time bounded traffic class. These contributions include end-to-end differentiation analysis as well as individual flow behaviors study. Although the work in the area focuses mainly on the Relative Differentiation approach, such as proportional differentiation, the present proposal focuses on an hybrid queuing model of traffic differentiation. Some different schemas, such as EDD [23] (also denoted as EDF), also try to limit queuing packets delays but they are more suitable for scenarios of strong per node admission control procedures in order to ensure the necessary feasibility conditions [24] for their operations. Instead the mechanism proposed here is more adequate for scenarios where admission control procedures are more relaxed and operate at network edges devices. In this context, in [25] the EDD schema is modified in order to differentiate the probability of queuing delay violations under a congested network. Furthermore, is fundamental to differentiate the relative value of such violations, i.e. under general class congestion ensure that the excess queuing delays of high priority classes are not higher than the obtained by lower ones. Additionally, the present mechanism is based on a simple queuing selection procedure which does not use additional class state information, reducing the differentiation node complexity.

3 An Hybrid Priority Queuing Model

3.1 Model Construction

The Upper Time Limit (UTL) model belongs to the class of Priority Queuing models [14] where each queue is ruled by a priority function that varies over time (Time-Dependent Priorities). The nature of the priority function and its configuration parameters define the behavior of the service assigned to each queue. This study considers N classes $Class_{i(0 \leq i \leq N-1)}$ having $Class_0$ the highest priority. The UTL model is a more rigid schema than the additive and proportional models as it imposes a finite queuing delay. The main idea is to define a boundary (reflected by U_i) for the packet queuing time (Eq. (1)). In this model, the lower the boundary, the higher the priority function slope will be. When $t - t_{0_i} \geq U_i$, i.e. on or over the limit, the server is *forced*[1] to dispatch the packet awaiting service[2]. This model protects the high priority classes aiming that packets remain in queue for a maximum value U_i with $U_i < U_{i+1}$ (Fig. 1(a)).

$$p_i^a(t) = \begin{cases} \dfrac{t - t_{0_i}}{U_i - t + t_{0_i}} & \text{if } t < t_{0_i} + U_i \\ \infty & \text{if } t \geq t_{0_i} + U_i \end{cases} \tag{1}$$

Fig. 2 illustrates two examples of the model behavior for three CBR and Exponential sources[3] contending for a common 100Mbps capacity link and for specific U_i parameters. Within this model some additional considerations can be made regarding its behavior: *i)* as seen in Fig. 2, and as expected, under heavy load conditions the *upper time* parameters of the classes can be violated; *ii)* under a violation it is not possible to control the spread of excess queuing delay; *iii)* starvation of low priority classes may occur (e.g. $Class_C$ between 100-150 server transmission times). This shows that the model is not able to distinguish excess queuing delays in the traffic classes. The reason for this behavior is that when a class violates the respective upper time limit, the priority function assumes an

[1] Obviously when congestion occurs packets can be dropped or the waiting time limit exceeded.
[2] t_{0_i} is the arrival time of the heading packet of $Class_i$.
[3] The results were obtained implementing the native UTL model in the *Network Simulator-2*.

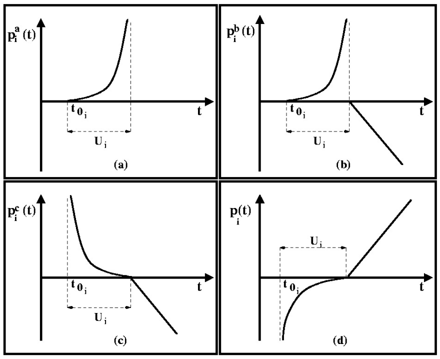


Fig. 1. a) Native UTL model b) c) d) Different stages leading to the hybrid queuing model.

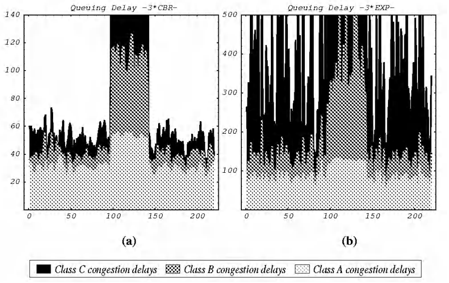


Fig. 2. a) Three CBR classes $(U_A, U_B, U_C) = (30\mu s, 40\mu s, 50\mu s)$ b) Three exponential classes $(U_A, U_B, U_C) = (100\mu s, 150\mu s,$-$250\mu s)$.

infinity value and from that instant the system acts as a strict priority model. Furthermore, it is common that under a high delay violation of a class the other classes also become overloaded due to starvation and, as consequence, all the priority functions assume an infinite value (i.e. a cascade effect). This means that it is useless to use the priority values to differentiate classes as the decision is based again on static assumptions (as to serve the highest priority class first). These cascade effects are prominent in high speed networks dealing with high traffic loads and strict bounds for queuing delays. In order to overcome these problems, an additional mechanism allowing class differentiation in such scenarios is needed. The first modification introduced in the original model was to activate an additional function for $t \geq t_{0_i} + U_i$. The idea was to allow congested classes to be differentiated instead of using infinity values hindering the initial objective. Moreover, this additional function should assume priority values allowing the scheduler to differentiate the congested situation from the normal priority function behavior for $t < t_{0_i} + U_i$. This is achieved using a multiplicative function for $t \geq t_{0_i} + U_i$ as presented in Eq. (2) (see Fig. 1(b)). This function, as explained later, will allow the *proportional differentiation* of excess queuing delays between traffic classes. The excess queuing delay (the difference of the total and upper time delay, i.e. $t - t_{0_i} - U_i$) is multiplied by a scale parameter C_i which guides the priority function behavior. In this region the priority function assumes negative values.

$$p_i^b(t) = \begin{cases} \dfrac{t - t_{0_i}}{U_i - t + t_{0_i}} & \text{if } t < t_{0_i} + U_i \\ -(t - t_{0_i} - U_i) * C_i & \text{if } t \geq t_{0_i} + U_i \end{cases} \tag{2}$$

The drawback of Eq. (2) is the complexity of the queue selection procedure. In fact, it will be necessary to compute all $p_i^b(t)$ values; verify if a negative value exists; if so, select the lowest one, otherwise select the highest positive priority value. To simplify the selection procedure Eq. (3) was applied to Eq. (2) resulting in Eq. (4). In this modified function the selection procedure consists of a simple selection of the queue with the lowest $p_i^c(t)$ value as the one to be served next by the scheduler (see Fig. 1(c)).

$$p_i^c(t) = \frac{1}{p_i^b(t)} \quad for \quad t < t_{0_i} + U_i \tag{3}$$

$$p_i^c(t) = \begin{cases} \dfrac{U_i - t + t_{0_i}}{t - t_{0_i}} & \text{if } t < t_{0_i} + U_i \\ -(t - t_{0_i} - U_i) * C_i & \text{if } t \geq t_{0_i} + U_i \end{cases} \tag{4}$$

In order to maintain the normal semantics of priority queuing, where the class with the highest priority function is selected first, the symmetric function (5) is used as the normalized priority queuing function. This model is then configured with two distinct sets of parameters: *Upper time* $(U_0, .., U_{N-1})$ and *Congestion*[4] parameters $(C_0, .., C_{N-1})$. The final priority function is given by (6) and Fig. 1(d) illustrates its behavior.

$$p_i(t) = -p_i^c(t) \tag{5}$$

$$p_i(t) = \begin{cases} \dfrac{\delta_t - U_i}{\delta_t} & \text{if } \delta_t < U_i \\ (\delta_t - U_i) * C_i & \text{if } \delta_t \geq U_i \end{cases} \tag{6}$$

with $\delta_t = t - t_{0_i}$ and $0 \leq i \leq N - 1$.

In order to define the selection task, let C be the set of all (i, p_i) pairs in the system, where p_i is the priority to serve $Class_i$, i.e. $C = \{(i, p_i) \mid 0 \leq i \leq N - 1\}$. The selector Sel, defined according Eq. (7) and (8), determines the index of the class to be served, i.e. taking the maximum priority value, p_{max}, the corresponding minimum i is chosen. The total delay[5], d_i, affecting $Class_i$ can be divided in two components: one induced by priority function when it assumes negative values, i.e. $t < t_{0_i} + U_i$, which we call *upper time delay*, d_i°, and other when the function assumes positive values, which we call *congestion delay*, $d_i^\bullet$ (see Eq. (9)). The magnitude of d_i° is controlled by the *upper time parameter*, U_i, whereas C_i controls the magnitude of $d_i^\bullet$. This means that fundamental differentiation relations among classes, i.e. $d_0 \leq d_1 \leq ... \leq d_{N-1}$, can be achieved through different combinations of d_i° and $d_i^\bullet$, and consequently by different combinations of parameters U_i and C_i. The next section discusses this aspect.

$$p_{max} = max\{y \mid (x, y) \in C\} \tag{7}$$

$$Sel = min\{x \mid (x, y) \in C \wedge y = p_{max}\} \tag{8}$$

$$d_i = d_i^\circ + d_i^\bullet \tag{9}$$

3.2 Parameter Configuration Modes

Fig. 3(a) presents three distinct behaviors of the hybrid queuing model (configuration mode I, II and III). For each configuration, the relations between the *upper time delay*

[4] We use the *congestion* term in a relaxed way as it may reflect heavy load conditions in the server, heavy load conditions in $Class_i$ impairing the expected upper time limit or feasibility problems in the differentiation parameters.

[5] In the remaining of the paper, d_i is also used to denote the average queuing delay of $Class_i$.

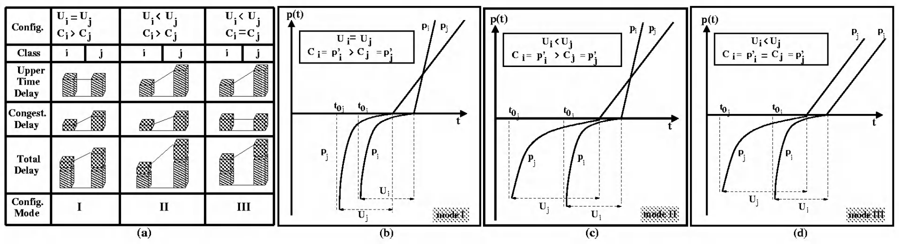


Fig. 3. (a) Combinations of U_i, C_i parameters (b)(c)(d) Representation of configuration modes.

and *congestion delay* are presented for two generic classes i and j with $i < j$. As shown, the *total delay* differentiation behavior for all configuration modes obeys the *Relative Differentiation* relation: $d_0 \leq d_1 \leq ... \leq d_{N-1}$.

Configuration I: In this configuration mode identical upper time limits are configured for the two traffic classes. This means that both classes share the same priority function as the packets stay in queue for a time limit below the configured U_i parameter. In this configuration the traffic classes are differentiated by C_i which means that in case of congestion the priority function for the higher priority class assumes higher values than for the other. Fig. 3(b) illustrates the priority function evaluation in this configuration mode for traffic classes i and j, with the assumption of heading packet time arrivals $t_{0_i} > t_{0_j}$. As shown, identical priority function shapes for both classes are obtained for $t < t_{0_i} + U_i$. On the other hand, in the positive priority function region, the function slope C_i is greater than function slope C_j eventually leading to the expected switch in priority values for these queues. This configuration mode may be appropriated for real-time classes with the same upper time limit for queuing delay and distinct capabilities to absorb possible delay violations. The expected behavior of this model is that under feasible conditions the specified upper time limits for both classes are achieved, i.e. $d_i = d_i^\circ = d_j = d_j^\circ < U_i$ or $< U_j$. However, if the server becomes overloaded and the upper time limit delays of the classes are violated the maximum difference between the queuing delays is given by (10)[6] and (11). Recall that within this configuration mode and due to similar *upper time* configurations, the queuing delay difference of *congested* classes is only influenced by the C_i parameters.

$$d_j - d_i = d_j^\bullet - d_i^\bullet \approx \frac{C_i}{C_j} \cdot d_i^\bullet - d_i^\bullet \tag{10}$$

$$d_j - d_i \approx \underbrace{d_i^\bullet \cdot \left(\frac{C_i}{C_j} - 1\right)}_{congestion\ part} \tag{11}$$

Configuration II: In this configuration mode the traffic classes are distinct as regards *upper time differentiation parameters* and *congestion differentiation parameters*. As result, the priority function associated with higher priority classes has a larger increase

[6] Note that for the proportional model $\frac{d_j^\bullet}{d_i^\bullet} \approx \frac{C_i}{C_j} \Rightarrow d_j^\bullet \approx d_i^\bullet \cdot \frac{C_i}{C_j}$.

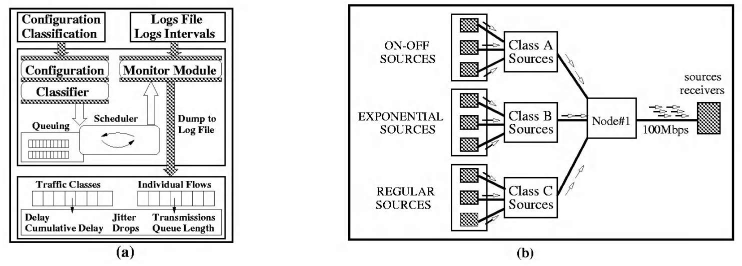


Fig. 4. (a) Components implemented in each output link (b) Simulation scenario.

than the lower ones for both negative and positive values of $p_i(t)$. This means that under congestion both *congestion delay* and *upper time delay* associated with high priority classes should be lower than the ones associated with the other classes. This configuration is appropriate to differentiate high delay sensitive applications with low capacity to absorb excess queuing delays. Fig. 3(c) presents an example of the priority function evaluation in this configuration model for two classes. Again, if the server becomes overloaded and under *upper time limits* violations, the delay differentiation is given by (12) and (13). As presented by Eq. (13) the delay difference has two components: one resulting from the U_i parameters and the other from the C_i parameters, having these ones the same characteristics as in configuration mode I.

$$d_j - d_i = (d_j^{\circ} - d_i^{\circ}) + (d_j^{\bullet} - d_i^{\bullet}) \tag{12}$$

$$d_j - d_i \approx \underbrace{U_j - U_i}_{upper\ time\ part} \underbrace{+ d_i^{\bullet} \cdot \left(\frac{C_i}{C_j} - 1\right)}_{congestion\ part} \tag{13}$$

Configuration III: This mode differentiates traffic classes only by U_i parameters. This configuration is used to distinguish a class by its maximum queuing delay limit and, in case of violation, the classes share the same priority behavior for the excess queuing delays (see Fig. 3(d)). The delay differentiation achieved by this model is presented in Eq. (14). As expected the differentiation is caused only by U_i as C_i and transitively the behavior of $p_i(t)$ are the same for both classes and for $t \geq t_{0_i} + U_i$.

$$d_j - d_i = (d_j^{\circ} - d_i^{\circ}) \approx \underbrace{U_j - U_i}_{upper\ time\ part} \tag{14}$$

4 Performance Evaluation

The differentiation mechanisms were implemented and tested in the *Network Simulator (NS-2)*. Specific queues and monitors were also developed in order to collect results from the tests. Fig. 4(a) shows the implemented architecture associated with the output link of a differentiation node. At Otcl level, the scheduler is selected, the differentiation

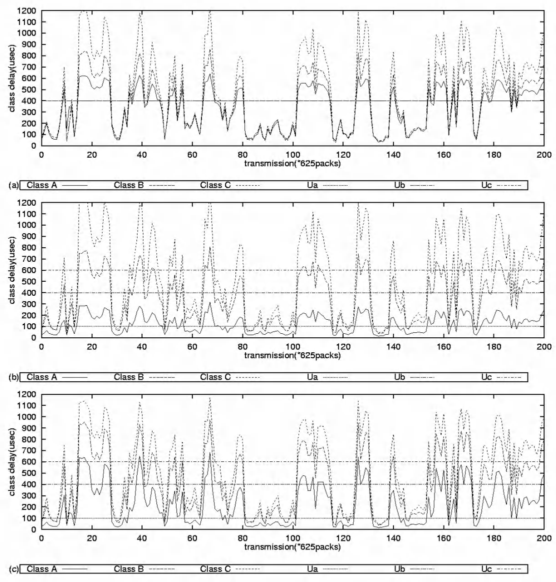


Fig. 5. Delay differentiation for a) $(U_A, U_B, U_C) = (400\mu s, 400\mu s, 400\mu s)$, $(C_A, C_B, C_C) = (4, 2, 1)$ (mode I) b) $(U_A, U_B, U_C) = (100\mu s, 400\mu s, 600\mu s)$, $(C_A, C_B, C_C) = (4, 2, 1)$ (mode II) c) $(U_A, U_B, U_C) = (100\mu s, 400\mu s, 600\mu s)$, $(C_A, C_B, C_C) = (1, 1, 1)$ (mode III).

parameters of the queues/classes are defined and classification data is provided, i.e. $(packet_{flowid}, queue_{id})$ pairs. At the same level, the state information granularity to be logged at scheduling time is indicated. In the architecture core, the monitor module logs periodically state information about flows and classes for subsequent analysis. Fig. 4(b) shows the simulation scenario used to test the scheduler behavior. In this example different traffic patterns were mixed converging to the differentiation nodes. The scenario includes *on-off*[7], exponential and isochronous traffic sources which are mapped to different classes (A, B and C) contending for a common link. Each class contributes evenly to the overall load, i.e., have similar long term rates, and generates mean packet lengths of 500 bytes uniformly distributed over $[250, 750]$. Similar queuing resources were allocated for all classes and $Class_A$ has the highest priority[8].

4.1 Single-Node Differentiation

Fig. 5 shows three distinct differentiation examples obtained using the scheduler in configuration modes I, II and III. The results are presented graphically where the x-axis represents the server packet transmission times with a plot granularity of $25ms$.

[7] On-off periods follow a Pareto distribution with $\alpha = 1.2$.

[8] The reason for this choice is that this class carrying Pareto related traffic causes high variability on queue lengths being more demanding on the differentiation algorithm.

Fig. 5(a) shows a configuration where all classes share an upper time limit of $400\mu s$, i.e. $(U_A, U_B, U_C) = (400\mu s, 400\mu s, 400\mu s)$, and distinct *Congestion Differentiation Parameters*, in this case $(C_A, C_B, C_C) = (4, 2, 1)$. As plotted, all classes have similar queuing delays in the non-congested scheduling region $(d_A, d_B, d_C \leq 400\mu s)$. However, in the congested regions the scheduler switches to proportional differentiation. As a result when the queuing delays are higher than $400\mu s$ the excess queuing delays on $Class_A$ are lower than those on $Class_B$, and both lower than $Class_C$ delays. Additionally, the proportional relation between the excess queuing delays may be easily visualized, as excess delays in $Class_C$ are approximately twice the delays in $Class_B$, which in turn double $Class_A$ delays. Consequently $Class_C$ excess delays are four times longer than in the highest priority $Class_A$. This satisfies the proportional relations defined by the *Congestion Differentiation Parameters*. These results show that the proposed behavior for the configuration mode I is feasible and can be used by classes sharing the same queuing delay constraints but with different capabilities to absorb excess queuing delays.

Fig. 5(b) plots the differentiation behavior for configuration mode II. In this case different upper time limits are assigned to each class, $(U_A, U_B, U_C) = (100\mu s, 400\mu s, 600\mu s)$. All classes are also differentiated by the *Congestion Differentiation Parameters* with $(C_A, C_B, C_C) = (4, 2, 1)$. As a result, as plotted, for congested periods there is an excess queuing delay in all congested classes following the proportional differentiation approach. For example, when the upper time of the highest class is violated ($d_A > 100\mu s$) the remaining queuing delay is approximately two times lower than the obtained by $Class_B$ (relative to its upper time of $400\mu s$). The same applies to relations between $Class_A$ and $Class_C$ and to $Class_B$ and $Class_C$. This configuration is useful when aiming of a fully differentiation schema, which means that highest classes have lower upper time delays and simultaneously are more sensitive to excess queuing delays, resulting in a higher *Congestion Differentiation Parameter*.

Fig. 5(c) illustrates the differentiation behavior for configuration mode III. In this case, different upper time limits are assigned to each class, $(U_A, U_B, U_C) = (100\mu s, 400\mu s, 600\mu s)$, but classes are not differentiated by the *Congestion Differentiation Parameters* as $(C_A, C_B, C_C) = (1, 1, 1)$. As a result, and as illustrated, for congested periods there is an excess queuing delay in all congested classes following a fair distribution. For example, when the upper time of the highest class is violated ($d_A > 100\mu s$) the remaining of the queuing delay is similar to the obtained by $Class_B$ (relative to its upper time of $400\mu s$). The same applies to relations between $Class_A$ and $Class_C$ and to $Class_B$ and $Class_C$. This configuration is useful for partial differentiation schemas, which means that higher priority classes have lower upper time delays but simultaneously have similar sensitivity to excess queuing delays as the lower priority classes.

4.2 Flow Granularity and Fairness

In this section the model behavior is studied at the flow level. The aim is to examine how the delay-oriented QoS offered to each class is extended to the flow level. Knowledge of the flow characteristics can be useful to check whether applications can also expect a fair delay differentiation as a complement to the class differentiation behavior of the model. Furthermore, it is expected that the scheduler provides per flow queuing delay

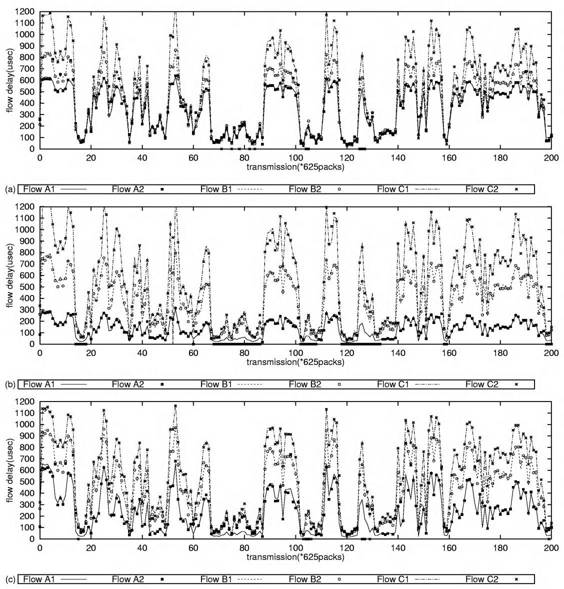


Fig. 6. Differentiation at Flow Level a) Configuration I b) Configuration II c) Configuration III.

consistency. In other words, it is expected that flows sharing a common traffic class at a given time also share identical average queuing delays. To check this, two flows of each traffic class were selected and their queuing delay behavior plotted together as shown in Fig. 6. It can be seen that the differentiation behavior is also valid at flow level, i.e. delay violations at flow level behave similarly to their traffic classes. Furthermore, the differentiation mechanism keeps per flow queuing delay consistent with their traffic classes having the same queuing delay. Inspection of Fig. 6 (a),(b) and (c) shows that the plots of delay per flow coincide within each class for all configurations.

4.3 Verification of Results

In this section we verify if the results obtained by simulation satisfy the delay difference equations devised in Sec. 3.2, representing the maximum queuing delay spread which may occur for each configuration mode under heavy load conditions. Fig. 7 plots the current delay differences and the maximum theoretical differences expressed by equations (11), (13) and (14), for each configuration mode. The latter are represented by lines, during the congested periods only, and the former represented by dots. Only the differences $d_B - d_A$ and $d_C - d_A$ are plotted. It can be seen that in congested periods, i.e. when the lines show up, the plots of delay differences follow the lines, thus confirming that the scheduler is performing appropriately. In fact, during the congestion periods the dots approximately follow the lines and for uncongested periods the observed differences are smaller. This corroborates equations (11), (13) and (14).

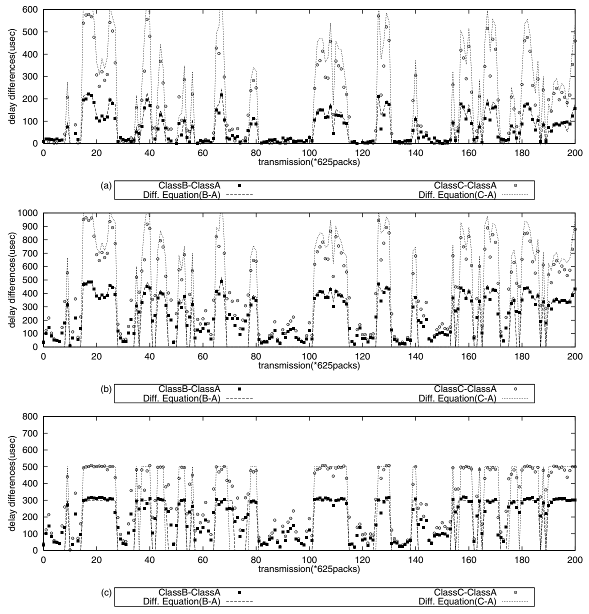


Fig. 7. Delay differences compared to maximum values given by a) Eq. (11) for configuration I b) Eq. (13) for configuration II c) Eq. (14) for configuration III.

4.4 Mixed Configurations

The discussion so far has focused on the scheduler operating under three specific configuration modes, however, it also supports mixed configurations, i.e. any subset of the traffic classes can be configured under a distinct mode. This point has major relevance in the scheduling area because it allows supporting a number of differentiation requirements. The example in Fig. 8(a) corresponds to an hybrid configuration involving configuration mode I and II. $Class_B$ and $Class_C$ are configured under configuration mode I, i.e. with the same U_i and different C_i parameters, whereas $Class_A$ and the other classes follow configuration mode II, i.e. different U_i and C_i parameters. As result, a mixed differentiation behavior is obtained. $Class_B$ and $Class_C$ share a common upper time constraint of $500\mu s$ and due to different congestion parameters, 2 and 1 respectively, suffer different delay violations. $Class_A$ is protected by a lower upper time constraint of $100\mu s$ and a much higher congestion parameter of 20 which causes very low violation values for this class. This example illustrates a possible configuration for a class with a behavior similar to EF PHB [26], consisting of a low latency and jitter[9] traffic class. The second example in Fig. 8(b) corresponds to mixed configuration of modes II and III. $Class_B$ and $Class_C$ are configured in mode III with upper time constraints of $500\mu s$ and $700\mu s$ and the same sensitivity to delay violations through identical congestion parameters. $Class_A$ and the other classes follow configuration mode II, as this class has different upper time and congestion parameters. Again, a different mixed behavior can be seen

[9] $Class_A$ delay oscillations are very low which means that jitter also assumes a low value.

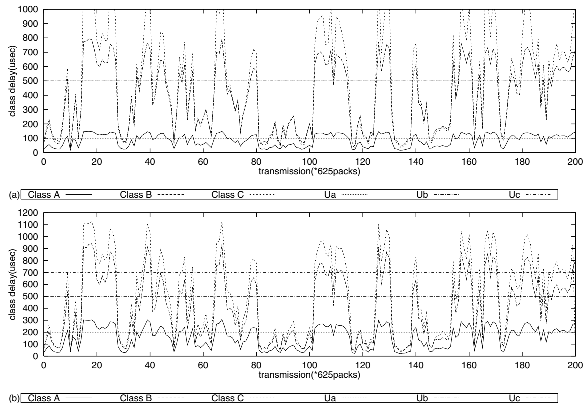


Fig. 8. a) $(U_A, U_B, U_C) = (100\mu s, 500\mu s, 500\mu s)$, $(C_A, C_B, C_C) = (20, 2, 1)$ (Conf. I+II) b) $(U_A, U_B, U_C) = (200\mu s, 500\mu s, 700\mu s)$, $(C_A, C_B, C_C) = (5, 1, 1)$ (Conf. II+III).

in Figure 8(b). When $Class_B$ and $Class_C$ are under congestion they suffer identical delay violations while $Class_A$ due to its higher congestion parameters achieves a delay violation which can be as lower as five times the obtained by the other classes. These examples illustrate the configuration capabilities of the scheduler which is able to achieve mixed differentiation behaviors.

5 Conclusions

This study proposes an hybrid queuing model to provide real-time traffic differentiation. The novel characteristic of the presented scheduler is the capability to simultaneously achieve an upper time limit for queuing delays and, under heavy load conditions, deny class starvation by providing an expectable differentiation schema for excess queuing delays. Through simple configuration is possible to control the expectable queuing delay and the congestion queuing delay of each traffic class. The hybrid behavior of the differentiation mechanism is obtained resorting to simple queue selection procedures, without using additional node state information. The scheduler allows three configuration modes each one involving distinct differentiation semantics. Mixed configurations modes are also possible, which improve the differentiation semantics of the proposed mechanism. Moreover, the differentiation achieved by the scheduler is fair at flow level, obeys the corresponding theoretical formulation and has reduced impact on the node complexity.

References

1. G. Armitage. *Quality of Service in IP Networks: Foundations for a Multi-Service Internet*. Macmillan Technical Publishing, April 2000.
2. S. Lima and P. Carvalho and A. Santos and V. Freitas. A Distributed Admission Control Model for CoS Networks using QoS and SLS Monitoring. In *ICC'2003 – IEEE International Conference on Communications*, May 2003.

3. R. Braden et al. Resource reservation protocol (rsvp). *RFC2205*, Sep. 1997.
4. I. Stoica and H. Zhang. Providing guaranteed services without per flow management. In *Proc. of SIGCOMM'99*, 1999.
5. Z. Zhang et al. Decoupling qos control from core routers: A novel bandwidth broker architecture for scalable support of guaranteed services. In *Proc. of SIGCOMM'00*, 2000.
6. M. Baldi. End-to-end delay analysis of videoconferencing over packet-switched networks. *IEEE/ACM Trans. on Networking*, 8(4), Aug. 2000.
7. W. E. Naylor et al. Stream traffic communications in packet switched networks: Destinations and buffer considerations. *IEEE Transactions on Communications*, COM-30(12), 1982.
8. P. Sousa and V. Freitas. A framework for the development of tolerant real-time applications. *Computer Networks and ISDN Systems*, 30:1531–1541, Dec. 1998.
9. T. Nandagopal and N. Venkitaraman. Delay differentiation and adaptation in core stateless networks. In *Proc. of INFOCOM 2000*, Tel Aviv, Israel, 2000.
10. I. Busse et al. Dynamic QoS control of multimedia applications based on rtp. *Computer Communications*, 19(1):49–58, Jan 1996.
11. R. Braden et al. Integrated services in the Internet architecture: an overview. *RFC1633*, June 1994.
12. S. Blake et al. An architecture for differentiated services. *RFC2475*, Dec. 1998.
13. L. Kleinrock. *Queueing Systems, Volume 2*. John Wiley and Sons, 1976.
14. G. Bolch et al. *Queueing Networks and Markov Chains – Modeling and Performance Evaluation with Computer Science Applications*. John Wiley and Sons INC., 1998.
15. C. Dovrolis and P. Ramanathan. A case for relative differentiated services and the proportional differentiation model. *IEEE Network Magazine*, 1999.
16. C. Dovrolis and D. Stiliadis. Relative differentiated services in the Internet: Issues and mechanisms. In *Proc. ACM SIGMETRICS'99*, 1999.
17. C. Dovrolis et al. Proportional differentiated services: Delay differentiation and packet scheduling. In *Proc. of ACM SIGCOMM'99*, 1999.
18. C. Dovrolis et al. Proportional differentiated services: delay differentiation and packet scheduling. *IEEE/ACM Transactions on Networking*, 10(1), Feb. 2002.
19. L. Essafi, G. Bolch, and H. Meer. Dynamic priority scheduling for proportional delay differentiated services. Technical Report TR-14-01-03, Univ. Erlangen-Nuremberg, March 2001.
20. M. Leung, J. Lui, and D. Yan. Adaptive proportional differentiated services: Characterization and performance evaluation. *IEEE/ACM Transactions on Networking*, 9(6), Dec. 2001.
21. P. Sousa, P. Carvalho, and V. Freitas. End-to-end delay differentiation of IP traffic aggregates using priority queueing models. In *Proc. of the IEEE Workshop on High Performance Switching and Routing (HPSR2002)*, pages 178–182, Kobe, Japan, May 2002.
22. P. Sousa, P. Carvalho, and V. Freitas. Tunning delay differentiation in IP networks using priority queueing models. In E. Gregori et al, editor, *Proc.* 2^{nd} *International IFIP-TC6 Networking Conference*, pages 709–720. LNCS 2345, Springer-Verlag, 2002.
23. C. Liu and J. Layland. Scheduling algorithms for multiprogramming in a hard real-time environment. *Journal of the ACM*, 20(1):44–61, Jan 1973.
24. V. Sivaraman. Statistical analysis of delay bound violations at earliest deadline first (EDF) scheduler. *Performance Evaluation*, 36(1):457–470, 1999.
25. S. Bodamer. A scheduling algorithm for relative delay differentiation. In *Proc. of the IEEE Conf. on High Performance Switching and Routing*, pages 357–364, Heidelberg, June 2000.
26. B. Davie et al. An expedited forwarding PHB (per-hop behavior). *RFC3246*, March 2003.

Authentication Issues in Multi-service Residential Access Networks

Judith Rossebø[1], John Ronan[2], and Kristian Walsh[3]

[1] Telenor Research & Development, Telenor Communication II AS, N-1331 Fornebu, Norway, judith.rossebo@telenor.com
[2] Telecommunications Systems Software Group, Waterford Institute of Technology, Waterford, Ireland, jronan@tssg.org
[3] Department of Mathematics and Computing, Cork Institute of Technology, Cork, Ireland, krwalsh@cit.ie

Abstract. Multi-service residential access networks allow residential customers to choose amongst a variety of service offerings, over a range of Core Networks and subject to user requirements such as QoS, mobility, cost and availability. These issues place requirements on authentication for network access, with a need for mutual authentication of the residential gateway (RG) to the local access point (LAP). The EU-IST project TORRENT is building a testbed providing for multi-service residential access networks in order to demonstrate the benefit of intelligent control, both for the customer and for the network operators and service providers. Adequate security measures are essential in order to secure access to the TORRENT system and services and for QoS provisioning to authorised users. This paper examines the authentication issues for the TORRENT system and presents a public key based authentication protocol for mutually authenticating the RG and the LAP.

Keywords: Authentication, Public Key Infrastructure, Encryption, Residential Gateway

1 Introduction

Multi-service residential access networks are concerned with exploiting the use of shared physical access networks for a range of different services and traffic types optimising bandwidth utilisation in existing access networks while meeting user requirements related to QoS, security, cost and availability. In order to deliver such offerings to the residential customers, an infrastructure is required mapping user requirements to the appropriate networks, services, and applications. The infrastructure typically incorporates a residential gateway in the home and one or more serving local access point(s)(LAP) integrating access technologies with services and core networks. The residential gateway(RG) connects the home network technologies (especially WLAN, Ethernet) to the access network(s), e.g. cable, ISDN, *x*DSL, wireless, LMDS or any combination thereof.

Functions of the local access point may include providing customer negotiation facilities and host accounting and security functionality (e.g., AAA services) of customer access to e.g., metering, security, and monitoring.

This work has evolved from investigations in the TORRENT project of authentication issues in multi-service residential access networks. In future multi-service residential access networks, users may have the opportunity to choose between several network operators and service providers. Authentication requirements of the different network operators and service providers can lead to the situation that a user has to keep track of several different usernames and passwords and different methods of authenticating to the networks and services. There is a need for a single mechanism for authentication, which gives the user access to all services. For this, the mechanism should be suitable for services requiring strong authentication (e.g. signing your loan application electronically). This paper discusses the issues involved and presents our scheme for public-key based authentication for which a smart card is used as certificate and key container and may also be used for authentication to many services.

1.1 TORRENT Overview

TORRENT[8] is an EU-supported Framework V project, aiming to build a test-bed for multi-service residential access networks. This test-bed (Figure 1) will allow demonstration of the benefit of intelligent control for the customer, network operators and service providers. An important goal is to optimise the bandwidth utilisation in existing access and core networks, while at the same time meeting user requirements in an optimal manner. These requirements include Quality of Service (QoS), security, cost, and availability. Security is of major importance, and adequate security measures are essential in order to secure access to the TORRENT system and services, and also for provisioning QoS to authorised users.

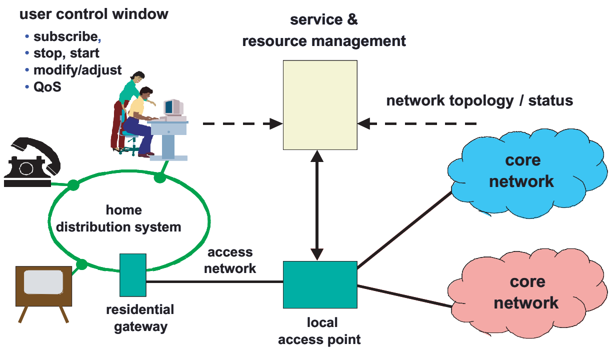


Fig. 1. TORRENT Architecture

IPv6 is integrated in TORRENT as a transport protocol and the IP Security Protocol (IPsec) is used as a service for securing the data between the Residential Gateway (RG) and the Local Access Point (LAP). Investigations of various authentication and key agreement schemes have been carried out in the IPsec performance trials, as documented in [2].

2 Background and Objectives

It is a requirement of TORRENT to mutually authenticate the RG and the LAP; to provide users with secure access to TORRENT services; and also to authenticate users accessing the system e.g. to make changes to the user profile and user preferences (service subscription, QoS, cost, etc.) It foreseen that if a hardware (HW) token were used as key holder of the authentication exchange, then this HW token could also be used for user authentication (of the RG user) and eCommerce applications. The same token could be used for authentication and key exchange for an IPsec VPN tunnel service. Certificates and associated private keys for authentication and encryption of agents and agent communication could also be stored on the HW token. In fact, certificates and associated private keys for the services of user authentication, electronic signature, and encryption can be reused by TORRENT's agent based Service to Resource Management system for authentication of the user agents. Therefore, public key techniques were explored early in the TORRENT project.

2.1 Assumptions

We make the following general assumptions about the TORRENT system:

- The RG is vulnerable
- The link between the RG and LAP is vulnerable.
- The user access to the user profile interface is vulnerable.
- The link (not shown) between service domains (e.g., LAP to LAP) is vulnerable.

2.2 Security Objectives

The authentication requirements for the TORRENT system have been determined by a threat analysis. It was determined that the threats of masquerading by LAP or RG can be mitigated by mutually authenticating the LAP and the RG. Authentication is important e.g. to ensure that the authorised customer behind an RG is getting the QoS that was requested, to reduce the likelihood of fraud and also as a baseline for avoiding repudiation of messages e.g. payments. Users and providers of networks and services will thus benefit from this security service. A scheme for providing this will be presented later in the paper.

Authentication of agents to each other in the agent system is also required to prevent manipulation of traffic sent between agents. Authentication of agents is addressed in [4].

2.3 Enabling Quality of Service

Authentication alone will not guarentee that the customer receives the QoS that was requested. However, assuming that QoS mechanisms are in place to maintain and control the quality of service, mutual entity authentication is an important countermeasure to a number of threats to QoS provisioning.

For example, if authentication of the RG to the LAP is insufficient, an attacker may make a distributed attack from many false RGs allocating a substantial number of QoS enabled flows, which binds up the resources and degrades the performance for authorised user flows. An important mitigating measure to counter this attack is mutual authentication of the RG and the LAP. In the case of the TORRENT project, the user may select a level of QoS on a per service basis via a GUI hosted on the LAP. Sufficient authentication of the user for access to the GUI (and integrity protection of the communication on the link between the RG and the LAP) is required to prevent unauthorised users from masquerading as authorised customers in order to obtain a higher level of QoS without paying for it.

3 Authentication and Key Agreement Based on Public Key Techniques

In this section we will discuss the case for public key authentication techniques for network access.

3.1 Background

Historically, authentication for network access has been done using shared secret techniques and this has proven to be scalable. For ISDN and now ADSL the residential user is issued a terminal per service (NT box for ISDN, ADSL modem for ADSL). ADSL modem authentication is at best, password based and open to dictionary attacks when the password is small or insufficiently random. Requirements have been lax: it is well known that most ADSL modems can be hacked without requiring sophistication [5]. 2G and 3G mobile telecommunication systems also use secret key techniques for network access.

As the number of access networks increases scalabity issues become more significant. In GSM, for example, scalability is achieved using brokers and roaming agreements. A small operator has an agreement with a roaming broker, which establishes roaming agreements with a lot of other operators. It can be argued that as the number of operators and roaming agreements grows significantly, then these bilateral agreements may be inefficient and costly to maintain. But essentially, scalability issues alone do not provide a strong argument for the case for public key techniques for network access.

Public key techniques were ruled out early on in the 3G design and standardisation process as these were considered too complex and seemed to require too much computational overhead. Since then, however, public key based mechanisms have been successfully implemented in the GSM SIM card for e.g. mCommerce[9] and in smartcards[6], demonstrating that computational overhead is no longer an issue.

The strong arguments for public key techniques are scalability and to some extent mobility (of the user to choose freely between networks and service providers), and the elegance of the reusability of public key techniques for a multitude of services such as authentication, and key exchange, notary public services, eCommerce, mCommerce, and electronic signatures. Use of public key techniques is also motivated in part by stronger requirements for user anonymity — public key authentication offers possibilities for providing strong user identity and location confidentiality as the user id and location information does not need to be transmitted in the clear over the network. For example, the public key of the authorised receiving party can be used to encrypt the identity and other private information belonging to the user.

4 Authentication in TORRENT

In this section we will propose a scheme for public key authentication for securing network access from the user of the RG to the LAP.

The LAP is fitted with a hardware key container with a (possibly several) server certificate(s) and associated private key(s). A smartcard as certificate and key container is inserted in the RG. The user behind the RG has a relationship with a trusted third party (Òcertificate validator/brokerÓ).

At least one X.509 certificate and associated pair of keys are stored on the smartcard for authentication and encryption purposes (actually, the public key is contained in the certificate, while the private key is not). The trusted third party's public key certificate is also stored on the smartcard to enable validation of foreign incoming certificates.

The RG user Smartcard contains a certificate binding the RG user to a public key suitable for both encryption and signature verification (or it contains two separate certificates). The LAP has a certificate binding the LAP operator to certificate issued by a Certificate Authority (CA). Both must have X.509 certificates that can be validated by a trusted third party, which we call the Certificate Validation Clearinghouse/Broker (CVC). The CVC functions as a trusted third party and has an agreement with the CA that issued the certificate to the LAP allowing it to perform certificate validation services on behalf of the CA, and it also has such an agreement with the CA that issued the certificate(s) to the RG user. The CVC public key is installed on the RG user smartcard a priori. Note, it is feasible that the RG user may have the public keys of several CVCs installed on their smartcard.

The aim is to mutually authenticate the RG and the LAP using public key techniques. The motivation for this is the case that the user of the RG does not have to be bound to a Service Provider or Operator (by a subscription) but is free to shop around for network access and services. In this case, shared secret keying techniques are not appropriate. It should be noted that this algorithm can be applied to both wired and non-wired network access.

Use of public key techniques for network access has been studied in the SHAMAN project [7] and two methods for authentication and key agreement using public key techniques are described. The protocol presented in this paper is different from those presented by SHAMAN. In the method described in SHAMAN the mobile node has a subscription to a home operator, and must establish network access with an access network, which has an agreement with the home operator or a roaming broker. This network access point (e.g. LAP) has a number of pre-installed public key certificates (signed by each trusted third party or home network with which it has an agreement). The access network sends the appropriate one to the node (e.g. RG) and this is used by the node to assure the node that a roaming agreement exists with the home network.

Similarly to SHAMAN, public key techniques are used to mutually authenticate the LAP to the RG and the RG to the LAP using the the CVC. In the TORRENT case, however, the user does not necessarily have a relationship/subscription to a home network, nor does the LAP have to have an agreement with other access network operators. It is the CVC's public key that is pre-installed in the RG user's smartcard and also on the LAP, and which is used in the validation process. The CVC can function as a broker and a clearinghouse.

The CVC validates the RG user's certificate on behalf of the Certificate Authority for the LAP, and the LAP's certificate on behalf of the RG. . How this validation is performed is outside of the scope of this paper: a thorough explanation can be found in [3].

Once the certificate validation process is completed, the type of payment for services (e.g., bandwidth, QoS, VPN) can be agreed using the preferred payment method such as credit card, online banking, billing directly based upon some method associated with the smartcard, etc. (again, the exact method used is outside of the scope of this paper).

5 Protocol RRW Strong Two-Way Entity Authentication

The basic protocol involves A (the RG User Smartcard), B (the LAP), and T (the CVC server). The initial checking of certificates incorporates the X.509 protocol for strong Three-Way Authentication[1].

At outset the RG User Smartcard contains at least one public key pair suitable for both encryption and signature verification and in accordance with X.509 standards. The RG user's Smartcard must also acquire (and authenticate) the CVC encryption public key *a priori*.

The LAP has its public key pair for signature and encryption. The CVC server has a public key pair for signature and encryption. The CVC has an agreement with the CA that issued the certificate to the LAP, allowing the CVC to validate on behalf of the CA which issued the LAP certificate. The CVC also has an agreement with the CA that issued the card to the user.

SUMMARY: RG User Smartcard interacts with a Trusted CVC server and LAP.
RESULT: mutual entity authentication.

1. Notation

   A denotes the RG User Smartcard.
   B denotes the LAP
   T denotes the CVC server.
   I_T denotes the identification of the CVC so that the authentication request can be sent to the correct CVC.
   $P_X(y)$ denotes the result of applying X's encryption public key to data y.
   $S_X(y)$ denotes the result of applying X's signature private key to data y.
   r_X denotes a random number generated by X .
   d_X denotes a random number generated by X.
   $Cert_X$ is a certificate binding party X to a public key suitable for both encryption and signature verification. Remark: A good practice is to avoid using the same cryptographic key for multiple purposes.
   $cert_X ok$ denotes the "$Cert_X$ has been validated" message.

2. System Setup

   (a) Each party has its public key pair for signature and encryption.
   (b) The encryption public key of the CVC is installed on both the the RG user Smartcard and the LAP *a priori*.
   (c) The CVC server has an agreement with the Certificate Authority (CA) that issued A's public key certificate(s) allowing the CVC to validate A's certificate on behalf of the CA, and similarly with the Certificate Authority that issued B's public key certificate.

3. Protocol messages

$$A \rightarrow B : Cert_A, r_A, S_A(r_A) \tag{1}$$
$$A \leftarrow B : Cert_B, S_B(r_A, r_B), r_A, r_B \tag{2}$$
$$A \rightarrow B : (r_B, S_A(r_B)), P_T(Cert_A, Cert_B, d_A, r_A, r_B), I_T \tag{3}$$
$$B \rightarrow T : P_T(Cert_A, Cert_B, d_A, r_A, r_B), S_A(r_B) \tag{4}$$
$$A \leftarrow T : P_A(cert_B ok, S_T(d_A)) \tag{5}$$
$$B \leftarrow T : P_B(cert_A ok, S_T(r_B)) \tag{6}$$

4. Protocol actions

   a) A generates r_A, signs it using A's own private key, and sends to B message (1).
   b) B extracts A's public key from $Cert_A$ and uses this to check the signature on the data r_A. If the check succeeds, B is satisfied that A's posesses the private key corresponding to the certificate $Cert_A$, but not that B and A are authorised to communicate.

c) B generates r_B, signs the concatenation of this and r_A and passes message (2) back to A.

d) A extracts B's public key from $Cert_B$ and uses this to check the signature on the data (r_A, r_B). If the check succeeds, A is satisfied that B possesses $Cert_B$ and its corresponding private key, but not that A is permitted to communicate with B.

e) A signs the message r_B. A also generates d_A, a new challenge value to be answered by the CVC. A encrypts data for the CVC server T containing $Cert_A$, $Cert_B$, r_A, r_B and d_A. This encrypted message is sent along with $(r_B, S_A(r_B))$ and the plaintext CVC identification I_T to B (for relaying to T).

f) B checks the signature on the data r_B, then B uses the cleartext identifier I_T in message (3) to relay (4) to T. Before relaying the message, B appends $S_A(r_B)$, (A is sending r_B inside the encrypted message).

g) T decrypts (4) using its private decryption key. T validates $Cert_A$ and $Cert_B$. T extracts A's public key from $Cert_A$ and uses it to verify that $S_A(r_B)$, as appended by B is a valid signature of the r_B value which was encrypted by A. This check prevents A from exchanging one certificate with B initially, but sending a different one to T for authentication.

h) T signs r_B and d_A, and sends the messages in (5) and (6). Each message is encrypted using the recipient's public key.

i) A decrypts (5) and verifies T's signature on the challenge d_A. If this signature is valid, A declares authentication of B successful.

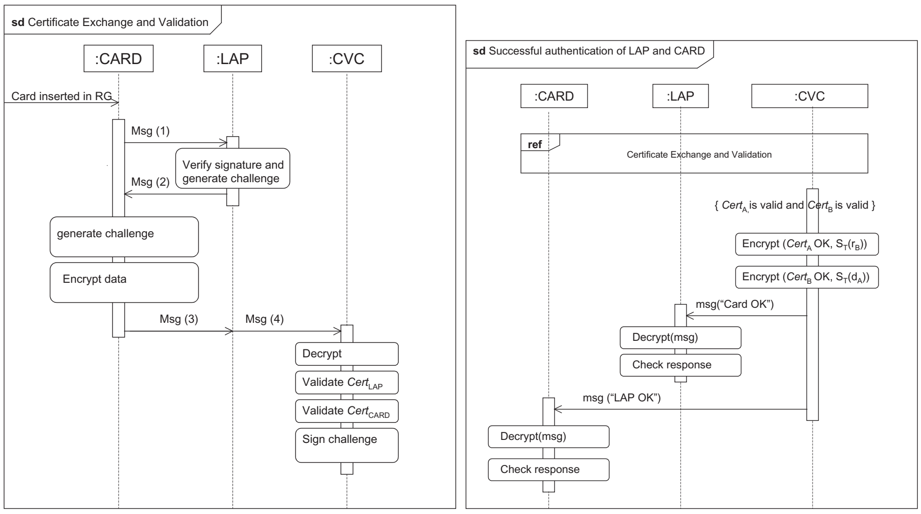
j) B decrypts (6) and verifies T's signature on the challenge r_B. If this signature is valid, B declares authentication of A successful.

Note: Upon authentication, A and B may proceed directly with a key-exchange for IPsec. Key exhcange was deliberately decoupled from the authentication process to allow operators to offer VPN as a value-added service if they wish.

5.1 Periodic Validation

To prevent redirection of the communications channel once authentication has been completed, it is necessary to periodically ensure that the authenticated parties are still in control of the communications channel. The method used here is a periodic "challenge-response" from LAP to the Card in the RG.

If the LAP does not receive a signed response from the card within a reasonable amount of time, the LAP will cease routing traffic to or from the RG containing the non-responsive card. Conversely, if the Card does not receive a challenge from the LAP within a reasonable time, it will disable the RG interfaces on the assumption that the LAP has either failed or has been compromised.


sd Certificate Exchange and Validation
:CARD
:LAP
:CVC
Card inserted in RG
Msg (1)
Verify signature and generate challenge
Msg (2)
generate challenge
Encrypt data
Msg (3)
Msg (4)
Decrypt
Validate $Cert_{LAP}$
Validate $Cert_{CARD}$
Sign challenge
sd Successful authentication of LAP and CARD
:CARD
:LAP
:CVC
ref
Certificate Exchange and Validation
{ $Cert_A$ is valid and $Cert_B$ is valid }
Encrypt ($Cert_A$ OK, $S_T(r_B)$)
Encrypt ($Cert_B$ OK, $S_T(d_A)$)
msg("Card OK")
Decrypt(msg)
Check response
msg ("LAP OK")
Decrypt(msg)
Check response

(a) (b)

Fig. 2. Authentication Sequence: (a) initial LAP/Card certificate exchange, (b) sucessful authentication process

6 Sequence Diagrams

Figure 2 shows the authentication progression. The two alternative sequences in Figure 3 show possible deviations in authentication progression, depending on the validity of the LAP or RG smartcard holder's certificates.

7 Feasibility

The greatest barrier to implementation of this system is the ability of the smart card processor to perform the necessary encryption and decryption operations.

The protocol presented here requires the smart card to perform one signature, one encryption, one decryption and one signature validation. Of these, the encryption (of the combined certs and challenges message) involves by far the largest amount of data. These operations must be performed on-card, as the RG in which the card sits is a non-trusted system.

7.1 Encryption Algorithm

Because of the limited computational power of the smart card processor, a computationally-light, but still secure encryption scheme is required. Elliptic Curve Cryptography (ECC)[15] offers strength at least equal to the more widely-deployed RSA. However, ECC requires shorter key lengths and lower computational load[16], and as such would be an ideal candidate for use in the system described here.

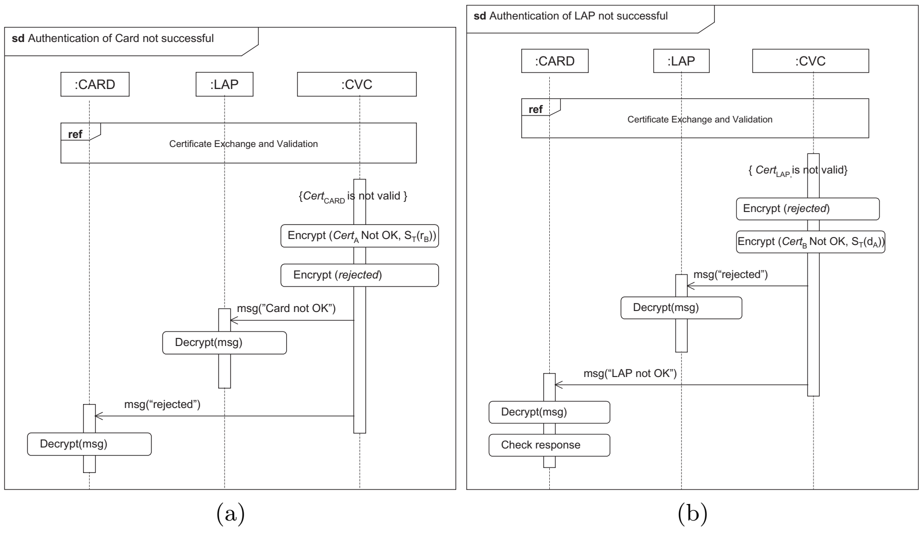

sd Authentication of Card not successful
:CARD
:LAP
:CVC
ref
Certificate Exchange and Validation
{Cert_CARD is not valid }
Encrypt (Cert_A Not OK, S_T(r_B))
Encrypt (rejected)
msg("Card not OK")
Decrypt(msg)
msg("rejected")
Decrypt(msg)
(a)
sd Authentication of LAP not successful
:CARD
:LAP
:CVC
ref
Certificate Exchange and Validation
{ Cert_LAP, is not valid}
Encrypt (rejected)
Encrypt (Cert_B Not OK, S_T(d_A))
msg("rejected")
Decrypt(msg)
msg("LAP not OK")
Decrypt(msg)
Check response
(b)

Fig. 3. Sequences for failed authentication process — (a) invalid card, (b) invalid LAP

7.2 Smart Card Performance

The encryption and decryption operations performed by the card are not time-critical: because authentication is not performed regularly, a time of up to three seconds from card insertion (a similar performance to most pay TV systems) is acceptable and easily achievable.

Current microprocessor smart cards can already be used for key generation and encryption using ECC techniques, and so are well able to perform the encrypt/decrypt operations required by the protocol presented here.[17][18].

7.3 Threat Analysis

The primary weakness in this system, as in all trusted-party systems, is the integrity of the CVC server. However, assuming that the operator of the CVC Server takes adequate measures to protect it from subversion, destruction or replacement, a potential attacker is left with only three other targets: the smart card, the RG, and the LAP. Each of these is examined in turn.

Smart Card. The Smart card is a closed computer whose only means of communication with the RG is via a simple serial data link[19]. The firmware running on the smart-card card CPU controls what information is sent along this serial link to the RG.

The smart card private key and public key pair are written into on-card read-only memory during manufacture. The software on the smart card will never cause the private key to be sent out of the card, and there is no other way to access the on-card memory without resorting to industrial disassembly of the card chip.

Messages from the smart card to the LAP and CVC are encrypted on the card before being passed to the RG for transmission. The exception to this is the initial exchange of certificates between smart card and LAP, when the smart card sends its X.509 certificate as plaintext. However, the only information of consequence in this certificate is the card's public key, disclosure of which does not compromise the safety of the card's private key.[13]

RG. The RG is envisioned as a mass-produced, dedicated computer system, based on open-standards. As such, the RG is susceptible to substitution: a malicious user can create their own fraudulent RG using a general purpose PC and a smart-card interface. It is precisely for this reason that the protocol as presented here assumes that the RG is a hostile party in the system.

No critical messages are passed to the RG without first being encrypted. The RG can choose not to forward these messages, but this will result in a denial of service, as the LAP will not enable the RG to LAP link until authentication is complete.

It is conceivable that a tampered RG could be re-programmed to perform authentication once with a legitimate card inserted, and then remain on-line indefinitely, regardless of the presence of the smart card. The periodic validation scheme (§5.1) is designed to defeat this attack: without a legitimate card in the RG to answer the challenge of the LAP, the RG will quickly be disconnected from the network.

LAP. The LAP is a dedicated computer system owned and maintained by the network operator, and located on their property. As such, the LAP is a less attractive target for direct attack than the RG or smart card. However, where a shared-medium network (e.g., wireless) serves the LAP and RGs, the risk of LAP substitution arises.

In order to defeat a user who sets up a fraudulent LAP and replays the genuine LAP's contributions to the authentication protocol, the genuine LAP passes a random challenge message to the the smart card during the initial LAP/Card certificate exchange.

If a LAP's private key is somehow disclosed, the compromised LAP can be issued with a new key pair, and the CVC server can reject any messages signed with the old, compromised, private key. No change is necessary to the smart cards, as they do not store LAP information.

8 Comparison with Existing Schemes

It was our intention to use existing, open-standard protocols and technology wherever possible. For this reason, the protocol presented here incorporates the X.509 Strong Three-Way mutual Authentication protocol[1], and extends it for use with a trusted third party.

The proposal is similar to Kerberos[12] in that it involves a party A interacting with a trusted server T and a party B. However, The encryption algorithm of Kerberos is symmetric, and public key techniques are not involved. A ticket is generated which allows A to re-use the ticket for multiple authentications to B without involving T.

In our approach, multiple authentications are not necessary. The periodic validation function (§5.1) allows the RG and LAP to communicate over long periods of time without requiring re-authentication. Once authentication is finished, the Card still keeps listening on a socket for an occasional "heartbeat" request from the LAP/Authentication server. So even if the card is removed without being able to send the "Disable Services" message to the RG (or if the RG has been tampered with to ignore this message), the LAP will quickly realise the card is no longer in the RG, and will stop routing the RG's traffic.

9 Re-usability of the Certificates and Keys

In this section we give a few examples of how the certificates and keys can be reused for securing other functions of the TORRENT system, for user authentication, application security, eCommerce, etc.

As explained above, the customer behind the RG has a smartcard inserted in the RG. The smartcard contains at least one X.509 certificate and associated asymmetric keys. The certificate belongs to the customer and is linked to the customer profile database. Depending on the certificate policy, the customer may have to present themselves in person to the CA to register and receive the certificate and keys.

The card and accompanying certificate(s) and keys may be used for key exchange when setting up an IPsec VPN to securing the access link between the RG and the LAP [2]. Using Qualified Certificates the smartcard may even suitable for creating electronic signatures [14] [20].

In a system where users can make changes to their profile on-line (e.g., QoS requests, billing details, etc.), the X.509 certificate can serve as authentication to these services. If the operator requires electronic signature of requests from users, the user can also sign them electronically at this point, provided that the card contains a certificate that can be used for electronic signature. It can also be foreseen that pay-per-use services can be provided using this scheme and that the user can provide an electronic signature for payment purposes.

10 Conclusion

10.1 Discussion

In this research we focus on mutual authentication procedure between the customer's residential gateway and the Local Access Point using a proposed authentication protocol that combines techniques that are already proven to be reliable (X.509 certificates, smartcards, public key cryptography). The motivation comes from a requirement of the TORRENT IST project for a method of mutually authenticating a residential gateway (in the home) and a local access point (in the operator's premises). Bearing in mind that with future multi-service residential access networks, users may have — or demand — the facility to choose between several network operators and service providers.

The authentication requirements of the different operators and providers can lead to a situation where the user has to keep track of several combinations of usernames and passwords which becomes much more difficult to manage as time progresses. The process described in this paper proposes a scheme in an attempt to solve these issues for the TORRENT project and other deployments of LAP/RG types of systems. The process can also be applied to authenthicate to, and access mobile access networks, e.g. for use in 3G and future networks where the smart card may be used for authentication to many services.

10.2 Related Work

Use of public key authentication and key exchange for network access has been studied in the IST-SHAMAN project [7]. The first method proposed by SHAMAN is described in §4, and requires each network access point (e.g. LAP) to have a number of pre-installed public key certificates (signed by each trusted third party or home network with which it has an agreement). As the LAP operates in a multi-provider environment, this scheme could quickly lead to a situation for which the LAP is overloaded by certificate processing.

Our method does not require any pre-existing agreement between the users and the networks they attempt to join, and pre-installation of home network operator certificates on the LAP is also not required.

The second method described in SHAMAN differs also from our work in that it does not require client authentication. This method is concerned with providing anonymous access to the network based on immediate payment for services and therefore does not require authentication of the client. This method is purely concerned with authenticating the network.

The WLAN Smartcard Consortium[11], which was established in February 2003, is working towards defining specifications for world-wide access to WLAN networks using smartcard security in order to provide privacy, roaming and related capabilities. As this group is recently established, the specifications are not publicly available at this time. It is projected by the Consortium that they will be available by the end of the 2003.

Similarly, the ETSI AT NGN@Home has announced plans for work on a deliverable entitled "Access and Terminals (AT); Home Area Networks and the support of Next Generation Services; Part 6: Security and Copyright issues" [10]. The intention is to outline the security and copyright (including Digital Rights Management and Privacy) issues related to the support and delivery of Next Generation Services and applications both to and within a Home Area Network.

10.3 Future Work

Future work will include schemes for using the Smartcard to gain secure access to a service at a chosen QoS class. Methods for protection of personal privacy will also be investigated. In the age of full IPv6 deployment, it is envisioned that protection of personal privacy can be made much easier. With IPv6, and certificates, IP addresses can be public, but the certificate, coupled with public key techniques, can be used to govern what traffic is allowed in, and what traffic should be prohibited. Only users with pre-approved certificates can reach (and communicate with) the user at the destination IP address.

Acknowledgements. The research on which this paper reports has partly been supported by the IKT 2003 project SARDAS (15295/431) funded by the Research Council of Norway; and the European Union-supported TORRENT project (IST-2000-25187).

References

1. Menezes, A. J, P. C. van Oorschot, S. A. Vanstone, *Handbook of Applied Cryptography*, CRC Press, Florida, 1997
2. Ronan J, Malone P, Ó Foghlú M, *Overhead Issues for Local Access Points in IPsec enabled VPNsÓ*, IPS Workshop, Salzburg, February 2003. Retreived: 3. April, 2003 from `http://www.ist-intermon.org/workshop/papers/09_01_vpn-overhead.pdf`
3. Ølnes J, *Trusted Certificate Validation Services – Breaking the PKI Deadlock*, jon.olnes@validsign.com , `http://www.validsign.com`, 10. October 2002
4. Houmb, Siv Hilde. *Security Issues in FIPA Agents*, paper in progress.
5. Cert Advisory CA-2001-08, *Multiple Vulnerabilities in Alcatel ADSL Modems*, 12. April 2001. Retreived: 3. April, 2003 from `http://www.cert.org/advisories/CA-2001-08.html`
6. Schlumberger, *Schlumberger Smart cards Cryptoflex Home Page*, Retreived: 3. April, 2003 from `http://www.cryptoflex.com/index.html`
7. IST-SHAMAN Deliverable D09: *ÒDetailed Technical Specification of Security for Heterogeneous AccessÓ*, June 2002
8. TORRENT (Technology for a Realistic End User Access Network Test-bed), IST-2000-25187. `http://www.torrent-innovations.org`
9. Telenor Mobil *MobilHandel^{TM}overview.* Retrieved: 10. June, 2003 from `http://telenormobil.no/tjenester/mobilhandel/index.jsp`
10. ETSI AT MGN@Home *Access and Terminals (AT); Home Area Networks and the support of Next Generation Services; Part 6: Security and Copyright issues*, work in progress. Retrieved 10. June 2003 from `http://portal.etsi.org/Portal_Common/home.asp`
11. WLAN Smartcard Consortium, homepage. Retrieved: 10. June, 2003 from `http://www.wlansmartcard.org/`.
12. Neuman B C and Ts'o T, "Kerberos: An Authentication Service for Computer Networks", *IEEE Communications* **32**(9),*pp 33–38.* September 1994
13. Diffie W and Hellman M E, "New directions in cryptography", *IEEE Transactions on Information Theory* **22**(1976), *pp 644–654*

14. EU Directive 1999/93 on Electronic Signatures *ETSI TS 111 456 for Qualified Certificates*
15. Satoh T, Araki K, Miura S. "Overview of elliptic curve cryptography", *Proc. PKC'98*, LNCS **1431**,*pp. 29-49*, Springer-Verlag, 1998.
16. Gupta V, Gupta S and Chang S *Performance Analysis of Elliptic Curve Cryptography for SSL* ACM Workshop on Wireless Security (WiSe), Mobicom 2002, Atlanta, Georgia, USA September. 2002. Retrieved 10. June 2003 from `http://research.sun.com/projects/crypto/performance.pdf`
17. Aydos M, Yanık T, and Koç Ç, *An High-Speed ECC-based Wireless Authentication Protocol on an ARM Microprocessor*, Annual Computer Security Applications Conference, New Orleans, 2000. Retreived: 10. June, 2003 from `http://acsac.org/2000/papers/24.pdf`
18. Woodbury A, *Efficient Algorithms for Elliptic Curve Cryptosystems on Embedded Systems*, Worcester Polytechnic Institue, Massachusetts. Retrieved 10 June, 2003 from `http://www.wpi.edu/Pubs/ETD/Available/etd-1001101-195321/unrestricted/woodbury.pdf`.
19. International Organisation for Standardization, *ISO/IEC 7816-3:1997: Information technology – Identification cards – Integrated circuit(s) cards with contacts – Part 3: Electronic signals and transmission protocols* , Geneva, 1997.
20. Santesson S et al, *RFC 3039: Internet X.509 Public Key Infrastructure Qualified Certificates Profile*, January 2001. Retreived: April 3, 2003 from `http://www.ietf.org/rfc/rfc3039.txt`

Secure Inclusion of Phones into Online E-meetings

Peter Parnes

Luleå University of Technology
Department of Computer Science and Electrical Engineering
Media Technology Division
971 87 Luleå
Sweden.
`Peter.Parnes@sm.luth.se`

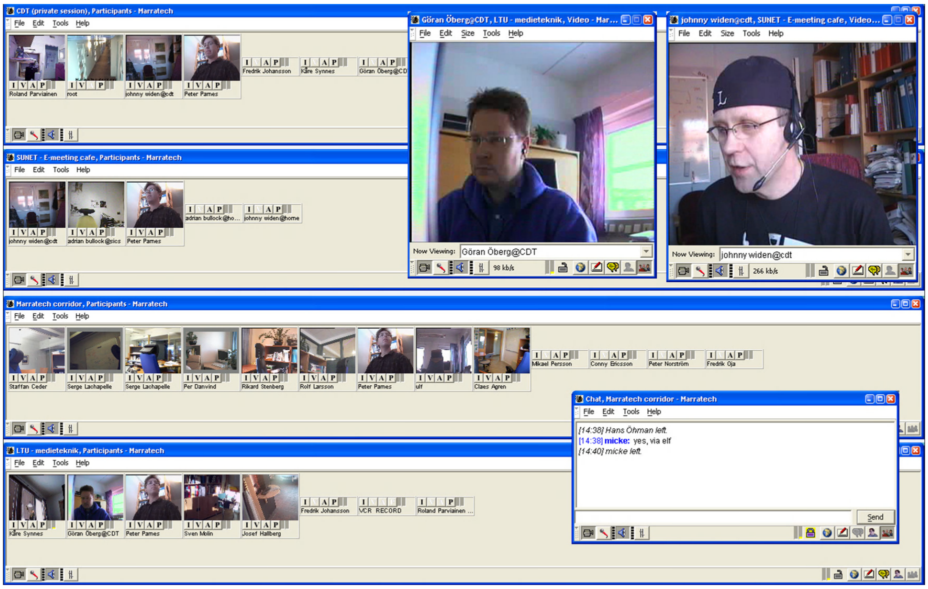
Abstract. Online Internet based e-meetings for synchronous communication is becoming more and more common and the need for secure communication is a strong requirement from both corporate and private users. At the same time not all users can always be available via the Internet and as phones (fixed and mobile) are still the dominant communication method there is a need for general inclusion of phones into online e-meetings, without compromising the security of the online session. This paper presents an architecture supporting three different scenarios for including normal phones into e-meetings, and at the same time keeping the security of the session intact. It is also shows how an e-meeting portal is used for simple inclusion of phones into e-meetings even if the inviting client is behind a NAT gateway or a firewall.

1 Introduction

The need for people to meet over the Internet is becoming pervasive. Meeting scenarios include everything from scheduled meetings for project groups and net-based learning to 24-hour connected e-corridors for shared group awareness and an increased sense of presence. The latter allows users to exchange real-time media via a common group area were all connected users can see the transmitted video and as the number of users increase the need for scalable distribution increases. As a reference, it can be mentioned that the author is daily part of 3-5 different e-meetings or e-corridors and is constantly receiving between 0.5 and 2 Mbps of real-time video data distributed over 10-50 simultaneously sending users.

One such system is the Marratech Pro[1], which allows users to interact synchronously via audio, video, text based chat, shared whiteboard and shared WWW browsing. The client supports both data distribution using IP-multicast [1] for direct peer-to-peer communication and in the case that IP-multicast is not available the Marratech E-Meeting Portal is used as a data distribution server. Figure 1 shows the application user interface for 4 simultaneous e-corridors.

[1] <URL:http://www.marratech.com/>


CDT (private session), Participants - Marratech
SUNET - E-meeting cafe, Participants - Marratech
Marratech corridor, Participants - Marratech
LTU - medieteknik, Participants - Marratech
Göran Öberg@CDT, LTU - medieteknik, Video - Mar...
johnny widen@cdt, SUNET - E-meeting cafe, Video...
Chat, Marratech corridor - Marratech
File Edit Tools Help
File Edit Size Tools Help
Now Viewing: Göran Öberg@CDT
Now Viewing: johnny widen@cdt
98 kb/s
266 kb/s
Send
Roland Parviainen
root
johnny widen@cdt
Peter Parnes
Fredrik Johansson
Kåre Synnes
Göran Öberg@CDT
adrian bullock@sics
adrian bullock@ho...
johnny widen@home
Staffan Ceder
Serge Lachapelle
Per Danvind
Rikard Stenberg
Rolf Larsson
ulf
Claes Agren
Mikael Persson
Conny Ericsson
Peter Norström
Fredrik Oja
Sven Molin
Josef Hallberg
VCR RECORD
Roland Parviainen ...

[14:38] Hans Öhman left.
[14:38] micke: yes, via elf
[14:40] micke left.

Fig. 1. Screen shot of 4 simultaneous e-corridors

Something going hand in hand with on-line meetings is the user requirement for secure communication, where all exchanged media data is encrypted, including the real-time audio and video. It is here obviously attractive that in the case a portal is part of the e-meeting, it should not have to be able to decrypt the media, as the portal might be hosted by a third party (e.g. an ISP or a computer support department) not trusted by the members of the online group. In general, it is always advisable to share a common secret with as few other parties as possible, if it should remain secret. Symmetric encryption is here used due to a private/public encryption scheme does not scale to a large number of users when exchanging high bandwidth real-time media.

All users do not always have access to a local computer to allow for interaction with e-meeting groups and thus it is attractive to be able to join an e-meeting via a normal phone (wired or wireless). This inclusion of phones should be done without compromising the security of the session.

This paper presents an architecture for secure inclusion of phones into online meetings using three different scenarios, where each scenario has certain benefits over the other two. The architecture is primarily designed around the assumption that an existing member of the e-meeting invites phone users into the session, i.e. the phone will get a call. The opposite where the phone user calls directly into the meeting is left for further work and its current status is further discussed in section 5.2.

The rest of this paper is organized with section 1.1 discussing related work and continues with an overview of the proposed system in section 2. Section 3 presents the architecture and its implementation, while section 4 presents the evaluation of the work. The paper is concluded in section 5 together with a discussion about future work.

1.1 Related Work

Usage of IP based online e-meetings have grown over the last 15 years and the first usable applications, such as VIC [10] and VAT [8] were all part of the MBone [2] suite of tools. The authors own work in this area lead to the mStar suite [13,12] that in turn lead to the creation of the Swedish company Marratech AB. The research around the MBone suite has led to a number of IETF and ITU standards recommendations for IP based online communications, but unfortunately neither the MBone suite nor the ITU H.323 [3] provide a full environment for supporting scalable group communications and a number of incompatible systems have been created over the years. The audio and video parts though are compatible and [5] presents one solution for merging these two conferencing domains using the Session Initiation Protocol [14].

2 Overview

This section presents three different alternatives for solving the problem of including phones into an e-meeting.

First of all, an interface component between IP networks and the phone system has to be included. This can be easily be solved by a dedicated computer with a phone interface (i.e. modem), but that only gives a single phone line and is not a production quality solution.

A more robust solution is to use a dedicated phone gateway that can be accessed via a standardized protocol, such as H.323 or the Session Initiation Protocol, SIP. The latter is a text based open protocol and thus selected for this system. A further advantage of using SIP is that several Internet phone operators exist today that sell IP telephone services based on SIP and it has been the goal of the work presented in this paper to be inter-operable with these Internet phone operators.

2.1 Key Management in E-meetings

A basic requirement for secure e-meetings is that the actual exchange of real-time media is secure and this is handled via the Secure Real-time Transport Protocol, SRTP [11] which is an extension of the Real-time Transport Protocol, RTP [15]. The shared knowledge (i.e. the key) that is used for encryption can be exchanged in a number of ways. It can be stored in the e-meeting description file, provided by the user at startup or it can be included via an external reference using a URI. The last is the method that is both secure and convenient as the key can be updated regularly[2] and the user does not need to store the key in his/her mind or noted locally. It also means that the key is not shared with the administrator of the e-meeting portal, even if it provides the session information. The remote key should of course be protected by e.g. a normal secure WWW protection scheme.

2.2 Integrating Phone Audio Media into E-meetings

The phone-gateways are normally constrained to only support one or a few audio encodings as well as not being able to mix several audio streams or even to receive more than audio stream from the IP network. In an e-meeting, audio from each participant is sent in its own media stream to both allow for peer-to-peer communication (i.e. no central unit that mixes all active audio streams) and to allow the user to mute individual audio senders. This leads to the requirement that the e-meeting to SIP gateway has to support both optional two-way transcoding of audio, mixing of audio streams, and to maintain the security it also has to handle the encryption and decryption of the audio media.

The integration can be done by the following approaches:

A *Client only:* Let the client take care of all the security coding, audio mixing and audio transcoding. The advantage is that no portal is required. The disadvantage is that any computer glitches (such as running an application that takes a lot of CPU, e.g. starting a large word processor application) will affect the audio quality, not only to the local user but also to the rest of the group if audio is actively sent between the phone and the group. Degradable audio for the local user is usually accepted as the user is the one that causes the degradation. The handling of the security information (encryption key) is all done locally.

B *Portal with shared secret:* Move all the audio mixing and audio transcoding to the portal and share the e-meeting encryption key with the portal. The advantage here is that inclusion of the phone into the e-meeting is not dependent on a client host and its CPU as mentioned in approach A. The disadvantage is that the portal has to know the shared secret and once it has the key it will be able to decode the audio even after the call is terminated.

C *Portal without shared secret:* Same as approach B but one client does the decryption, re-encrypts the audio and sends it to the portal for further handling. Here the one client and the portal exchange a common secret to be used for encrypting the media between the portal and the one client. The resulting difference between this and approach B is that the portal only gets access to the audio media, and only during the call is active.

A special case which somewhat unifies approaches B and C, is when different keys are used for each media in a session. This is fully possible, but makes it more cumbersome to set up and potentially more cumbersome for the user. Using

[2] If the key is changed it will only take effect after the client is restarted but that is an implementation issue.

different keys is not the common case in today's e-meeting usage. Also, obviously if no encryption is used in the session then there is no shared secret problem at all.

No commercial Phone SIP gateway service allows clients to connect without providing correct credentials and this has to be included in the architecture. The transport of the credentials (i.e. authentication) is handled by SIP.

These different alternatives are further discussed in conjunction with the proposed architecture in section 3 and further evaluated in section 4.

2.3 Handling of NAT and Firewalls

Network Address Translation, NAT gateways and firewalls are very common in today's Internet. If a client is behind a NAT gateway then a Portal can be used to participate in an e-meeting and the only thing required is that the NAT gateway and firewall is setup in so called "allow return" mode, meaning that if traffic has first been sent out through the gateway/firewall then traffic can come back through the gateway/firewall on the same IP port. Allow return is usually the default mode on gateways/firewalls making it very easy to setup an e-meeting.

The authorization process between the client and the e-meeting portal is a two step process. First a secure TCP connection is established using SSL over TCP (i.e. normal secure WWW interaction) and a shared secret as well as public identifier is created by the portal and shared with the client. Session network information, described using the Session Description Protocol, SDP [4] is also passed from the portal to the client including which ports the portal is listening on for each media. The client, then for each media sends a UDP packet to the corresponding port on the portal with the public identifier. This maps the port on the client side (i.e. the outgoing port on the NAT/firewall gateway) to the client, but as this can easily be compromised a challenge/response transaction is initiated based on the shared secret. If the challenge/response transaction was successful the client becomes a full member of the e-meeting. The same port mapping authorization scheme is utilized even if no NAT/firewall gateway is used.

3 Architecture

This section goes further into depth by discussing the architecture of the proposed system.

3.1 A: Client Only

The client only solution does not involve the portal in the process of including the phone into the e-meeting and all SIP messaging is handled directly by the client.

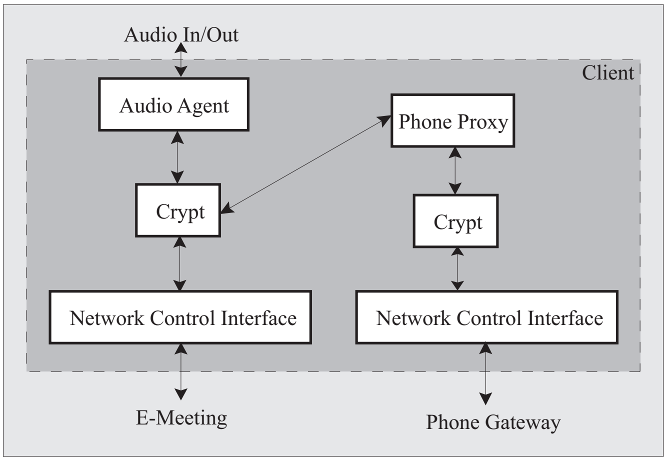


Fig. 2. Internal architecture of approach A.

Audio to and from the phone gateway is handled by a *phone proxy* software component embedded into the application and call interaction with the phone gateway is handled by the *SIP agent*, acting as a SIP User Agent Client. The process of inviting a phone into an on-going e-meeting is as follows (simplified):

1. The user opens a user interface dialog and enters a phone number (directly or via a phone book).
2. The SIP agent sends an INVITE message to the phone SIP proxy.
3. The phone SIP proxy answers with an authorization required answer.
4. The SIP agent retrieves the credentials needed (from local storage or by asking the user) and resends the INVITE message to the SIP proxy.
5. The SIP proxy then establishes a connection to the phone itself and sends an acknowledge message to the SIP agent.
6. The application then sets up the phone proxy that handles en-/decryption, transcoding and mixing of the audio.
7. Audio data can now be exchanged between the phone and the e-meeting.
8. At the end of the conversation the call is terminated by either the phone user (the SIP proxy sends a BYE message to the client) or by the local client user (the SIP agent sends a BYE message to the SIP proxy).

This process is very similar to how a normal SIP based IP telephone call is set up. Note, that if a NAT gateway or firewall (see section 2.3) is present between the client and the phone SIP proxy the call setup will not work without special configuration. Figure 2 show the internal software architecture of approach A.

3.2 B: Portal with Shared Secret

There are a number of advantages by using a portal for the inclusion of phones into the e-meeting, but it also makes the architecture more complex. The same

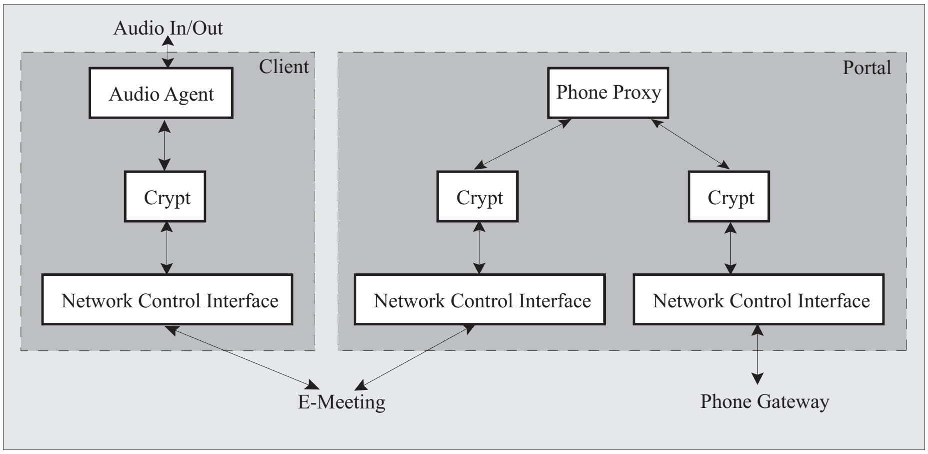


Fig. 3. Internal architecture of approach B.

elements as mentioned in section 3.1 are used, but the phone proxy is now moved into the e-meeting portal and the session setup is done via secure communication as described in section 2.3. A new element is also introduced, a *portal SIP proxy*, which is a special kind of SIP proxy as shown below. We also assume that the session key is not known to the portal. The process then becomes as follows:

1. The user opens a user interface dialog and enters a phone number (directly or via a phone book).
2. The SIP agent sends an INVITE message to the portal SIP proxy.
3. The portal SIP proxy modifies the media description included in the INVITE so the portal becomes the new data end point instead of the client.
4. The portal SIP proxy forwards the modified INVITE to the phone SIP proxy.
5. The phone SIP proxy answers with an authorization required answer, which the portal SIP proxy either forwards back to the client or uses credentials stored locally.
6. In the former case the SIP agent retrieves the credentials needed (from local storage or by asking the user) and resends the INVITE message to the portal SIP proxy which then forwards the INVITE to the phone SIP proxy.
7. The phone SIP proxy then establishes a connection to the phone itself and sends an acknowledge message to the portal SIP proxy.
8. The portal SIP proxy then sends back the acknowledgment to the SIP agent together with a request for encryption credentials for the e-meeting.
9. The SIP agent encrypts the session credentials with the shared secret, that was earlier exchanged via SSL (see section 2.3), and sends that to the portal using a SIP extension message[3].

[3] This extension message is currently not standardized and should be seen as application specific in this context.

10. The portal then sets up the phone proxy that handles en-/decryption, transcoding and mixing of the audio.
11. Audio data can now be exchanged between the phone and the e-meeting.
12. At the end of the conversation the call is terminated by either the phone user (the phone SIP proxy sends a BYE message to the portal SIP proxy) or by the local client user (the SIP agent sends a BYE message to the portal SIP proxy which forwards it to the phone SIP proxy.

This method allows for media handling in the portal independent of the end client application, but it still means that the portal need to get a copy of the shared key used within the session. Figure 3 show the internal software architecture of approach B.

3.3 C: Portal without Shared Secret

The algorithm described in the previous section can be further enhanced where the session key does not have to be shared with the portal, but instead the inviting client re-encrypts the audio data.

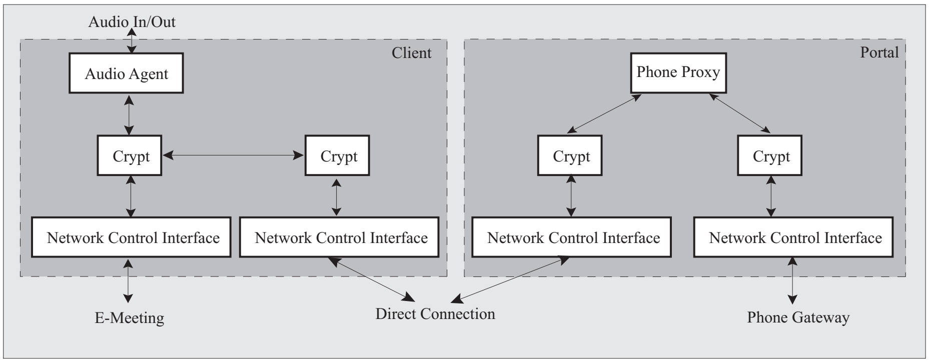


Fig. 4. Internal architecture of approach C.

Steps 1– 8 are the same as in the previous section, and it continues as:

9. The SIP agent sends a new SIP INVITE to the portal SIP proxy requesting a point-to-point session. The encryption key used for this separate session is the same as was earlier exchanged in the initial setup between the client and the portal.
10. The portal then sets up the phone proxy that handles en-/decryption, transcoding and mixing of the audio but handles audio data directly to and from the client only.
11. Termination is handled as before.

The advantage here is that the audio data will only be available to the portal while the phone call is active. As soon as it is terminated the portal will not receive any more audio data. Figure 4 show the internal software architecture of approach C.

3.4 User Membership

A very important security related issue is how to handle the situation where the inviting user wants to leave the session. In scenario A and C this will obviously force the phone user to leave the meeting (as the traffic is passing through the user software), but in alternative B no such strict need exists. In the realization behind the architecture, it is left up to the leaving user to decide, but the default value is not to let the phone user stay behind in the meeting.

4 Performance Evaluation

The various alternatives introduce different amounts of delay and bandwidth utilization. These amounts vary depending on the audio encoding used and as reference can be assumed that PCM encoded audio is used for communication with the phone gateway. This of the most common non-proprietary audio encoding and has a data rate of 64 Kbps which translates to effective network 71 Kbps (if 40 ms frames are used). A common proprietary audio codec with good robustness and high audio quality is the GIPS iPCM-wb codec with a variable bit rate of an average 80 Kbps (89 Kbps effective). Note that all modern networked audio applications utilize silence suppression (or user controlled "click to talk"), and audio data is only sent when there actually is something to send.

4.1 Bandwidth Utilization

When a phone is included into an e-meeting, data has at least to be sent to and from the phone gateway, and as all traffic to the phone is mixed before transmission the network utilization will be 71 Kbps independent of the amount of simultaneous active audio senders.

In cases A and B no extra bandwidth is used, but in case C an extra $(n+1)*X$ Kbps, where n is the number of active audio senders and X is the bandwidth of the audio encoding used, will be utilized between the client and the portal.

4.2 Delay

As with all synchronous communication tools delay is very important. In the architecture presented in this paper the only component that generates any significant extra delay is the audio mixer due to algorithmic delay in the audio codec.

Using a Pentium4 2.8 GHz PC an extra delay of audio sent to the phone gateway is a minimum 44 ms of which 2*20 ms is the decoding and encoding delay in the audio codec. The other 4 ms are general application handling and encryption/decryption. Note that the optional transcoding between different audio types is included in the mixing process as all audio streams, independent of type are handled in linear 16 bit encoding. On top of the 44 ms is a dynamic jitter buffer that can vary from 0 ms to 100 ms depending on packet-loss and jitter between the arrival times of the packets in the audio stream. The delay introduced by the phone gateway is not included here as it depends which type of hardware the gateway provider is using, but it is typically 20 ms is each direction as packets has to be stored and forwarded (unless the gateway is doing direct bit forward to the phone system in which case the delay can be neglected). Data from the phone gateway is only re-encoded if the default audio encoding of the session has a lower bandwidth than the one used with the phone gateway. In most cases no re-encoding will be necessary.

On top of the audio coding delay there is also a network propagation delay that obviously varies a lot between different network setups. In the evaluation presented here the network propagation delay to the phone commercial gateway from the client or the portal was measured to 22ms.

All in all, the extra delay introduced by including a phone into an e-meeting is low enough to allow users to use it for real meetings. An interesting observation is that several phone users have stated that they get better perceived audio quality when talking with users in e-meetings than if they talk with another phone user. The rationale for this is that the audio equipment on a modern computer is of higher quality than that in a normal phone.

5 Discussion

This paper presents an architecture for secure inclusion of phones into online e-meetings including three different approaches for handling coding and key management. In approach A everything is handled by the end client and the session key information does not have to leave the client. In approach B the session key is shared with a portal that does the media handling. If it is not acceptable to share the media key with the portal then approach C shows how re-encryption of the audio media is done at the client and then all audio streams are forwarded to the portal, during the duration of the phone call.

The interaction with the phone gateway is done using the Session Initiation Protocol and the paper shows how the portal can handle SIP invitations using credentials only stored at the client, while handling the actual audio data at the portal.

The reason for including a portal into the architecture is that running all the audio transcoding, mixing and decryption/encryption is dependent on what the client host is doing. I.e. if the user starts a larger program that momentarily takes a lot of CPU, then that will create glitches in the audio for the phone connected user and/or e-meeting members listening to the phone user.

In the commercial world everything is not always as perfect as it is in theory or in an academic setting. Most[4] generally available commercial SIP based phone gateways today do not support encrypted media transport, leaving a big hole in an otherwise secure online e-meeting. This leaves the option of operating a phone gateway of your own or accepting that the mixed audio data is actually transmitted un-encrypted over the IP network between the audio proxy and the phone gateway.

An evaluation of the delay was presented and user tests have shown that the delay is low enough to make the system usable and some users perceived the resulting audio quality better than using only phones.

5.1 Implementation

All three approaches presented in this paper have been implemented into the Marratech Pro and Marratech E-Meeting Portal solution. These products are mainly developed using the Java language with about 10% of the Marratech Pro application code developed in C/C++ due to performance issues (e.g. audio and video coding).

The audio coding component is produced by Global IP Sound where the wide-band GIPS iPCM-wb codec is the most commonly used for e-meetings. Note that this codec is GIPS proprietary.

The SIP User Agent in the client and the SIP proxy in the portal are based on the JainSIP1.1 [6] reference implementation provided by the National Institute of Standards and Technology, NIST.

The prototype presented here has been verified and used on several different operating systems including Microsoft Windows (Windows 98 and up), Apple MacOSX, Linux and Solaris.

5.2 Future Work

In the current architecture phone users cannot call into an e-meeting because of the problems of choosing which e-meeting to dial into as well as the problem of providing credentials to enter a secure e-meeting. The approach we are working on right now is based on the user using either her phone via WAP [9] or a Java J2ME MIDP Midlet [7], or via a WWW page using a separate computer[5] for choosing which session to enter and providing credentials to the portal to be authorized to enter the meeting. The portal then sends back a pin-code (a password) and a phone number to call. The latter is the number to an incoming phone-gateway registered to forward all calls to the portal. The user calls the number and enters the pin-code provided earlier by the portal. An alternative, depending on the setup is that the portal makes a call directly to the phone via a

[4] All that we have found actually.

[5] Obviously, it would be better to use the local computer for the communication, but it might lack the right computer audio equipment and it might be used in conjunction with the phone.

phone gateway. This introduces the question of who will pay for the call and one further alternative is that the phone users provides phone gateway credentials to the portal (when registering initially or for each call) and that phone gateway account is used for calling the phone users. The latter might also be attractive to do depending on different call rates in different directions (i.e. it might be cheaper to call *to* the phone than *from* the phone). Storing the credentials or forwarding them to the portal might be seen as non attractive if the user does not trust the portal administrators.

Acknowledgments. The author would like to thank Hans Öhman, Marratech AB for helping with the architecture design and Per Danvind, Marratech AB with helping with implementing the audio proxy and the anonymous reviewers for comments, suggestions, and feedback

This work is supported by the Centre for Distance-spanning Technology (CDT) under the VITAL project, Marratech AB, and the Swedish Research Institute for Information Technology.

References

1. S. E. Deering. *Multicast Routing in a Datagram Internetwork.* PhD thesis, Stanford University, 1991.
2. H. Eriksson. Mbone: The multicast backbone. *Communications of the ACM*, 8:54–60, 1994.
3. Packet-base Multimedia Communication Systems, September 1999. ITU-T Recommendation H.323.
4. M. Handley and V. Jacobson. SDP: Session Description Protocol, April 1998. RFC2327.
5. Jiann-Min Ho, Jia-Cheng Hu, and Peter Steenkiste. A Conference Gateway Supporting Interoperability between SIP and H.323. In *Proceedings of ACM Multimedia'2001*, pages 421–430, October 2001.
6. Sun Microsystems, Inc. JAIN SIP API Specification. JSR-000032.
7. Sun Microsystems, Inc. Mobile Information Device Profile Final Specification 2.0. JSR-000118.
8. V. Jacobson and S. McCanne. vat - LBNL Audio Conferencing Tool. URL[6].
9. Wireless Application Protocol Forum Ltd. Wireless Application Protocol Architecture Specification, July 2001. WAP-210-WAPArch-20010712-a.
10. S. McCanne and V. Jacobson. vic: A flexible framework for packet video. In *Proceedings of ACM Multimedia*, 1995.
11. Baugher, McGrew, Oran, Blom, Carrara, Naslund, Norrman. The secure real-time transport protocol, June 2002. draft-ietf-avt-srtp-05.txt, IETF work in progress.
12. Peter Parnes. *The mStar Environment - Scalable Distributed Teamwork using IP Multicast.* Luleå University of Technology, December 1999. Doctoral Thesis, ISSN 1402-1544 / ISRN LTU-DT–99/31–SE / NR 1999:31.
13. Peter Parnes, Kåre Synnes, and Dick Schefström. mStar: Enabling Collaborative Applications on the Internet. *Journal of IEEE Internet Computing*, September/October 2000.

[6] <URL:http://www-nrg.ee.lbl.gov/vat/>

14. J. Rosenberg, H. Schulzrinne, G. Camarillo, A. Johnston, J. Peterson, R. Sparks, M. Handley, and E. Schooler. SIP: Session initiation protocol, June 2002. IETF RFC3261.
15. H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson. RTP: A transport protocol for real-time applications, 1996. IETF RFC1889.

Evaluating the Accuracy of Active Measurement of Delay and Loss in Packet Networks

John A. Schormans[1] and Tijana Timotijevic[2]

[1] Department of Electronic Engineering
Queen Mary, University of London, London E1 4NS, UK,
[2] Technology Development - Network Systems, Vodafone UK.
John.schormans@elec.qmul.ac.uk ,
Tijana.Timotijevic@VF.Vodafone.co.uk

Abstract. Performance measurement of packet (e.g. IP) networks is a vital element in the commercial viability of broadband. Active measurement by injection of probing packets will be very widely used as a method of performance measurement. In this paper we quantify the error when using probing. We discover that, when measuring the mean packet delay across a WAN through a representatively loaded (i.e. 50%-90% utilised) access link, the measurements often have an error of many tens, hundreds or even thousands of milliseconds. Furthermore, we apply results from queueing analysis to show that probing for packet loss will require that the probes be about the same size as the data packets; if small packets are used the measured packet loss probability will be many orders of magnitude smaller than it actually is. When this is accounted for, i.e. by using probing packets the same size as the data packets, then, for constant probing load (i.e. a smaller number of larger packets), the error in the returned delay measurements becomes considerably worse.

1 Introduction

IP and MPLS packet networks are now carrying a heterogeneous mix of traffic, with widely differing Quality of Service (QoS) requirements. The service model of emerging multiservice packet networks, including the packet backbones for 2.5G and 3G mobile networks, is based on the network's ability to guarantee QoS to user applications. In a commercially competitive environment there are two main reasons for performance measurement: a) Traffic Engineering will rely on the accuracy of the performance monitoring capabilities and b) Network Operators and Service Providers must monitor performance to maintain their Service Level Agreements (SLAs) with customers, [1, 2].

Active probing is a technology whereby 'probing' packets are injected into the network, and travel from source to destination along with the packets containing the user information, allowing the delays across the network to be measured. In order to measure in a way that is representative of the users experience during periods when most users are active (i.e. busy hours) we adopt, in this paper, the recent recommendation from Cisco Systems [3] that measurements are carried out over busy hours. In conjunction with the probing rate this gives us the number of measurement probes. We report results using 2 sizes of probe packet: small (100bytes), in order to reduce the measurement load while probing delay, and large packet probes to ensure that the monitoring probes experience the same loss probability as the data traffic being monitored. For a fuller justification of using representatively 'large' packets, see Section 4.

IP networks are evolving in a manner that is essentially heterogeneous – there is no single global network covering the world, rather an interconnected collection of different networks with different owners, and this is itself acting to encourage the use of active network probing. In this paper we address specifically the accuracy that should be expected from using active measurements. Why is this an issue? Any measured probe delay is an experiment whereby the network 'path' is tested. In order to know even the mean end to end delay many probes must be used, and the related questions are: 1) how many probes are needed for a desired level of accuracy, and 2) given N probes, what accuracy should be expected (and is it good enough?).

It is known from sampling theory that the greater the variability of the sampled data the more samples are needed for accurate estimation. In the case of actively probing packet networks the variability is dependent on two main factors: the load on the network, and the type of traffic being carried. Highly bursty packet network traffic results in very large variances associated with the number of packets in queues, and hence the packet delays. This is critically important, as using packet networks to carry this sort of traffic implies that a very large number of probes may be used to achieve the required levels of accuracy.

Previous work in assessing the effectiveness of active probing is limited, in [4, 2] to the technological requirements, and in [5] to issues associated with the types of sampling that are available (random, stratified, deterministic etc) and their particular advantages and disadvantages. A study is reported, [6], which addresses the question of whether probing is accurately measuring packet delays. However, this was done by simulation, so the results are necessarily confined to simulations of small numbers of traffic sources in order that the simulations reach steady state. However where there are clear conclusions they tend to support ours.

So this paper is arranged as follows. In Section 2 we combine sampling and queueing theory to find the solution for the variance of the queue depth in a typical buffer at an access link to a WAN. Once the variance is known the number of sampling (probe) packets can be calculated for any utilisation of the access link. In Section 3 we apply these solutions to typical traffic multiplexes at typical access link loads (which will probably be about 70%→80% during peak hours). We use Section 4 to justify why we are concerned to test both short probing packets and longer ones, i.e. probes that are about the same size as the informational packets whose performance they are measuring. In the final section we conclude, attempting to draw together the most significant results for the practice of packet traffic measurement.

2 Analysis for Generic Queueing Scenarios

IP networks are evolving in a manner that is essentially heterogeneous, which is acting to encourage the use of active probing. And network heterogeneity is matched by traffic heterogeneity: the growth in user applications from VoIP and picture messaging to file transfer – both real time and non-real time – all cause different patterns of traffic to appear in packet networks. However, despite this apparent complexity queueing theory has shown that there is a generic envelope of queueing which has been found to be ubiquitous: that is packet scale and burst scale queuing [7, 8, 9] (at this point we ignore the issues specifically associated with self-similar traffic that strays from this envelope, touching on it in section 2.4).

2.1 Markovian Traffic Buffering: Combining Packet and Burst Scale Queueing

Packet delay is caused by waiting behind other packets in buffers, and packet loss occurs when an arriving packet finds a full buffer. So our focus in this paper is to use mathematical analysis to find the required number of samples (probes) in scenarios in which we know, or can calculate, the probabilities associated with a particular number of packets in a buffer. The known scenario we use is Markovian queueing, i.e. combined packet scale and burst scale queueing [7, 9, 8], which is shown diagrammatically in Figure 1.

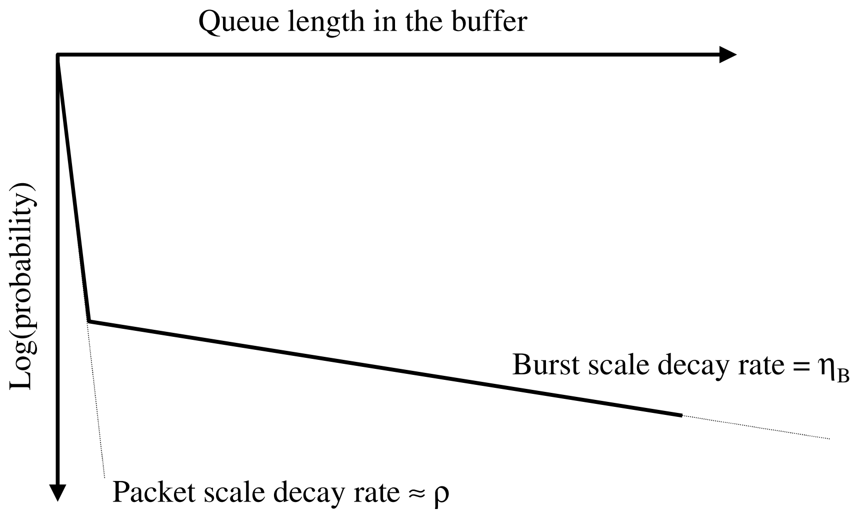

Queue length in the buffer
Log(probability)
Burst scale decay rate = η_B
Packet scale decay rate ≈ ρ

Fig. 1. Schematic representation of packet scale and burst scale queueing

In real networks probing will measure delays directly. However, as delays are the result of queueing behind other packets, we must evaluate the number of probes required to estimate with sufficient accuracy the mean of the number of packets in a buffer when a probe arrives.

The previously unexplored aspect of active measurement is whether even the mean of these distributions can be estimated accurately given that they may have very high variance. To quantify this problem in terms of probing accuracy we need an expression for the variance of the queue length in the presence of typical packet network traffic. In section 2.2 we derive the expression we need. In section 2.3 we discuss what is likely to be the result of going beyond probing the mean of the delay to probing the jitter. In section 2.4 we discuss the potential problems when the traffic is not Markovian but self-similar.

2.2 Variance of the Measured Buffer Queue Level, Markovian Traffic

Combined packet scale and burst scale queueing is a function as follows, [7, 9, 8]:

$$p(k) = (1 - P_{BS}).P_P(k) + P_{BS}.p_B(k) . \quad (1)$$

where:

$p_P(k)$ = Prob(arriving packet encounters queue length of 'k' packets conditional on packet scale queueing)
$p_B(k)$ = Prob(arriving packet encounters queue length of 'k' packets conditional on burst scale queueing)
P_{BS} = Prob(the buffer is experiencing burst scale queueing)

This can be expressed as:

$$p(k) = (1 - P_{BS}).(1 - \eta_P).\eta_P^{\,k} + P_{BS}.(1 - \eta_B).\eta_B^{\,k} . \quad (2)$$

where :

ρ = the load on the buffer
η_B = the decay rate in the burst scale part of the queueing distribution;
η_P = decay rate in packet scale queueing part of the distribution
($\eta_P \approx \rho$, see also [8, 9] for more accurate expressions)

Define:

N = the number of probe packets (measurements) available
$t_{N-1,\,1-\alpha/2}$ = the Student t-distribution value for N-1 degrees of freedom
(i.e. sample size = N), and (100-α)% confidence interval for the estimate of
the mean queue size (L)
δ = the standard error in the measurements
L_N = mean number of packets in a queue delaying an arriving probing packet, estimated from N measurements
S_N = standard deviation of the measurements
S^2_N = variance of the measurements

In order to establish a relationship between the measurement accuracy and the number of probes (samples) needed for that accuracy, we will use the Student t-distribution. This will allow us to relate the standard deviation of the queue length, the accuracy of the measurement and the number of probes. Then, once we establish the relationship between the measurement accuracy, number of probes and the standard deviation, we can use the results from queueing theory to find the queue length variance (and hence standard deviation) for specific queuing scenarios. Queueing scenarios are described by both scheduling discipline and a specific traffic arrival pattern. We therefore use specific traffic arrival characteristics to find the queue length variance in these well known scenarios, and therefore the number of probes (i.e. measurement load) needed to estimate the variance with the desired accuracy.

For the number of probes needed to estimate the mean L_N, the 100(1-α)% confidence interval for the true mean is given by:

$$L_N \pm t_{N-1,1-\frac{\alpha}{2}} \cdot \frac{S_N}{\sqrt{N}} . \tag{3}$$

For the sampled mean L_N to be within the error $\pm\delta$ of the true mean, N must be such that:

$$t_{N-1,1-\frac{\alpha}{2}} \cdot \frac{S_N}{\sqrt{N}} \leq \delta . \tag{4}$$

Equations (3) and (4) follow from sampling theory. Now we must use queueing theory to find a formula for the variance in the measured queue length, from which we can establish the number of probes needed per hour for a given level of accuracy (or the accuracy achieved from a given number of probes per hour).

As noted already, we have to imitate the packet traffic in a realistic way, and this means Markovian models with a packet scale and burst scale queueing envelope (see figure 1). In this case we need equation (4), but must account for the fact that the unconditional mean is the weighted sum of both the packet scale and the burst scale:

L = mean queue length for combined effect of packet and burst scale queueing

$$L = (1 - P_{BS}) . L_P + P_{BS} . L_B . \tag{5}$$

In (5) L_P and L_B are the mean number of packets delaying an arriving packet conditional on packet scale queueing and on burst scale queueing respectively. We now need to move from an expression for the mean to an expression for the variance, and thereby the standard deviation so that we can apply the results from sampling theory.

If 'q' is the random variable representing the size of the packet queue delaying the packet or probe:

$$S^2_N(q) = E(q^2) - (E(q))^2 . \tag{6}$$

$$E(q) = L = (1 - P_{BS}).L_P + P_{BS}.L_B . \tag{7}$$

From (8) and (9), and the basic definitions of Moments, we find:

$$\begin{aligned} S_N^{\,2} = {} & (1-P_{BS}).L_P^{\,2}/\eta_P + P_{BS}.L_B^{\,2}/\eta_B + (1-P_{BS}).P_{BS}.L_P^{\,2} \\ & + (1-P_{BS}).P_{BS}.L_B^{\,2} - 2(1-P_{BS}).P_{BS}.L_P.L_B . \end{aligned} \tag{8}$$

In equation (8) we need to find P_{BS}, L_P, L_B, η_B and η_P in order to find the standard deviation, S_N. The mean number of packets in the queue conditional on burst scale queuing is [9]:

$$\text{E[number of packets | burst scale queueing]} = L_B = \eta_B / (1 - \eta_B) . \tag{9}$$

Convenient expressions for η_B, and P_{BS} are (given homogenous On-Off multiplexing), [7, 9]:

$$\eta_B = \exp[-(N_O / b) . (1 - \rho)^3 / (4\rho + 1)] . \tag{10}$$

where:
b = average number of packets in an ON period of an individual source
N_0 = number of active sources needed to have burst scale queueing, [9, 7].
P_{BS} can be found as:

$$P_{BS} = \frac{\exp[-(\rho.N_O)] . (\rho . N_O)^{\lfloor No \rfloor}}{(1 - \rho)^2 . N_O .(\lfloor N_O \rfloor !)} . \tag{11}$$

Combining equations (8), (9), (10) and (11) we can find the variance and standard deviation of the number of packets in the queue, and substituting this into (4) we can find how this related to the required number of probes for the desired accuracy. We concentrate on the effect of the packet queue in the access buffer, as this is where the bulk of the packet queueing will take place.

2.3 Error When Measuring the Prob(delay >T) for Delay Variation (Jitter)

Prior work, e.g. [10], has shown the importance of keeping the probing overhead as low as possible. [11] shows that to measure the delay jitter will probably require orders of magnitude more probes per unit time than does measuring the mean. Therefore attempts to get delay jitter estimates from the number of probes assumed here implies errors considerably greater than would be experienced when measuring the mean delay.

2.4 Measuring Networks Carrying Self-Similar Traffic

Recent literature indicates that the presence of self-similar traffic in packet networks is now well appreciated by most of the networking community. Results, both from simulation and analysis, clearly show that where the input traffic has active periods that are power law distributed, the overall traffic 'pattern' is likely to be self-similar, and therefore the distributions associated with packet queueing are likely to be power law distributed too. Power law distributions may feature infinite variance, and this would cast doubt on the ability of active traffic measuring technologies ever to determine mean values within a desired level of accuracy. However, further work is required here, as results recently reported on passive queue monitoring have shown some promise in resolving queue state probabilities, albeit without the analytical support specifically aimed at guaranteeing the statistical accuracy of the results [12].

3 Results – Application to Realistic Network Examples

Probing across a WAN involves passing the measurement packets through a series of buffers in routers end to end. However, it is now well understood that the buffers/routers within the WAN core will experience a very low utilisation compared to the network access lines, while the access lines will be highly utilised during busy hours, as they represent the bottlenecks. For this reason we concentrate our attention on these bottleneck access lines.

3.1 Buffering VoIP Traffic

In order to obtain some practical estimate of the accuracy we can expect when using probing, we predict measurement accuracy for 2 standard traffic models, against increasing load. The 1st model we adopt is the standard VoIP model (exponential On and Off periods, mean on time = 0.96 seconds, mean off time = 1.69 seconds), which is very well accepted and widely used. This gives the following parameter values:

- b = 80 pkts per sec * 0.96 (average ON period in seconds, or activity factor) = 76.8
- N_0 = capacity in pkts per sec / 80 pkts per sec
- packets of length 100 bytes.

Figure 2 shows the value of the error, δ, on the measurements for 128kbps access lines. This shows that at least 512kbps, or 2Mbps should be used. It can be seen that, for probe packet injection rates corresponding to the use of 1%, 2% and 5% of the 128kbit/sec access channel capacity (i.e. non-trivial bandwidth available for measurements) the error gets very large as the load increases.

Naturally, given constant measurement bandwidth, i.e. 10 times fewer probes when 10 times larger probes are being used, this effect is much more marked when the 1000 byte probes are used. Figure 3 repeats this example for 34Mbit/sec access lines, and here significant error is apparent at high loads and with the larger packet size.

3.2 Traffic More ‘Bursty’ than VoIP

In this section we repeat the previous studies, but this time replace the VoIP traffic model with the burstier ‘data’ model, i.e. burstier than VoIP. To obtain such a model we use the following parameters:

- average source active (On) period = 0.1 seconds
- packet generation rate when in the On periods = 1250 packets /second
- b = 1250 packets / second * 0.1 = 125
- N_0 = Capacity (in packets per second) ÷ b
- Data packets = 1000 bytes in length

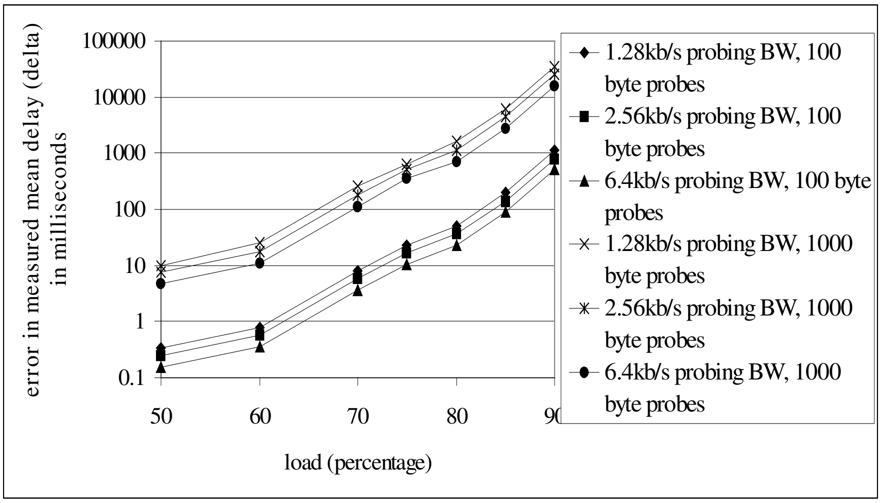


Fig. 2. Error (δ) in the probing measurements for VoIP access scenarios of 128kbit/sec with 100 byte and 1000 byte probing packets

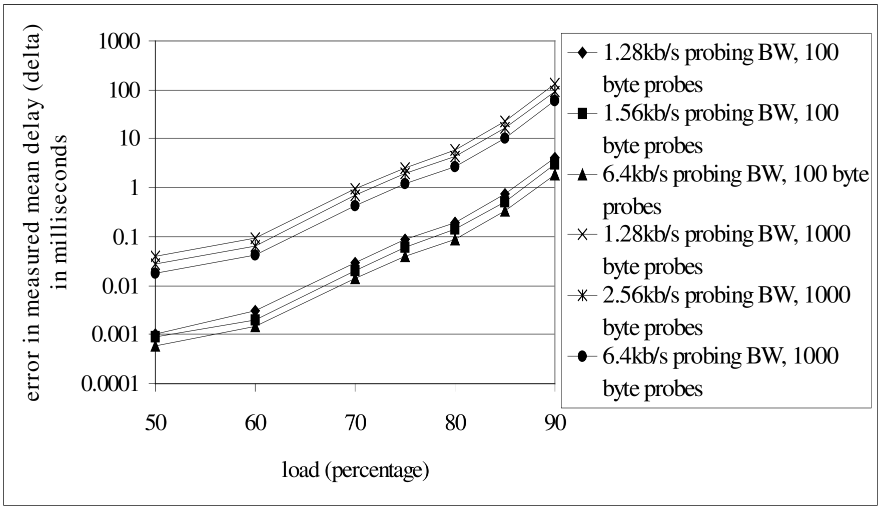


Fig. 3. Error (δ) in the probing measurements for VoIP access scenarios of 34Mbit/sec with 100 byte and 1000 byte probing packets

It can be seen that, for both cases we used (i.e. 128kbps and 34Mb/s access line rate), see Figures 4&5, the absolute error (δ) in the measurements is larger for the data packets than for the VoIP at all utilisations. This is intuitive: the burstier the traffic the more variable the queueing/delay distributions and hence the more probes needed to accurately estimate the mean delay.

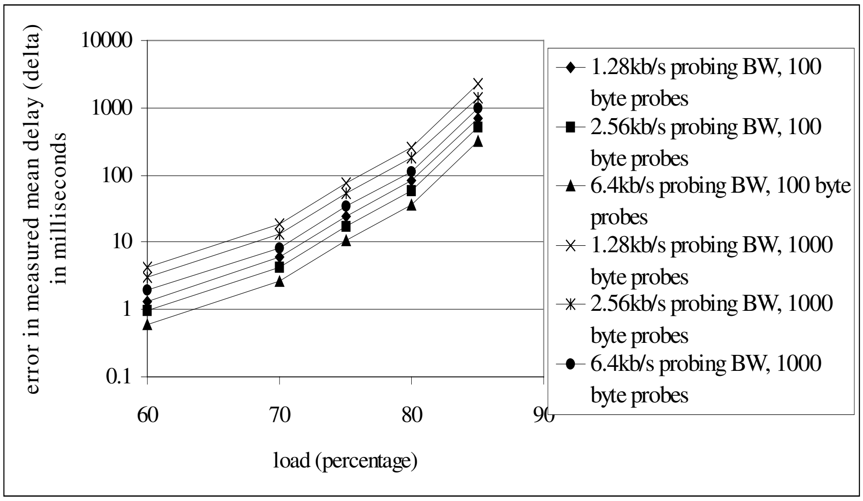


Fig. 4. Error (δ) in the probing measurements for generic 'data' access scenarios of 128kbit/sec with 100 byte and 1000 byte probing packets

It is important here to note that the probing bandwidths we have used are actually quite large: 1%, 2% and 5% of 128kbit/sec in all cases. Many organisations are using probing rates equivalent to 1 packet per second, and it is clear that at such low rates measurement accuracy will be significantly worse than predicted here.

3.3 Example Using a 3rd Party Video Trace

In this example we do not use a traffic model but instead use the results of a previous experiment [13] with a real traffic trace of a video sequence (based on the MPEG Star Wars video trace). The parameter values were: channel access link capacity = 17Mbps, video packet size = 424 bytes, and the (measured) utilisation ≈ 0.8. From these results: $\eta = 0.99924$, and $P_{BS} = 0.01$, and therefore the unconditional variance of the queue length = 34359. (It should be noted that this figure for unconditional variance in queue length is broadly in line with the values generated by the traffic models we used in sections 3.1 and 3.2.) The error when using probing packets is given in Table 1:

Table 1. Error [millisecs] in the measurements (δ) of mean delay (video trace example)

Probe size =	**100 byte**		**1000 byte**	
Access link rate =	***128kbps***	***34Mbps***	***128kbps***	***34Mbps***
Probe rate = 1.28 kb/s	δ=30 [ms]	δ=0.1 [ms]	δ=305 [ms]	δ=1.1 [ms]
Probe rate = 2.56 kb/s	δ=22 [ms]	δ=0.08 [ms]	δ=216 [ms]	δ=0.8 [ms]
Probe rate = 6.4kb/s	δ=14 [ms]	δ=0.05 [ms]	δ=136 [ms]	δ=0.5 [ms]

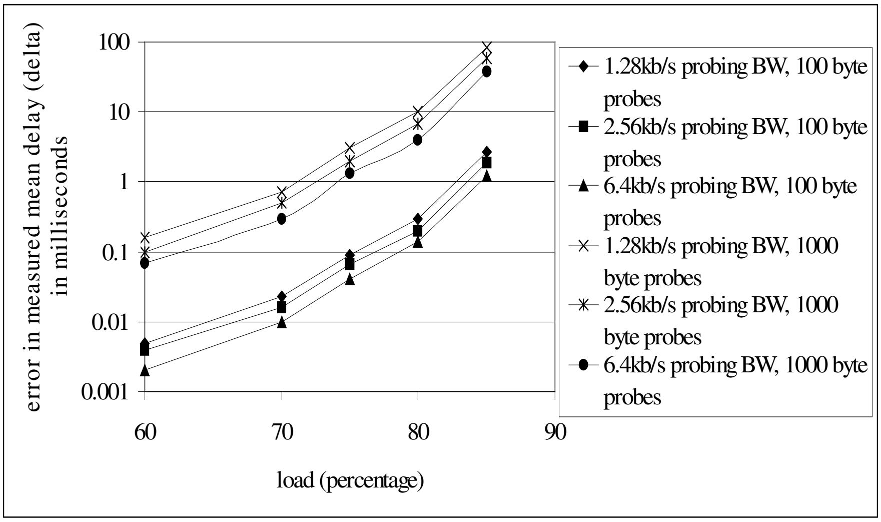


Fig. 5. Error (δ) in the probing measurements for generic 'data' access scenarios of 34Mbit/sec with 100 byte and 1000 byte probing packets

4 Accuracy When Probing Loss

It is important for the effectiveness of network measurement that the number of probes, and hence bandwidth overhead, is kept to a minimum. To do this the stream of probes that are being used to measure delay should also be used to measure loss probability. However this will require the use of probe packets that are of equivalent length to the data packets for which the packet loss probability is to be quantified. To see that this is the case consider loss resulting from overflow in a buffer which is multiplexing large packets, of L bytes, and small packets of S bytes. Perhaps the large packets represent TCP data while the small are TCP ACK packets (they could also be VoIP, however for QoS reasons it is unlikely that such dissimilar services would pass through the same buffers). The total capacity (in bytes) of the buffer is X, where $X = N_L.L = N_S.S$, i.e. N_S , N_L are the number of (respectively) small and large packets that the buffer can hold in the absence of the other type. Clearly once the buffer contains at minimum $((N_L - 1)*L + S)$ bytes it is 'full' as far as the large packets are concerned, and further arrivals of large packets will be lost. However the small packets can still be accepted: these will not be lost until the buffer contains the equivalent in bytes of N_S small packets, i.e. N_SS bytes. Therefore if the probes are small packets they will not necessarily be lost when large packets are.

So it is easy to see that smaller probe packets cannot accurately be used to measure the loss probability for larger packets. While it is easy to see this intuitively, it is important to discover how significant this inaccuracy will be. To do this, we use the method given in [14], as this allows us to evaluate the ratio:

$$R = \frac{\text{[measured packet loss probability found by using small probes]}}{\text{[Actual packet loss probability]}}$$

We use the results of an analysis of multiplexing large and small packets through a classical M/G/1 queue model with 2 packet sizes, small and large. In order to perform these calculations we have had to make an assumption about the relationship between X, L and S, so for simplicity, and in the absence of further information, we have assumed that the extra space, on average, available to the small packets is = $\lfloor L/2 \rfloor$, which is ½ of the length of a large packet.

Table 2. The value of the error ratio 'R' for varying proportions of small packets, with small packets also used as probes

Total load (%)	10% small packets	20% small packets	50% small packets
50	R = 6.00E-10	R = 9.22E-09	R = 3.20E-07
60	R = 1.30E-09	R = 1.87E-08	R = 7.10E-07
70	R = 2.20E-09	R = 3.52E-08	R = 1.23E-06
80	R = 3.90E-09	R = 5.90E-08	R = 2.09E-06
90	R = 6.30E-09	R = 9.60E-08	R = 3.35E-06

These results for the ratio 'R' are given in Table 2. Percentage "small packets" refers to the TOTAL traffic carried by small packets. These results show that the measured packet loss probability, in this standard buffer model, will be between 10^{-6} and 10^{-10} SMALLER than the actual packet loss probability for large packets. Clearly measuring packet loss probabilities as being smaller by such large amounts will be a very significant problem. This is the justification for presenting results from Section 3 that use 1000 byte probing packets as well as 100 byte packets.

5 Conclusions

In this paper we have shown that the standard error in the measurements of mean packet delay, for traffic with VoIP type characteristics, or similar bursty data, will be significant: at non-trivial but representative access link load the measurement error may reach many hundreds or even thousands of milliseconds. The effect of this may be to invalidate any SLA based on guaranteeing mean delays within reasonably tight limits.

It is important to note that the probing bandwidths we have used are actually quite large: 1%, 2% and 5% of 128kbps in all cases. Furthermore, when using packet sizes that accurately represent the data packet size, and therefore the packet loss probability of the data traffic, the measurement inaccuracy may be considerable worse (for constant measurement bandwidth).

These results pertain only to measuring the mean delay. Also of clear significance is the achievable accuracy when measuring the delay jitter and packet loss probabilities. It is clear from the work in this paper, and others [10, 11], that either a) much more bandwidth needs to be made available these for these measurements, or b) inaccuracy so severe as to invalidate them must be accepted when attempting their measurement.

In QoS oriented networks there are two more reasons for suspecting that measurement accuracy will be worse than presented here: a) queueing in the WAN will add to the effect of the bottleneck queueing and b) traffic will be divided into (probably about 4) Classes of Service, the effect of which will be to subdivide the available probing bandwidth, meaning a smaller number of measurements is actually available for each calculated average in each CoS class. If, in addition to this, SLAs are to be guaranteed on a per packet size, or per application type basis, then the problems of inaccuracy raised in this paper will become yet more severe.

References

1. Verma, D: "Supporting Service Level Agreements on IP Networks". MacMillan Technical Publishing, ISBN 1-57870-146-5.
2. Cisco Systems – Service Assurance Agent. www.cisco.com/univercd/cc/td/doc/product/software/ios120/120newft/120t/%120t5/saaoper.htm, 2000.
3. Service-Level Management: Defining and Monitoring Service Levels in the Enterprise. Cisco White Paper, 2003, available from: http://www.cisco.com/en/US/products/sw/cscowork/ps2428/products_white_paper09186a0080091ba5.shtml
4. Nevil Brownlee and Chris Loosley: "Fundamentals of Internet Measurement: a tutorial", CMG Journal of Computer Resource Management, Issue 102, 2001. Also available at www.keynote.com/solutions/assets/applets/Fundamentals_of_Internet_Measurement_A_Tutorial.pdf
5. Claffy, K.C., Polyzos, G.C., and Braun, H-W: "Application of sampling methodologies to network traffic characterisation". SIGCOMM 1993, 194–203, and also from: citeseer.nj.nec.com/claffy93application.html
6. Hill, J.: "Assessing the Accuracy of Active Probes for Determining Network Delay, Jitter and Loss". MSc Thesis in High Performance Computing, The University of Edinburgh, 2002.
7. Roberts, J. (Editor): "Performance Evaluation and Design of Multiservice Networks". COST224 Final Report, 1991.
8. Schormans, J.A. and Pitts, J.M.: "From Erlangs to Excess Rate", Journal of the IBTE Oct-Dec 2001.
9. Pitts, J.M. and Schormans, J.A.:" Introduction to IP and ATM Design and Performance", Wiley, 2000.
10. Timotijevic, T. and Schormans J.A.:" Bandwidth overhead of probe technology guaranteeing QoS in packet networks". To appear in Electronics Letters, 2003.
11. Schormans, J.A.: "Bandwidth overhead of packet probing". Queen Mary internal report, available from john.schormans@elec.qmul.ac.uk
12. Leung, C.M., Schormans, J.A. and Ma, A.H.I.: "Measurement-based queue length distribution estimation for power law traffic". Electronics Letters, 21st November, 2002, Vol. 38, No. 24, pp. 1608–1610.

13. Choe, J. and Shroff, N.B.: “Queueing Analysis of High-Speed Multiplexers including Long-Range Dependent Arrival Processes”. IEEE INFOCOM'99, New York, NY, March 1999, pp. 617–624.
14. Pitts, J.M., Schormans, J.A., Phillips, C.I. and Griffiths, J.M.: “Accurate Delay Bounds For Real-Time IP Services in Presence of Variable Packet Size”. Electronics Letters, 1st February 2001, Vol.37, No. 3, pp. 166–167.

On Using Multi-agent Systems in End to End Adaptive Monitoring

Leila Merghem [1], Dominique Gaiti [1,2], and Guy Pujolle [1]

[1] LIP6-Université de Paris 6, 8 rue du Capitaine Scott – 75015 Paris, France,
{Leila.Merghem,Guy.Pujolle}@lip6.fr
[2] LM2S: Université de Technologie de Troyes, 12 rue Marie Curie – 10010 Troyes Cedex, France.
Dominique.Gaiti@utt.fr

Abstract. The complexity (in terms of services and multimedia streaming) and dynamicity of telecommunication networks are continually growing, making network management and control more and more difficult. Such a management must be adaptive, dynamic and smart. In order to be sure of getting the best management mechanisms' choice, a monitoring operation becomes fundamental. This paper presents a new adaptive monitoring approach based on a multi-agent system, taking into consideration the features of the networks namely flexibility, dynamicity, heterogeneity of users generated traffic, etc. A two-layers monitoring architecture is proposed in this paper. The first layer is responsible for dealing with monitoring of some local parameters, which are useful for the second layer. Indeed, this latter is adaptive, autonomous and able to make new decisions- about the data to monitor and the management mechanisms to activate- based on the current node's state.

1 Introduction

Telecommunication networks represent a very dynamic and complex area, in which every day managers are facing new problems and challenges. Increases in network complexity and information volume make resources and network control more and more difficult. Users ask for more and more services and multi media applications where the network is not conceived to carry these data. We argue that in such an unpredictable, changing and open environment, intelligent agents give the opportunity to obtain an optimized network management and monitoring. In fact, the main features of agents, namely the autonomy, the ability to communicate with the others, the ability to solve common problems in a decentralized manner, in addition to learning aptitudes, allow agents to operate in telecommunication network dynamic environment.

In this paper, we will present a new adaptive monitoring approach of telecommunication networks based on agents which monitor the network state and control its different components. Our aim is to include some intelligent and dynamic control, thanks to agents, allowing us to guarantee a QoS and to give a better management and global performance of the network.

The remainder of this paper is organized as follows. We first present the features of the monitoring approach we propose. Then, we describe Multi-Agent Systems and explain why we have chosen this technique. Section 4 introduces the two monitoring levels within a network's node, followed by a detailed description of the different kinds of events and actions in the model. In section 6, we give examples of adaptive monitoring. Section 7 presents related work and section 8 concludes the paper.

2 Usefulness of Adaptive Monitoring

The monitoring we perform is:

- scalable: our monitoring approach is scalable because it is based on a multi-agent system which scales well with the growing size of the monitored network. For that, one has to integrate an agent (or a group of agents) on the new node to be controlled and the monitoring of this node is realized;
- distributed: each agent is responsible for a local monitoring. There is no centralization of the information collected by the different agents, and the decisions the agent performs are in no way based on global parameters. This feature is very important as it avoids having bottlenecks around a central monitoring entity;
- adaptive: the agent adapts its actions depending on the monitored data and entities according to the incoming events and the vision of the current system state. The monitoring we adopt is adaptive because of the following: (1) the agent modifies the monitored parameters: the agent decides, at every moment, which parameters must be monitored and which ones are no longer important under the current conditions; (2) the agent adapts the current management mechanisms (MM) and the actions undertaken when a certain event occurs. The actions the monitoring process executes may become no longer valid and must therefore be replaced by other actions. These new actions are considered more suitable to the current observed state;
- local: the agent monitors only local parameters. However, the agent can use information sent by its neighbors (from other nodes) to adapt the monitoring process;
- selective: the agent filters the received events and reacts only to those it recognizes. Event classification (see section 5) is used to trigger the appropriate actions.

3 Multi-agent Approach

A multi-agent system is composed of a set of agents which solve problems that are beyond their individual capabilities. Ferber [9] defines an agent as being an entity which: (1) can communicate directly with other agents, (2) possesses its own resources, (3) is capable of perceiving its environment (but to a limited extent), (4) has only a partial representation of its environment (and perhaps none at all), (5) has a behavior which tends towards satisfying its objectives, taking account of the resources and skills available to him and depending on its perception, its representation and the communications it receives.

Multi-agent systems have been used in numerous areas like: (1) the road traffic control ([4], [14]); (2) biologic phenomena simulation like the study of eco-systems

[7] or that of ant-colonies [8], for example; (3) social phenomena simulation like the study of consumer behaviors in a competitive market [5]; (4) industrial applications like the control of electrical power distribution systems, the negotiation of brands, etc.; (5) etc. Multi-agent approach is well suited to control distributed systems. Telecommunication networks are good examples of such distributed systems. This explains partly the considerable contribution of agent technology when introduced in this area. The aim was mainly to solve a particular problem or a set of problems in networks like: the discovery of topology in a dynamic networks by mobile agents ([13],[17]), the optimization of routing process in a constellation of satellites [19], the fault location by ant agents [20], and even the maximization of channel assignment in a cellular network [6].

Our approach consists in integrating agents in the different network nodes. These agents optimize the networks QoS parameters (delay, jitter, loss percentage of a class of traffic, etc.), by adaptively monitoring the network elements, the traffic nature and volume, and the user profile and his (her) habits. Agents can be reactive, cognitive, hybrid or adaptive [3], [7], [22]. Reactive agents are suitable for situations where we need less treatment and faster responses (actions). Cognitive agents, on the other side, allow making decisions and planning based on deliberations taking into account the knowledge of the agent about itself and the others. Adaptive agents can adapt their actions and parameters to the changing situations. Hybrid agents are composed of several concurrent layers. In INTERRAP [15], for example, three layers are present: a reactive layer, a local planning layer, and a cooperative layer. The approach we propose is different. In fact, every node has one cognitive agent that supervises, monitors, and manages a set of reactive agents. Each reactive agent has a specific functioning aiming to optimize some monitored parameters. The cognitive agent (we call it Master Agent) is responsible for the MMs' selection of the different reactive agents it monitors, regarding the current situation and the occurring events.

4 The Two Levels Monitoring Process

Two kinds of agents are defined in our model: (1) Master agent: which monitors the other agents in addition to what is happening at the level of the node; (2) the other agents: which monitor some local parameters like loss percentage, etc.

We can distinguish two levels of monitoring within a network node. These two levels are the following:

Management Mechanisms' Level (Level 0)

This level is composed of the different node's management mechanisms, which are currently activated. Each management mechanism has its own parameters, conditions and actions, which can be monitored and manipulated by the entity lying at the level 1 (the Master Agent). The functioning of a MM is limited to the execution of the loop (conditions $\rightarrow$ actions); therefore, this is represented by a method of a reactive agent. We can find different agents at this level (Scheduler Agent, Queue Manager Agent, Admission Controller Agent, Routing Agent, etc.). Each one of these agents is responsible for a specific task within the node. So each agent responds to a limited set of events and performs actions ignoring the treatments handled by other agents lying in the same node or in the neighborhood.

Adaptation Level (Level 1)
This level supervises, monitors, and manipulates the entities of level 0. A Master agent is lying at this level and is responsible for the different interactions with the other agents like the cooperation, negotiation, messages processing, etc. This agent possesses a model of its local environment (its neighbors) that helps him to take its own decisions. It chooses the actions to undertake by consulting the current system's state (neighbors nodes state, percentage of loss, percentage of the queue load, etc.) and the meta-rules it has at its disposal in order to have only the most relevant management mechanisms activated with the appropriated parameters. The node, thanks to the two monitoring levels (which are in the same time its decision levels), responds to internal events (loss percentage for a class of traffic, load percentage of a queue, etc.) and to external ones (a message sent by a neighbor node, reception of a new packet, etc.).

Actions of the Master Agent

The actions of the Master Agent adapt the node's MMs and may consist in:

- inhibiting the MM (action (3) on C in Figure 1): the inhibition happens when this MM becomes useless regarding the current node's situation and rules;
- modifying the internal functioning of the MM (action (2) on the MM B): this modification appears by updating the parameters on which the MM depends like the thresholds (MinTH and MaxTH) for the RED management mechanism or the weight of each queue for the WFQ management mechanism, for example;
- activating the MM (action (4) on the MM D): the activation takes place if the Master Agent considers that this management mechanism is appropriate to the current node conditions. This activation may be accompanied by the inhibition of other MMs to avoid the coexistence of contradictory MMs;
- letting the active behavior running (action (1) on MM A): this occurs when the MM is still relevant to the current conditions. So, the Master agent can continue to make the same actions and monitor the same set of data and parameters.

The Master Agent uses meta-rules in order to decide on actions and monitoring to perform. Each Master Agent possesses a set of rules allowing to select the appropriate management mechanisms to activate, and therefore to select the best actions to execute. These rules respond to a set of events and trigger the actions which affect the MMs supervised by that Master Agent. Their role is to manage a set of management mechanisms in order to provide the best functioning of the node and to avoid incoherent decisions within the same node. These rules give the node the means to guarantee that the set of actions executed, at every moment, by its agents are coherent, in addition to be the most relevant to the current situation. The Master Agent owns some modules and each of them is responsible for a particular task (figure 2).

The actions undertaken by the node have local consequences but may influence the decisions of the other nodes. In fact, by sending messages bringing new information on the sender node's state, a receiver's Master Agent rule may be triggered. This can

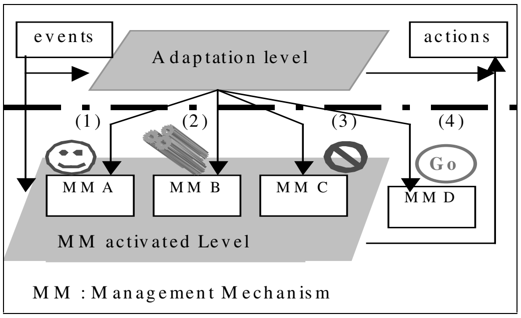


Fig. 1. Operations on management mechanisms

involve a change within the receiver node (the inhibition of an activated management mechanism, or the activation of another one, etc.). This change may have repercussions on other nodes, and so forth until the entire network be affected.

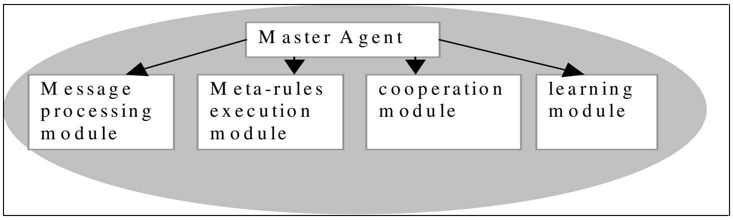


Fig. 2. Master Agent's main modules

This dynamic process aims to adapt the network to new conditions and takes advantage of the agents' abilities to alleviate the global system. We argue that these agents will achieve an optimal adaptive monitoring process because of the following two points: (1) each agent holds different processes (management mechanisms and adaptive monitoring of these mechanisms) allowing to take the most relevant decision at every moment; (2) the agents are implicitly cooperative in the sense that they possess rules that take account of the neighbors' state in the process of management mechanisms' selection and monitoring (section 5, example 1).

The Master Agent meta-rules and the dynamic they create are represented by an Augmented Transition Network (ATN) [21]. In an ATN, two concepts are fundamental: states and transitions. For us, a state represents the current activated management mechanism, while the transition represents a rule, that is, a set of events and the actions they cause. Each Meta-Behavior has its own ATN. In figure 3, we can see the queue management rule-based ATN. Three rules, and consequently three transitions, are represented. Each transition is labeled by the rule engendering it. The state PRIOR means that the activated queue MM is PRIOR with the parameters given by the rule action. The Master Agent has dedicated ATNs to represent the management mechanisms dynamic. One can argue that it is unnecessary to associate an ATN to each management task and that we can have only one ATN. In this case, a state will look like: FIFO_StaticRouting_PQScheduling. This implies an ATN with a very important size and we are not favorable for such a solution.

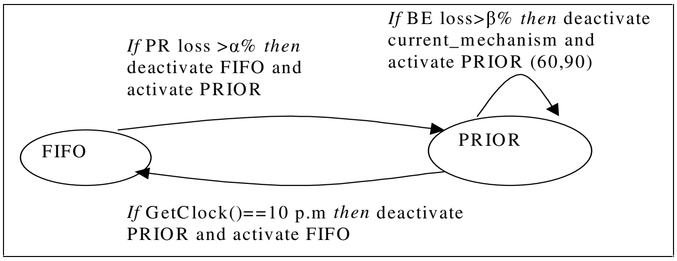

If BE loss>β% *then* deactivate current_mechanism and activate PRIOR (60,90)
If PR loss >α% *then* deactivate FIFO and activate PRIOR
FIFO
PRIOR
If GetClock()==10 p.m *then* deactivate PRIOR and activate FIFO

Fig. 3. Queue Management Meta-behavior ATN

The current ATN are provided by the system's designer but it is possible to enrich them by a learning process. This issue is beyond the paper purpose. The Meta-rules-Execution_Module is responsible for deciding when the meta-rule must be executed regarding the events it receives.

5 Events and Actions Model

Each agent holds a set of rules defining the manner of realizing the tasks. A rule is of the following form: *if Events then Actions* and its formal definition is given in figure 4. An event can be: a message, the value of a parameter or time-dependant.

```
Rule::=<Events>; <Actions>
Event::=<message>|<parameter_value>|<time_based>
Message::=<Simple_message>|<Normalized_message>
Simple_message::=<Source>; <Destination>; <Msg_Type>
Normalized_message::=<KQML_message>|<ACL_message>
parameter_value::=<attribute>;<operator>;<value>
time_based::=<agent.current_clock>;<operator>;<value>
```

Fig. 4. Rule description

Message-Based Event

This message carries information likely to be interesting to the receiver. It can be simple (and light, by the way) carrying only its type (Message M1 in Rule 1 means that a congestion has occurred or will soon occur) as it can be more developed. In that case the message is enclosing an order, an advice, etc. and is respecting a formalism like KQML [23], ACL [24], etc. A message can be broadcasted to all agents or to a given group of agents, or sent in an end-to-end manner.

Rule 1: *if the message received is of type M1 then*
add the message source to congested_nodes_list
use dynamic routing avoiding this node until it recovers from its congestion

Each Master Agent owns a message base containing the messages to which it responds. When the message arrives, it is inqueued before being removed from the messages queue by the get_msg method. The match_msg method (figure 5) has the role of deciding if the event belongs really to those events the agent deals with. In this case, the Meta_rule_Execution_Module will be notified. It is neglected and the treatment of the next event starts otherwise.

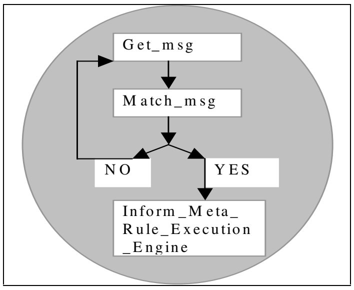


Fig. 5. Message treatment process

Parameter-Value Based Event

This event can be a single value or an interval. Therefore, the rule can look like: *if X= α then actions* (Rule 2) or *if X in [α, β] then actions* (Rule 3).

Rule 2: *if all the received packets are Best Effort (their percentage = 100%) then*
deactivate Prior queue management mechanism
activate FIFO queue management mechanism

Rule 3: *if Premium loss percentage > 0.0001% then*
deactivate FIFO queue management mechanism
activate Prior queue management mechanism with parameters (α, β)

Time-Based Event

This parameter becomes more and more important, especially because of the personalization of the users contracts allowing predicting -to some extent-, the nature and the amount of traffic within the network. Therefore, decisions may be achieved based on the current time (in addition to the date). In Rule 4, the node knows that, from 10 p.m., the clients (which have already negotiated their QoS), requiring a Premium class [16] in rush hours are no longer Premium after 10 p.m. It is consequently obvious that a simple FIFO mechanism has to be activated.

Rule 4: *if getClock() = 10 p.m. then*
deactivate Prior queue management mechanism
activate FIFO queue management mechanism

Reactive Agents Events and Actions

Events to which the reactive agents respond are of type 2: they depend on a value parameter. CAREFUL (see [10] for a more detailed description) for example, a PRIOR queue MM, depends on the current queue load parameter. As we can see in figure 6, following the current queue load, the agent state changes causing a different response to a particular event. Let consider the event we are interested in is a best effort packet arrival. If the current agent's state is *stable queue*, the action to execute will consist in putting the packet in the queue according to its priority. But if the current state is *close congestion* then the packet will be dropped. This proves that the context in which an event occurs influences a lot the system's response to this event.

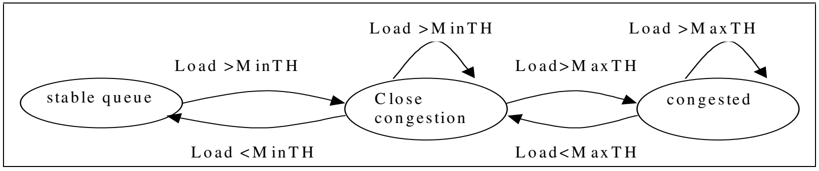


Fig. 6. PRIOR queue management mechanism

6 Examples of Adaptive Monitoring

In order to illustrate our monitoring approach, we propose some examples and the results of their simulations with the oRis Multi-Agent simulator [25].

Example 1

The management mechanism we are interested in is the queue management one with three rules (Rules 3 and 4 of the section 5, in addition to the rule 5 below). We suppose that each node has one queue and that the simulation starts with the activation of a FIFO mechanism.

Rule 5: *if Best Effort loss percentage > 35% then*
deactivate current_QueueManagement_mechanism
activate Prior queue management mechanism with parameters (60, 90)

The node, by monitoring the loss percentage of Premium packets, comes to notice that this percentage is very important (Figure 7), and because it can not tolerate such a bad performance, it will execute the action consisting in activating PRIOR (30,50) (rule 3 triggered, with (α= *30*,β=*50*). This mechanism, detailed in [10, 11, 12] and consisting in monitoring two thresholds of the queue load and changing the actions concerning the incoming packets according to the current queue load, is able to guarantee a better treatment of these packets. By adopting PRIOR mechanism, the node begins to monitor a different parameter (in addition to those already monitored). This parameter is the Best Effort packets' loss. After a certain time, the node notices that there is an important loss of Best Effort packets.

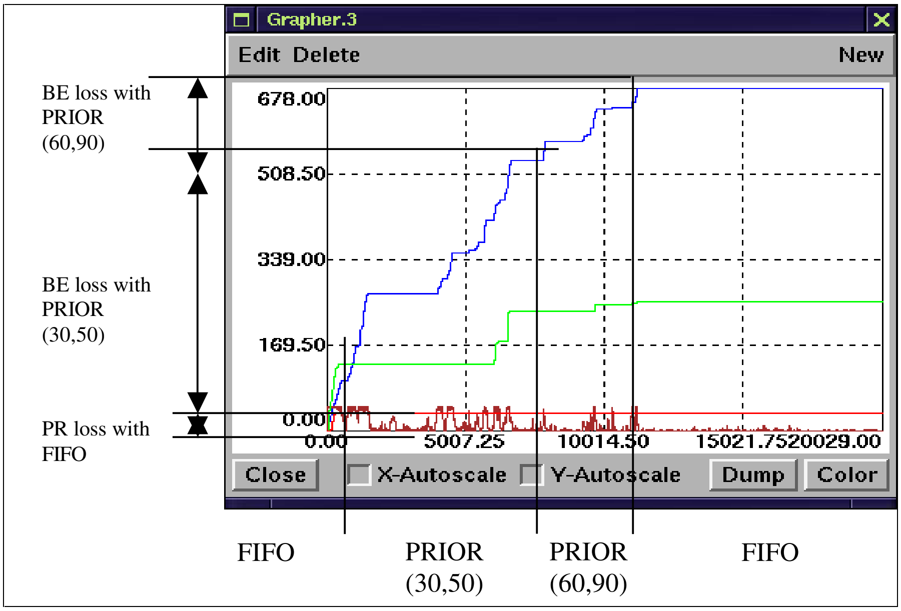


Fig. 7. Loss with an adaptive queue management

In order to respond to such an event, the node executes the action consisting in adapting the parameters of the currently activated queue management mechanism. As a result, it will use PRIOR (60,90) (Rule 5) which will permit to more Best Effort packets to be accepted in the queue. But at 10 p.m., all the packets are considered to belong to the same class of traffic: Best Effort one. Subsequently, it is unnecessary to keep the current queue MM because it is no more suitable; only one class is, from now on, present in addition to the fact that PRIOR mechanism is more complex and, consequently more time-consuming, than a simple FIFO mechanism.

One can notice that we used fixed parameters for PRIOR activation. This is done only in order to give simple and clear examples. The action may look like ”activate PRIOR with (α,β) α is randomly chosen in [0,95], and β in [3,100]” or “activate PRIOR (current_MinTH- α, current_MaxTH – β).

Example 2

In this example, we will demonstrate the gains introduced by adaptive monitoring of scheduling mechanisms. We suppose that the scheduling algorithm used at the simulation beginning is Round Robin, each class of traffic has its own queue managed by FIFO mechanism and the nodes possess the following rule:

Rule 6: *if (Premium loss >0.0001% OR Olympic Loss is >3%) then*
deactivate Round Robin
activate Priority Queuing

Figure 8 shows that the Olympic packets loss becomes almost insignificant from the moment that Rule 6 is applied. An important point, we would like to attract your attention on, is the fact that by monitoring the loss Olympic parameter, we could optimize another important parameter which is the Olympic delay (Figure 9). This proves that by carefully choosing the rules, one can optimize many parameters at the same time.

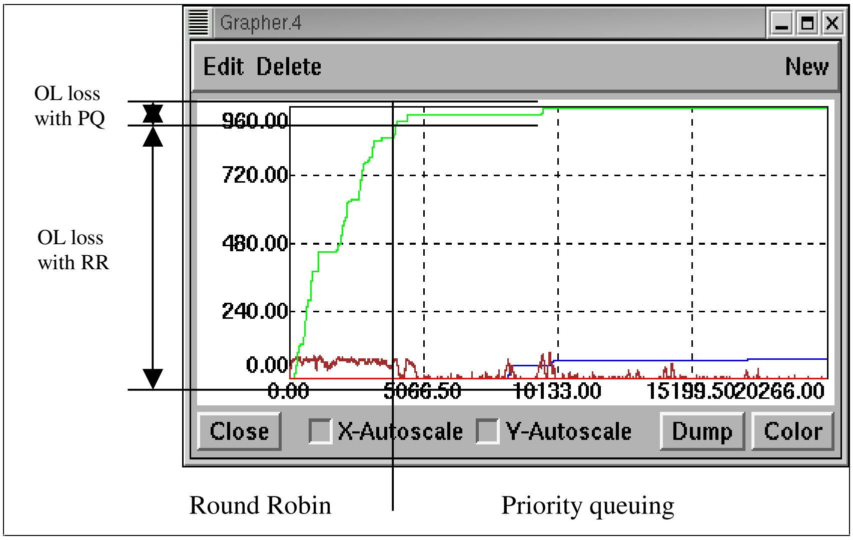


Fig. 8. Loss with adaptive scheduling

Example 3

Routing is another management mechanism that we think adaptive monitoring can optimize. In this simulation (Figure 10), PRIOR (60,90) is used as the queue MM (each node has one queue for the 3 classes of traffic) and the Rule 1 (section 5) is the one we are interested in. We can see that, after the reception of the message M1 by the neighbors of the node in question, this one minimizes Olympic packets loss. This demonstrates to what extent an adaptive routing can be advantageous.

These examples show the following aspects: (1) the importance of the monitored parameters is influenced by the current node condition and its current management mechanisms. Best effort packets loss was not important before the activation of PRIOR mechanism, for example; (2) the adaptive selection of the mechanisms controlling the node tasks depends on the occurring events; (3) the adaptive selection of the parameters that must be optimized is very important in order to achieve better end-to-end network management.

7 Related Work

An overview of the main features of on-line monitoring in addition to the description and the analysis of some monitoring systems is presented in [18]. These systems do not use a multi-agent approach. Schroeder classifies events in three classes: hardware-level events, process-level events and application-dependant events. Our system supports these three classes, even if we classify them differently. [1,2] propose a multi-agent dynamic distributed monitoring by using a set of local monitoring agents (LMA) and domain monitoring agents (DMA). The DMA delegates monitoring tasks to LMA by dynamically constituting and re-configuring groups of agents. The proposed approach uses reactive agents monitoring a particular event, while in our model, the same agents are in charge of nodes control (routing, scheduling, etc.) in addition to the monitoring task.

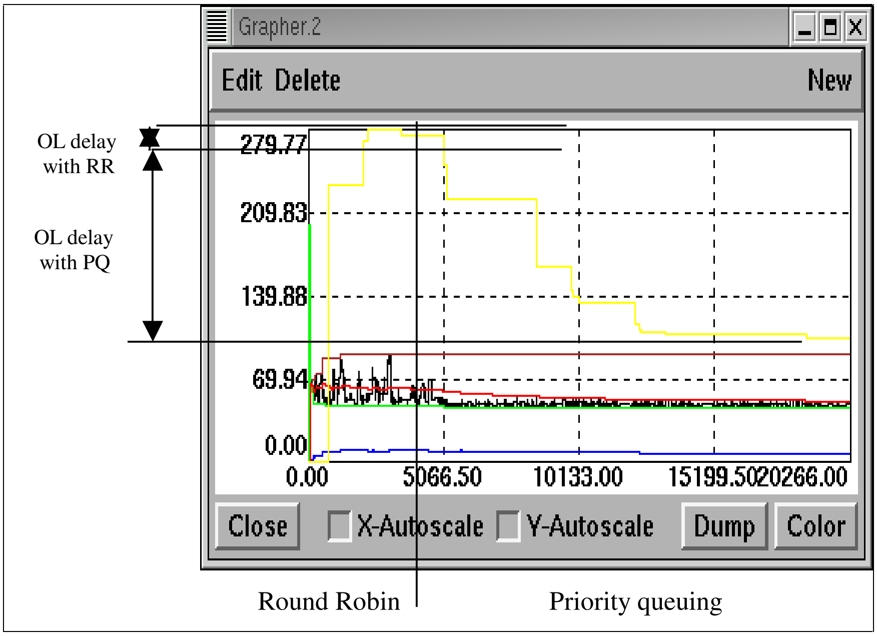


Fig. 9. Delay with adaptive scheduling

The originality of our contribution lies in the fact that we use a purely agent-oriented approach in order to guarantee an adaptive monitoring, by providing an agent-based model and an agent-based simulation. The agent model we propose respect the definition of [9] as: (1) the used agents can communicate by message sending; (2) each agent possesses it own resources (execution space, knowledge base, rules, etc.); (3) the Master Agent knows its environment: the node it belongs to, the reactive agents it is responsible for, etc. and the reactive agents also know the Master Agent and the node in which they reside; (4) the Master Agent possesses some data about the Master Agents of its nearest neighbors like their address, some of their current management mechanisms (further to a message reception), etc.; (5) each agent has an objective, even implicit. The Routing Agent objective is to affect the packet to the appropriate output queue, taking account of the packet's destination, and the current routing mechanism.

Messages transiting between agents are inspired from the agent-based approach. Even if in the current state the used messages do not appeal for the definition of advanced syntax and semantic, our system is open to extensions like KQML or FIPA-ACL. Aspects like cooperation and negotiation often used by the agent community- since the agents evolve in a common society and have common goals beyond their individual abilities- is supported by our model. We chose a multi-agent simulator for our experimentations in order to minimize the bias between the model and its simulation. In fact, the model is more respected because the multi-agent simulator offers the majority of functionalities and features that the agent-based model requires.

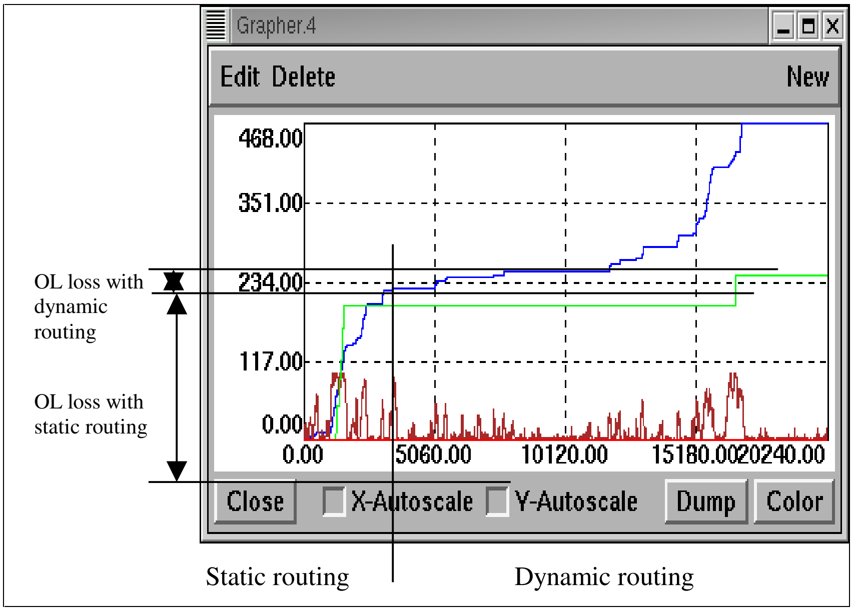


Fig. 10. Loss with adaptive routing

8 Conclusion

A new approach of adaptive monitoring in dynamic networks has been presented in this paper. This approach, based on agents, relies on their abilities like: the autonomy, the decentralization and especially, the adaptability to guarantee a continuous distributed, scalable and adaptive monitoring, aiming to optimize the network performance and guarantee the end to end QoS for the network's users.

We have described examples which clarify our model, and prove also the benefits of our adaptive rule-based selection of management mechanisms. As future work, we intend to use learning techniques to allow more adaptability in the rules' selection.

Anticipation on events is an important issue we want to explore in future work. In fact, activating a given management mechanism occurs only when a QoS parameter has already a poor performance. So, it will be interesting to have anticipative actions instead of corrective ones.

References

[1] Al Shaer E. Adaptive Multicast Group Management for Distributed Event Correlation. International Journal of Networking and Information Systems, pp. 75–88, June 1999.

[2] Al Shaer E. A Dynamic Group Management for Scalable Distributed Event Correlation, IEEE/IFIP Integrated Management (IM'2001), May 2001.

[3] Barber S. and Martin C. Dynamic Adaptive Autonomy in Multiagent Systems: Representation and Justification. IJPR&AI, Vol. 15, N° 3, pp 405–433, 2001.

[4] Bazzan A.L.C., Wahle J. and Klügl F. Agents in Traffic Modelling – From Reactive to Social Behaviour. KI'99, Bonn, Germany, LNAI 1701, pp 303–307 September 1999.

[5] Bensaid L., Drogoul A., and Bouron T. Agent-Based Interaction Analysis of Consumer Behavior. AAMAS'2002, Bologna, Italy, July 2002.

[6] Bodanese E.L. and Cuthbert L.G. A Multi-Agent Channel Allocation Scheme for Cellular Mobile Networks. ICMAS'2000, USA. IEEE Computer Society press, pp 63–70, July 2000.

[7] Doran J. Agent-Based Modelling of EcoSystems for Sustainable Resource Management. 3rd EASSS'01, Prague, Czech Republic, LNAI 2086, pp 383–403, July 2001.

[8] Drogoul A., Corbara B. ad Fresneau D. MANTA : New experimental results on the emergence of (artificial) ant societies".in Artificial Societies: the computer simulation of social life, Nigel Gilbert & R. Conte (Eds), UCL Press, London, 1995.

[9] Ferber J. Multi-Agent Systems: An Introduction to Distributed Artificial Intelligence. Addison Wesley Longman, 1999.

[10] Gaïti D. and Merghem L.: Network modelling and simulation: a behavioural approach, Smartnet conference, Kluwer Academic Publishers, pp. 19–36, Finland, April 2002.

[11] Merghem L. and Lecarpentier H. Agents: A Solution for Telecommunication Network Simulation. NetCon'2002, Paris, France, Kluwer Academic publishers. October 2002.

[12] Merghem L. and Gaïti D.: Behavioural Multi-agent simulation of an Active Telecommunication Network, STAIRS 2002, France. IOS Press, pp 217–226, July 2002.

[13] Minar N., Kramer K.H. and Maes P. Cooperating Mobile Agents for Mapping Networks. http://www.media.mit.edu/~nelson/research/route-coopagents/

[14] Moukas A. *et al.* Trafficopter: A Distributed Collection System for Traffic Information. CIA'98, Paris, France, LNAI 1435 pp 34–43, July 1998.

[15] Müller J.P and Pischel M. Modelling Reactive Behaviour in Vertically Layered Agent Architecture. ECAI'94, Amsterdam, Netherlands. John Wiley & Sons, pp 709–713, 1994.

[16] Pujolle G. "Les réseaux". Eyrolles Editions, 2003.

[17] Roychoudhuri R., *et al.* Topology discovery in ad hoc Wireless Networks Using Mobile Agents. MATA'2000, Paris, France. LNAI 1931, pp 1–15. September 2000.

[18] Schroeder B. On-line Monitoring: A Tutorial. IEEE Computer, Vol. 28, pp 72–78, 1995.

[19] Sigel E., *et al.* Application of Ant Colony Optimization to Adaptive Routing in LEO Telecommunications Satellite Network. Annals of Telecommunications, vol.57, no.5-6, pp 520–539, May-June 2002.

[20] White T. *et al.* Distributed Fault Location in Networks using Learning Mobile Agents. PRIMA'99, Kyoto, Japan. LNAI 1733, pp 182–196. December 1999.

[21] Woods W. Transition Network Grammar for Natural Language Analysis. Communication of the Association of Computing Machinery, Vol. 10, N 13, pp 591–606, 1970.

[22] Wooldridge M. Intelligent Agents. In « Multiagent Systems : a Modern Approach to Distributed Artificial Intelligence » Weiss G. Press, pp 27–77, 1999.
[23] http://www.cs.umbc.edu/kqml
[24] http://www.fipa.org/specs/fipa00061/SC00061G.pdf
[25] http://www.enib.fr/~harrouet/oris.htm

A New Available Bandwidth Measurement Technique for Service Overlay Networks

Cao Le Thanh Man, Go Hasegawa, and Masayuki Murata

Graduate School of Information Science and Technology, Osaka University
1-3, Machikaneyama-cho, Toyonaka, Osaka 560-8531, Japan
Phone: +81-6-6850-6863, Fax: +81-6-6850-6868
{mlt-cao, hasegawa, murata}@ist.osaka-u.ac.jp

Abstract. We introduce a new measurement algorithm for the available bandwidth in a network path between two endhosts. This algorithm is an adaptation from existing active measurement methods for the purpose of being applied in *inline network measurement*. Inline network measurement method measures the available bandwidth by using packets which an active TCP connection transmits, instead of using probe packets, as in the existing methods. The measurement is performed by inferring network characteristics information from transmitting and receiving intervals of TCP data and ACKnowledgement packets. Inline network measurement is useful in service overlay networks, in which up-to-date information on the available bandwidth in the underlying IP network should be known as early as possible. In this paper, we first introduce the measurement algorithm and then discuss the problems with applying the proposed measurement algorithm to an active TCP connection. We evaluate the proposed measurement algorithm using simulation experiments and show that it can provide an estimation of available bandwidth every 2-4 RTTs. Moreover, the accuracy of measurement results can be maintained considerably well while the number of packets in a measurement is reduced by approximately 90% in comparison with existing methods.

1 Introduction

Network measurement techniques have received a great deal attention, and numerous measurement tools have been developed, as reported in [1,2,3,4,5,6,7,8,9,10]. These tools observe and/or monitor network characteristics such as physical link bandwidth [5,6,7,8], available bandwidth [1,2,3], delay [7], loss [9] and topology of the network [10]. The observed results are often used for network trouble-shooting, isolation of fault location, and network provisioning [11]. Measurement techniques can be categorized into *active* and *passive* approaches. The active approaches [1,2,3,4,7,8,9,10] inject test packets into the network, and utilize the feedback information to derive the measurement results. The passive approaches [5,6] do not use extra packets, but rather monitor packets traversing a router interface or a link.

The population of Internet users and network diversity grow rapidly, and various types of service oriented networks, called *service overlay networks*, are emerging. Such networks include peer-to-peer (P2P) networks, Grid computing networks, Content Delivery/Distribution Networks (CDNs), and IP-VPNs. These networks are upper-layer networks that provide special-purpose services that are built upon a lower-layer network, i.e., the IP network. Therefore, in order to improve the performance of these service overlay networks, accurate and rapid information concerning the total and remaining resources of the lower IP network is important when a data transfer request for a particular service is initiated.

For example, in a CDN service such as Akamai [12] or Exodus [13], measurement techniques can be used to estimate the available bandwidth and regulate the sending rate of transmissions for Web prefetching [14] in order to avoid degrading the performance of other traffic. The network measurement technology can also be used to realize adaptive control mechanisms in various service overlay networks, including the following examples:

- In P2P networks, when the resource discovery mechanism finds multiple peers having the requested contents, the measurement results help to determine from which peer the contents are transmitted
- In data grid networks, when multiple sites have the same data, the measurement results help to determine from which site data will be copied or read.

However, we cannot directly employ existing measurement tools when we measure network characteristics on a service overlay network. One reason for this is that we want to avoid utilizing the test packets used by active measurement approaches for this measurement. This is because using the test packets would degrade the performance of other traffic when the measurements repeat continuously. Another reason is that we want to obtain the latest, and hence the most accurate, information about network characteristics as quickly as possible, because the volume of IP network traffic fluctuates greatly. However, the existing measurement techniques require a long time to obtain one measurement result.

Based on the considerations mentioned above, we are now developing a new network measurement technique to resolve these problems and to improve the quality of service overlay networks. In particular, we focus on a method to measure the *available bandwidth* for a path between two endhosts. Although our method is based on existing measurement approaches, we do not use extra packets for the measurement. Instead, the measurement is performed by collecting the information about the network characteristics obtained from the packets which a regular TCP connection transmits in providing a particular service (we call this approach *Inline Network Measurement*). The method is a sender-based measurement method; it does not require any change in the TCP receiver.

The idea of using packets in TCP for network measurements has attracted a lots of studies [9,15,16,17]. The most advantage of using TCP is that the measurement tool is able to work on a single host. Sting [9], a sender-side only modification of the TCP stack, takes advantage of TCP behavior to measure packet loss rate. In TCP Vegas [18] and TCP Nice [19], the sender uses RTT values to estimate the bandwidth available for the connection. TCP Westwood [17] and a study by Hoe [16] are the most similar to our approach. In these studies, the sender exploits ACK arrival intervals for bandwidth estimation, and decides the value of *ssthresh* according to the estimated values.

In the existing inline network measurement methods, there is a lack of validity in bandwidth measurement algorithms. The measurement algorithms in [16,17] are good in terms of simplicity but, as a result, fail to deliver good measurement results. A study in [20] mentions that the *packet pair* technique used in [16] is not suitable for estimating available bandwidth. Moreover, the packet pair techniques are evaluated in [15] and appear to yield bad effect to the connections in some cases, "due to an inability to form an estimate or overestimating". The same problem can be seen in TCP Westwood [17]. The measurement algorithm in TCP Westwood is similar to the *packet train* technique first proposed in [1]. The technique overestimates and leads to starvation and fairness disruption [21]. To mitigate the high values of the measurement results of TCP Westwood, some low-pass filters have been developed [22] . The filters successfully adjust the measurement results but considerably slow down the measurement speed. According to the experiment results given in [22], TCP Westwood, with filters, cannot give a good measurement result within longer than 40RTTs since the connection starts.

In this paper, therefore, we propose a valid measurement algorithm which becomes the fundamental mechanism of our inline network measurement method. The algorithm gives measurement results periodically; the algorithm reports an estimated value of available bandwidth every 2-4 RTTs (RoundTrip Times) so as to rapidly reflect changes in the IP network. The basic concept behind realizing such a rapid measurement is limiting the measurement range of the bandwidth by using statistical information of previous estimated results, rather than searching from 0 Mbps to the upper limit of the physical bandwidth, as in existing algorithms. By this mechanism, the proposed algorithm requires a rather small number of packets and a short time for measurement as compared to existing algorithms. Consequently, even when the available bandwidth of the path changes dramatically, the proposed algorithm can show the change after only a few measurements. The simulation results show that the proposed measurement algorithm can provide accurate results quickly and periodically, while using a rather small number of probe packets. In addition, we discuss the problems of applying the proposed measurement algorithm to an active TCP connection.

2 Inline Network Measurement in IP-Based Service Overlay Networks

2.1 Requirements

In accordance with the above discussion, we consider the following factors to be the requirements of the measurement algorithm of inline network measurement:

- Small number of packets used
  Because our method uses TCP packets for the measurement, there is a restriction on the number of packets available for transmission at any one time. This is because of the TCP window size. Since the TCP window size is relatively small and changes dynamically, the measurement algorithm should use as small a number of packets as possible.
- No effect on other traffic on the network
  Since the goal of measurement is to improve the quality of services of the service overlay network, the measurement should not affect either the traffic of the supported services itself or the external traffic. The measurement may adversely affect the

network in two ways: by sending numerous probe packets and by sending probe packets at a high rate.

- Providing results continuously
  Since the characteristics of the IP network changes constantly and dynamically, measurement should provide periodic estimation results. Furthermore, the interval should be as small as possible in order to provide an accurate depiction of the rapidly network change.
- Providing results quickly
  The measurement should be performed quickly in order to obtain up-to-date information of the IP network. In the proposed method, we therefore assign a higher priority to measurement speed than to measurement accuracy.

2.2 Existing Network Measurement Methods

As mentioned in Section 1, the existing measurement methods can be divided into two groups: passive measurement methods and active measurement methods. The passive methods, represented by SPAND [5] and Nettimer [6], observe passing traffic at some certain points in the network and use the monitored information to obtain the measurement results. These approaches require quite a long time to gather information for accurate measurement results and many measurements are necessary in order to estimate the characteristics of the end-to-end path. Furthermore, passive approaches cannot provide high-accuracy measurement results because the available information is very limited.

On the other hand, the active measurement methods inject probe packets into the network and collect the feedback information from monitored results including transmission delay, packet arrival-interval time, packet loss ratio and so on. Therefore, we can expect a higher accuracy of measurement results in an end-to-end fashion than is possible by passive methods. Cprobe [1], Topp [2], and Pathload [3], are representative tools to measure the available bandwidth of the network path between two endhosts. These algorithms work on endhosts and require no change inside the network, so that they seem suitable for application to measurement in service overlay networks. However, these algorithms also have fundamental disadvantages. One is that many probe packets are sent at a high transmission rate. For instance, Topp sends 5000 packets to obtain only one measurement, and Cprobe injects 100-200 probe packets at the physical bandwidth speed of the link connected to the sender host. The probe traffic can affect other traffic along the path, for example by degrading traffic throughput and increasing the packet loss ratio and packet transmission delay. Existing active measurement algorithms also require a long time to obtain one measurement result (for example, 50-100 RTTs are necessary to obtain one estimation value for the available bandwidth in Topp and Pathload). Long-term measurement can provide an accurate result but cannot follow the dynamic changes on the IP network.

3 Proposed Measurement Algorithm

Figure 1 shows an outline of the proposed measurement algorithm. A sender host transmits measurement packets to a receiver host, which immediately sends received packets

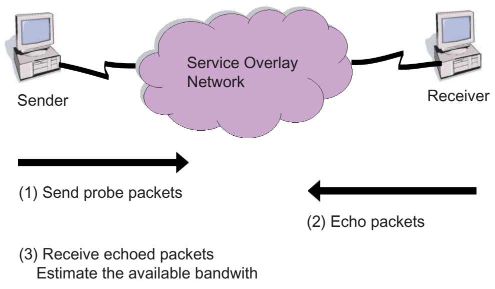


Fig. 1. Outline of proposed measurement algorithm

back to the sender host. The sender then estimates the available bandwidth of the path using the arrival intervals of the echoed packets.

In every measurement, we use a *search range* to find the value of the available bandwidth. Search range I= (B_l, B_u) is a range of bandwidth which is expected to include the current value of the available bandwidth. The proposed measurement algorithm searches for the available bandwidth only within the given search range. The minimum value of B_l, the lower bound of the search range, is 0, and the maximum value of B_u, the upper bound, is equal to the physical bandwidth of the link directly connected to the sender host. By introducing the search range, we can avoid sending probe packets at an extremely high rate, which seriously affects other traffic. We can also keep the number of probe packets for the measurement quite small. As discussed later herein, even when the value of the available bandwidth does not exist within the search range, we can find the correct value in a few measurements. The following are the steps of the proposed algorithm for one measurement of the available bandwidth A:

1. Set the initial search range.
2. Divide the search range into multiple sub-ranges.
3. Inject a packet stream into the network for each sub-range and check the increasing trend of the packet inter-arrival times of the received stream.
4. Find a sub-range which is expected to include the correct value of the available bandwidth using the increasing trends of sent streams.
5. Calculate the available bandwidth by means of linear regression analysis for the chosen sub-range.
6. Create a new search range and return to Step 2.

A packet stream is a group of packets sent at one time for the measurement. In what follows, we explain in detail the algorithm by which to implement the above steps.

1. Set initial search range.
   We first send a packet stream according to the Cprobe algorithm [1] to find a very rough estimation of the available bandwidth. We set the search range to $(A_{cprobe}/2, A_{cprobe})$, where A_{cprobe} is the result of the Cprobe test.
2. Divide the search range.

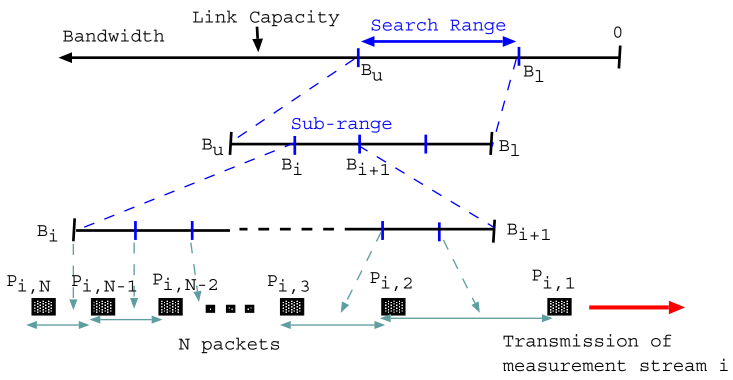


Fig. 2. Relationship of search range, sub-ranges, streams, and probe packets

We divide the search range into k sub-ranges $\mathrm{I}_i = (B_{i+1}, B_i)$ $(i = 1, 2..k)$. All sub-ranges have the identical width of the bandwidth. That is,

$$B_i = B_u - \frac{B_u - B_l}{k}(i-1) \;\; (i = 1, ..., k+1)$$

As k increases, the results of Steps 4 and 6 become more accurate, because the width of each sub-range becomes smaller. However, a larger number of packet streams is required, which results in an increase in the number of used packets and the measurement time.

3. Send packet streams and check increasing trend.
For each of k sub-ranges, a packet stream i $(i = 1...k)$ is sent. The transmission rates of the stream's packets vary to cover the bandwidth range of the sub-range. We denote the j-th packet of the packet stream i as $\mathrm{P}_{i,j}$ $(1 \leq j \leq N$, where N is the number of packets in a stream) and the time at which $\mathrm{P}_{i,j}$ is sent from the sender host as $S_{i,j}$, where $S_{i,1} = 0$. Then $S_{i,j}$ $(j = 2..N)$ is set so that the following equation is satisfied:

$$\frac{M}{S_{i,j} - S_{i,j-1}} = B_{i+1} + \frac{B_i - B_{i+1}}{N-1}(j-1)$$

where M is the size of the probe packet. Figure 2 shows the relationship between the search range, the sub-ranges and the packet streams. In the proposed algorithm, packets in a stream are transmitted with different intervals, for this reason the measurement result may not be as accurate as the Pathload algorithm [3], in which all packets in a stream are sent with identical intervals. However, the proposed algorithm can check a wide range of bandwidth with one stream, whereas the Pathload checks only one value of the bandwidth with one stream. This reduces the number of probe packets and the time required for measurement. By this mechanism, the measurement speed is improved at the expense of measurement accuracy, as described in Subsection 2.1.

We then observe $R_{i,j}$, the time the packet P$_{i,j}$ arrives at the sender host, where ($R_{i,1} = 0$). We then check if an increasing trend exists in the packet arrival intervals $(R_{i,j} - R_{i,j-1})$ $(2 \leq j \leq N)$ according to the algorithm used in [3]. As explained in [3], the increasing trend of a stream indicates that the transmission rate of the stream is larger than the current available bandwidth of the network path.
Let T_i be the increasing trend of stream i as follows:

$$T_i = \begin{cases} 1 & \text{increasing trend in stream } i \\ -1 & \text{no increasing trend in stream } i \\ 0 & \text{unable to determine the existence of an increasing trend in stream } i \end{cases}$$

As i increases, the rate of stream i decreases. Therefore, T$_i$ is expected to be 1 when i is sufficiently small. On the other hand, when i becomes large, T_i is expected to become -1. Therefore, when neither of the successive streams m or $m+1$ have an increasing trend ($T_m = T_{m+1} = -1$), the remaining streams are expected not to have increasing trends ($T_i = -1$ for $m+2 \leq i \leq k$). Therefore, we stop sending the remaining streams in order to speed up the measurement.

4. Choose a sub-range.
Based on the increasing trends of all streams, we choose a sub-range which is most likely to include the correct value of the available bandwidth. First, we find the value of a $(0 \leq a \leq k+1)$, which maximizes ($\sum_{j=0}^{a} T_j - \sum_{j=a+1}^{k} T_j$). If $1 \leq a \leq k$, we determine the sub-range I$_a$ is the most likely candidate of the sub-range which includes the available bandwidth value. That is, as a result of the above calculation, I$_a$ indicates the middle of streams which have increasing trends and those which do not. If $a = 0$ or $a = k+1$, on the other hand, the algorithm decides that the available bandwidth does not exist in the search range (B_l, B_u). We determine that the available bandwidth is larger than the upper bound of the search range when $a = 0$, and that when $a = k+1$ the available bandwidth is smaller than the lower bound of the search range.
In this way, we find the sub-range which is expected to include the available bandwidth according to the increasing trends of the packet streams. We avoid using the values of packet receiving intervals like TOPP or Cprobe because their use may lead to serious estimation errors when the transmission rate of probe packets is much larger than the available bandwidth [3,20].
5. Calculate the available bandwidth.
We then derive the available bandwidth A from the sub-range I$_a$ chosen by Step 4. We first determine the transmission rate and the arrival rate of the packet P$_{a,j}$ ($j = 2...N$) as $\frac{M}{S_{a,j}-S_{a,j-1}}$, $\frac{M}{R_{a,j}-R_{a,j-1}}$, respectively. We then approximate the relationship between the transmission rate and the arrival rate as two straight lines using the linear regression method, as shown in Figure 3. Since we determine that the sub-range I$_a$ includes the available bandwidth, the slope of line (i) which consists of small transmission rates is nearly 1 (the transmission rate and the arrival rate are almost equal), and the slope of line (ii) which consists of larger transmission rates is smaller than 1 (the arrival rate is smaller than the transmission rate). Therefore, we determine that the highest transmission rate in line (i) is the value of the available bandwidth.

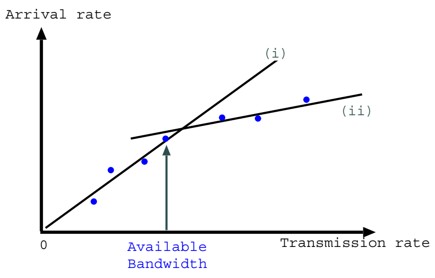


Fig. 3. Finding the available bandwidth within the sub-range

On the other hand, when we have determined that the available bandwidth value does not exist in the search range (B_l, B_u) in Step 4, we temporarily set the value of available bandwidth as follows:

$$A = \begin{cases} B_l & a = 0 \\ B_u & a = k+1 \end{cases}$$

6. Create a new search range.
When we have found the value of the available bandwidth from a sub-range I$_a$ in Step 5, we accumulate the value as the latest statistical data of the available bandwidth. The next search range (B_l', B_u') is calculated as follows:

$$(B_l', B_u') = \left(A - max\left(1.96\frac{S}{\sqrt{q}}, \frac{B_m}{2}\right), A + max\left(1.96\frac{S}{\sqrt{q}}, \frac{B_m}{2}\right)\right)$$

where S is the variance of stored values of the available bandwidth and q is the number of these values. Therefore, we use the 95% confidential interval of the stored data as the width of the next search range, and the current available bandwidth is used as the center of the search range. B_m is the lower bound of the width of the search range, which is used to prevent the range from being too small. When no accumulated data exists (when the measurement has just started or just after the accumulated data is discarded), we use the search range of the previous measurement.
On the other hand, when we can not find the available bandwidth within the search range, it is possible to consider that the network status has changed greatly. Therefore, we discard the accumulated data because this data becomes unreliable as statistical data. In this case, the next search range (B_l', B_u') is set as follows:

$$B_l' = \begin{cases} B_l & a = 0 \\ B_l - \frac{B_u - B_l}{2} & a = k+1 \end{cases}$$

$$B_u' = \begin{cases} B_u + \frac{B_u - B_l}{2} & a = 0 \\ B_u & a = k+1 \end{cases}$$

This modification of the search range is performed in an attempt to widen the search range in the possible direction of the change of the available bandwidth.

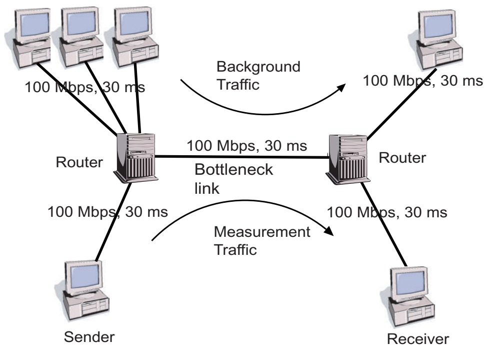


Fig. 4. Network model for simulation experiments

By this statistical mechanism, we expect the measurement algorithm to behave as follows: when the available bandwidth does not change greatly over a period of time, the search range becomes smaller and more accurate measurement can be obtained. On the other hand, when the available bandwidth varies greatly, the search range becomes large and the measurement can be restarted from the rough estimation. That is, the proposed algorithm can give a very accurate estimation of the available bandwidth when the network is stable, and a rough but rapid estimate can be obtained when the network status changes.

4 Simulation Results

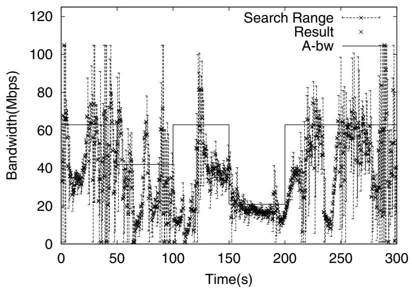
This section shows some simulation results in ns [23] and validates the measurement algorithm proposed in Section 3. Figure 4 shows the network model used in the simulation. A sender host connects to a receiver host through a bottleneck link. The capacity of the bottleneck link is 100 Mbps and the propagation delay is 30 msec. All of the links from the endhosts to the routers have a 100-Mbps bandwidth and a 30-msec propagation delay.

There is background traffic generated by endhosts connecting to the routers. The background traffic is made up of UDP packet flows, in which various packet sizes are used according to the monitored results reported in [24]. The correct value of the available bandwidth of the bottleneck link is calculated as:

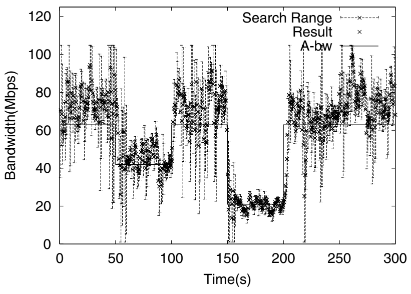
$$Bottleneck\ link\ capacity - Total\ rate\ of\ background\ traffic$$

We make the available bandwidth on the bottleneck link fluctuate by changing background traffic rates.

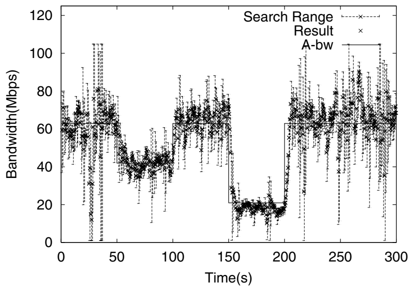
The sender host sends probe packets to the receiver host and the receiver host echoes the packets to the sender. The sender, using the algorithm proposed in Section 3, measures the available bandwidth of the path between the two hosts. In this situation, the result corresponds to the available bandwidth of the bottleneck link between the routers.



(a) N=3



(b) N=5



(c) N=8

Fig. 5. Simulation results

The number of sub-range k, which a search range is divided into, is decided according to the width of the search range and the latest result of the measured available bandwidth, A_{prev};

$$k = \begin{cases} 2 & (0 \leq \frac{B_u - B_l}{A_{prev}} < 0.15) \\ 3 & (0.15 \leq \frac{B_u - B_l}{A_{prev}} < 0.2) \\ 4 & (0.2 \leq \frac{B_u - B_l}{A_{prev}}) \end{cases}$$

B_m, the lower bound of the width of search ranges, is set to 10% of A_{prev}. The probe packet size is 1500 Bytes.

Figure 5 shows the measurement results of the available bandwidth and the search ranges for a simulation time of 300 sec. During the simulation, the background traffic is changed so that the available bandwidth of the bottleneck link is 60 Mbps from 0 sec to 50 sec, 40 Mbps from 50 sec to 100 sec, 60 Mbps from 100 sec to 150 sec, 20 Mbps from 150 sec to 200 sec and 60 Mbps from 200 sec to 300 sec. We also plot the correct values of the available bandwidth in all figures. Figures 5(a)-5(c) show the results when the number of the probe packets in a stream (N) is 3, 5 and 8, respectively. These figures indicate that when N is 3, the measurement results are far from the correct values. When N becomes larger than 5, on the other hand, the estimation result accuracy increases. The proposed measurement algorithm can determine the available bandwidth rapidly, even when the available bandwidth changes suddenly. When N is very small, we can not determine the increasing trend of the streams correctly in Step 3 in the proposed algorithm, which leads to the incorrect choice of subrange in Step 4. Although the accuracy of measurement results increases as N is increased from 5 to 8, since we place a higher priority on measurement speed than on measurement accuracy, as described in Subsection 2.1, $N = 5$ is judged to be the better setting. Actually, finding a suitable value of N is a difficult problem because selection depends on many factors, such as the available bandwidth, the bandwidth size of changes of the available bandwidth and the background traffic. A solution to this problem will require further study. From these simulation results, we can conclude that the proposed algorithm can quickly estimate the available bandwidth, independent of the degree of change in available bandwidth.

5 Problems with Applying the Proposed Algorithm to an Active TCP Connection

In TCP, the TCP sender sends data packets to the receiver and the receiver sends corresponding ACK packets back to the sender. Therefore, we can apply the measurement algorithm in Section 3 to TCP by considering data and ACK packets as probe packets. Due to various characteristics and mechanisms of TCP, however, there are some problems when applying the proposed measurement algorithm.

– Window size
  In a TCP connection, the number of packets which can be transmitted at one time is limited by the window size. The maximum number of packets sent for one measurement stream, therefore, is also limited by the window size. This limitation seriously affects the measurement algorithm, especially when the TCP has a small window size. Moreover, the TCP window size always fluctuates greatly due to the congestion control mechanism of TCP, which also causes difficulty in choosing the optimal number of packets to send at one time for the measurement.
  In order to overcome the problem, the measurement algorithm should be able to dynamically adapt to the changes of the window size of TCP as follows. The measurement algorithm should not attempt to send a measurement stream when the current window size is smaller than the number of packets required for the stream. On the other hand, when the window size is sufficiently large, the algorithm should create as many measurement streams as is allowed by the window size in order to enhance the measurement speed.
– Degrading TCP transmission speed
  The TCP sender generally transmits data packets as soon as possible when it receives an ACK packet from the receiver. However, the measurement algorithm requires multiple packets to build up a measurement stream. Therefore, it is necessary for these packets to be stored briefly before transmission, which causes a delay in packet transmission.

We can avoid degrading TCP transmission speed, caused by storing data packets, by setting an appropriate timeout value for stopping the creation of a stream. There is a trade-off relationship between the speed of the measurement and the TCP transmission speed in choosing the timer length. That is, if the timer length is long, the measurement algorithm can create more measurement streams, so the measurement can be performed quickly. However, since numerous TCP data packets are stored at the intermediate buffer for a long time, TCP transmission speed may be deteriorated. Moreover, the long packet delay may lead to TCP timeout events.

– Behavior of TCP receiver
  The proposed measurement algorithm expects the TCP receiver to send an ACK packet back to the sender immediately when a data packet arrives. However, some TCP receivers utilize the delayed ACK option [25], in which the TCP receiver does not deliver an ACK packet for each data packet. In this case, the measurement algorithm may not work properly. In this case, Step 3 of the proposed algorithm should be changed so that intervals of three packets are used rather than intervals of two packets. That is, we calculate the arrival intervals $(S_{i,2j'+2} - S_{i,2j'})$ $(1 \leq j' \leq \lfloor N/2 \rfloor)$ for the probing packets in stream i in order to check its increasing trend. This has the same effect as halving the number of packets in one stream, which results in the degradation of measurement accuracy. Therefore, the number of packets in a stream should be increased appropriately.

6 Conclusion

In this paper, we introduced a measurement method for the available bandwidth of a path between two endhosts using an active TCP connection. We first constructed a new measurement algorithm which uses quite a small number of probe packets and yet provides periodic measurement results quickly. We presented a number of simulation results in order to validate the effectiveness of the proposed algorithm. We then discussed problems when applying the proposed algorithm to an active TCP connection and presented possible solutions. The proposed methods were evaluated by simulation experiments and we have observed that the proposed measurement algorithm can successfully measure the available bandwidth. As future research, we will integrate the measurement algorithm into TCP and validate its effectiveness.

References

1. R. L. Carter and M. E. Crovella, "Measuring bottleneck link speed in packet-switched networks," Tech. Rep. TR-96-006, Boston University Computer Science Department, Mar. 1999.
2. B. Melander, M. Bjorkman, and P. Gunningberg, "A new end-to-end probing and analysis method for estimating bandwidth bottlenecks," in *Proceedings of IEEE GLOBECOM 2000*, Nov. 2000.
3. M. Jain and C. Dovrolis, "End-to-end available bandwidth: Measurement methodology, dynamics, and relation with TCP throughput," in *Proceedings of ACM SIGCOMM 2002*, Aug. 2002.
4. M. Allman, "Measuring end-to-end bulk transfer capacity," in *Proceedings of ACM SIGCOMM Internet Measurement Workshop 2001*, ACM SIGCOMM, November 2001.

5. S. Seshan, M. Stemm, and R. H. Katz, "SPAND: Shared passive network performance discovery," in *Proceedings of 1st Usenix Symposium on Internet Technologies and Systems (USITS '97)*, pp. 135–146, Dec. 1997.
6. K. Lai and M. Baker, "Nettimer: A tool for measuring bottleneck link bandwidth," in *Proceedings of the USENIX Symposium on Internet Technologies and Systems*, Mar. 2001.
7. V. Jacobson, "pathchar-A Tool to infer characteristics of Internet paths," *http://www.caida.org/tools/utilities/others/pathchar/*, 1997.
8. A. B. Downey, "Using pathchar to estimate internet link characteristics," *Measurement and Modeling of Computer Systems*, pp. 222–223, 1999.
9. S. Savage, "Sting: A TCP-based network measurement tool," in *Proceedings of USITS '99*, Oct. 1999.
10. B. Hu, A Daniel and P. David, "Topology discovery by active probing," in *Proceedings of the 2002 Symposium on Applications and the Internet (SAINT)*, Jan 2002.
11. K. Matoba, S. Ata, and M. Murata, "Capacity dimensioning based on traffic measurement in the Internet," in *Proceedings of IEEE GLOBECOM 2001*, pp. 2532–2536, Nov. 2001.
12. Akamai Home Page, *http://www.akamai.com/*.
13. Exodus Home Page, *http://www.exodus.com/*.
14. L. Fan, P. Cao, and Q. Jacobson, "Web prefetching between low-bandwidth clients and proxies: Potential and performance," in *Proceedings of ACM SIGMETRICS '99*, May 2000.
15. Mark Allman and Vern Paxson, "On estimating end-to-end network path properties," in *Proceedings of SIGCOMM '99*, pp. 263–274, 1999.
16. J. C. Hoe, "Improving the start-up behavior of a congestion control sheme for TCP," in *Proceedings of the ACM SIGCOMM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, vol. 26,4, pp. 270–280, ACM Press, 1996.
17. M. Gerla, M. Y. Sanadidi, R. Wang, A. Zanella, C. Casetti and S. Mascolo, "TCP Westwood: Congestion window control using bandwidth estimation," in *Proceedings of IEEE Globecom 2001*, pp. 1698–1702.
18. M. Gerla, Y. Sanadidi, R. Wang, A. Zanella, C. Casetti and S. Mascolo, "TCP Vegas: New techniques for congestion detection and avoidance," in *Proceedings of the SIGCOMM '94 Symposium*, pp. 24–35, Aug. 1994.
19. A. Venkataramani, R. Kokku and M. Dahlin, "TCP-Nice: A mechanism for background transfers," in *Proceedings of the Fifth Symposium on Operating Systems Design and Implementation*, Dec. 2002.
20. C. Dovrolis and D. Moore, "What do packet dispersion techniques measure?," in *Proceedings of IEEE INFOCOM 2001*, pp. 22–26, Apr. 2001.
21. Luigi Alfredo Grieco and Saverio Mascolo, "TCP westwood and easy RED to improve fairness in high-speed networks," in *Proceeding of seventh International workshop on Protocols for High-Speed Networks (PfHSN 2002)*, pp. 130–146, 2002.
22. L. A. Grieco and S. Mascolo, "End-to-end bandwidth estimation for congestion control in packet networks," in *Proceedings of Quality of Service in Multiservice IP Networks, II Int. Workshop, QoS-IP 2003*, Feb. 2003.
23. NS Home Page, *http://www.isi.edu/nsnam/ns/*.
24. NLANR web site, *http://moat.nlanr.net/Datacube/*.
25. R. Wright and R. Stevens,, "TCP/IP Illustrated, Volume 1: The Protocols," *Addison-Wesley Publishing, USA 1994*.

Management of *Peer-to-Peer* Networks Applied to *Instant Messaging*

Guillaume Doyen, Emmanuel Nataf, and Olivier Festor

Equipe MADYNES
LORIA
Campus Scientifique
54500 Vandoeuvre les Nancy
{doyen, nataf, festor}@loria.fr

Abstract. *Peer-to-peer* networking has allowed the development of new applications that couln't have been built on the client/server model. Having a good operation of such applications implies a considerable support of underlying networks and services. A typical application is instant messaging and we present an instrumentation of one of the existing one. In order to do that, we use the abstract model for such a service defined in the IETF context. Finally, this work is materialized by the instrumentation of a basic service: the presence service. We have used a particular Instant messaging application that is JXTA based : JIM. This last is built on the JXTA development environnement and allows our instrumentation to be reusable for other instant messaging applications.

1 Introduction

1.1 *Peer-to-Peer* Networking

Peer-to-Peer (P2P) systems of services and infrastructures are of a large variety, both in the nature of applications and in used models. A taxonomy of P2P networks and typical applications given in [6] details the main actual works in this domain and proposes a first step towards the generalization of some concepts. As of today, the main applications are dedicated to file sharing (like Freenet, Napster or Gnutella), grid computing (like SETI@Home), collaborative work (like Groove or MAGI) and messages exchanges between buddies (like ICQ, AIM, Jabber). Moreover, some generic platforms enable the development of applications independently of their nature (like JXTA or .NET).

The main principle of P2P networking consists in runnning an application in a cooperative and identical way among several users. The underlying P2P infrastructure aims at ensuring them that the the communication is well available. The communication models detailled in [7] can be, for some of them, very close to the client/server one, as in SETI@Home where each client executes a part of a global calculation individually and returns its result to a fixed server. Other P2P models are pure, in the sence that they do not present any dedicated server, but only entities that can play the role of both client and server (called *servent* or *peer*), like in Freenet. Finally, hybrid models use dedicated servers whose role consist in grouping *peers* addresses so that a new *peer* wanting to join the community can do it efficiently. In the following of this paper, we will distinguish P2P applications from P2P networks, the first ones are using a P2P service supported by the second ones.

1.2 Network Management and P2P

The rapid success of services delivered by P2P networks implies the appearance of some potential risks for their own operation. Let's have an example with the file sharing where a major part of users are consumer first of all. In this situation, the over consumption makes resources harder to access and leads to service disruption. Moreover, danger is present too for underlying supporting networks and enterprises that foster the users (like network services in enterprises and Internet service providers). Indeed, the traffic due to P2P services can be and often already is very consequent (in particular with the exchange of multimedia documents like DivX movies) and bottlenecks of well-known servers are not so easy to locate because any user can become a server. The access facilities to these services often enable the users to bypass any authorization of the enterprise and thus to generate personal traffic accounted to the enterprise. As a consequence, it can become a potential weakness in security policies because both the tools and the exchanged content are out of any control process. Such dangers have been rapidly detected and there are several solutions to treat some of these aspects individually (for example by tracking any traffic towards a well-known *peer* server).

Functional areas of network management and P2P. Functional areas of network management can address different important points. In the following, we reuse the standard classification and apply it to P2P networks. Thus:

– **Fault managemement** that deals with discovering abnormal behavior of the network, identifying its cause and taking the appropriate actions to solve the fault, has to reconsider most of the definitions of a fault known in fixed networks or standard architectures to fit the P2P behavior.
– **Configuration management** that enable the coordination of the nodes paramaters of a network must not be located in the equipments but in the *peers*.
– **Accounting management** is in charge of associating a value to the use of resources and can be applied to the P2P networking. It consists in collecting data bout a service usage, defining its price, and finally bill the user.
– **Performance management** whose goal is to ensure a given traffic level and secondly to be able to plan the evolution of networks, devices and services, together with their configuration is dependant on the nature of the considered P2P application.

– **Security management** is in charge of the set of encoding and authentication mechanisms that allow access to resources and of those that ensure an optimal use of resources. In the case of P2P services, alternatives approaches based on trust relationship emerge.

Management solutions. There is no full managment solution for *peer–to-peer* services. Nevertheless, some architectures deals with a particular functional area of the network management. About performance management, [3] proposes to use an active network architecture to ensure performance management of a Gnutella like P2P application. [1] carry out accounting management by defining a fictitious money services that regulates exchanges between *peers*; gains are collected by the *peers* which the most participate to the service delivery.

P2P services have a strong need of management if they want to be more generally deployed and embrace new application domains. In the case of a free participation of any user, everyone benefits from delivering a way to get the whole managed correctly. In the case of private services (charged or restricted to a enterprise), the need to define policies implies the need to manage the delivered service by enforcing the specified policies.

In this context, we are seeking to propose a generic architecture dedicated to the management of P2P services. Thus, our approach consists in using a generic platform to develop P2P services. Its functional role is between the realization of services (Instant messaging, file sharing, ...) and a common execution environment for the all applications. A first experiment on JXTA [8] has shown its openess for management mechanisms; by using some basic functionalities available in each *peer*, these latters features an interface allowing their management.

The work presented in the remainder of this paper comes within the scope of a study to propose management models for various types of applications in the P2P world. We have chosen the Instant messaging one firstly for its simplicity enabling a very focused instrumentation, and a dependent integration of a service that is complex to isolate. Lastly, existence of several IM applications based on JXTA allows us to test the genericity of our approach and to establish some comparisons.

2 *Instant Messaging*

2.1 Instant Messages and Presence Models

The IETF has constituted a working group whose goal is to define a model for Instant Messaging applications[1]. Results of this group are a set of proposals in the draft state and some *informational* RFC. In the general model [2], two services are defined; one for the *peer* presence and another for the messages delivery. Figure 1 describes the main elements that compose these services.

The presence service (`PRESENCE SERVICE`) is charged to maintain the consistency between the state of a user and the one seen by others users. A user can

[1] `http://www.ietf.org/html.charters/impp-charter.html`

claim himself as being ready to receive messages or signal that he is away, busy or offline (only in the first two cases messages can be sent). The model proposes to represent a *peer* according to its use of the presence service; it is seen as a `PRESENTITY` when it notifies the service of its state. The users concerned by the state of an other can be informed by the `PRESENCE SERVICE` in different ways : (1) by examining (`WATCHER`) what can be done once (`FETCHER`) or regularly (`POLLER`), or (2) by notification (`SUBSCRIBER`).

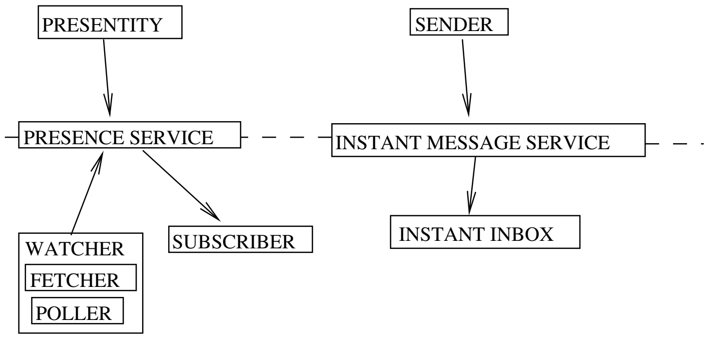


Fig. 1. IETF model for *instant messaging*

When a user wants to send a message to another one, he is described in the model as being a `SENDER` and has to use the message service (`INSTANT MESSAGE SERVICE`). This service is in charge of delivering a message into the environment of the receiver (`INSTANT INBOX`).

2.2 Measurement of the Presence Service Latency

Instant messaging application management, we aim at characterizing the quality of service delivered by all services among them, the `PRESENCE SERVICE`. This service is crucial because it ensures the consistent view that a *peer* has of the others. Whenever the service is not delivered or with too much latency, the state change of a *peer* will not be signaled to the other `SUBSCRIBER` `WATCHER` *peers*. Thus, a *peer* wishing to transmit a message to another one whose state appears as being able to receive messages will do it with the risk of a message loss. Conversely, when a *peer* sees another as being able to receive messages, it knows afterwards whether a message has been lost, that the recipient *peer* state has changed and that it cannot receive any message.

Different solutions are possible to measure the latency time of the presence service. Figure 2 illustrates the different elements that come in the scope of this functionality. In the case of a *peer* running as a `WATCHER` (*peer1*, according to the figure 2.a), when the *peer* 2 `PRESENTITY` signals its state to the presence service (arrow 1), it is the *peer2*'`WATCHER`'s role to launch a request towards the service (arrow 2) and to measure the time spent to recover the response (arrow 3). This time represents the delay needed for a *peer* to obtain the state of another one.

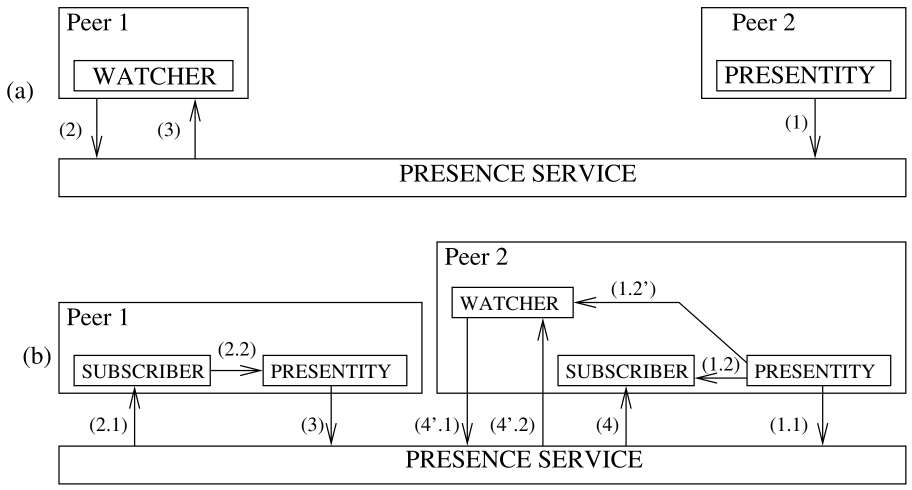


Fig. 2. Presence Service Latency Measurement

In the case of figure 2.b, one cannot measure the delay of the presence service to obtain the state of a *peer* because when the `SUBSCRIBER` of *peer*1 receives a signal from the presence service, he cannot have the knowledge of the date when *peer* 2 has notified the presence service of its state. The proposed solution for this configuration consists in measuring the time spent by the presence service to propagate the state of a *peer* towards another one, what is corresponding to a symmetric vision of the one seen just before. To realize this measurement, each `PRESENTITY` of the *peers* has to collaborate with the other constituting elements. In all cases, a *peer2* signals its state to the presence service (arrow 1.1) and the `SUBSCRIBER` of the other *peer1* is warned (arrow 2.1). In order to inform the *peer2* of the reception, the `SUBSCRIBER` has to use the `PRESENTITY` (arrow 2.2) to send an acknowledgement (arrow 3). On the *Peer 2* side, this acknowledgement can be recovered in different ways depending on whether it contains a `SUBSCRIBER` or a `WATCHER`. In the first case, reception gives immediately the measurement (namely the time spent from arrow 1.1 to number 4 one using 1.2), in the second case, it is necessary to recover the service request in the same way as in the precedent case (arrows 1.1, 4'.1, 4'.2 using 1.2'). This last measurement method is complex, implies a consequent overload of instumented elements and appears to be approximate. Indeed, transactions 1.1 and 2.1 can be done much faster than the issuance of the acknowledgement (arrows 3 and 4.x) implying a wrong increase of the RTT measurement.

2.3 Managing the Latency of Several *Peers*

Considering a global manager for the P2P application, as in the case of an enterprise or an Internet service provider, its interesting to know what are the parts of an application that can be improved or delivered with an associated service level agreement. For example, according to the figure 3, *peers p1* to *p4* form an IM application group, arrows between *peers* are as large as the time spent for the presence service to operate.

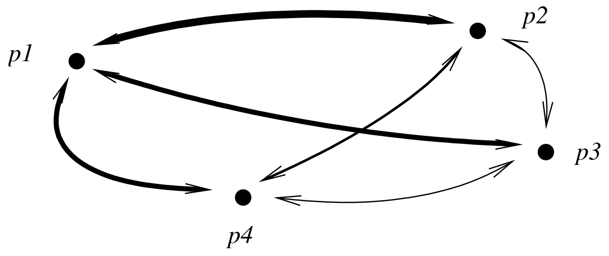


Fig. 3. *RTT* Global view of the presence service

Thus, *peer p1* seems to be more distant than the others ; *p2* and *p4* have slower links than with *p3*. Consequences of these remarks can be, at the underlying network level, to verify whether there is no element decreasing the traffic rate and to differently bill according to the presence of some of them (e.g. *p1* presence decreases the global service cost due to the decreasing of quality).

3 JIM

In order to build an instrumentation of the presence service of an IM application, we have used the JIM freeware (JXTA *Instant Messaging*)[2]. This one fits the IETF model presence by using `WATCHER/ POLLER` *peers*. Secondly, its design built over JXTA allows us to compare differents IM implementations over this environnement and to prepare an instrumentation of JXTA itself.

We chose JIM, because contrary with other IM applications it uses a P2P model. Indeed, AOL Instant Messenging or ICQ uses a client-server model while Jabber presents a functionment very similar to the mail: clients under a domain send and receive instants messages through a well-known server, and the routage between the different domains is done by a server-to-server communication.

3.1 States in JIM

A *peer* running JIM can present several states: *online* when it is ready to receive and answer messages, *busy* when it is ready to receive but cannot answer

[2] http://visis-www.informatik.uni-hamburg.de/jim

immediately, *away* when it is ready to receive but will answer in a long time (more than in *busy*). Finally, the *offline* state means that the *peer* is unable to receive messages ; any attempt at sending will result in a refusal from the message service.

3.2 JIM and JXTA

Figure 4 locates the JIM application in relation to JXTA, which provides several building blocks and APIs to build P2P applications. Among these ones, *pipes* are logical channels through which a *peer* can send data towards one or several *peers*. The communication between *peers* can be effectively done insofar the *peers* belong to the same *peerGroup* (which represents a set of *peers* in JXTA). Security and monitoring services of the JXTA core are not directly used by JIM, but are used by other JXTA elements in an implicit way.

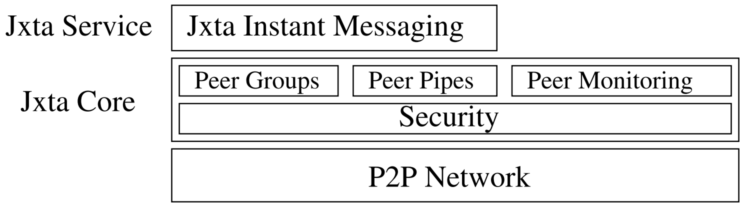


Fig. 4. JIM / JXTA Architecture

JIM uses JXTA to create a users *peerGroup* for the IM application and *pipes* to implement messages and presence services. Figure 5 illustrates the presence service mechanism; one can first notice that each *peer* contains the whole set of presence management elements. *Peer p1* owns a list of *peers*, called the *buddylist* connected to the JXTA *PeerGroup* created for JIM and uses a *pipe* towards each of them. The `PRESENCE SERVICE` is not delegated to a centralized persistent element. In this way, the operational model is the following : at regular intervals, the Poller sends a `PING` message to each *peer* belonging to the *buddylist*. Each *peer* replies to this message by issuing a `PONG` containing its state.

According to the IETF model, the *peer* sending such requests plays the role of `POLLER` towards the presence service that is represented by the reply of each *peer* contacted by the *pipe*. The model instantiation is represented in figure 5. Entities *Status Updater, Controler and User* are software parts of JIM that respectively plays the role of `WATCHER`, `PRESENCE SERVICE` and `PRESENTITY`. The *StatusControler* plays a double role because it deals with both `PONG` messages (`WATCHER`) and `PING` ones (`PRESENCE SERVICE`).

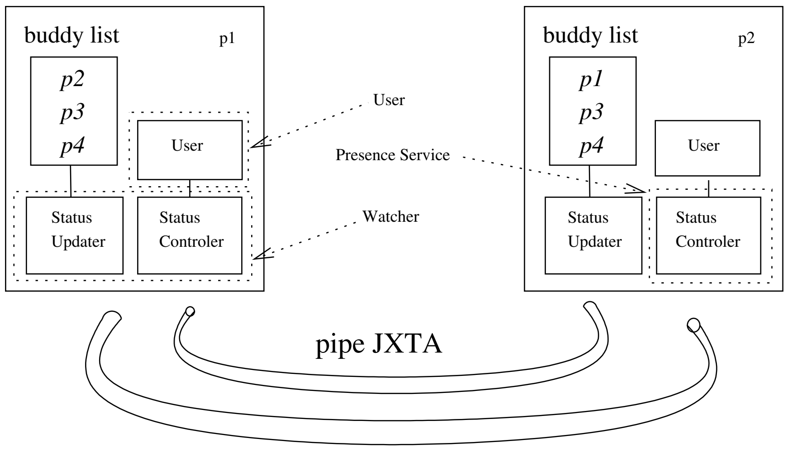


Fig. 5. JIM *peers* and JXTA *pipes*

4 Instrumentation

In order to measure the service presence performance, we chose to instrument the RTT of a `PING(Buddy)/PONG(BuddyStatus)` exchange. The collected result represents the time a *peer* has to wait to be notified of one buddy status. We use the ARM[3] (*Application Response Measurement*) java API and implementation to instrument these exchanges as transactions. Each transaction record will be registered in a management framework, namely the JMX (*Java Management eXtension*) java API.

4.1 Application Response Measurement

ARM is designed to evaluate service levels of single-systemss and distributed applications by measuring the availability and performance of transactions.

Figure 6 illustrates the ARM architecture. In order to carry out the measurement of a transaction, an application has to define transaction objects that best fit the required measurement. For example, some long-lasting transactions may have to be regularly updated while other ones may require additionnal metric informations. The entity responsible for creating and fostering transactions is the ARM Producer. The ARM agent is responsible for setting record objects properties for each finished measurement and communicating them to any ARM consumer that has subscribed to this type of measurement. Indeed, to catch the records of transactions, a consumer populates a pool with empty ARM record objects. Thus, when a transaction is finished, the ARM agent extracts an object from the pool, sets its different properties and pushes it in the queue. By calling a `get` method, the consumer can in its turn extract the record from the queue, exploit it and put it back in the pool for any further use.

[3] http://www.opengroup.org/tech/management/arm/

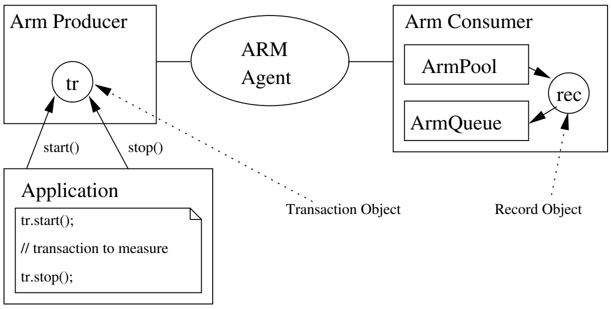


Fig. 6. The ARM architecture

4.2 Definition of a Transaction

We have defined a transaction as a `PING(Buddy)/PONG(BuddyStatus)` exchange between two buddies. Nevertheless, a simple response time value didn't provide us with enough information for our management objective. Thus we have used the metric definition feature of ARM to cope with this lack of information.

ARM allows the specialization of transactions by the use of metrics. A metric is a data that can provide an additionnal measurement (e.g. the size of a transaction) or any informational value (e.g. an indentifier). For our instrumentation, we have equipped our transactions with the following metrics :

– *the receiver identifier* : indicates the ping receiver;
– *the receiver status* : indicates the status of the ping receiver;
– *the transaction status* : is a boolean information that indicates if the measurement has stopped by receiving a PONG response or by being interrupted of a timeout signal managed by the ARM producer;
– *the transaction correlator* : uniquely identifies a PING/PONG exchange between two *peers*.

4.3 The Measurement Process

The way we measured the time spent for a PING/PONG exchange is described on figure 7. At first, in the source code of JIM we have add calls to `start` and `stop` methods respectively immediately before sending a PING message and immediately after receiving a PONG one. The `start` and `stop` methods are called on a Transaction Pool object that allow us to measure several transactions simultaneously. Moreover, the application cannot know which transaction object is actually started or stopped. When a transaction object has been started and stopped, for each subscribed consumer, the ARM agent extracts a free record object from the pool, sets the response time and the different metrics, and finally stores it in the queue. Considering the exploitation of the results, in the case of the CVS consumer, it gets all the record object properties, write it in a CSV (*Comma Separated Value*) file and put the record object back in the pool. In the

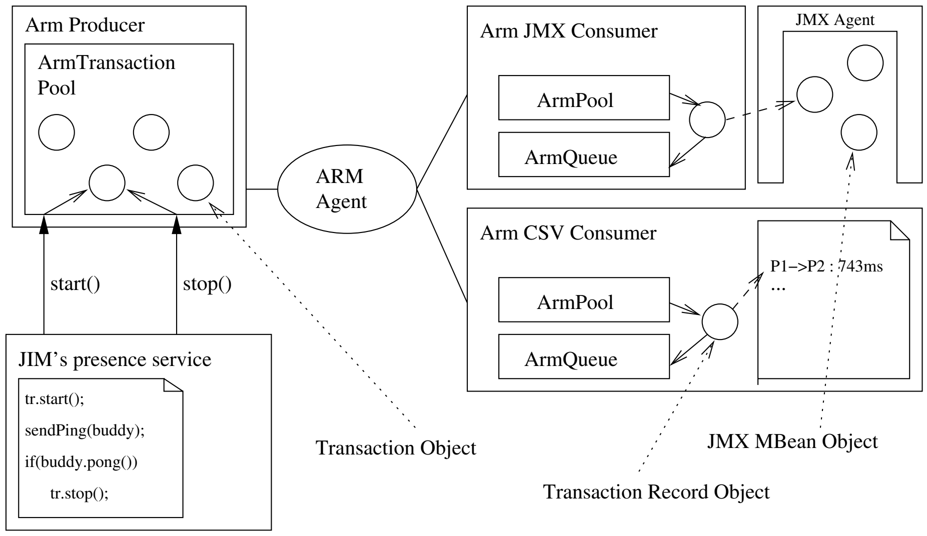


Fig. 7. The instrumentation process

other hand, in the case of the JMX consumer, for each record extracted from the queue, it creates a new standard MBean object containing all the record object properties and registers it in a JMX agent. These instances are accessible by contacting the agent (by using a HTML adaptor or a RMI connector provided with the JSR 160[4] JMX implementation, for example) or can be serialized to be transported in a JXTA service dedicated to the management (see [8]).

4.4 Correlation of the Measurement

In our instrumentation, we have to measure the time spent for the exchange of a PING / PONG message couple. Ping messages are sent every 10 seconds. PONG messages are received in an unbounded time. Thus, whenever we had used a single transaction we could not have had the certainty that a received PONG message was the one corresponding to the last PING emitted. We coped with this problem by (1) creating a transaction pool that contains enough transaction objects so that several PING can be sent simultaneously, and (2) provisioning each transaction and PING/PONG couple with a correlator that ensures that the receipt of a PONG message will stop the corresponding transaction.

4.5 Observed Results

Figure 8 show the response times of PING/PONG exchanges between two *buddies*. They have been obtained from a CSV file containing measurements on a

[4] http://www.jcp.org

100Mb Ethernet local area network. A *rendezvous peer* located outside the LAN (provided by the JXTA Developers community) was used. Our experiment consisted in measuring the response time in a permanent state, after a transient phase where *peers* discover themselves for the first time. Figure 8.a, shows the occurence of peaks that are due to the underlying P2P network. A instrumentation of JXTA will give us explanations about these peaks. Nevetheless, according to figure 8.b, we may assume that the response times present a certain regularity; the average notification time of the state of a *peer* to one of its buddy is about 850 ms with a standard deviation of 300 ms.

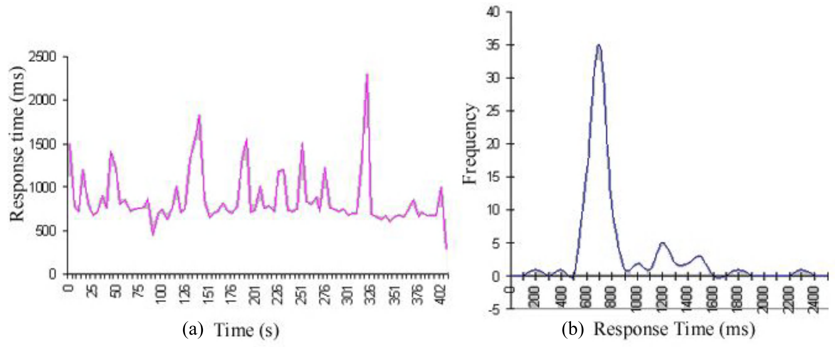


Fig. 8. Response time and distribution measurement

This first experiment was not to have a complete statistical study of the performances of a P2P application and the underlying virtual network (complete studies can be found in [5] and [4]).

4.6 Overhead Analysis

Instrumenting JIM has implied an overhead that we have always tried to minimize. Nevertheless, we have characterized it in terms of :

- *Network traffic* : Our instrumentation has required to add a correlator, represented by a unique long value, in the PING/PONG messages exchanged between two *peers*. The XML encoding specified by JIM has implied the definition of a new couple of tags, namely `<correlator>` and `</correlator>` for the insertion of our correlator in the messages. Thus, we may assume that, in terms of network traffic, the overhead introduced by our instrumentation doesn't exceed 50 bytes, which represents about 10 percent of the total size of a PING message.
- *Memory space* : the memory use of our instrumentation is due to the record of the results in a CSV file. Each result is about 150 bytes large. In this way, the file will grow at the speed of 60 ko per hour. This growth seems to be reasonnable. Moreover, further mechanisms of data compression, or filing

may be deployed in case a contiuous and very long time of use of JIM, but they are out of scope.
Now, considering the read-write memory overhead, our instrumentation increases the used memory space of the application of about 20 percent, with a total of 40 Mo. This huge overhead is due to the different threads running simultanously (ARM agent, Transaction producer, 2 transactions consumers and the JMX agent).
- *Processing time* : We have tried to have the simplest code as possible in terms of code algorithms and number of threads inducing the smallest overhead of processing time.

5 Integrating the Measurements in a Management Framework

We have chosen to use the JMX framework in order to integrate our measurement in a management architecture. JMX is a Java infrastructure that enable the management of Java applications. At the instrumentation level, JMX uses managed Beans (*MBeans*) as bases components of the architecture. A *MBean* represents a Java managed object that implements a specific interface. At the agent level, a JMX agent is container of *MBeans* and allow a remote application to access them by using a dedicated protocol adaptor or a connector.

5.1 Use of Record Objects as *MBeans*

In our experiment we have focused on the performance of the presence service of JIM and especially on the PING/PONG exchanges. Nevertheless, others aspects of JIM may require performance measurements implying the definition of multiples types of transactions and thus of *MBeans* interfaces. For this reason, we have chosen to use dynamic *MBeans* in our framework. It allows *MBean* objects to present a dynamic interface corresponding to the type of the measured transaction.

When launching JIM, we populate the pool of the JMX consumer with objects that inherit from `ArmRecord` and implement our dynamic *MBean* interface. When a record is present in the queue, we extract and register it in the JMX agent. Besides, when a record has to go back in the pool, we simplely unregister it from the JMX agent enabling it to be reused later.

5.2 The Naming Scheme

Each *MBean* object registered in a JMX agent is named according to a convention thats presents the following form :

```
Domain : attr1 = value1, attr2 = value2, ...
```

where `Domain`, `attr`*i* and `value`*i* are strings characters. In our instrumentation, we use the the following convention to uniquely indentify a *MBean* representing a transaction measurement :

- the domain identifier is the name of the *peer* currently running JIM;
- attributes and values are defined as:
  - dest = the name of the ping receiver;
  - corr = the value of the correlator used in the PING/PONG exchange.

6 Conclusions and Future Works

In this article, we have presented an instrumentation of a simple P2P application: the instant messaging. From an abstract model we have expressed how to measure the quality of a particular service used by such an application : the presence service. A well operating of this service ensures the quality of the application and needs to be managed. By registering measurement results in a JMX agent, we allow a global management application to manage the different *peers* that participate to the service. Our future works will consist in extending the definition of *peers* characteristics that need to be managed independantly of any application. We aim to define an abstract model for P2P services that enables us to design manageable *peers*. These ones will present a standard management interface that a centralized or distributed manager could use in order to manage a particular service. Finally, we plan to instantiate this model on JXTA and use JMX as a management framework.

References

1. P. Antoniadis and C. Courcoubetis. Market models for p2p content distribution. In *AP2PC'02*, July 2002.
2. M. Day, J. Rosenberg, and H. Sugano. A model for presence and instant messaging. RFC 2778 (Informational), IETF Network Working Group, February 2000.
3. H. deMeer and K. Tutschku. An application-level active networks based architecture for the performance management of peer-to-peer services. OPENSIG 2001 Workshop: Next Generation Network Programming, September 2001.
4. T. Hong. *in [7]*, chapter Performance, pages 203–241. Number 14.
5. R. Matei. Peer-to-peer architecture case study: Gnutella network. http://people.cs.uchicago.edu/m̃atei/PAPERS/gnutella-rc.pdf.
6. Dejan S. Milojicic, Vana Kalogeraki, Rajan Lukose, Kiran Nagaraja, Jim Pruyne, Bruno Richard, Sami Rollins, and Zhichen Xu. Peer-to-peer computing. Technical Report HPL-2002-57, HP Laboratories, Palo Alto, March 2002.
7. Andy Oram, editor. *Peer-to-peer: Harnessing the power of Disruptive Technology.* O'Reilly & Associates, 2001.
8. Radu State and Olivier Festor. A management platform over a peer to peer service infrastructure. In *8 th. IEEE Intl. Conference on Telecommunications*, Tahiti French Polynesia, February 24 March 1 2003.

A Remote Resource Monitoring Approach in Active Networks

Yuhong Li, Lars Wolf, and Fangming Xu

Institute of Operating Systems and Computer Networks
Braunschweig University of Technology
38106 Braunschweig, Germany
{li,wolf}@ibr.cs.tu-bs.de, f.xu@tu-bs.de

Abstract. Monitoring the resource status in active network nodes is important both for the network administrator to maintain and deploy active networks and for the normal end-users to acquire better performance. Hence, a method that can monitor the resource status at a remote node is needed. Moreover, due to the complexity of the resource usage in active networks, the corresponding method should be able to monitor the status of various resources at remote node dynamically and in a user-specified way. This paper presents the design and implementation of such a resource monitoring method. The method adopts the active application technique to obtain the desired resource status information at a remote network node and transfer them back to the user in the user-specified way. It also allows the user to control the content, the procedure and the precise of the monitoring.

1 Introduction

In active networks, the resource status information at an active node is important for both the end-users and the network administrators. Active applications use the networks more aggressively than the simple applications in the traditional passive networks, for example, they need various types of resources to inject and execute their programs in the networks. Therefore their performance becomes more sensitive to the network conditions, such as the available resources at the network nodes. In order to acquire better performance, these applications may need sufficient information about the resource status within the networks. For the network administrators, for instance, if they have enough information about the different types of resources at the concerned network nodes, they can decide dynamically when, where and what kind of services can or cannot be executed, so that the congestion and scarcity of a specific type of resources in a certain time can be avoided. In addition, the resource status may be also beneficial for the provision of new services, such as routing at an active node according to the resource status at the neighbor nodes etc. Hence, a method that can monitor the resource status at a remote node is necessary in active networks.

Due to the complexity of the resource usage at active nodes, e.g., more types of resources are needed by an application simultaneously, it is difficult to gather, describe and express the resource status information. Moreover, different end-users and network administrators are interested in different resource status information at different time. This requires that no constraints should be employed on the content and format of the resource information needed to be observed using this monitoring method. Furthermore, users should also be able to control the procedure of the monitoring, e.g, to specify or change the content of the information desired to be observed or the precise of the monitoring at run time.

Some efforts have been made in the traditional IP-based networks to gather information in the networks, such as using the implicit network feedback (e.g., the packet loss rate), the well-known traceroute program and the SNMP (Simple Network Management Protocol). However, they are not satisfying because of limitations concerning the amount, content or authority of the information. Some systems designed for collecting the network information, such as the Globus [8], Prophet [18] and NWS [20], can still not satisfy the need of the complex and intelligent users more or less. We will consider these methods in more detail in section 5.

This paper presents a new method to monitor the resource status at a remote network node. Different types of resources, such as computation, memory and network bandwidth are taken into account. The system resources at different levels, as well as the resources consumed by each active application at a remote network node, can be observed dynamically. Moreover, users can also specify the content of the information needed to be monitored and control the procedure of monitoring. The remainder of the paper is organized as follows: section 2 analyses the requirements to the resource monitoring approach in detail and introduces our implementation mechanism. Then in section 3, the design and implementation of our mechanism are presented, including the active node architecture and the RemoteMonitor application. Following this in section 4, the evaluation of the method is given. Section 5 discusses some related work and finally a summary is made in section 6.

2 Requirements to the Resource Monitoring Approach

In active networks, users, including the normal end-users and network administrators may have the following wishes with regard to the resource monitoring.

1. Resources can be monitored remotely and dynamically. As introduced above, a user needs to be able to monitor the resource status at a network node for the purpose of performance and management. Moreover, due to the continual arrival and departure of applications at the network node, as well as the start and end of other applications and services inside the network node system, the resource status of the node varies permanently. Hence, the users should be able to acquire the resource information at any remote network node and at any time point as they want.
2. Resources can be monitored in a user-specified way. This involves two meanings. One is the content of the resource information needed to be monitored. Different users may care about different resource information. Even for the same user, he may be interested in different resource information at different times. The other is the procedure of the monitoring. A user may also want to control the procedure of the resource monitoring at runtime. One example is, at the beginning the network administrator wants to observe the resource status at a network node every 30 seconds, after some time, when he believes that the network situation is stable, he

may want to change the interval to 2 minutes. He can also decide to suspend the monitoring for 20 minutes and then resume the monitoring.

3. Observing multiple types of resources. Because of the introduction of application-specific computation into the networks, multiple types of resources, such as computation, memory and network bandwidth, play an important role for applications at the active nodes. Therefore, a resource monitoring approach should be able to supervise multiple types of resources simultaneously.
4. Observing at fine granularity. A user should be able to observe the resource status not only at the system level, e.g., resources available in the node system, but also at application level, e.g., resources consumed by a single active application.

Considering the first two requirements, we adopt the active application mechanism to monitor the resource status at a remote node. Active networks allow applications to carry their own programs in the packets to the network nodes and maintain the infrastructure to load and execute the programs. Therefore they provide the possibility for users to specify what kind of resource information they want and how to get the information. Moreover, since the parameters to control the execution of the program can be sent to the nodes in separate packets during the execution of the programs, the procedure to obtain the resource status at the remote nodes and the way to deliver them back to the users can be controlled in a user-specified way.

The last two requirements need help from the underlying active nodes. Some basic functions and services regarding the resource status in the system should be provided by the nodes, just like other normal network services. Since normally users are not permitted to access the resource status at the network nodes directly for the safety and security reasons. Through invoking these basic functions and services, users can obtain the needed information about the resource status without threatening the system's security.

In the following we describe our implementation of the resource monitoring method, beginning with the architecture of the active nodes, which can provide the basic functions related to the resource status to the end-users.

3 Implementation

3.1 An Active Node Architecture

Basically, an active network consists of a group of network nodes. Some of them own a mechanism for loading and executing user-specific programs, these nodes are called active nodes. The general architecture for active nodes [5] consists of three components: node operating system (NodeOS), execution environment (EE) and active applications (AA). The NodeOS [2] can access all the node's communication, memory and computation resources and provides some basic services and functions for the applications from end-users. The EE acts as an interface between applications and the NodeOS. It provides an execution environment for the applications. The basic services and functions provided by the NodeOS can only be invoked by the active applications through the EE.

Fig.1 illustrates the active node architecture (ANwithRM) providing some resource query functions. It follows the general three-component active node architecture. Janos [17] and ANTS2.0 [3] have been selected as the basis of the NodeOS and EE, respectively. The most important reasons are that Janos provides some basic resource control functions and ANTS2.0 bases on Janos and takes also the resource control functions from Janos. A resource management subsystem with three modules is integrated in the NodeOS part. Among them, the resource management (RM) module processes all the system resource information and therefore is responsible for providing all the functions regarding the resource status information.

The basic functions regarding resource query provided by the RM module includes appsResUsed(), anNodeResUsed(), and osResAvailable(), osResUsed() etc. Applications from the users can invoke these functions through the EE. Currently, only the three major types of resources, i.e., the network bandwidth, memory and computation, are considered in our system. After invoking these functions, resource status about all the three types of resources are returned. The osResUsed() returns the total resources consumed in the underlying operating systems, and the osResAvailable() returns the resources still available in the whole system. The anNodeResUsed() returns the total amount of resources used by the ANwithRM system, including the system costs and the resources consumed by all the active applications running in the ANwithRM. And the appsResUsed() returns a list of all the active applications running within the ANwithRM, together with the amount of resources consumed by each application. Application's programmers can invoke these functions to acquire the corresponding resource information and select the desired information that should be sent back to the user through programming.

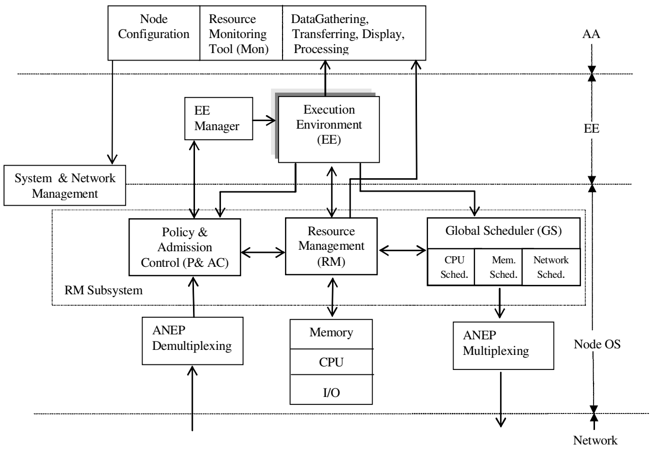


Fig. 1. The active node architecture with resource management

In our implementation, the resources consumed by an active application running within the ANwithRM consists of two parts, one is those consumed by the node system on behalf of the application, such as the Janos and ANTS threads. The other is the resources consumed by the threads initiated due to the execution of the application from the user. The information about the former can be acquired by the RM module with the help of the NodeOS and the underlying operating system. The information about the later can only be acquired through accounting, namely in order to be able to provide resource consumption information about each active application running in the node system, the RM module has to count the various resource consumptions during the running of the applications in the node system.

The RM module uses the bytecode rewriting and a few native code methods to acquire the various resource statuses. Bytecode rewriting is used to signal the allocation and deallocation of memories, the initiation of application's threads, and when needed, some native codes are used to record the amount of the resources.

Note that in Fig.1, we illustrate the resource monitoring tool in the AA part. In practice, the program code for this application, with the functions of, e.g., resource information gathering, transferring, display and processing, can be injected by the end-users, as explained in the next section. In addition, providing basic resource query services to users is only a part of the functions of the node architecture introduced here. The node architecture can also realize some resource management functions such as admission control. More details about this architecture can be found in [12].

3.2 Remote Monitor Application

3.2.1 Principle of the Application

We have developed the RemoteMonitor application to monitor the resource usage at a remote active node. In order to transfer the user-specific programs and control parameters as well as the resource status data, following types of packets are designed with regard to this application.

MonitorStart: this is the first packet to ask for the remote resource monitoring. It carries the program code, which tells the remote node what kinds of resources are desired and how to acquire them. The program can also process all the packet related to the RemoteMonitor application. In addition, this packet carries also a dynamic precise parameter, namely a value denoting the time interval, at which the resource status information at the remote node should be inquired and sent back periodically to the user.

When this packet arrives at a network node, an execution environment (EE) is created for the RemoteMonitor application, and the program code carried is loaded into the execution environment. Then the program begins to run to gather the specified information. Fig.2 illustrates part of the program code. In this example, the user wants to obtain the resources consumed by the whole ANwithRM system and the resources consumed by the first active application running within it. Here RV stands for resource-vector, it presents all types of resources concerned in the node system. Note that we have adopted the general methods for developing active applications using ANTS [19], which is the basis of our EE.

ResourceData: this packet carries the resource status data. As the result of the execution of the program, the needed data about the resource status are acquired. The data are sent back to the user in the ResourceData packet. In order to be able to observe the resource information in "real-time", we send back a ResourceData packet immediately after the resource information is acquired in our example. In practice, users may also select to send the resource information gathered in several time points together with the according time stamp in one ResourceData packet.

MonitorSuspend/MonitorResume: since a user may not be interested in the resource status at a remote node for some time, we have designed the

```
public boolean evaluate(Node n) { //n is the node where the packet arrives
   if ( n.address()!= concernedNodeAddr() ) {
      return n.routeForNextNode(getDst());
   }     // ensure that only the information at the desired node are collected

   else if (n.address()==concernedNodeAddr ) {
      while (!STOP) {
         sendResourceDataCapsule(n, destination);
                // destination is equal to the source address from where
                // the MonitorStart capsule is received
         sleep(INTERVAL);
      }
   }
}

public void sendResourceDataCapsule(Node n) {
    ResourceDataCapsule cap= new ResourceDataCapsule()
    RV r1= n.anNodeResUsed();
    RV r2=n.appsResUsed().elementAt(0);
    cap.pack(r1);
    cap.pack(r2);
    n.send(cap);
}
```

Fig. 2. Program code carried in the MonitorStart packet

MonitorSuspend and MonitorResume packet to the application. After receiving the MonitorSuspend packet, the program stops gathering and transferring the status information until the MinitorResume packet is received. This action can reduce the resource overhead caused by the RemoteMonitor application itself.

Note that using the MonitorSuspend/Resume, the user may, e.g., store the acquired data in the same file for analysis after the resumption. In addition, the resume procedure is faster than to start the application again. However, since the RemoteMonitor application does not end, the resources it occupied at the remote node are not released.

FrequencyChange: after the program carried in the MonitorStart packet begins to run at the remote node, the user receives the ResourceData packet periodically from the remote node. Obviously, the smaller the interval between the two ResourceData is, the more precise can the resource status at the remote node be reflected at the user. We call this time interval the monitoring precise. For some reasons, the user may want to change the monitoring precise. E.g., the network administrator may want to lower the monitoring precise when he thinks that the resource status at a network node is stable enough. Therefore we designed the FrequencyChange packet. After receiving this packet, the program continues to gather and deliver the resource status data according to the new frequency. Note that if the program is already suspended, the program will record this new value and begin to use it when the program is resumed. The monitoring precise parameter has also been integrated in other packets such as the MonitorStart and MonitorSuspend/Resume packet.

MonitorStop: this packet stops the execution of the program at the remote node. As the result, no more resource status data will be transferred to the user.

Fig.3 illustrates the general format of these packets. We have defined a ConcernedNode option field in the ANEP header [1] to stress the addresses of the nodes, where the resource information should be monitored. The program code and data are the payload of the ANTS packet [19]. The TimeStamp field is used in the ResourceData packet to indicate the time when the status data are gathered. This is helpful to recovering the resource statuses. The program code is used only in the

MonitorStart packet to specify the content of the information to be monitored and how the information can be acquired and sent back to the user.

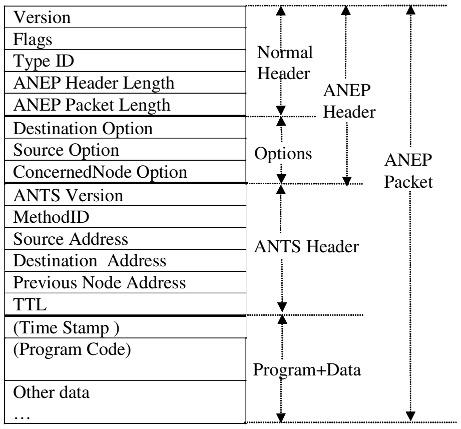


Fig. 3. General packet format

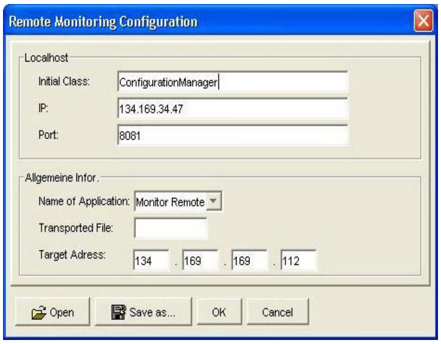
Note that here the program code and the data to be processed are carried in the same packet. We called this mechanism the associated code loading. ANTS [19] adopts an on-demand code loading mechanism to propagate code needed to process a packet in the networks. I.e., the program code (called method in ANTS) for processing various packets related to an application is sent separately from the packets. When a node receives a packet, it checks if the method needed for processing it is already available in the node according to the method identifier carried in the packet. If yes, the method is invoked and the packet is processed. If no, the packet is first saved in a waiting queue, and the node asks for the corresponding method from the previous node using a simple code loading protocol. After the needed method is received and cached, the queued packet will be processed. This mechanism limits the distribution of the code to where it is needed. However, the startup performance of this mechanism is relative poor. There exists a relative long delay until the first packet of an application is processed if the program code has not been cached in the node to be monitored. We have added an associated code loading mechanism to the ANTS EE. It allows each packet to carry the program code needed to be executed in the network nodes together with other data, and therefore omits the separate code loading procedure. This mechanism is suitable for short programs (all the code can be carried in one capsule). The startup performance comparison between the two code-loading methods can be found in [11].

Since the active application mechanism is used to monitor the resource information and the content and format of the information needed to be monitored are specified by the program code, to some extent, this method seems more complicated for users than the methods that need only to specify that users want to monitor some network information. However, due to the flexibility, the method is more applicable to the situation, where the user-specific information needed to be monitored, e.g., the information not defined by the SNMP[6] etc.

Furthermore, in our current implementation, besides invoking the resource query functions, the RemoteMonitor application acts just like other normal active applications. That means the application may be rejected if the active nodes implement an admission control mechanism and there are not enough resources for this application. In practice, according to the implementation of the active nodes, the resources consumed by the RemoteMonitor application can either be considered as the system resource cost and therefore already reserved by the system node, or the application can be given a suitable priority, such that it has a certain possibility to be accepted by the active nodes.

3.2.2 User-Interface

Besides injecting code into the remote node to instruct it to collect the needed resource information, the RemoteMonitor application has also a graphical user-interface at the local system. Through this interface, users can configure the execution of the RemoteMonitor application and the acquired resource data can be displayed graphically in time or stored in files for later analysis and statistics. The main functions of this interface include:


Remote Monitoring Configuration

Localhost

Initial Class: ConfigurationManager

IP: 134.169.34.47

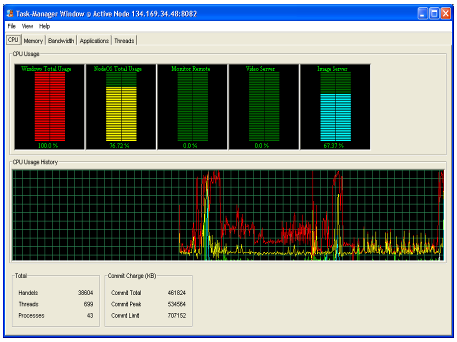
Port: 8081

Allgemeine Infor.

Name of Application: Monitor Remote

Transported File:

Target Adress: 134 . 169 . 169 . 112


Task-Manager Window @ Active Node 134.169.34.48:8082

Windows Total Usage 100.0 %; NodeOS Total Usage 76.72 %; Monitor Remote 0.0 %; Video Server 0.0 %; Image Server 67.37 %

Total	
Handels	38604
Threads	699
Processes	43

Commit Charge (KB)	
Commit Total	461824
Commit Peak	534564
Commt Limit	707152

Fig. 4. User-application interface **Fig. 5.** Display resources dynamically

- Configure and start the RemoteMonitor application. E.g., to specify the address of the node where the resource should be monitored. Fig.4 illustrates this window. Since the RemoteMonitor is an active application, it runs atop an active node. If the active node ANwithRM is not yet running, it can be started together with the application. After the application begins, a MonitorStart packet will be sent to the remote node
- Send different control packets to the remote node, such as MonitorSuspend/ Resume, FrequencyChange, during the resource monitoring.
- Display the resource status information received from the remote node in time. After the ResourceStatus packet is received, the time stamp and the resource data are unpacked, passed to the interface, and displayed graphically in time, as shown in Fig.5. Users can also select which type of resources they want to observe.
- Store and process the received data. The data displayed in Fig.5 can also be saved as XML and Java Object file in order to be processed or analyzed later.

4 Evaluation

4.1 Test Examples

To evaluate our resource monitoring approach, we have used two machines in the same city to construct a small active network; both run our implementation of the active node architecture. One acts as the end-user and has the IP address of 134.169.34.47. The other acts as the remote active node with the address of 134.169.169.112. The end-user starts the RemoteMonitor application by sending the MonitorStart packets to the destination 134.169.34.112. In order to be able to illustrate the results clearly, we do not run any other active applications at the remote node.

In this example, the user wanted to observe the resource usage of the underlying operating system, the ANwithRM, as well as the RemoteMonitor application itself. At the beginning, i.e., in the MonitorStart packet, he specified a precise of one second, namely the resource status at the remote node should be checked and transferred back every one second. After two minutes, the user suspended the program for one minute and then resumed it. At the same time, the precise was changed to two seconds. After two minutes, the suspending and resumption procedure was repeated and the monitoring precise was set to four seconds.

To compare the resource status data observed remotely and locally, we implemented also a simple local application, which inquired directly the local resource status every 1s and display the results on time. Since the CPU usages vary relative heavily, we illustrate only the CPU usages at the remote active node (134.169.169.112) in Fig.6 and Fig.7, which are observed locally at the node 134.169.169.112 and by the user remotely at the node 134.169.34.47, respectively. From these figures we can see that when the monitoring precise is high, the resource usage at a remote node can be well reflected at the user. Otherwise the dynamic variation can be observed roughly. Note that there is still a little difference between Fig.6 and Fig.7, this is because the sampling points of the data displayed on the figures are a little different, they are not strictly synchronized.

In the next sections, we evaluate the monitoring method in the context of delay and the resource costs caused by the method itself.

4.2 Delay

We have studied two types of delay concerning the resource monitoring method. The first is the starting delay, i.e., the time interval between the MonitorStart packet is sent and the first ResourceData packet is received. The average value of tests for 5 times is 6269 ms. The value includes the transmission delay, i.e., from the user to the network node and back to the user, and the time used for loading and executing the RemoteMonitor program in the network node. The second is the observation delay, i.e., the time interval between the resource information is acquired at the remote node and when it is displayed locally, namely the time when the ResourceData packet is received and unpacked at the end-user. This value includes the transmission delay from the remote node to the user and the processing time at the remote node and at the

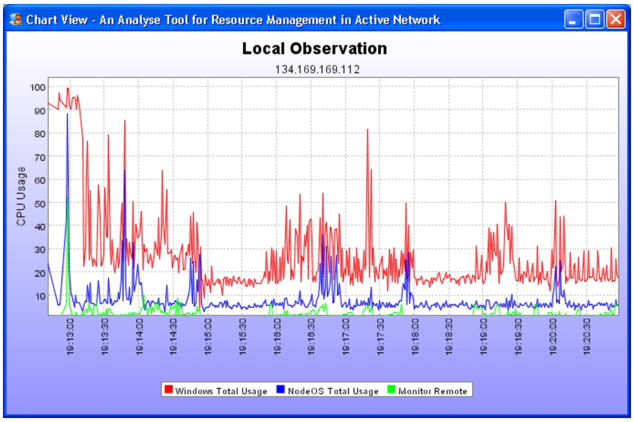


Fig. 6. CPU usages observed locally (in the node 134.169.169.112)

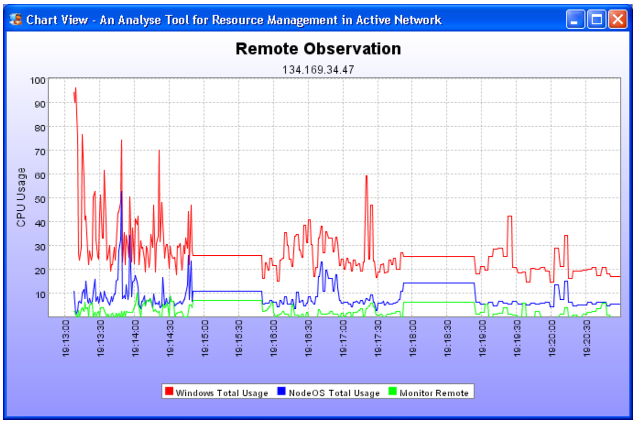


Fig. 7. CPU usages observed by the user remotely (in the node 134.169.34.47)

user, such as the resource data is packed at the remote node and unpacked at the local node. In each of our test, 210 ResourceData packets are received by the local user as the result of the 6 minutes monitoring with the frequency of 1s, 2s and 4s respectively. The average value of the delay of these 210 packets is 21ms after 5 times test. Now we are doing the tests and study these values in the wide area.

4.3 Monitor Precise vs. Resource Cost

Since we adopt the active application mechanism to realize the remote resource monitoring, the method itself has some resource cost. Generally, the cost of the three main types of resources depends on the monitoring precise, i.e., the frequency that the resource status data are gathered and transferred. The network bandwidth cost depends mainly on the length of the ResourceData packet and the number of the packets sent per unit time, because the length and number of other types of packets are relative small and fixed. The length of each ResourceData packet varies with the content of the resources that end-users want to monitor. And the number of the packets depends on the monitoring precise. In our case, i.e., the user wants to monitor the resource usage of the underlying operating system, the ANwithRM and there is only one active application running in the ANwithRM, and the required dynamic monitoring precise is 1 second for 2 minutes, 2 seconds for 2 minutes and 4 seconds for 2 minutes, and together with 2 minutes suspend, the average network bandwidth is 765bit/s. The total computation cost consists of the time used for the load and execution of the RemoteMonitor code at the remote node for gathering and transferring the resource status data, as well as for processing the controlling packets, such as the MonitorSuspend/Resume and FrequencyChange. It is mainly related to the monitoring precise. The lowest curve in Fig.6 illustrates the computation cost of our example. The memory is mainly used for the buffers for packet receiving and sending, thread stacks, heap memory for objects during the program code execution and heap memories for "soft states", such as inside communication etc. In our example, the average memory usage by the RemoteMonitor application at the remote node is 4.3MB.

5 Related Work

Because of the importance of the network information, several methods related to the monitoring and measurement of network information have been developed in the traditional IP-based networks. [10] presents an architecture to monitor the network resource. It consists of a sensor director, several network sensors and a database. The sensor director initiates the collection of data by the network sensors in response to requests from the resource manager. To some extent, each user in our method acts as both the resource manager and the sensor director, and the program sent by the user and executing at the remote node has the same function as the network sensors. However, since our method benefits from the active networks, it is much simpler and more flexible than the approach presented in [10]. Moreover, our method permits any user in the networks to monitor the resources at any other node in the networks. Remos [7][13][14] is a system specially designed for collecting different kinds of information in networks. It uses different methods such as SNMP [6] and benchmarks to collect network information and provides a standard interface format independent of the details of any particular type of network. One important characteristic of this system is that it provides a query-based interface for its applications and allows them to specify what kind of information they need. Our approach can also provide this function: users can specify what kind of information on which nodes they want. However, the Remos system provides the collected information only in the format of a logical topology. Though it is more attractive and direct than other methods such as listing bandwidth or latency for each pair of endpoints, however, to some extent, it cannot meet the needs of all users because different applications have different requirements. In this point, our approach is more flexible: the user can specify the desired format of the network information in the program code sent to the network. NWS [20] and Prophet [18] provide applications with benchmark-based predictions by sending messages to make measurements between pairs of computation nodes.

These methods concentrated on the predictive accuracy and forecast of the performance of available resources but pay little attention to the requirements of the users. Globus [8] and Legion [9] provide a wide range of functions such as resource location and reservation, authentication and remote process creation mechanisms, but they do not focus on supporting application level access to network status information. Compared with these works, our method is more flexible. It allows the users to specify both the content and the format of the information they need. The users can also observe and process the information in their specified way and in time.

In recent years active networks have gained a rapid development [15]. Many works related to the enabling techniques, such as platforms, languages as well as reliability etc., have been done in order to implement and integrate active networks into the current networks. On the contrary, works related to the applications in the active networks are relative few. The existing applications, such as congestion control [4], network management [16], multicasting and caching, are just to use the idea of the active network technology to solve some hard problems in the network. From this point of view, our work presents an application in the active networks.

6 Summary

This paper suggests a method for monitoring the resource status at remote active nodes. The active application technique is used during the gathering and transferring the resource status data, in order to satisfy the various resource monitoring requirements from users. The RemoteMonitor application has been developed, which maintains different types of packets to notify the remote nodes what kinds of resource data are needed by the user, how to acquire and transfer them back to the user etc. Moreover, the application also allows users to control the procedure of the monitoring, such as specifying the monitoring precise, suspending and resuming the monitoring. In addition, the active node architecture that implements this method has been presented, including the consisting modules and basic functions regarding resource information query. Some initial test results about this method have been given. Now tests related to this method in the wide area are being performed.

References

[1] D.Alexander, B.Braden, C.A.Gunter, A.W.Jackson, A.D.Keromytis, G.J.Minden and D.Wetherall, "Active Network Encapsulation Protocol (ANEP)", July 1997.

[2] AN Node OS Working Group. Node OS Interface Specification. Jan. 10, 2001.

[3] http://www.cs.washington.edu/research/networking/ants/

[4] S.Bhattacharjee, K.Calvert and E. Zegura, "Congestion Control and Caching in CANES". In ICC'98, 1998.

[5] K.Calvert et al. "Architectural Framework for Active Networks", available from http://www.dcs.uky.edu/~calvert/arch-docs.html.

[6] J.Case, K.McCloghrie, M.Rose and S.Waldbusser. "Structure of management information for version 2 of the simple network management protocol (SNMPv2)". RFC1902, Jan. 1996.

[7] Tony Dewitt, Thomas Gross, Bruce Lowekamp, Nancy Miller, Peter Steenkiste and Jaspal Subhlok, "ReMos: A Resource Monitoring System for Network Aware Applications", Tech. Rep. CMU-CS-97-194, Carnegie Mellon University, December,1997.
[8] I.Foster and C.Kesselman, "Globus: A Metacomputing Infrastructure Toolkit", International Journal on Supercomputing Applications, vol.11, no.2, 1997.
[9] A.Grimshaw, W.Wulf and Legion Team, "The Legion vision of a worldwide virtual computer". Communications of the ACM. Jan. 1997, Vol. 40, No. 1.
[10] Philip M. Irey IV, Robert W. Hott, and David T. Marlow, "An Architecture for Network Resource Monitoring in a Distributed Environment". Lecture Notes in Computer Science 1388, Springer-Verlag, 1998. Pages 1153–1163.
[11] Yuhong Li and Lars Wolf, "Collection of Network Information in Active Networks", Operating Systems Review 35(4): 39–49 (2001).
[12] Y.Li and L.Wolf, "An Active Network Node System with Adaptive Resource Management". In Proc. of ICT2002, Beijing, China.
[13] B.Lowekamp, N.Miller, D.Sutherland, T.Gross, P.Steenkiste and J.Subhlok "A Resource Query Interface for Network-Aware Applications", proceeding of IEEE symposium on High-Performance Distributed Computing, July, 1998.
[14] Nancy Miller and Peter Steenkiste, "Collecting Network Status Information for Network-Aware Applications", IINFOCOM 2000.
[15] Konstantinos Psounis, "Active Networks: Applications, Security, Safety, and Architectures", IEEE Communications Surveys, First Quarter 1999.
[16] D.Raz and Y.Shavitt, "Active Networks for Efficient Distributed Network Management", IEEE Communications Magazine, Mar. 2000.
[17] P.Tullmann, M.Hibler and J.Lepreau. "Janos: A Java-oriented OS for Active Network Nodes". IEEE Journal on Selected Areas of Communication. Vol.19, No.3, Mar.2001.
[18] J.Weissman and X.Zhao, "Scheduling Parallel Applications in Distributed Networks". Cluster Computing, vol.1, No.1, pp. 95–108, May, 1998.
[19] D.Wetherall, J.Guttag and D.Tennenhouse, "ANTS: A Toolkit for Building and Dynamically Deploying Network Protocols", IEEE OPENARCH'98, San Francisco, Apr. 1998.
[20] R.Wolski, N.Spring and C.Peterson, "Implementing a performance forecasting system for metacomputing: The network weather service", Tech.Rep.TR-CS97-540, University of San Diego, May 1997.

SCAMPI: A Scalable and Programmable Architecture for Monitoring Gigabit Networks

Jan Coppens[1]*, Steven Van den Berghe[1], Herbert Bos[2],
Evangelos P. Markatos[3], Filip De Turck[1], Arne Oslebo[4], and Sven Ubik[5]

[1] INTEC, Ghent University, Gent, Belgium.
[2] LIACS, Leiden University, Leiden, The Netherlands.
[3] ICS FORTH, Heraklion, Greece.
[4] UNINETT, Trondheim, Norway.
[5] CESNET, Prague, Czech Republic.

Abstract. Effective network monitoring is vital for the growing number of control and management applications typically found in present-day networks. Increasing link speeds and the diversity of monitoring applications' needs have exposed severe limitations of existing monitoring techniques. As a response, the EU IST SCAMPI project designs and implements a scalable and programmable architecture for monitoring multi-gigabit networks. The SCAMPI architecture has an expressive programming interface, uses intelligent hardware, provides user policy management and resource control, and achieves scalability through parallelism. This paper addresses the problems with current high-speed network monitoring and presents the system architecture and components of the SCAMPI platform.

Keywords: High-speed Network Monitoring, Programmable Monitoring, Open Monitoring Platform.

1 Introduction

With the widespread use and deployment of the Internet, traffic monitoring has been increasingly used as the mechanism to improve the performance and security of computer networks. In modern networks, different kinds of users would like to monitor different aspects of the network traffic for different purposes, e.g. traffic engineering, intrusion detection, denial of service detection, monitoring of service level agreements (SLAs), charging, etc. Some of these applications require aggregate traffic statistics only, while others may be interested in monitoring each and every byte that travels through the network.

Existing tools tend to focus on a single subclass of network monitoring problems, while ignoring the others. Each of these monitoring tools is generally based on a specific set of primitives and functions, different from the ones used in other

* Corresponding Author: `Jan.Coppens@intec.UGent.be`; Sint-Pietersnieuwstraat 41, 9000 Gent, Belgium.

tools. To make matters worse, commercial vendors frequently use their own libraries and standards.

Besides the plethora of incompatible monitoring tools, current solutions are finding it increasingly difficult to keep up with modern link rates. As networks get faster and traffic analysis more complex, network monitoring gets slower. Monitoring techniques that used to be effective at low speeds (10 to 100Mbit) are becoming less and less useful when applied to current high-speed backbones [8].

The European Union IST project known as *SCAMPI* [13] addresses both of these problems by developing an affordable, high-performance and open network monitoring platform. SCAMPI started in April 2001 with the aim to implement all the necessary infrastructure to monitor networks at speeds exceeding 10 Gbps, including: (1) hardware (NIC), (2) middleware, (3) monitoring API or *MAPI*, and (4) applications that validate the SCAMPI approach. The key ideas behind SCAMPI are that programmable hardware is used to thin the datastream feeding into the host processor, while the middleware offers support for running multiple high-speed monitoring applications simultaneously (including resource control and policy management), and the MAPI offers a standardised API to applications that is much more expressive than what is offered by existing solutions. The MAPI, definition and implementation, is one of the main contributions of the project. While simple enough to support most existing network monitoring tools without having to change the code (for example, we offer a `libpcap` interface on top of the MAPI), it is expressive enough to allow applications to specify advanced queries on the data streams (e.g. 'scan all packets in a flow for a signature and return the packet if there is a match'). To our knowledge, such a query is impossible to express in existing APIs. The applications that will be implemented in SCAMPI are derived from real-world problems and chosen so as to cover a wide range of measuring requirements (e.g. traffic sampling, header capture, full payload capture of every packet, etc.).

The SCAMPI platform can be used by Internet Service Providers (ISPs) and Application Service Providers (ASPs), to provide for instance more advanced billing mechanisms. At the same time, it may improve the end-user experience through better network performance achieved by informed network management and traffic engineering. It also allows administrators to enhance the security of computer systems connected to the Internet by providing a better defence against cyberattacks, such as Denial-of-Service attacks and Intrusion attempts [2].

The remainder of this paper describes the problems of current monitoring architectures and the solution provided by SCAMPI. Section 2 addresses high-speed network monitoring and the hard- and software limitations when dealing with these kind of speeds. In section 3 the overall SCAMPI architecture is presented, while section 4 describes individual components such as the MAPI, the Module Organizer and the Job Control subsystem. After the Related work in section 5, a conclusion is presented in section 6.

2 High-Speed Network Monitoring

Monitoring is complicated by the fact that network speed increases at a rate that exceeds Moore's Law. For example, some research suggests that bandwidth doubles roughly every 9-18 months [3], an observation which is usually referred to as "Gilder's Law". At the same time, network monitoring applications tend to become more complex and demanding. Where early monitoring applications commonly required little information from the network (e.g. aggregated traffic or statistics), more recent tools may need a much more significant amount of information, possibly including both header and payload for each and every packet. Worse still, the amount of processing needed on this data tends to increase. For example, for detecting Internet worms and various other forms of cyberattacks, we may need such computationally intensive processing as string matching on the entire payload.

2.1 Hard- and Software Optimizations

In order to monitor gigabit networks, we have to overcome the hardware limitations of current monitoring devices. In PC-based monitoring platforms, captured packets are sent over the PCI bus to main memory. Present-day PCI buses are 64 bits wide and run at 66Mhz, yielding a maximum throughput of 0.5 Gbps at best, which is way too little to monitor gigabit backbones. Even though new PCI standards are being developed (e.g PCI-133 and PCI-533, designed for a maximum of roughly 1 Gbps and 4.3 Gbps, respectively), the latter one is currently not commonly available [1]. In fact, it is doubtful whether PCI-X 533 will be widely available before the year 2005, by which time high-speed link rates may well have increased to 40 Gbps and beyond. Even ignoring the bus speed, the memory is another bottleneck, which is unable to keep up with the growth of link speed.

The obvious solution to this problem is to limit the number of packets and bytes that are transferred to the main CPU. By using configurable hardware devices or network processors (e.g. Intel IXPs [12]), the captured data that is not of interest to any of the applications can already be discarded in the hardware itself. To reduce the network flow even further, techniques such as packet sampling, counting and calculation of statistics can also be applied in the hardware. SCAMPI therefore aims to provide optimizations by configuring the hardware on behalf of the applications' requests.

In environments without such programmable hardware (e.g. lower-speed networks with traditional network cards), SCAMPI is still useful, albeit less efficiently. Note this is not a problem for lower-speed networks, as these are not capable of saturating the bus and memory bandwidth anyway. It is important to realise, however, that the SCAMPI programmable hardware may take many different forms and may not look like network cards at all. For example, Juniper routers allow for traffic filtering based on header fields [15]. The filtered traffic, in turn, can be mirrored to a router's port, on which we may connect a PC that has its interface in promiscuous mode. This way, Juniper routers can be used to make an effective first-pass monitor, and delegate the rest of functions to a general-purpose computer (e.g. a PC). In addition, Juniper routers provide possibilities for obtaining counters and statistics which can be exploited by SCAMPI. Similarly, although the cards by themselves offer fixed functionality, there are efforts underway to make ENDACE's DAG cards programmable, by way of daughter cards [20].

Besides programmable hardware, software optimizations, and in particular packet buffering techniques, can provide other means in order to keep up with high-speed networks. Commonly used applications for network monitoring purposes, such as the intrusion detection mechanisms and the network protocol analyzers exploit the libpcap API to communicate with the network devices. Libpcap and other user-level monitoring APIs gain access to the captured packets by copying them from kernel- to user-space. By doing so, a lot of valuable processing time is wasted due to this redundant operation. As processing time is vital in high-speed networks, copying packets in memory will degrade the performance of the network monitor. In the proposed architecture, zero-copy packet handling, i.e. *mmap* [18], and advanced packet buffering techniques are used to transfer packet from hardware or kernel-space to user-space. In the SCAMPI project we implemented a driver that mmaps the IXP buffer to the application, as well as a driver, for commodity network interfaces, that provides packet filtering in the Linux kernel, with a buffer that is mmapped to the applications.

2.2 SCAMPI Goals

Addressing the challenges posed by the state-of-the-art tools in network monitoring, SCAMPI represents a step towards building an affordable network monitoring system for high-speed networks that will enable the development of portable applications. SCAMPI employs a three-pronged approach in order to achieve its goals:

- **Standard Monitoring API (MAPI).** SCAMPI defines and implements a set of monitoring calls/primitives that are collectively called the MAPI. Monitoring applications are written using this MAPI. The MAPI is implemented on top of several platforms, decoupling the development of the monitoring applications from the monitoring environment on top of which the applications will be executed. Monitoring applications are written once, and are able to run on top of any monitoring environment without the need to re-write or re-compile the application.
- **Expressive power.** Current monitoring application programming interfaces provide little (if any) expressive power to application programmers. Application programmers are not able to communicate their monitoring requirements to the underlying network monitoring system. As a result, frustrated application programmers end up receiving all network packets in the address space of their application where they perform the operations they need. As a simple example of the poor expressive power of current network monitoring systems, consider a user that wants to sample one out of every

100 packets in order to find the most popular applications that use his/her network. Current network monitoring systems (like libpcap/Berkeley Packet Filters [14][17], and Linux Socket Filters [11]) do not enable users to express such simple sampling requirements. Therefore, users that are interested in receiving just one out of every 100 packets are required to read all packets, and just discard 99 out of every 100 of them. To overcome these limitations, SCAMPI's MAPI will enable monitoring application programmers to express their requirements to the underlying monitoring system, which in turn will decide how these requirements are more efficiently implemented. The MAPI will be discussed in more detail in Section 4.1.
– **Scalability through special-purpose hardware and parallelism.** Although network monitoring can be performed on top of traditional network adapters, SCAMPI, wherever possible, will exploit specialized network adapters that provide some monitoring functionalities in hardware. These devices contain on-board processors and FPGAs that can be programmed to perform monitoring functions and off-load the host processor, memory system, and I/O bus from much of their load. An important part of the SCAMPI project is the design, development and manufacturing of our own specialized network adapter. This adapter will be able to operate at 10 Gbps speed and will have integrated programmable packet filtering, sampling and statistics functionality. The adapter will support multiple concurrent monitoring application requiring different sets of packets and specifying different monitoring capabilities. These features make the SCAMPI adapter significantly different than other monitoring adapters, such as DAG cards.

3 The SCAMPI Architecture

To monitor a high-speed backbone, a SCAMPI system will be connected to a splitter, which mirrors all the traffic of that link. The SCAMPI system will be composed of a regular PC coupled with a Hardware Monitor connected to the system's I/O (e.g. PCI) bus. Different instantiations of the SCAMPI system will require different hardware monitors. Low-speed monitoring systems will probably use a regular network interface, while high-speed systems may employ a special-purpose adapter. This adapter has the computing capacity to not only capture packets, but also to process them and perform some simple (but fast) monitoring functionalities. In the SCAMPI monitoring system, software will run on the host processor (a PC), both at kernel-level and at user-level, as well as on the SCAMPI monitoring hardware if possible.

User-level software will be composed of the following major modules: the job control system, the protection subsystem and the system that translates MAPI calls into the underlying system calls.

Figure 1 depicts the overall SCAMPI system architecture. The MAPI separates the monitoring applications from the monitoring infrastructure. Independent of the underlying infrastructure, monitoring applications will be written using the function calls provided by the MAPI. The MAPI enables programmers to write their application once and run them on top of different monitoring

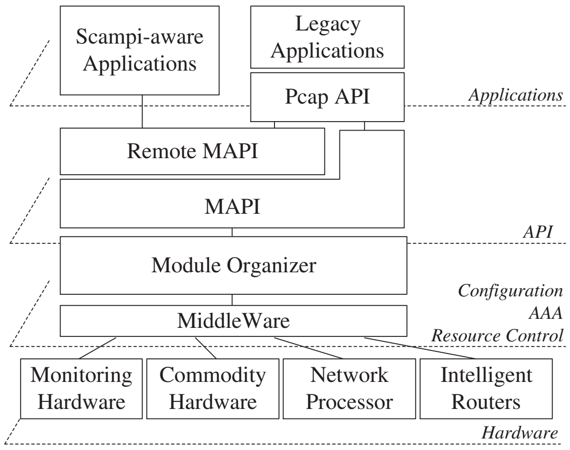


Fig. 1. Generic Architectural Decomposition

infrastructures without any changes. Generally speaking, the MAPI is a library which is linked in with the application, providing (optionally distributed) access to the SCAMPI system. A special subset of the functions offered through this subset provide the same signature as `libpcap` in order to support legacy applications.

The MAPI communicates with the module organizer in order to aggregate the requests of the different clients into a single configuration. Other tasks handled by the module organizer are:

– optimize the configuration to handle current and future requests, taking into account the capabilities of the available hardware and exploit possible commonalties in the different simultaneous monitoring jobs;
– perform admission control based on user policies and current state of the system.

The actual monitoring jobs are executed in either software or in hardware, which can be either internal or external to the monitoring agent. From the implementations point of view, the functions in the MAPI give a view of the different capabilities below, without saying that a given function is available in soft- or in hardware. This means that the MAPI should offer three categories of functions:

1. Monitoring Job Control: including definition of monitoring jobs, installation, removal and result handling.
2. Programmability: adding and removing user-defined analysis functions.
3. User Handling: including access control, authentication and profiles (or policies as discussed later)

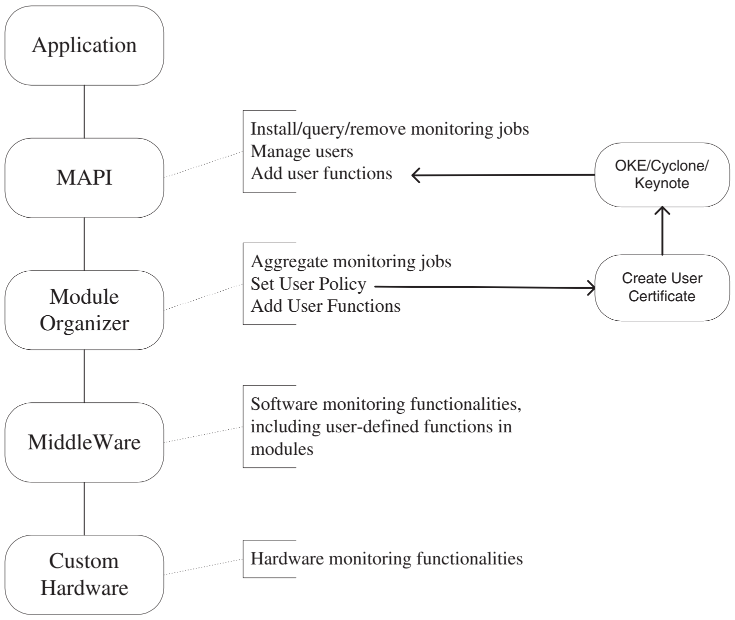

Application
MAPI
Module Organizer
MiddleWare
Custom Hardware
Install/query/remove monitoring jobs
Manage users
Add user functions
Aggregate monitoring jobs
Set User Policy
Add User Functions
Software monitoring functionalities, including user-defined functions in modules
Hardware monitoring functionalities
OKE/Cyclone/ Keynote
Create User Certificate

Fig. 2. Implementation view on the architecture

While the first two categories will have repercussions in either hardware or software, the management of users is done in software only by the module organizer. The implementation view on the architecture is illustrated in figure 2. An application can use the MAPI to install, remove and query monitoring jobs. By calling the right MAPI function, user-defined functions can be loaded in the monitoring agent. To allow secure access to the monitoring agent and manage the operational SCAMPI system, administration applications can add and remove authorized users and configure their privileges. If one of the previous functions is called, the MAPI propagates it to the module organizer. If for example an administrative application adds a new user, the module organizer will create a credential containing the privileges and credentials of this new user. This credential is sent back to the requesting application using the same mechanism as currently found in the OKE trust management scheme, which in turn builds on KeyNote [5][4].

4 A Programmable Monitoring Architecture

4.1 Definition of the Monitoring API

Most applications will interact with the SCAMPI system through its Monitoring API or MAPI. The Monitoring API is a library that can be compiled into any application to encapsulate the communication with the centralized SCAMPI components. The MAPI can either run co-located with the SCAMPI monitoring device or can be accessed remotely. The following functions form the basic infrastructure of the MAPI.

- Network flow creation: creates a new flow containing only the packets of interest.
- Apply function: append one of the functions defined below or a user-defined function added to the system, to the chain of operations to be executed on the network flow.
- Network flow removal: releases the resources related to the network flow.
- Information retrieval: gets information from a given network flow, including statistics, packet traces or flow records.
- Events: get asynchronous information from the given network flow.

Once an application has created the subflow, containing only those packets the application is interested in, one or more functions can be executed on this flow of packets. These functions can be subdivided in the following categories:

- Cooking: re-order and select packets in such a way that a transport or application-level flow can be analyzed, e.g. to recreate a TCP flow all duplicate packets have to be discarded, packets that are out of order have to be re-ordered and fragmented packets need to be de-fragmented.
- Flow selection: this category classifies incoming packets into a network flow. This can be based on IP/TCP/UDP header fields, a fixed bit pattern or a pattern to be matched in a cooked stream. Also a set of sampling selectors are available, i.e. periodic, probabilistic or based on the hashing of packets header fields [10]. Flow selections can be concatenated to refine the selection. The resulting selected packets are sent to the following steps in the chain.
- Flow analysis: get analysis results from a network flow. This includes byte and packet counters and bandwidth and packets per second analysis.
- Flow export: export resulting packets or statistics into a flow record conform to the IPFIX standard [9], a tcpdump file or proprietary format.

The first two functions can be used to create the subflow of packets of interest, the last two functions will be applied on this subflow.

4.2 The Module Organizer

The module organizer is a component that translates a monitoring job (created by calling MAPI functions) to a equivalent module representation. A first step in creating a monitoring job is obtaining the required network flow, i.e. all packets are selected that are needed in the monitoring job. Modules such as classifiers, samplers and pattern matchers will be used to create this target network flow. Once the monitor obtains the target network flow, several functions will operate on that subflow. These functions will process the flow in order to obtain

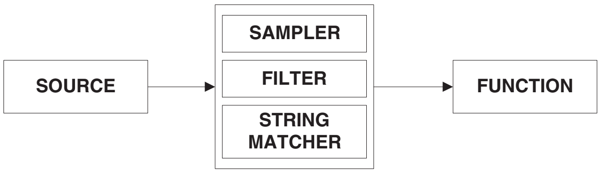


Fig. 3. Monitoring Job Representation

the required monitoring information. Functions such as 'return the entire network flow', 'return only packet headers', 'count packets or bytes' and 'measure bandwidth of the flow' will be supported.

Figure 3 depicts the module representation of a generic monitoring job. It consists of different packet processing components that are linked together. If a packet conforms the selection criterion in a certain component, it is sent to the next component in the chain. In order to create such a graph representation of a monitoring job, a graph library is developed. This library is implemented in C++ and supports all functions needed in order to add, delete, replace and find nodes and merge and clone existing graphs.

Because a SCAMPI monitoring agent will process multiple monitoring jobs simultaneously, a global monitoring graph will be created. The global monitoring graph will be a tree with the Raw Network Flow as root and consists of all merged monitoring jobs. This tree has to be optimized in order to support as many monitoring jobs as the monitor (hardware and software) can handle. Every node in the graph has an associated cost or processing time, which conforms the number of processing cycles needed to process a packet. A patternmatcher for example has a higher cost than a simple counter, a filter will have a higher cost than a periodic sampler. Each packet will traverse the tree top-down. It is possible that a packet will branch in a certain node and travel on multiple paths simultaneously. The ultimate optimization goal is to minimize the total processing time for each packet.

Module Optimizer. A straightforward solution to create the global monitoring graph is to replicate all packets in the raw network flow for each monitoring job. This approach is very easy to implement, but will definitely not be the most efficient solution. Because all packets are duplicated for each monitoring job, the number of packets in memory raises linearly with the number of configured jobs, even if multiple jobs require the same set of packets. Techniques and algorithms are studied to optimize the monitoring graph in such a way that the total number of duplicate packets in memory is minimized.

Module Mapper. The module mapper will map a monitoring graph to the corresponding hard- and/or software. In the ideal case, the whole monitoring graph will be mapped to the hardware, because this will result in the fastest packet processing. Because hardware resources are limited or not available in the case of off-the-shelf network interfaces, part of the graph will be mapped to software, i.e. the Click modular router [16]. In SCAMPI we use the Click framework to do the packet handling and processing. The Click kernel module can be configured by writing the configuration to the "/proc" filesystem. A Click configuration is described in "Click language". Due to the fact that both the monitoring graph and Click configurations are directed graphs of elements, the conversion is very straightforward.

4.3 Job Control

If SCAMPI is to support multiple applications that may be active simultaneously, it needs to provide some form of resource control, both for safety (in terms of not being able to bring the system in a corrupt or inconsistent state), sharing (i.e. ensuring applications only use those resources that are allocated to them) and security (in terms of not being able to touch other applications' sensitive data/packets). We consider monitoring applications to be the clients of the SCAMPI platform, which acts as a server. In this section, we focus on admission control (AC), which deals with questions like:

1. does the client have the required privileges to perform a specific operation at the server?
2. given the total resource capacity and the resources currently in use, can we accommodate the client's request?

The former aspect is handled by explicit trust management and in essence depends only on the credentials presented by a client. As we shall see, the latter is dealt with in a similar fashion, but with the subtle difference that per-request state may be required at the server side. Besides AC, we also need to enforce the resource control.

Trust Management. All authorization in SCAMPI will be based on delegated trust management via explicit credentials as implemented by Keynote [4]. In such a scheme, a server initially contains a policy, stating explicitly which clients are allowed to do what. Whenever a client wants to perform an operation (e.g. `createFlow()`), it has to provide the corresponding credential, proving that it was authorised to perform this operation. In SCAMPI, trust may also be delegated, i.e. an authorising party may delegate part of its privileges to another party (the licensee), with or without extra constraints.

Operation authorisation. For security reasons, each operation in the MAPI should be authorised. In this way, it is possible to restrict the access to resources (including packets) to specific clients only. However, we do not want to send a specific set of credentials for each and every request. We therefore use an explicit authorisation operation which provides the appropriate authorisation settings:

At admission-control time, we may also check whether or not sufficient resources are available in the system to accommodate the new request. For example, it may be specified in a policy that the maximum number of flows that can be created by a client should not exceed N, or the number of functions applied to a flow should not exceed M. This also means that when a request is accepted by admission control, the available resource capacity changes and should be updated. In other words, whichever resources are taken into consideration (simple examples could be number of flows, or data rates in the flows), some accounting of resources is needed, e.g. to keep track of the reserved resources (on a system-wide or per-request basis). This is also needed for the admission control component, which needs to keep state about the currently available resource capacities.

For this purpose, a resource database component is added to the SCAMPI server. The resource database is responsible for keeping track of the resources in the server's domain. Each request that leads to an explicit resource reservation (e.g. loading code that instantiates a queue of specific size) may be entered in the database, together with the resources reserved/allocated on its behalf and dependencies between requests. All of this is transparent to the clients. It is only relevant to the admission control procedure.

Custom Code Loading. One of the goals of SCAMPI is to allow new functionality to be loaded below the kernel-userspace boundaries by clients with the appropriate privileges. There may be two ways in which this is done. If security is not an issue, clients should be able to insert new modules that automatically link with the predefined functions. For example, in a Click-like software model, it may be possibly to link a new component in between two existing elements. In case security and strict resource control is needed, the components that will be linked will be components in the Open Kernel Environment (OKE) [5]. The OKE allows third parties to load and execute fully optimised native code in environments without explicit support for code isolation (such as the kernel of an operating system, or microengines on a network processor) without jeopardising safety. The resources that can be accessed by such modules is restricted by the OKE. A version of the OKE for the Linux kernel is available under GPL. In SCAMPI we have implemented an interface between the OKE and a Click configuration, that allows one to extend a Click configuration with OKE modules at runtime. In fact, using an implementation that combines OKE with a Click-like software model, known as Corral, we are able to maintain a consistent software model throughout the middleware [6].

We distinguish between code loading and code application. When loading a new function, we only add (and link) some object code that was previously unknown to the system, without actually executing it (beyond what is needed for registration). Both operations need to be appropriately authorised.

The load operation takes two parameters that define what sort of code this is and where it is supposed to be running: (1) the type of code to be loaded, which can be OKE code or plain code, and (2) the place where this code will run (e.g.

in the kernel on an x86 host processor, or in the kernel of the StrongArm control processor on an IXP). Whether or not one is permitted to load such code at this location is a separate issue and handled by the credential scheme. Whether or not the appropriate execution environment exists will be kept in the resource database.

5 Related Work

Of existing APIs, CoralReef [7] is probably the one that has the most in common with MAPI. CoralReef is a software suite for developing applications for capturing and analyzing network traffic. Analysis can be done in real time or from trace files and multiple hardware adapters are supported. The expressiveness of CoralReef is however limited to specifying BPF filters for removing unwanted traffic and there is currently no support for sampling, flow records or utilizing intelligent network adapters.

IPFIX [9] and PSAMP [10] are two IETF working groups that are working on standardizing IP flow export and sampling. PSAMP concentrates on defining methods for sampling based passive measurement of flows, while IPFIX is a new effort to define what information flow records should contain and how they are exported to collectors. The SCAMPI project follows the standards that these groups develop. Comparison with other related work can be found in [19].

6 Conclusion

In this paper we have addressed the problems that arise when using today's monitoring tools in future networks. The EU IST SCAMPI project boldly proposes a solution for these emerging problems by designing and implementing a flexible, scalable and programmable monitoring architecture. We defined and partially implemented an expressive Monitoring API (MAPI), which scales to multi-gigabit networks by using configurable hardware and parallelism. Moreover, the MAPI is backward compatible with current libpcap-based monitoring applications.

In the remainder of the project life-time, multiple monitoring applications, such as pure packet capture, traffic summary reporting, threshold alerting, QoS monitoring and security related applications will be developed on top of the MAPI. These portable applications will illustrate the ease-of-use of the SCAMPI architecture, while keeping up with high-speed networks. We will evaluate the architecture by conducting isolated and multi-domain experiments. These experiments will validate and provide feedback to the architecture design and demonstrate the scalability of the SCAMPI platform.

Acknowledgement. This work was supported by the IST project SCAMPI (IST-2001-32404) funded by the European Union. Work of the fifth author is also supported in part by the Fund Of Scientific Research – Flanders (F.W.O.-V., Belgium).

References

1. S. Arramreddy and D. Riley, “PCI-X 2.0 White Paper”, April 5 2002.
2. R. Bace and P. Mell, “Intrusion Detection Systems”, National Institute of Standards and Technology (NIST), Special Publication 800-31, 2001.
3. J. S. Bayne, “Unleashing the power of networks”, http://www.johnsoncontrols.com/ Metasys/articles/article7.htm.
4. M. Blaze, J. Feigenbaum, J. Ioannidis and A. D. Keromytis, “The KeyNote trust-management system version 2”, Network Working Group, RFC2704, September 1999.
5. H. Bos and B. Samwel, “Safe kernel programming in the OKE”, In Proceedings of OPENARCH’02, New York, USA, June 2002.
6. H. Bos and B. Samwel, “The OKE Corral: Code Organisation and Reconfiguration at Runtime using Active Linking”, Proceedings of IWAN’2002, Zuerich, Switzerland, December 2002.
7. CoralReef, http://www.caida.org/tools/measurement/coralreef/.
8. G. Iannaccone, C. Diot, I. Graham, and N. McKeown, “Monitoring Very High Speed Links”, ACM SIGCOM Internet Measurement Workshop, 2001.
9. IETF IPFIX Working Group, http://www.ietf.org/html.charters/ipfix-charter.html.
10. IETF PSAMP Working Group, http://www.ietf.org/html.charters/psamp-charter.html.
11. G. Insolvibile, “Kernel korner: The linux socket filter: Sniffing bytes over the network”, The Linux Journal, 86, June 2001.
12. “Intel IXP1200 Network Processor Family – The Foundation of a Total Development Environment for Next-Generation Networks”, Intel Product Overview, 2001.
13. IST-SCAMPI: A Scaleable Monitoring Platform for the Internet, http://www.ist-scampi.org.
14. Jacobson V., Leres C. and McCanne S., “pcap manual page”, Lawrence Berkeley National Laboratory, University of California, Berkeley, CA, 2001
15. Juniper, M5 and M10 Internet Routers Hardware Guide, 2002, http://www.juniper.net.
16. E. Kohler, R. Morris, B. Chen, J. Jannotti and M. F. Kaashoek, “The Click Modular Router”, IEEE/ACM Transactions on Computer Systems 18(3), pages 263–297, August 2000.
17. S. McCanne and V. Jacobson, “The BSD packet filter: A new architecture for user-level packet capture”, In USENIX Winter, pages 259–270, 1993.
18. A. Rubini and J. Corbet, “Linux Device Drivers, 2nd Edition”, O’Reilly, 2nd Edition , June 2001.
19. SCAMPI D0.1, “Description and analysis of the state-of-the-art”, http://www.ist-scampi.org/publications/deliverables/D0.1.pdf.
20. The DAG Project, http://dag.cs.waikato.ac.nz/.

Architecture for Efficient Monitoring and Management of Sensor Networks

Mohamed Younis[1], Poonam Munshi[1], and Ehab Al-Shaer[2]

[1] Dept. of Computer Science and Elect. Eng.
University of Maryland Baltimore County
Baltimore, MD 21250
{younis, poonam1}@cs.umbc.edu
[2] School of Computer Sc.,Telecom. and Info. Sys.
DePaul University
Chicago, IL 60604
ehab@cs.depaul.edu

Abstract. Wireless sensor networks are poised to increase the efficiency of many military and civil applications, such as disaster management. Typically sensors collect data about their surrounding and forward that data to a command center, either directly or through a base-station. Due to inhospitable conditions, these sensors are not always deployed uniformly in an area of interest and some sensors can be unreachable because they are too distant from the base-station or simply because there exist obstacles in their path. This paper focuses on reducing the sensitivity of the operation and monitoring of sensor networks to the ambiguity of the propagation model of the radio signal. We define 'agent' sensors, which monitor the health and relay messages to and from unreachable sensors. We form groups of sensors around these agents while considering the load on each agent. An energy-aware routing of data collected by and relayed by these agents is performed. Our approach localizes communication which reduces the amount of sensor energy expended in transmission, enables efficient monitoring of sensor resources and health status and allows optimal management of deployed sensors for increased network lifetime. The approach is validated by the simulation results.

Keywords: Sensor networks, Energy efficient design, Network monitoring, Energy-aware communication

1 Introduction

Advances in microelectronics have enabled the development of small sensing devices equipped with signal processing and wireless communication capabilities. Such sensors are usually deployed in an ad-hoc manner in the area of interest to track events and gather data about the environment. Networking unattended sensors is expected to have significant impact on the efficiency of many military and civil applications, such as combat field surveillance, security and disaster management12345. Sensors in such applications are typically disposable and expected to last until their energy drains. Therefore, energy is a very scarce resource for such sensor systems and has to be managed wisely in order to extend the life of the sensors for the duration of a particular mission.

Sensors probe the surrounding environment and generate reports of the collected readings. Such reports are sent via the radio transmitter to a command center, usually through a gateway, deployed in the physical proximity of the sensors (Fig. 1). The gateway aggregates and analyzes the sensed data prior to forwarding to the command center. The gateway can perform fusion of the sensed data in order to filter out erroneous data and anomalies and to draw conclusions from the reported data over a period of time. For example, in a reconnaissance-oriented sensor network, sensor data indicates detection of a target while fusion of multiple sensor reports can be used for tracking and identifying the detected target 6.

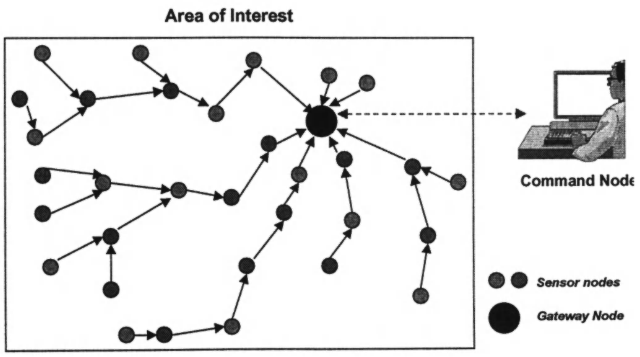


Fig. 1. Three-tier sensor network architecture

Signal processing and communication activities are the main consumers of sensor's energy. Since sensors are battery-operated, keeping the sensor active all the time will limit the duration that the battery can last. Therefore, optimal organization and management of the sensor network are very crucial in order to perform the desired function with an acceptable level of quality and to maintain sufficient sensors' energy to last for the duration of the required mission. On the other hand effective operation of sensor networks requires knowledge of the status of available sensors. Based on the remaining energy, field of view and communication range the gateway can assess sensor coverage and balance the load on the sensors in order to extend the life of the network. Therefore, the gateway needs to monitor the health status and remaining energy of the sensors. Monitoring the sensor network is a challenging problem due to the resource limitation and the large network size. Monitoring protocols for wired networks introduce large overhead and are thus deemed unsuitable for such resource constrained sensor nodes.

In order to minimize communication energy, multi-hop routes are usually employed for collecting sensor's reports. Transmission power of radio circuits is generally proportional to distance squared or even higher order for environment rich with obstacles and interference sources. Multi-hop paths minimize the total transmission power by shortening the distance a radio signal needs to travel from a transmitter to a receiver. The sensors are ideally able to communicate with the gateway through short-haul communication under all circumstances either directly or by using multiple hops. This type of model assumes that the gateway is directly reachable to all sensors under all conditions. However, sensor deployment can be non-uniform due to inhospitable conditions and the terrain may not always be low. In fact, most practical deployment scenarios involve regions that have various obstacles (e.g. buildings, trees, etc.) preventing effective signal propagation. This makes the sensor network sensitive to the propagation model. Consider for example the two cases depicted in figures 2a and 2b. In Fig. 2a, the sensors and gateway are within the communication range of each other. However the gateway is not able to directly communicate with some sensors due to an obstacle on the communication path between them. Another possible case is shown in Fig. 2b where some sensors are not directly reachable to the gateway because they are located too far from the gateway.

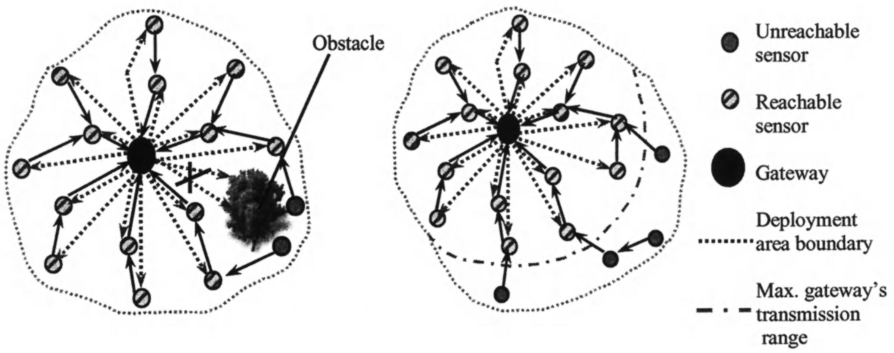


Fig. 2. a) Some sensors are not reachable by the gateway due to an obstacle between them

Fig. 2. b) Some sensors lie outside the transmission range of the gateway

In this paper, our main objective is to reduce the sensitivity of the operation of the sensor network to the ambiguity of the propagation model and to enable effective monitoring of sensors status. We introduce an additional tier in the network by forming groups of sensors. An agent sensor is designated for each group of sensors that cannot be directly reached by the gateway. Agents relay commands from the gateway, report the status of group members to the gateway and forward collected reports within the group to next assigned hops. We present an efficient technique for sensor grouping that considers sensor proximity and reachability to the gateway while balancing load on picked agent sensors. An energy-aware multi-hop routing is further employed among agent sensors to minimize communication energy. Our approach localizes communication within the group and limits the effect of poor propagation of radio signals. Furthermore, sensor health and resource status can be efficiently monitored. While clustering and energy-conscious routing are widely pursued in sensor networks, we are not aware of other work that efficiently handle both monitoring and management of the network resources.

1.1 System Model

A set of sensors is spread throughout an area of interest to detect and track events/targets in this area. The sensors are battery-operated and are empowered with limited data processing engines. Their mission is dynamically changing to serve the need of a command center. A significantly less energy-constrained gateway node is deployed in the physical proximity of sensors. Both the sensor nodes and the gateway are stationary. The gateway is assumed to know the geographical location of deployed sensors. Sensors are discovered through repeated beacons [7]7. The gateway is responsible for organizing the activities at sensor nodes to achieve a mission, fusing collected data, coordinating communication traffic and interacting with the command node. The gateway node sends to the command node reports generated through fusion of sensor readings, e.g. tracks of detected targets. The command node presents these reports to the user and performs system-level fusion of the received reports for an overall situation awareness. The system architecture for the sensor network is depicted in Fig. 1.

The sensor is assumed to be capable of reporting its remaining energy and operating in an active mode or a low-power stand-by mode. The sensing and processing circuits can be powered on and off. In addition both the radio transmitter and receiver can be independently turned on and off and the transmission power can be programmed for a required range. It is worth noting that most of these capabilities are available on some of the advanced sensors, e.g. the Acoustic Ballistic Module from SenTech Inc. [9]. It is also assumed that the sensor can act as a relay to forward data from another sensor. We refer to the gateway's selection of a subset of the sensors for probing the environment as network organization and to the data routing and medium access arbitration as network management. In our architecture, network organization and management are energy aware and rely on the knowledge of energy reserve at each sensor. Network monitoring refers to the process of querying the health status of sensors and checking their remaining energy.

1.2 Related Work

Routing of sensor data has been one of the challenging areas in wireless sensor network research. It usually involves multi-hop communications and has been studied as part of the network layer problems 101112[13]. Despite the similarity between sensor and mobile ad-hoc networks, routing approaches for ad-hoc networks proved not to be suitable for sensors networks. This is due to different routing requirements for ad-hoc and sensor networks in several aspects. For instance, communication in sensor networks is from multiple sources to a single sink, which is not the case in ad-hoc networks. Moreover, there is a major energy resource constraint for the sensor nodes. As a consequence, many new algorithms have been proposed for the problem of routing data in sensor networks. These routing mechanisms can be classified as data-centric [10], hierarchical 10 or location-based 12. Although current research on routing of sensor data mostly focused on protocols that are energy aware to maximize the lifetime of the network, we are not aware on any approach the handle both energy-aware management and efficient monitoring of sensor status.

Achieving energy saving through activation of a limited subset of nodes in an ad-hoc wireless network has been the goal of some recent research such as SPAN [14], GAF 14 and ASCENT 15. Both SPAN and GAF are distributed approaches that require nodes in close proximity to arbitrate and activate the least number of nodes needed to ensure connectivity. Nodes that are not activated are allowed to switch to a low energy sleep mode. While GAF uses nodes' geographical location to form grid-based cluster of nodes, SPAN relies on local coordination among neighbors. In

ASCENT, the decision for being active is the courtesy of the node In our approach node's state is determined at the gateway while considering processing duties in the sensor's state transition. In addition, we monitor sensor's status to enhance the effectiveness of network operation.

The unique requirements of monitoring sensor networks are well analyzed in [17]. An approach has been proposed that empowers neighbors to monitor each other. Although ideally neighbors should notify the network controller upon agreement on the faulty status of a particular sensor, they may not be able to reach the controller due to radio signal propagation problems. Instead such neighbors collaboratively handle the failure. Such model is different from ours and does not handle the core issue of sensor reachability. Another mechanism for distributed identification of faulty sensors is reported in 17. Sensors are assumed to be on all time and involved in a rather multiphase complex protocol. Instead of monitoring the individual sensor, the approach proposed in 18 track energy in small areas. Energy reserves are continuously scanned and aggregated locally and only a distribution of available energy on the network level is formed. Such approach requires sensors to be active all time and introduces additional aggregation at the sensor level.

2 Multi-tier Sensor Network Architecture

As discussed earlier, sensors are typically deployed randomly for probing the environment in a particular area of interest and transmit collected reports to a command center via radio. Since sensors are usually battery-operated, their energy has to be managed wisely. In addition, given the limited resources the sensor has, sensor status has to be closely watched in order to ensure a good coverage by enough functional sensors. We propose a multi-tier architecture for monitoring and management of sensor networks. A gateway node is to be deployed in the vicinity of sensors in order to organize the sensors according to the application requirements and to manage the collection of sensors data. To overcome weak propagation of radio signals caused by terrain and high level of signal interference and which prevents some sensors from receiving gateway messages, we form groups of sensors in close proximity with at least one reachable node.

For simplicity, we assume that the gateway can reach all deployed sensors in at most 2 hops. We consider all sensors directly reachable by the gateway as relays that can forward sensor's data to the gateway. From these relays we pick some 'agent' sensors, which act as communication hops for sensors that cannot receive gateway messages. We form groups of sensors around these agents while considering the processing and communication load on each agent. A group should have an agent, which will be responsible for interaction with gateway on behalf of that group(Fig. 3).

This approach guarantees higher level of node reachability for the gateway and reduces the energy consumed by the sensors in communication. The agent selection and grouping alleviates load on sensors for transmission by localizing the communication for the uplink. The sensors within the group need to transmit only to the agent. In addition, the selection of the path for the downlink from the gateway is more deterministic since the gateway needs to simply convey messages to the assigned agent who can then forward it to the recipient sensors within their group in 1-hop.

The next three subsections discuss network bootstrapping, group formation and finally energy-aware data routing and efficient monitoring of sensor's resources.

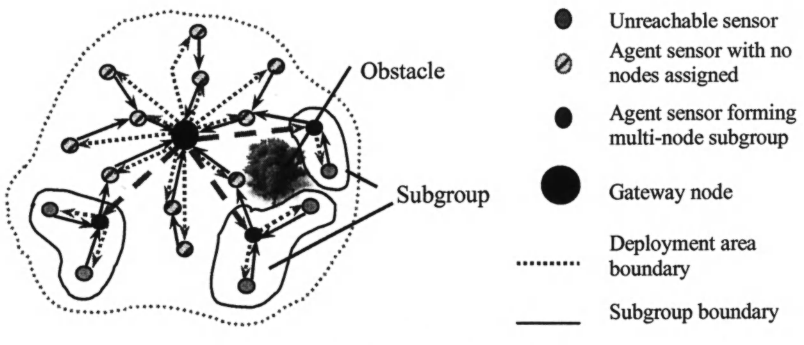


Fig. 3. A group can have only the agent or can include some unreachable sensors as well

2.1 Network Bootstrapping

Network bootstrapping refers to the initialization phase, in which the gateway finds out the location of the deployed sensors and the quality of the links among these sensors and between them and the gateway. Sensors discovery is assumed to be performed using repeated beacons, after which each sensor would know the set of sensors whose radio transmission can be heard 6. The gateway also makes multiple announcements to assess the quality of its links to the deployed sensors 7. The gateway then probes the deployed sensors to query the status of the communication links. The gateway is less energy constrained as thus expected to possess a larger transmission range than sensors. Therefore, sensors whose transmission range covers the gateway convoy location of other sensors that cannot reach the gateway and whether they can hear the gateway announcement.

Let S be the set of all deployed sensors with |S| = n. Let S_R and S_{UR} be the set of sensors that respectively can and cannot receive messages that are directly transmitted by the gateway. S_R and S_{UR} can be defined as follows:

$$S_R = \{s_i \in S : G \rightarrow s_i \text{ for all } i\}, \qquad S_{UR} = \{s_j \in S : G \not\longrightarrow s_j \text{ for all } j\}.$$

It is worth noting that $S_R \cup S_{UR}$ may not equal to S since there can be some sensors whose transmission cannot be heard by any node, e.g. fell in a deep hole. At the end of the bootstrapping phase, the gateway establish for every sensor the following attributes:

- *Gateway-link-status*: a variable set to 0 or 1 depending whether the sensor belongs to S_R or S_{UR} respectively.
- *Neighbor-list*: all sensors, which can hear its transmission.
- *Hop-list*: all neighbors, which are in the set S_R.

These attributes are used by the gateway to form groups of sensors as explained next.

2.2 Group Formation

Grouping of sensors is geared mainly for ensuring a connection between the gateway and sensors that belong to S_{UR}. The idea is to designate one or more sensors in S_R, called agents, to be responsible for convoying gateway messages to sensors in S_{UR} and relaying data collected by these sensors to the gateway (Fig. 4). Our approach for group formation that balances the load among selected agents in order to maximize their life. To facilitate such hierarchy, the gateway adds another attribute called *agentId* for every sensor, to identify the agent node to which it is assigned. Sensors that belong to S_R are assigned (agentId =0) since they do not need an agent to hear the gateway transmission. In addition, the gateway designates a *group-list* for every potential agent sensors to track members of its group. Group formation is performed as follows:

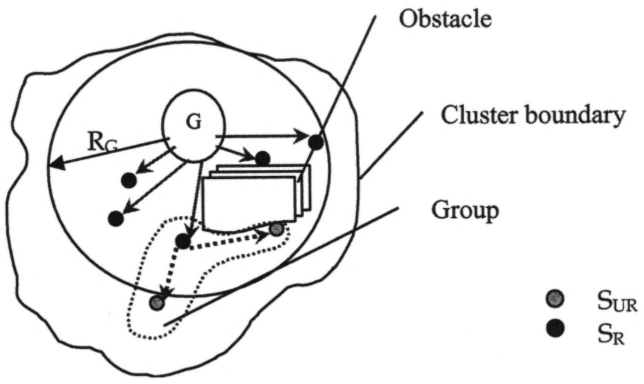


Fig. 4. Formation of groups by selection of 'agents' from the set of reachable sensors $S_{R.}$

Step 1: For each sensor $S_j \in S_{UR}$, the gateway calculates the *Hop-list* of sensors by picking neighbors that belong to the set S_R. Sensors in S_{UR} are sorted in ascending order according to the cardinality of their *Hop-list*.

Step 2: Starting from the sensors with lowest cardinality of the hop-list, we assign an agent sensor sensors S_i in S_R for every sensor $S_j \in S_{UR}$. There are two cases:

```
CASE 1: |Hop-list(S_j)| = 1            // node has a link to only one sensor in S_R
        Assign sensor S_j to the only reachable sensor S_i by making agentId(S_j) = S_i.
        Add S_j to group-list(S_i).
CASE 2: |Hop-list(S_j)| > 1            // node has a link to multiple sensors in S_R
        Calculate the assignment cost (AC) for each sensor S_i in the Hop-list of S_j.
        Assign sensor S_j to the sensor S_k in Hop-list(S_j) with least assignment cost
    (agentId(S_j) = S_k).
        Add S_j to group-list(S_k).
```

For a node $S_j \in S_{UR}$, which can reach more than one sensor in S_R, we select the node that appears to be the best i.e. has least cost at the time of assignment. The cost is a quantification of two important factors reflecting the interest of both group members and the agent (group leader). Ideally a node $S_j \in S_{UR}$ is to be assigned to a group, whose agent is the closest to S_j compared to other sensors in the Hop-list(S_j) so that communication energy is minimized. Also, we would like to ensure that the load is balanced among agents to increase their lifetime and avoid repeating the grouping process very often to recover from an agent failure. The time for the first node to die is one of the most important performance metrics for sensor networks 112. We define the assignment cost (AC) as a linear combination of both factors as follows:

$$AC_i = a_1 \times \text{Communication cost on the link } (S_j \rightarrow S_i) + a_2 \times |\textit{group-list}(S_i)|$$

The communication cost on the link ($S_j \rightarrow S_i$) is calculated based on the distance between S_j and S_i. The factors a_1 and a_2 are weighting constants that give us the flexibility of deciding which of the above two parameters is more important when forming groups. The load on sensor S_i is directly proportional to the number of sensors assigned to it (including S_j), assuming that all sensors produce data at the same rate. It is worth noting that not all sensors of S_R will be designated as a group leader. In this case they will be considered agents without group members in order to unify route setup. We next explain how sensor grouping enables efficient monitoring and data routing.

2.3 Routing and Monitoring

Since the gateway is significantly less energy-constrained compared to sensors, it is poised to efficiently manage the network. In our architecture, the gateway takes charge of sensor organization and network management based on the application mission and available energy in each sensor. Because the sensor is committed to data processing and communication, it is advantageous to offload management decision from the resource-constrained sensor nodes. In addition, since the gateway has a wide view of the network, the routing decisions should be simpler and more efficient than decisions based on local views at the sensor level. Moreover, knowledge of network-wide sensor status enhances the robustness and effectiveness of media access control because the decision to turn a node receiver off will be more accurate and deterministic than a decision based on a local MAC protocol 1920.

Sensor organization enables the activation of only a subset of the deployed sensors for probing the environment allowing the conservation of the energy of the other sensors by turning off their signal processing circuitry. Based on the remaining energy at sensors, the gateway assigns the role of environment probing to the sensors with highest remaining energy. Knowing which sensors need to be active in signal processing, the gateway can dynamically adapt the network topology to reduce the energy consumed in communication, thus extending the life of the network while achieving acceptable performance for data transmission. To achieve this, the gateway must analyze sensor reports, monitor sensors heath status and above all control the operational mode of each sensor.

Our proposed architecture enables monitoring and controlling of sensor's operation. If the gateway cannot reach some sensors, they will operate continuously and deplete their energy rather quickly. In addition, redundant data traffic will cause collisions among the transmissions of different sensors and lead to unacceptable high rate of packet drop and waste significant sensor energy. Agent nodes will enable gateway commands to reach all sensors and prevent uncontrolled redundant data generation and transmission. The gateway will be able as well to instruct some sensors to switch to a low energy sleep mode and save energy. Moreover, agent nodes can further relay data from unreachable nodes to the gateway and thus enhance the coverage that the sensor network provides.

Monitoring potentially large number of sensors in such energy-constrained environment is challenging. Status updates can generate high communication traffic in such large network and can be an energy burden, especially if done frequently. On the other hand, the quality of the sensor organization and network management decisions can be negatively affected if based on wrong sensor status. Our approach calls for modeling and tracking consumption in communication and signal processing. The gateway's knowledge of active sensors and data routes enables the estimation of energy depletion at each sensor and thus limits the need for frequent status update.

Sensors within a group communicate only with their group leader (agent). There is single-hop routing within the group i.e. the agent and the group members communicate directly. Routing data from agents to the gateway use multi-hop paths. In our routing approach, we model the sensor network as a graph with agent sensors and gateway nodes that are connected by bi-directional wireless links with a cost associated with each direction. Given that the gateway organizes the sensors, it can combine the consideration for energy commitments to data processing, remaining sensor energy, link quality and sensor location in the link cost to achieve efficient setup of message routes. Each link may have a different cost for each direction due to different energy levels of the nodes at each end. The cost of a path between two nodes is defined as the sum of the costs of the links traversed. The routing decisions take into consideration the agent's load as a function of the inbound traffic.

The cost function for a link between nodes *i* and *j* is defined as follows:

$$\sum_{k=0}^{7} CF_k \;= c_0 \times (distance_{ij})^l + c_1 / energy_j + c_2 \times Active\ inbound\ links + c_3 + c_4$$

Where: $distance_{ij}$ is the distance between the nodes *i* and *j*, $energy_j$ is the current energy of node *j* , and CF_k are cost factors defined as follows:

- CF_0: Communication cost = $c_0 \times (distance_{ij})^l$, where c_0 is a weighting constant and the parameter *l* depends on the environment, and typically equals to 2. This factor reflects the cost of the wireless transmission power, which is directly proportional to the distance raised to some power *l*.
- CF_1: Energy stock = $c_1 / energy_j$ for node *j*. This cost factor favors nodes with more energy. The more energy the node has, the better it is for routing.
- CF_2: relaying load = $c_2 \times$ number of inbound active link for node *j*. This factor extends the life of overloaded relay nodes by making them less favorable. Since these relay nodes are already critical by being on more than one path, avoiding overloading these nodes would increase the reliability of paths through them.
- CF_3: Relay enabling cost = c_3, where c_3 is a constant reflecting the overhead required to switch an inactive node to become a relay. This factor makes the algorithm favors the relay-enabled nodes for routing rather than inactive nodes.
- CF_4: Sensing-state cost = c_4, where c_4 is a constant added when the node *j* is in a sensing-sate. This factor does not favor selecting sensing-enabled nodes to serve as relays, since they have committed some energy for data processing.

For each sensing-enabled node, we find a least-cost path from the agent of that node, or the node itself if it is not a member of a group, to the gateway using Djikstra's Shortest Path algorithm. Again a sensing-enabled node forwards its data to the agent using direct links. The typical operation of the network consists of three alternating cycles: status update cycle, routing cycle and data cycle. In the status update cycle every agent queries the status of its group members and sends along with its own status directly to the gateway. The status mainly reflects the remaining energy. According to the status of nodes, the gateway adjusts the use pattern or even stops counting on it if it fails. In case of an agent failure, the gateway performs a limited regrouping during which the group members of the failed agent are reassigned to other agents. During the routing cycle, the state of each node in the network is determined by the gateway based on the application and routing algorithm and the nodes are then informed about their newly assigned responsibilities and how to route the data. In the data cycle, the nodes, which are sensing the environment, send their data to the gateway over the designated paths.

3 Experimental Validation

The effectiveness of our approach is validated through simulation. This section describes performance metrics, simulation environment and experimental results. We use the following metrics to capture the performance of our approach:

- *Node reachability*: The number of sensors that can be reached by the gateway. It is an indication of the effectiveness of the architecture and the use of sensor grouping.
- *Time for first node to die*: Losing an active node typically trigger either reorganization or re-routing or both. This metric gives an indication of network lifetime since the network can get partitioned and become hard to manage.
- *Average and standard deviation of node lifetime*: This also gives a good measure of the network lifetime. An approach, which minimizes the standard deviation of node life, is predictable and thus desirable.
- *Average energy consumed per packet*: A routing algorithm that minimizes the energy consumed per packet will, in general, yields better energy savings.

3.1 Environment Setup

In the experiments a set of 100 sensors is randomly placed in a 1000×1000 m^2 area. The gateway is randomly positioned within the boundaries of the deployment region. A free space propagation channel model is assumed with the capacity set to 2Mbps [212]. Packet lengths are 10 Kbit for data packets and 2 Kbit for routing and status packets. Each node is assumed to have an initial energy of 5 joules and a buffer for up to 15 packets 22. A node is considered non-functional if its energy level reaches 0. Some links among sensors in close proximity and between the gateway and some sensors are randomly selected to be broken to simulate terrain effects. For a node in the sensing state, packets are generated at a constant rate of 1 packet/sec 8. Each data packet is time-stamped when generated to allow tracking delays. In addition, each packet has an energy field that is updated during the packet transmission to calculate energy per packet. A packet drop probability is taken equal to 0.01. This is used to make the environment more realistic and to simulate the deviation of the gateway energy model from the actual energy. The energy model of sensors is described in Appendix A.

We assume that the network is tasked with a target-tracking mission in the experiment. The initial set of sensing nodes is chosen to be the nodes on the convex hull of the deployed sensors. The set of sensing nodes change as targets move. Since targets are assumed to come from outside the deployment area, the sensing circuitry of all boundary nodes is always turned on. The sensing circuitry of other nodes are usually turned off but can be turned on according to targets movement. Targets are assumed to start at a random position outside the convex hull and move linearly through the deployment area. These targets are characterized by having a constant speed chosen uniformly from the range four to six m/sec and a constant direction chosen uniformly depending on the initial target position in order for the target to cross the convex hull region.

3.2 Simulation Results

Experiments were run for 5 seed values varying the number of deployed sensors each time. The gateway's transmission range was taken as 500 m while the sensors' transmission range was set to 50 m. The values chosen for a_1 and a_2 in the simulation are 0.6 and 0.4, respectively. For the purpose of our simulation experiments the values for the parameters $\{c_i\}$ are initially picked based on sub-optimal heuristics for best possible performance. As mentioned earlier, some communication links among sensors in close proximity and between the gateway and sensors were broken to simulate the presence of obstacles. We compared the performance of our system with a system without grouping incorporated. The results (figures 5 through 8) show that the algorithm gives relatively good performance for all metrics especially as the number of deployed sensors is increased. We calculated the "Node Reachability" ratio as follows:
Node Reachability = Number of reachable sensors (using grouping) / Total number of sensors.

The figures show that our algorithm performs better when the sensors are densely populated since the probability of an unreachable sensor finding an agent in its vicinity increases. Our approach outperforms a system without grouping by approximately a factor of 85% in terms of node reachability.

Also in the new system model the gateway receives periodic status updates from the sensors; hence it can manage the sensors and perform educated routing decisions thus increasing the average lifetime of the nodes by a factor of 1.28 over the other system. Since unreachable sensors need to communicate only to their one-hop agents and the load on agents is taken into consideration while routing, it leads to better energy savings. Hence our system model gives a marginal improvement of 9% over a system without any grouping in terms of average energy consumed per packet.

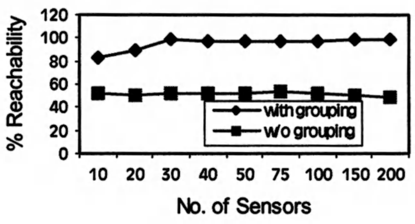


Fig. 5. Effect of grouping on node reachability

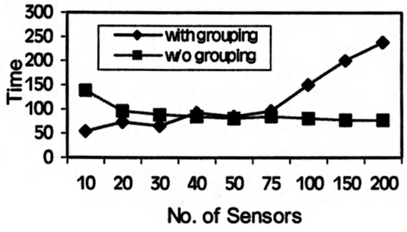


Fig. 6. Comparison of time to network partition

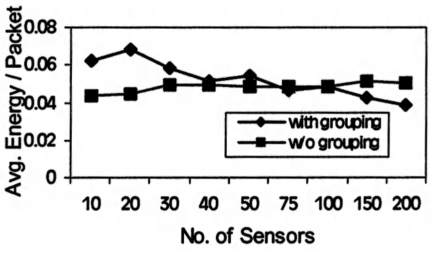


Fig. 7. Effect of grouping on average energy of Packet

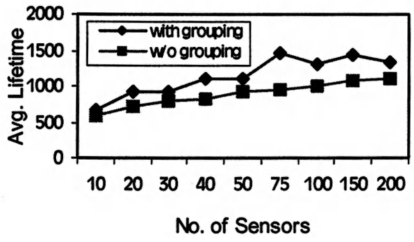


Fig. 8. Effect of grouping on average lifetime of node packet

4 Conclusions

Over the last few years, the design of unattended sensor networks has gained increasing importance due to their potential for some civil and military applications. Sensors are typically deployed randomly in an area of interest to probe the environment and transmit collected reports to a command center via radio. Due to terrain and high level of signal interference some sensors can be unreachable. In addition, sensors are usually battery-operated and their energy has to be managed wisely.

In this paper, we have introduced a novel approach for efficient monitoring and energy-aware management of wireless sensor networks. A less energy-constrained gateway node is used to manage network operation. A gateway node controls sensor functions, monitors sensor status and sets routes for sensor data. The gateway tracks energy usage at every sensor node and changes in the mission and the environment. Sensors are grouped to ensure that they can be reached via a member of its group. To limit monitoring traffic, energy consumption at the sensor is accurately modeled and tracked. Simulation results demonstrate that our architecture increases sensor reachability and the network lifetime along with better energy savings especially in densely populated sensor networks.

In the future we plan to extend the approach to address medium access issues related to the presented architecture, in particular time-based arbitration. In addition, we would like to investigate the problem of optimal placement of the gateway for enhanced sensor's reachability.

References

1. Akyildiz, I.F., et al.: Wireless sensor networks: a survey, Computer Networks, Vol 38 (2002), 393–422.
2. Estrin, D., et al.: Next Century Challenges: Scalable Coordination in Sensor Networks, Proceedings of the Fifth Annual International Conference on Mobile Computing and Networks (MobiCOM '99), Seattle, Washington, August 1999
3. Pottie, G. J., Kaiser, W.J.: Wireless integrated network sensors, Communications of the ACM, Vol. 43, No 5 (2000) 51–58
4. Sohrabi, K., et al.: Protocols for self-organization of a wireless sensor network, IEEE Personal Communications, Vol. 7, No. 5 (2000) 16–27
5. Min, R., et al.: Low Power Wireless Sensor Networks, in the Proceedings of Internation Conference on VLSI Design, Bangalore, India, January 2001
6. Burne, R., et al: A Self-Organizing, Cooperative UGS Network for Target Tracking, Proceedings of the SPIE Conference on Unattended Ground Sensor Technologies and Applications II, Orlando Florida, April 2000
7. Meguerdichian, S., et al.: Localized Algorithms in Wireless Ad hoc Networks: Location Discovery and Sensor Exposure, Proceedings of the IEEE/ACM Workshop on Mobile Ad Hoc Networking and Computing (MobiHOC 2001), October 2001
8. Mathew, R, Younis, M.: Energy-Efficient Bootstrapping Protocol for Sensor Network, Proceedings of the International Conference on Wireless Networks (ICWN'02), Las Vegas, Nevada, June 2003
9. Data sheet for the Acoustic Ballistic Module, SenTech Inc., http://www.sentech-acoustic.com/
10. Intanagonwiwat, C., Govindan, R., Estrin, D: Directed diffusion: A scalable and robust communication paradigm for sensor networks, Proceedings of the Sixth Annual International Conference on Mobile Computing and Networking (MobiCOM '00), Boston, Massachussetts, August 2000
11. Heinzelman, W., Kulik, J., Balakrishnan, H.: Adaptive protocols for information dissemination in wireless sensor networks, Proc. Fifth Annual ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom ‘99)
12. Heinzelman, W., Chandrakasan, A., Balakrishnan, H.: Energy-efficient communication protocol for wireless sensor networks, Proc. Hawaii Intl. Conf. System Sciences, Hawaii, (2000) 3005–3014
13. Shah, R., Rabaey, J.: Energy Aware Routing for Low Energy Ad Hoc Sensor Networks, Proceedings of the IEEE Wireless Communications and Networking Conference (WCNC’02), Orlando, FL, March 2002
14. Chen, B., et al: Span: an energy-efficient coordination algorithm for topology maintenance in ad-hoc wireless networks, Proceedings of MobiCom 2001, Rome, Italy, July 2001
15. Xu, Y., Heidemann, J., Estrin, D.: Geography-informed energy conservation for ad-hoc routing, Proceedings of MobiCom 2001, Rome, Italy, July 2001
16. Cerpa, A, Estrin, D.: ASCENT: adaptive self-configuring sensor networks topologies, Proceedings INFOCOM 2002, New York, June 2002
17. Hsin, C-f, Mingyan Liu: A Distributed Monitoring Mechanism for Wireless Sensor Networks, Technical Report, EECS Department, University of Michigan. Spring, 2002
18. Chessa, S., Santi, P.: Crash Faults Identification in Wireless Sensor Networks, Computer Communications, Vol. 25, No. 14 (2002) 1273–1282
19. Zhao, Y. J., Govindan, R., Estrin, D.: Residual Energy Scan for Monitoring Sensor Networks, Proceedings of the IEEE Wireless Communications and Networking Conference (WCNC'02), March 2002
20. Singh, S., Raghavendra, C.S.: PAMAS: power aware multi-access protocol with signaling for ad hoc networks, ACM Computer Communications Review, July 1998

21. Arisha, K, Youssef, M., Younis, M.: Energy-Aware TDMA-Based MAC for Sensor Networks, Proceedings of the IEEE Workshop on Integrated Management of Power Aware Communications, Computing and Networking (IMPACCT 2002), New York City, New York, May 2002
22. Andresen, J.B, Rappaport, T.S., Yoshida, S.: Propagation Measurements and Models for Wireless Communications Channels, In IEEE Communications Magazine, Vol. 33, No. 1, (1995)
23. Gerla, M., Pei, G., Lee, S.J.: Wireless, Mobile Ad-Hoc Network Routing,, IEEE/ACM FOCUS'99, New Brunswick, NJ, May 1999
24. Bhardwaj, M., et. al : Upper Bounds on the Lifetime of Sensor Networks, Proceedings of ICC 2001, June 2001 (12)
25. Min, R. , et. al : An Architecture for a Power-Aware Distributed Microsensor Node, IEEE Workshop on Signal Processing Systems (SiPS '00), October 2000
26. Heinzelman, W., et al: Energy-Scalable Algorithms and Protocols for Wireless Microsensor Networks, Proc. International Conference on Acoustics, Speech, and Signal Processing (ICASSP '00), June 2000
27. Sinha, A., Chandrakasan, A.: Energy Aware Software, Proceedings of the 13th International Conference on VLSI Design, Calcutta, India (January 2000) 50–55

Appendix A: Sensor's Energy Consumption Model

Since the presented routing algorithm uses model-based sensor energy consumption, it is important to use a fairly accurate model to ensure the effectiveness of the approach and minimize the correction made to the gateway-view of the sensor-energy level during the refresh phase. A typical sensor node consists mainly of a sensing circuit for signal conditioning and conversion, digital signal processor, and radio links 2324. The following summarizes the energy-consumption models for each sensor component.

Communication Energy Dissipation

We use the model of 112325. The key energy parameters for communication in this model are the energy/bit consumed by the transmitter electronics (α_{11}), energy dissipated in the transmit op-amp (α_2), and energy/bit consumed by the receiver electronics (α_{12}). Assuming a $1/d^n$ path loss, the energy consumed is:

$$E_{tx} = (\alpha_{11} + \alpha_2 d^n) * r \quad and \quad E_{rx} = \alpha_{12} * r$$

Where E_{tx} is the energy to send r bits and E_r is the energy consumed to receive r bits. Table 1 summarizes the meaning of each term and its typical value.

Computation Energy Dissipation

We assume the leakage current model of 242526. The model depends on the total capacitance switched by the program and the number of cycles the program takes. We used parameter values similar to those in 26.

Table 1. Parameters for the communication energy model

Term	Meaning
α_{11}, α_{12}	Energy dissipated in transmitter and receiver electronics per bit (Taken to be 50 nJ/bit).
α_2	Energy dissipated in transmitter amplifier (Taken = 100 pJ/bit/m^2.
r	Number of bits in the message.
d	Distance that the message traverses.

Sensing Energy Dissipation

We assume that the energy needed to sense one bit is a constant (α_3) so that the total energy dissipated in sensing *r* bits is 23: $E_{sensing} = \alpha_3 * r$

For the Ballistic Audio Module sensor 8, the energy dissipated for sensing a bit is approximately equal to the energy dissipated in receiving a bit. Therefore, the parameter α_3 is taken equal to α_{12}.

Unicast Probing to Estimate Shared Loss Rate

Dinh-Dung Luong[1], Attila Vidács[1], József Bíró[1], Daisuke Satoh[2], and Keisuke Ishibashi[2]

[1] Department of Telecommunications and Media Informatics
Budapest University of Technology and Economics
Magyar tudósok körútja 2, Budapest, H-1117 Hungary
{luong,vidacs,biro}@tmit.bme.hu
[2] NTT Corporation
3-9-11 Midori-cho, Musashino-shi, Tokyo, 180-8585 Japan
{satoh.daisuke,ishibashi.keisuke}@lab.ntt.co.jp

Abstract. The paper introduces a new receiver-based active end-to-end measurement technique, called the Single-Double Unicast Probing (SDUP), to estimate the rate of losses which occur on the shared network path of two flows. A comprehensive performance evaluation and a comparison between the SDUP and an existing technique having the same objective, the Striped Unicast Probing (SUP) [6], is carried out. We show that our proposed SDUP method has smaller bias (i.e., 20% smaller absolute error) and smaller variance compared to that of SUP. Besides the slightly higher accuracy, an important advantage of our method is that only one measurement equipment is needed at only one receiver, instead of deploying units to both receivers. Furthermore, with one sender and receiver pair we can measure not only one shared path, but any partial path that begins at the sender and locates on the path from the sender to the receiver. Our method is less intrusive and causes less bursty traffic compared to the SUP. The adaptation of these techniques into passive measurement is also considered.

1 Introduction

Packet losses inside communication networks are considered as the signals of traffic congestion or possible troubles, faults which need to be under control. Estimating different loss statistics has been a hot topic in network measurement. The well-known ping program, described in Chapter 7 of [9], sends ICMP *echo request* packets and receives *echo reply* packets to/from hosts in the networks. It can infer loss and delay statistics for the round-trip including forward and backward paths. The pathchar tool [5,3], by using TTL (Time-To-Live) field in IP header, is capable to estimate bandwidth, delay and loss properties of individual links in a network path. However, these estimated parameters are for two-way links. Besides, the pathchar has to face with several problems related to accuracy, measurement time and path asymmetry.

If one is interested in the loss rate of one-way path, he/she has to perform a receiver-based measurement. However, if the loss statistics of only one segment in a network path are required, the simple receiver-based measurement is insufficient. The task of estimating the “one-way” loss rate in a segment of a network path is usually reduced to solving the following simpler problem: we have two flows of probe packets from one sender S to two receivers A and B (see Figure 1). The packets of both flows will firstly pass a common path then fork into two separate paths. We call this common path the shared path. The question is that how we can estimate the loss rates in the shared and separate paths.

To address the problem described above, a multicast-based measurement technique is proposed in [7,8]. The sender sends multicast probe packets to two receivers. The measurement technique uses the observation that if a probe packet is lost in the shared path, none of them will receive it (perfect correlation). Then the joint loss statistics at the receivers (e.g., the probability that both of the receivers receive a probe packet) depend not only on the loss rate at the shared path but on the two separate paths as well. From the derived relations, the three loss rates can be explicitly computed. The measurement technique can be applied for several multicast flows in a tree structure and the loss rate of each path segment can be inferred.

With unicast end-to-end measurement, estimating loss statistics in the shared and separate paths could not be done with the original technique in multipath probing. The problem is that when a multicast packet is replaced by unicast packets there is no perfect loss correlation between successive packets. Authors in [6] have used striped unicast probes (i.e., with no delay between transmission of successive packets within a stripe) to enhance the loss correlation. It has been reported that the typical absolute error of an estimate was about less than one percent when the size of stripe was increased to four packets.

This paper introduces an alternative unicast probing technique called the Single-Double Unicast Probing (SDUP) which sends single and pairs of packets. We give a comprehensive comparison between the SDUP and the SUP. The adaptation into passive measurements is also discussed.

The rest of the paper is organized as follows. Section 2 revisits the SUP. We describe the SDUP in Section 3. We evaluate and compare these two techniques in Section 4. Their adaptation into passive measurement is considered in Section 5. We conclude the paper in Section 6.

2 Previous Works

In [6] the authors adapt the multicast inference technique proposed in [7]. The idea here is the construction of composite probes of unicast packets, i.e., striping a group of unicast packets and dispatching them back-to-back, whose properties closely resemble those of a multicast packet. However, correlations within stripes may be less than perfect in practice (e.g., dispersion of stripes as they travel, loss events that are narrower than the stripe, interleaving of background traffic, or packet-dropping on the basis of Random Early Detection (RED). We revisit the technique proposed in [6] next.

2.1 Binary Stripes

The main idea is illustrated using the two-leaf tree topology (see Figure 1) and a binary stripe. The server S sends packets back-to-back to clients A and B.

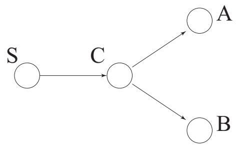


Fig. 1. Simple tree topology

Let $\alpha_k(D)$ denote the probability that packets in D ($D \subset \{1,2\}$ with binary stripes) are successfully transmitted to node k ($k \in \{C, A, B\}$ in our case), conditioned upon having reached its previous node on the path. For example, $\alpha_A(1)$ is the probability that the first packet from the stripe successfully arrives to its destination node A, given that the packet was not lost on its way to node C. Thus, the probability that the first (second) packet reaches its destination, $g(1)$ ($g(2)$), can be calculated as

$$g(1) = \alpha_C(1)\alpha_A(1), \quad g(2) = \alpha_C(2)\alpha_B(2). \tag{1}$$

Note, that the probabilities $\alpha_k(i)$ are not necessarily the same for all i. Let $\beta_k(i|j)$ ($i \neq j$) denote the conditional probability that packet i is successfully transmitted to node k, given that packet j is successfully transmitted, all packets having reached the previous node. Thus we have

$$\beta_C(2|1) = \alpha_C(\{1,2\})/\alpha_C(1), \quad \beta_C(1|2) = \alpha_C(\{1,2\})/\alpha_C(2). \tag{2}$$

With perfect correlations the various β_k would be 1. The multicast loss model of [7] is statistically equivalent to this special case and hence $\alpha_k(i)$ all equal some α_k. The probability that both packets successfully arrive to the destination is the following:

$$g(12) = \alpha_C(12)\alpha_A(1)\alpha_B(2). \tag{3}$$

Here we used the notation $\alpha_k(12)$ instead of $\alpha_k(\{1,2\})$ for simplicity. From Eqs.(2)-(3) we get the following relations

$$\alpha_C(1) = \frac{g(1)g(2)\beta_C(1|2)}{g(12)}, \quad \alpha_A(1) = \frac{g(12)}{g(2)\beta_C(1|2)}, \quad \alpha_B(2) = \frac{g(12)}{g(1)\beta_C(2|1)}. \tag{4}$$

Taking $\beta_C = 1$ yields the estimates

$$\hat{\alpha}_C = \frac{g(1)g(2)}{g(12)}, \quad \hat{\alpha}_A = \frac{g(12)}{g(2)}, \quad \hat{\alpha}_B = \frac{g(12)}{g(1)}. \tag{5}$$

However, with imperfect correlations β_C cannot be recovered from the $g(\cdot)$ success probabilities. Since $\beta_C \leq 1$, the estimations via Eq.(5) are biased, overestimating α_C and underestimating α_A and α_B.

2.2 n-Packet Stripes

The authors in [6] propose a modified striping scheme for which the value of β is closer to one. The idea is to use longer stripes. For example, using three-packet stripes $\langle A, B, B \rangle$, it is more likely that packet 1 reaches C, upon reception of packets 2 and 3 at receiver B, rather than reception of packet 2 alone. Replacing the reception of packet 2 with reception of packets 2 and 3, the analogs of the estimates in Eq.(5) are

$$\hat{\alpha}_C = \frac{g(1)g(23)}{g(123)}, \quad \hat{\alpha}_A = \frac{g(123)}{g(23)}. \tag{6}$$

(Note that α_B can be estimated using the complementary stripe $\langle B, A, A \rangle$.) These estimates introduce less bias than those from two-packet stripes. In general, the bias can be further reduced by lengthening the stripe. A potential problem with this approach is that the statistical properties of stripes may not reflect those of actual traffic if their width is not negligible compared with buffer sizes. Using measurements and simulation the authors of [6] observed good accuracy even with packet pairs, with a typical error of about 1%, which significantly decreases as stripe length is increased to four packets.

3 The Proposed Single-Double Unicast Probing

3.1 The Model

The Single-Double Unicast Probing (SDUP) is found upon our observation that the loss probability for the second packet in a back-to-back packet pair at a buffer is approximately twice as large as the probability that the first packet is lost. We analyze the reasons and introduce the measurement technique using the following model and approximations. (We use the notations of [6].) Let us consider a set of links between a server and two clients in Figure 2. Together these links form a tree rooted at the server. We refer to the set of links en route to client A (B) as L_A (L_B) and the set of links that they share as L_S. We assume that all buffers use Drop-Tail buffer management. A 3-packet probe sequence $S_{i,j,i}(\Delta, \varepsilon)$ is a sequence of packets destined to clients i, j, i, respectively. The packet spacing between the two first packets is Δ time units, and is at most ε between the second and third packets (see Figure 3). The size of probe packets is small enough as compared to the average size of packets in the network. Let p_i^k be the steady-state probability that the queue at L_i can receive exactly k packets, and p_i^{k+} is the probability that the queue at L_i can accept k or more packets. Denote $g(1)$ the probability that the first packet in packet triples is successfully received by client B. Similarly, $g(3)$ is the probability that the third packet in packet triples arrive at the client B. Suppose that the loss processes at different links are independent. Then we have

$$g(1) = \prod_{i \in L_B} p_i^{1+}. \tag{7}$$

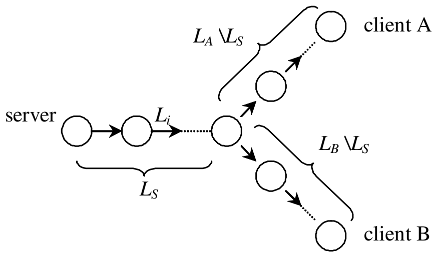


Fig. 2. Tree topology

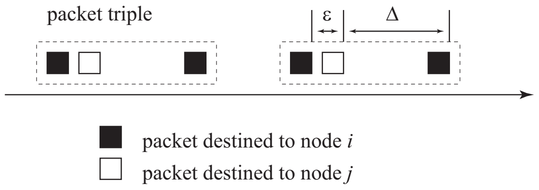


Fig. 3. 3-packet probe sequence

The third packet in each packet triple observes almost the same load condition at the shared path as the second packet does, since they have small size and are sent back-to-back. Then the third packet will pass the shared path if each queue in that path can accept at least two packets:

$$g(3) = \prod_{i \in L_S} p_i^{2+} \prod_{j \in L_B \setminus L_S} p_i^{1+}. \tag{8}$$

Combining Eq.(7) and Eq.(8) we get

$$Y \doteq \frac{g(1)}{g(3)} = \frac{\prod_{i \in L_S} p_i^{1+}}{\prod_{i \in L_S} p_i^{2+}} = \frac{\prod_{i \in L_S} (1 - p_i^0)}{\prod_{i \in L_S} (1 - p_i^0 - p_i^1)}. \tag{9}$$

The measure Y can be estimated from the estimates of $g(1)$ and $g(3)$. Eq. (9) deals with success probabilities. The probability that the first packet in a back-to-back packet pair is lost on link i is p_i^0, while the probability that the following back-to-back packet is lost is $p_i^0 + p_i^1$. Our observation is that the latter loss probability is approximately twice as large as the probability that the first packet is lost. Thus we use the approximation that $p_i^0 \approx p_i^1$. (We will verify this approximation by simulation later.) In other words, it means that the loss probability at a queue is approximately equal to the probability that the queue can accept exactly one packet. Then we get

$$Y \approx \frac{\prod_{i \in L_S} (1 - p_i^0)}{\prod_{i \in L_S} (1 - 2p_i^0)}. \tag{10}$$

The loss probability at the shared path is $p_{sl} = 1 - \prod_{i \in L_S} (1 - p_i^0)$. We estimate this loss probability in two separate cases:

Case 1: Shared Losses Occur Only on One Link
In this case Eq.(10) reduces to

$$Y = \frac{1 - p_{sl}}{1 - 2p_{sl}}. \tag{11}$$

From that,

$$p_{sl} = \frac{Y - 1}{2Y - 1} = \frac{g(1) - g(3)}{2g(1) - g(3)}. \tag{12}$$

Case 2: Shared Losses Occur on Several Links
In this case, we use a further approximation:

$$1 - 2p_i^0 \approx (1 - p_i^0)^2. \tag{13}$$

Then

$$p_{sl} = 1 - \prod_{i \in L_S} (1 - p_i^0) \approx 1 - \frac{1}{Y} = \frac{g(1) - g(3)}{g(1)}. \tag{14}$$

3.2 Implementation

Assume that we would like to measure the loss rate on the shared path L_s (see Figure 2). Then the 3-packet sequence of UDP packets $S_{A,B,A}(\Delta, \varepsilon)$ should be generated at the server and sent to the receivers. (Note that the sequence $S_{B,A,B}(\Delta, \varepsilon)$ is an alternative probing where we have our receiver at node B.) The sending times of probing packets are scheduled as described above. For the case of $S_{A,B,A}$, each probe packet to client A carries an unique sequence number with that we can determine at node A whether a packet was lost or not. The first and third probe packets could be sent to two different port numbers in order to distinguish them easily.

Let us define $Z_1^{(k)}$ and $Z_3^{(k)}$ as binary variables indicating whether or not the first and third probe packets in packet triple k arrive at the destination, respectively. $Z_1^{(k)}$ and $Z_3^{(k)}$ will be one if the considered packet reaches its destination, and zero otherwise. Thus we get $EZ_1 = g(1)$ and $EZ_3 = g(3)$. The estimates of $g(1)$ and $g(3)$ will be

$$\hat{g}(1) = \frac{1}{n} \sum_{k=1}^{n} Z_1^{(k)}, \quad \hat{g}(3) = \frac{1}{n} \sum_{k=1}^{n} Z_3^{(k)}, \tag{15}$$

where n is the sample size. Then the estimate of shared loss rate can be observed using Eq.(12) or Eq.(14). (Note that Eq.(14) can also be used when the shared losses occur only on one link, but Eq.(12) is expected to be more accurate.) Furthermore, loss statistics of only the first and the third packet in packet triples are considered, while we ignore what happens with the second packet. We call the first and the third packets the *concerned packets* (packets in black in Figure 3) and the second packet will be named as the *unconcerned packet* (in white).

4 Evaluation and Comparison

4.1 Simulation Scenario

The performance of the SDUP have been evaluated by simulation and compared with that of the SUP from different perspectives: accuracy with Drop-Tail buffers, measurement complexity, impact on background traffic, and adaptation possibility into passive measurement. The evaluation and comparison are mainly based on simulation results, using the ns simulator [1].

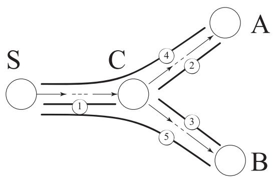


Fig. 4. Simulated tree topology with background flow groups

We set up a simple tree network configuration (see Figure 4) for simulation. The capacity of the shared links and the disjointed links were randomly chosen between 5 and 10 Mbps. The delay of each link was 50 ms. Five groups of background flows have been established (see Figure 4), the number of flows in each group was uniformly chosen from 10 to 20. A randomly chosen 75% of the flows use TCP, the others are CBR flows with on-off service times. The rate of each CBR flow was randomly chosen from 10 to 20 kbps, and the average on and off times were randomly chosen from 0.2 to 3.0 seconds. The background flows are started at time $t = -10s$, the probes start at time $t = 0$ and end at time $t = 120s$.

4.2 Queue Tail Approximation

Consider first the most important approximation in our loss estimation procedure. That is, to obtain Eq.(10), we assumed that $p_i^0 \approx p_i^1$, where p_i^0 is the probability that a queue at link i cannot accept any packet, and p_i^1 is the probability that a queue has space for exactly one more packet. We have verified this approximation by simulation using the scenario described above. Figure 5 shows the scatter plot representing p^1 versus p^0, indicating that the probability p^0 approximates p^1 accurately with very small error.

4.3 Accuracy with Drop-Tail Buffers

The packet triples for the SDUP had parameter Δ of 8 ms and parameter ε of 1 μs. The inter-packet time between the first two packets is exponentially distributed with expected value of Δ time units. The time distance between two

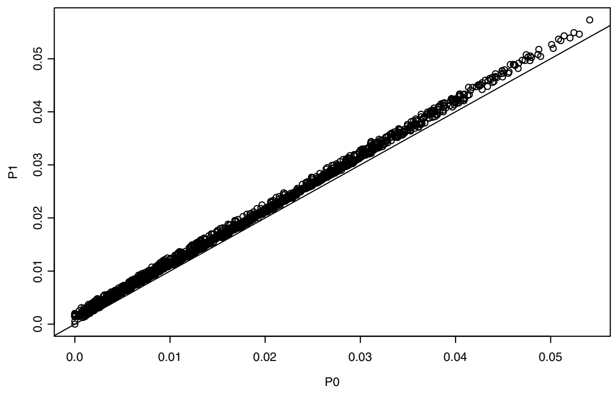


Fig. 5. Scatter plot of p^1 versus p^0, where p^0 is the probability that a queue cannot accept any packet, p^1 is the probability that a queue has space for exactly one more packet. (The straight line is the line of $y = x$.)

successive packet triples is also exponentially distributed with expected value of Δ. The probe packets' size was 40 bytes. For the SUP experiments, the inter-stripe and inter-packet times were set as in [6]: 16 ms and 1 μs, respectively. We considered 4-packet stripes for SUP since they are reported to give accurate estimations. We investigate the accuracy of the SDUP in two cases, we run 1000 experiments for each case:

Case 1: The shared path and the disjointed paths contain only one link. The estimated loss rate versus its actual value is shown in Figure 6.

Case 2: The number of links on the shared path as well as on the disjointed paths were randomly chosen between 1 and 5. The estimated loss rate versus its actual value is shown in Figure 7 for our SDUP method, and in Figure 8 for SUP. Comparing the results, there is no significant difference in accuracy between the two techniques, although the SDUP method seems to have smaller variance and less bias especially at low loss rates.

The average and the standard deviation of the absolute error (see Table 1) have changed slightly between the case of single link and the case of multiple links for the SDUP. Comparing with the results for the SUP, our proposed SDUP method has smaller bias (i.e., the absolute error is smaller by about 25%) and smaller variance compared to that of SUP.

Table 1. Statistics of the absolute error

Case	**Average**	**Standard deviation**
SDUP, DropTail, single shared link	0.0026	0.0032
SDUP, DropTail, multiple shared links	0.0033	0.0035
SUP, DropTail, multiple shared links	0.0044	0.0042

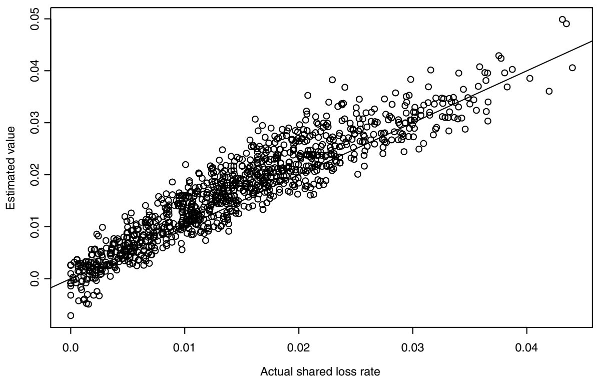


Fig. 6. The SDUP: Estimated loss rates versus their actual values for the case of single shared link. (The correlation coefficient of estimated and actual loss rate $R = 0.93$)

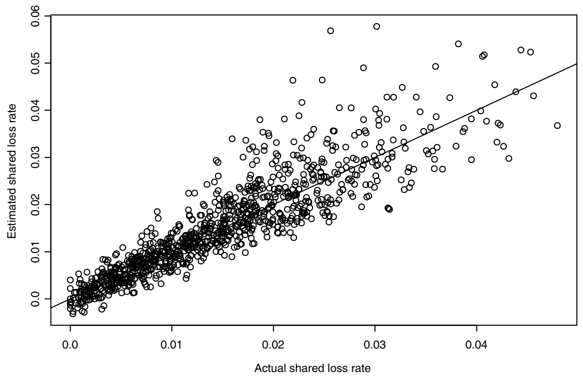


Fig. 7. The SDUP: Estimated loss rates versus their actual values for the case of multiple shared links. (Correlation coefficient $R = 0.89$)

4.4 Measurement Complexity and Flexibility

The measurement complexity intends to mean that how many measurement devices need to be set up and installed, and how complicated the measurement process is, including data collection and processing. The SDUP, as well as the SUP are receiver-based measurement techniques, then measurement equipments with installed measurement software are required at the receivers. The SUP requires measurement devices on both receivers. However, the SDUP needs it only on one receiver. Moreover, the processing of data is more simple since the SDUP takes into account only the packets received on one receiver.

An important advantage of the SDUP is that with one installed measurement device at the receiver, we can measure not only one shared path, but any partial path that begins at the sender and locates on the path from the sender to the receiver. We illustrate this in Figure 9: with a sender and receiver pair, we are

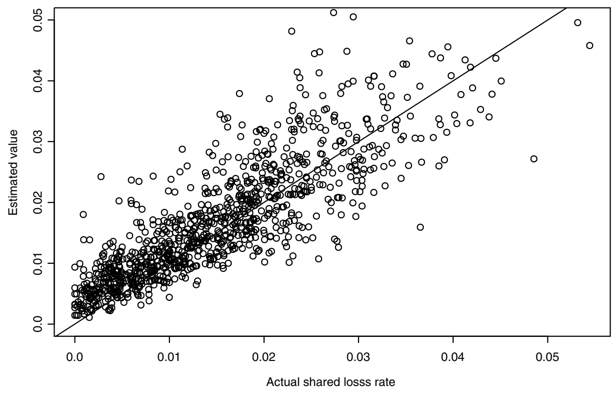


Fig. 8. The SUP: Estimated loss rates versus their actual values for the case of multiple shared links. (Correlation coefficient $R = 0.83$)

able to measure the loss rate of three shared segments: from S to node 1, from S to node 2 and from S to node 3. The implementation of SDUP in this case means that we need to deploy our measurement equipment only at the receiver, and generate probe packet triples at the sender. The only additional information needed is to choose the destination for the second (middle) probe packets such that they leave the shared path at node 1, 2 or 3, respectively, and register it in the probes that are to be received at the receiver. The advantage of the SDUP mentioned here is especially important if we cannot access other branches of the measurement tree, e.g., when there are branches belonging to another network.

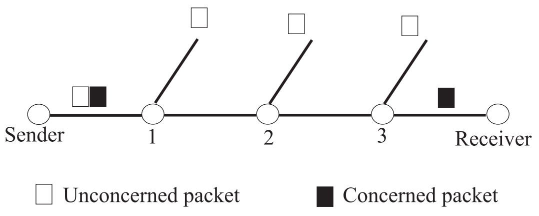


Fig. 9. With a sender and receiver set up once, SDUP can measure any shared path beginning at the sender and located on the path from the sender to the receiver

4.5 Impact on Background Traffic

In the SDUP, three packets are sent in each packet triple. In the SUP this number is at least the same, but four packets in a stripe would guarantee adequate accuracy. Suppose that the arrival rate of the triples is equal to that of the stripes, then the number of probe packets to be sent for the same measurement

Table 2. Some statistics of the extra loss rate caused by probe packets

Technique	Average extra loss rate	Standard deviation
SDUP	0.0041	0.0030
SUP	0.0049	0.0036

interval is larger in the case of SUP. Furthermore, the packets in stripes are back-to-back, then they would cause more bursty traffic than the triples do. We can guess that the SUP will cause at least an equal extra loss rate as the SDUP. In simulation experiments, we used 4-packet stripes and compared the extra loss rates generated by the two techniques in Table 2.

5 Adaptation Possibilities for Passive Measurement

Active measurements are intrusive since they inject probe packets into the network (e.g., they cause “extra loss”). With passive monitoring, we do not inject probe packets. However, we loose the ability to create packet patterns that are well controlled and fit our needs best. Instead, monitoring devices are placed inside or at the edges of the network. At the monitoring points packets can be captured, filtered and their collected statistics such as arrival time, packet size, header information (such as source and destination addresses) are stored and processed later.

There are mainly two kinds of packets in the Internet: UDP and TCP. The TCP header contains the sequence number and the source and destination information including address and port numbers that form the (nearly[1]) unique identification of a TCP packet. We can determine that a TCP packet was lost or not using this identification. UDP packets do not have a sequence number or similar header fields to identify them uniquely. To measure the loss rate on a path, a monitoring device needs to be placed at the beginning of that path. The determination of the fact that a TCP packet is lost or not can be done by waiting for its retransmission packet. If a TCP packet is retransmitted, we can infer that it was lost somewhere on the path from the monitoring device to its destination. A TCP packet is considered successfully transmitted if it has no retransmission packet.

The SUP can be adapted to passive environment as follows. The monitoring device tries to capture stripes which contain only TCP packets, and where the first packet is targeted to a predefined destination while the remaining packets in the stripe are for another target. The difficulty of this solution is that such stripes are expected to be unlikely.

In the passive version of the SDUP it is not required to capture triples which are all TCP, that is, the unconcerned packet in the middle can be UDP because

[1] In theory two different TCP packets can have the same source, destination address, port number and sequence number. However, in a short measurement interval, it is very unlikely.

Table 3. Parameters observed from measurement

Parameter	LBL-4	LBL-5	DEC-1	DEC-2	DEC-3	DEC-4
$S(40)$	0.47	0.49	0.37	0.35	0.40	0.38
v_{tcp}	0.65	0.50	0.64	0.67	0.64	0.66
$q(2)$	0.17	0.19	0.24	0.23	0.19	0.22
$q(3)$	0.06	0.10	0.10	0.09	0.05	0.07
$Ra(40,3)$	4.41	5.13	2.38	2.27	3.26	2.62
$Ra(40,4)$	12.09	9.45	5.58	5.98	11.76	7.88

we do not have to concern its loss event. Moreover, the first (single) packet and the remaining back-to-back packets (the double packets) can be monitored separately. Due to the separate monitoring of single and double packets, the probability of successfully capturing them is considerably larger than that of striped packets, therefore the SDUP is easier to adapt into passive measurement.

To characterize the frequency of filtered stripes for passive SUP and double packets for passive SDUP, we use the following notations. Let $r(A)$ and $r(B)$ be the probabilities that a captured packet is destinated to client A and client B, respectively. Let $S(s)$ denote the probability that a captured packet have smaller size than s (in bytes). Additionally, $q(n)$ denotes the probability that n successive captured packets have the same destination. For simplicity, we assume that these probabilities are mutually independent. Finally, v_{tcp} is the probability that a captured packet is TCP packet. Thus, the probabilities to capture appropriate SUP and SDUP probe packet sequences are:

$$Prob\{\text{double packets}\} = r(A)r(B)S(s)v_{tcp} \tag{16}$$

and

$$Prob\{\text{n-packet stripes}\} = r(A)r(B)q(n-1)v_{tcp}^2. \tag{17}$$

Then the ratio of the two probabilities above, denoted by $Ra(s,n)$, will be

$$Ra(s,n) = \frac{S(s)}{q(n-1)v_{tcp}}. \tag{18}$$

To evaluate Eq.(18) for real traffic, we used measured traffic traces. We analyzed the LBL-PKT and DEC-PKT traces provided by Lawrence Berkeley National Laboratory and are available from the Internet Traffic Archive [2]. The LBL-PKT traces were captured in 2 hours by monitoring all wide-area traffic into or out of the Lawrence Berkeley Laboratory, located in Berkeley, California, while the DEC-PKT traces are 5 hour-long wide-area traffic between DEC and the rest of the world, gathered at the primary Internet access point. Using these traces, the estimates of the related probabilities were calculated and are shown in Table 3. The results from the table indicate that if the size of the first packet within a SDUP packet pair must be smaller than 40 bytes, the probability of successfully capturing such packet pair is at least two times larger than that of capturing 3-packet stripes and at least five times larger than that of capturing 4-packet stripes.

6 Conclusion and Future Work

In the paper we introduced the Single-Double Unicast Probing (SDUP) to estimate the loss rate of shared path of two flows. We have evaluated the new technique and compared it with the Striped Unicast Probing (SUP)[6]. We have observed that our technique achieves slightly higher accuracy in the case of Drop-Tail buffers, but the main advantage of our proposed method is that SDUP provides higher level of measurement flexibility and low complexity, as well as it causes less measurement overhead than the SUP does. While the SUP requires measurement equipment devices on both receivers, the SDUP needs it only on one receiver. Moreover, with one installed measurement device we can measure not only one shared path, but any partial path that begins at the sender and is part of the sender-receiver path. The number of probe packets to be sent for the same measurement interval and accuracy is smaller with our method, and the generated probe traffic is less bursty, and thus less intrusive.

We also considered the adaptation of the two techniques into passive measurement. We have shown by analyzing measurement data that the probability of capturing appropriate packets for passive SDUP is considerably higher (e.g., ten times higher in certain cases) than that for passive SUP.

An important future work would be to examine and adapt the proposed method to the case when certain queue management methods (such as RED—Random Early Detection) are used in the network. In this case, packet losses not only occur when buffers overflow, and thus the correlation between back-to-back packets are weakened.

References

1. ns2: Network Simulator. http://www-mash.cs.berkeley.edu/ns/ns.html.
2. P. Danzig, J. Mogul, V. Paxson, and M. Schwartz. The internet traffic archive. http://ita.ee.lbl.gov/.
3. A. Downey. Using pathchar to estimate Internet link characteristics. In *Proceedings of the ACM SIGMETRICS International Conference on Measurement and Modeling of Computing Systems (SIGMETRICS-99)*, volume 27,1 of *SIGMETRICS Performance Evaluation Review*, pages 222–223, New York, May 1999. ACM Press.
4. S. Floyd and V. Jacobson. Random early detection gateways for congestion avoidance. *IEEE/ACM Transactions on Networking*, 1(4):397–413, August 1993.
5. V. Jacobson. Pathchar probe tool, 1997. ftp://ftp.ee.lbl.gov/pathchar/.
6. N.G. Duffield, F. Lo Presti, V. Paxson and D. Towsley. Inferring link loss using striped unicast probes. In *Proceedings of IEEE INFOCOM'01*, Anchorage, Alaska, April 2001.
7. R. Cáceres, N.G. Duffield, J. Horowitz and D. Towsley. Multicast-based inference of network-internal loss characteristics. *IEEE Transactions in Information Theory*, 45:2462–2480, 1999.
8. S. Ratnasamy and S. McCanne. Inference of multicast routing trees and bottleneck bandwidths using end-to-end measurements. In *Proceedings of IEEE INFOCOM'99*, New York, NY, March 1999.
9. W.R. Stevens. *TCP/IP Illustrated, Volume 1; The Protocols.* Addison Wesley, Reading, 1995.

VoD Service Traffic Monitoring with Neural Agent

Danielo G. Gomes[1*], Nazim Agoulmine[1], and J.N. de Souza[2]

[1] Laboratory of Complex Systems – LSC, University of Evry,
40, Rue du Pelvoux – CE 1455 Courcouronnes 91020 Evry Cedex, France
{dgomes,nazim}@iup.univ-evry.fr

[2] Teleinformatics Engineering Department – DETI, Federal University of Ceará - UFC,
Campus do Pici – Bloco 705, 60455-760, Fortaleza-CE, Brazil
neuman@ufc.br

Abstract. Video and LAN traffics can be modeled as self-similar processes and the Hurst parameter is a measure of the self-similarity of a process. The purpose of this work is to use this characteristic of Internet traffic allowing future Video on Demand Service Providers (VDSPs) to optimise their bandwidth utilization and consequently their communications cost. The work refers to one aspect of a global project that specifies intelligent agent architecture to manage the relationship among VDSP, Internet Service Providers (ISPs) and end-customers. In this paper, we also discuss the egress traffic aspect of the VDSP and propose a neural network approach to monitor and to estimate the nature of the future Value Added Service Provider (VASP) egress traffic using the Hurst parameter. This approach takes into account the real MPEG (Moving Picture Experts Group) streams in two scenarios: (i) with 13 individual MPEG streams; and (ii) with 6 mixed sequences.

1 Introduction

With the explosive growth of the Internet and of private networks related to it, a large number of new demands have increased. Low-volume Telnet conversations have been replaced by high-volume Web traffic, which it has been increasing the graphic intensity. Moreover, real-time applications have demanded an even quicker network response. In order to deal with these demands, we need nor only to make Internet capacity bigger but also to adopt accurate methods for managing the traffic. Several studies have claimed that different types of network can be accurately modelled using a self-similar process, i.e., a process capable to capture the Long-Range Dependence (LRD) phenomenon exhibited by such traffic. Furthermore, studies have demonstrated that the long range dependencies may have a pervasive effect on queuing performance. In fact, there is clear evidence that it can potentially cause massive cell losses. Such queuing system suffers from the buffer inefficacy phenomenon. The solution proposed

[*] This work is supported by CAPES (Brazil) under PhD Fellowship number 266/99-I.

by increasing the buffer size is not significantly effective for decreasing the buffer overflow probability [1].

This paper deals with the future Value Added Service Providers (VASPs), especially video on demand (VoD) VASP or VDSP, which is part of the work undertaken at the LSC Lab. This research goal is to define a global architecture based on intelligent agents whose intelligence derives from neural networks. The aim of that architecture is to allow Video on Demand Service Providers (VDSPs) to improve the use of the bandwidth as well as reduce the risk of the degradation of the Quality of Service (QoS). VDSPs are connected to a particular Internet Service Provider (ISP) that provides the necessary bandwidth that permit to end customers to access video service. The egress traffic from the VDSP domain is composed by different video streams which are sent to the end customers. The classical approach for VDSP to allow this communication is to establish a contract with an ISP that provides it with a certain bandwidth. However, the main problem with this approach is that the bandwidth is usually provided as leased line. Then, if the sum of traffic is inferior to the available bandwidth, the VDSP will pay for bandwidth without using it. On the other hand, if the number of customers increases, the allocated bandwidth may not be enough to keep the QoS at an acceptable level and the VDSP will have to refuse new connections. Thus, it is important for the VDSP to dynamically control the ISP allocated bandwidth to either increase or decreasing it according to the forecasting traffic. Since the Internet exhibit a long-range dependence, one possible approach to predict the traffic is to calculate in real time the nature of the VDSP egress traffic and detect whether it is possible or not to predict what the future traffic will look like. If the traffic significantly decreases, it is the time for the VDSP to negotiate on line a decrease of its bandwidth reservation. On the other hand, if the traffic goes over the maximum bandwidth reservation, the VDSP will negotiate an increase of its bandwidth. In the other cases, either the bandwidth allocation is optimal, or it is not possible to have a clear view of the future.

The Hurst parameter or parameter H characterizes the self-similarity process degree. The real-time calculation of this parameter will give a good view of the traffic on a particular link. The objective of this paper is identify the nature of the VDSP egress traffic by calculating the Hurst Parameter based on monitoring of traffic at the VDSP access router. In this paper, we evaluate the possibility to use a Neural Network (NN) to estimate in real time the Hurst parameter value. This neural network is managed by a agent that is placed in or near the VDSP access router and capable to take a decision regarding forecasting bandwidth reservation. The traffic for this analysis is composed of MPEG (Moving Picture Experts Group) real traces. Results indicate that the neurocomputation approach provides reasonably accurate results and is proper for real-time implementation.

The remainder of this paper is organized as follows. Section 2 provides some notions about Internet traffic, Hurst parameter, agent technology and neural networks. The proposed architecture is presented in Section 3, the MPEG streams used in Section 4 and the neural estimator in Section 5. Numerical results are shown in Section 6 and the Section 7 concludes the paper.

2 Background

2.1 Internet Traffic

The Internet Protocol (IP) is part of the TCP (Transmission Control Protocol)/IP protocol suite and is the most widely-used internetworking protocol. The IP function is to transfer data blocks, namely datagrams, transported from the source host to the destination host.

IP is a connectionless network protocol. Every IP datagram is seen as an independent unit. The communication is unreliable, i.e., there is no end-to-end recognition nor between intermediate nodes. In addition, there is no mechanism of error control of the data transmitted and no flow control is used.

In the old days, most of the data carried on networks was textual data. Today, with the rise of multimedia applications and network technologies, multimedia has become an indispensable feature on the Internet. Real-time voice and video applications become more and more popular on the Internet.

However, multimedia networking is not a simple task and, so far, the Internet has been following the *best-effort* delivery model. There is no admission control and the network makes *its best* to transmit information as quickly as possible but no assurance can be guaranteed about the delivery of the packets. Most of the applications on the Internet were elastic in nature, in that they tolerated packet delays and packet losses and they could be relatively served by the *best-effort* model. Nevertheless, the emerging multimedia traffic, such as voice and video, do not perform well under extreme variations in the delay and excessive dropping of packets.

In the current Internet model, the inelastic traffic does not perform adequately and interferes to the elastic traffic, leaving it with less bandwidth. To adapt inelastic traffic, it is necessary extending the Internet model to support the newer applications providing QoS control [2],[3].

2.2 The Hurst Parameter

Let $x(t)$, with $t = 0, 1, 2, ...$, a stationary stochastic process [4]. For each $m = 1,2,...$, let $x^{(m)}(k)$, $k = 1,2,3,\ldots$, denote a new series obtained by averaging the original series $x(t)$ over non-overlapping blocks of size m.

A process X is called *exactly second-order self-similar* with parameter $H = 1- \beta/2$, $0< \beta < 1$, if its autocorrelation function is [4]:

$$r^{(m)}(k) = \frac{1}{2}\left[(k+1)^{2-\beta} - 2k^{2-\beta} + (k-1)^{2-\beta}\right] \equiv g(k),\ 0<\beta<1,\ k=1,2,... \tag{1}$$

and X is called *asymptotically second-order self-similar* with parameter $H = 1- \beta/2$, $0< \beta< 1$, if for all $k=1,2,\ldots$,

$$\lim_{m\to\infty} r^{(m)}(k) = \frac{1}{2}\left[(k+1)^{2-\beta} - 2k^{2-\beta} + (k-1)^{2-\beta}\right] \equiv y(k) \tag{2}$$

In self-similar processes, the autocorrelations decay hyperbolically implying in a non-summable autocorrelation function $\Sigma_k r(k) = \infty$ (*long-range dependences*),

The Hurst parameter (H) gives the degree of self-similarity of a process, and, consequently, expresses the pattern of dependencies of a process. If $0.5 < H < 1$, the process is a Long-Range Dependent (LRD) process. If $0 < H < 0.5$ it is an anti-persistence process, and if $H = 0.5$ it is a Short-Range Dependent (SRD) process. Fig. 1 illustrates the auto-correlation decay for different values of the Hurst parameter.

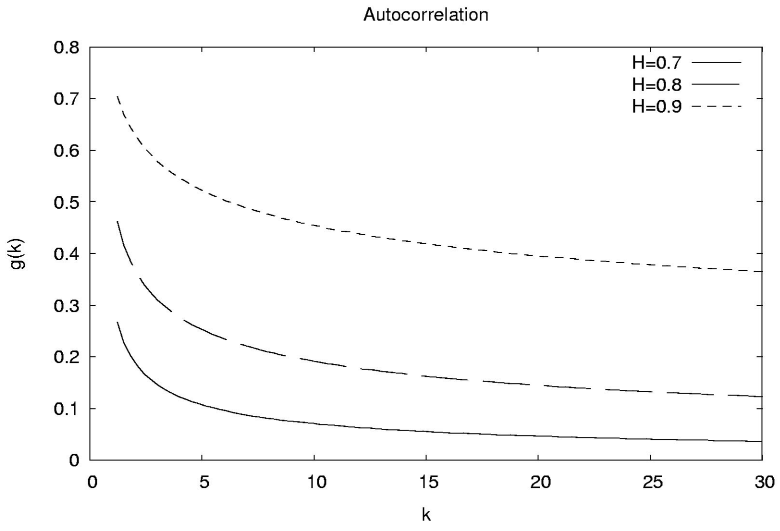


Fig. 1. Autocorrelation function (1) of an exact second-order self-similar with parameter $H = 1 - \beta/2$

2.3 Agent Technology

The agent concept has been widely proposed and adopted within both the telecommunications and Internet communities. Agent is a key tool in the creation of an open, heterogeneous and programmable network environment. This trend is motivated by the desire of using agents to solve some problems encountered in large scale distributed and real-time systems such as the volume and complexity of the tasks, latency, delays, and others. Generally, an agent can be faced as an assistant or helper, which performs routine and complicated tasks on the user's behalf. In the context of distributed computing, an agent is an autonomous software component that acts asynchronously on the user's behalf. Agent types can be broadly categorized as static or mobile. The majority of current communication system architectures employ the client-server method, which requires multiple transactions before a given task can be accomplished. This can lead to increased signalling traffic throughout the network. This problem can rapidly escalate in an open network environment that spans multiple domains. As a solution, mobile agents can move among computers and interact with other agents. For example, mobile agents can be delegated to complete specific tasks on their own, providing that a certain set of constraints or rules have been defined for them [5]. They can be dispatched across the network in the form of mobile program or mobile code that can be recompiled and executed in the remote host.

The main motivation of the use of agent technology herein is driven by the desire to automate the control and management processes by allowing for more programmability of the network to customize the provision of new information and telecommunication services Agents can also be used to implement Service Level Agreements (SLAs) between different actors of the network and service era. Agent can then be used as brokers or mediators between end users and a service provider in order to implement the SLA. In this way, complicated QoS metrics (from end user's point of view) can be communicated in a simplified manner. Service provider and network provider agents can then negotiate with users' agents to provide required service [6].

2.4 Neural Networks

The Neural Networks (NNs) appeared as an attempt of overcoming the sequential computers, based on the parallel processor structures, which can adapt the answer to the experience (training). A neural network attempts to emulate the way a human brain works.

A neural network is a system formed by a high number of simple processors (neurons or nodes), highly interconnected and based on a simplified model of the neuron. Neurocomputation is a computational approach of the neural networks for the processing of the information [7].

In fact, the weights represent the knowledge of the NN at the end of the training process and the learning is the result of all the process. Therefore, the learning is a process where the synaptic connections of the neural network are adapted by a continuous stimulus process from the environment where the network is inserted [8].

3 Proposed Architecture

The target architecture we intend to deploy is composed by a set of agents. In order to be operational, we suppose that all actors (VDSP, ISP and the end customer) will deploy an agent platform. At the ISP boundary, the platform will support four types of agents: a Policy agent responsible for the management of global ISP policies that defines high level specification of the system behaviour (pricing rules, allocation rules, etc), a Reservation agent responsible to interacting with physical equipments (routers) configuring them to provide the bandwidth agreed at the business level to the ISP customers (end customer and VDSP), an Accounting and Billing Agent responsible to accounting and billing network resource usage and allocating them to ISP customers and finally the Bandwidth Broker Agents responsible to interacting with ISP customer with the purpose of negotiating bandwidth allocation (Fig. 2).

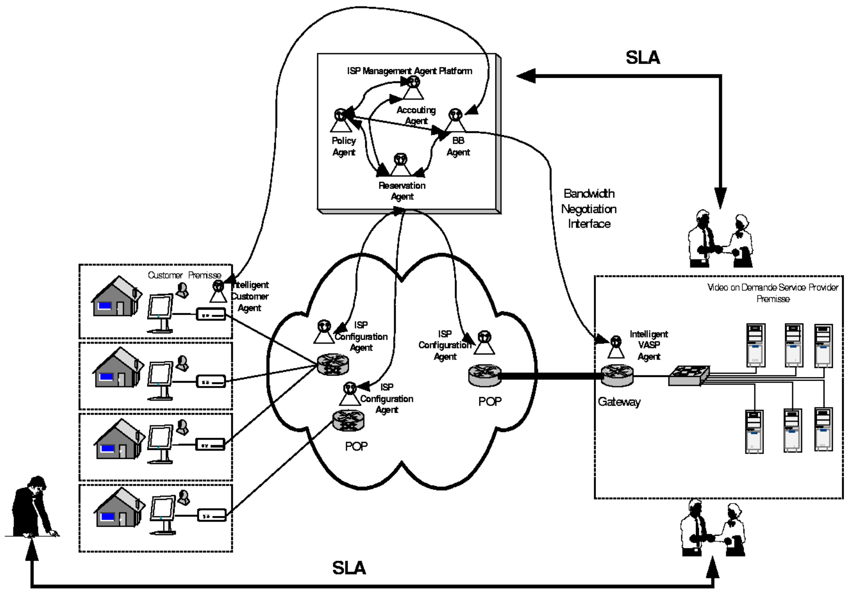


Fig. 2. Multi-agents environment

The VDSP intelligent agent is responsible for the monitoring of the egress traffic and the prediction of the forecasting traffic. At the end customers' premise, a negotiation agent allows the end customer to register at the VoD service. Also it is responsible for the end customer bandwidth availability.

The starting point of the process is a VDSP willing to offer a VoD service to a large number of end users. The VASP has established an agreement with an ISP to provide connectivity with a predefined bandwidth to end customers. We initially consider that end customers have already agreed an IP service with the same ISP using DSL technology. We suppose that the ISP is able to control the bandwidth allocated to each end user. Each customer can on-line register with the VoD service in order to access a particular movie stream at a certain date/time from the movie portfolio provided by the VDSP. Due to the bursty nature of the traffic that is sent by the video servers, the interest of the VDSP is to dynamically adapt its bandwidth reservation to a level compatible with the global traffic avoiding under provisioning or over provisioning and extra cost or quality of service degradation (Fig. 3 shows an example).

This approach is only possible if the VDSP is able to predict which traffic will appear in the middle term time scale. This task is assigned to the Intelligent VASP agent. Once the agent detects an important change in the traffic, it informs the BB agent of

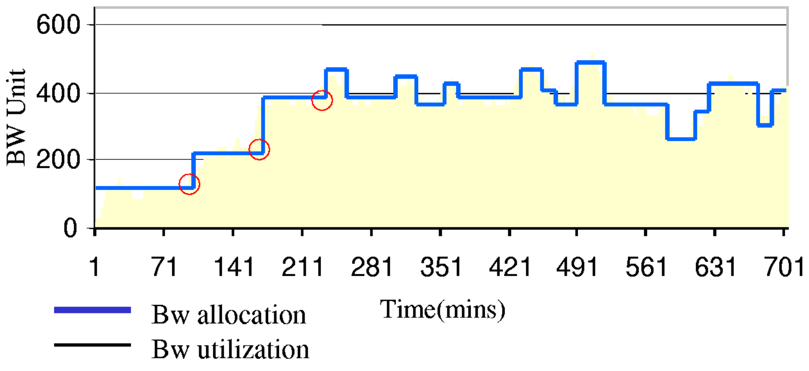


Fig. 3. Example of efficient bandwidth allocation

the ISP management system in order to request or release resources adapting to the new traffic profile. Therefore, the Intelligent VDSP Agent is at the heart of the suggested architecture.

In this context, we aim to investigate the possibility to predict the nature of the traffic so that to permit to renegotiate the allocated bandwidth. The renegotiation dialog is performed between the intelligent VDSP and the ISP BB agents (Fig. 4).

This aspect of the architecture and the internal interactions among the agents are not specified in this paper. Only the intelligent VDSP behaviour is mentioned. So the objective is using a neural network that allows the VDSP agent to take a decision regarding near future. Idea is to train the neural network with existing traffic profile and determine whether it is capable to calculate the Hurst parameter that gives information about the nature of the traffic in a relatively short period. In the real situation, it is possible to affirm that we can train the neural network including the existing video streams as the VDSP knows them initially. Every time a new movie is included in the movie portfolio, the agent is trained with its traffic.

4 MPEG Streams

Several coding algorithms for the compression of these streams were developed due to the high bandwidth needs of uncompressed video data streams. At the moment, the MPEG coding scheme is widely used for any type of video applications. Table 1 shows the MPEG sequences[1] used for Hurst parameter estimation as well as their respective H-values estimated by the R/S method [1]and the sequences were MPEG-1 encoded with sampling rate of 25 frames/sec .

[1] These traces were generated by O. Rose and are available in http://www3.informatik.uni-wuerzburg.de/~rose/.

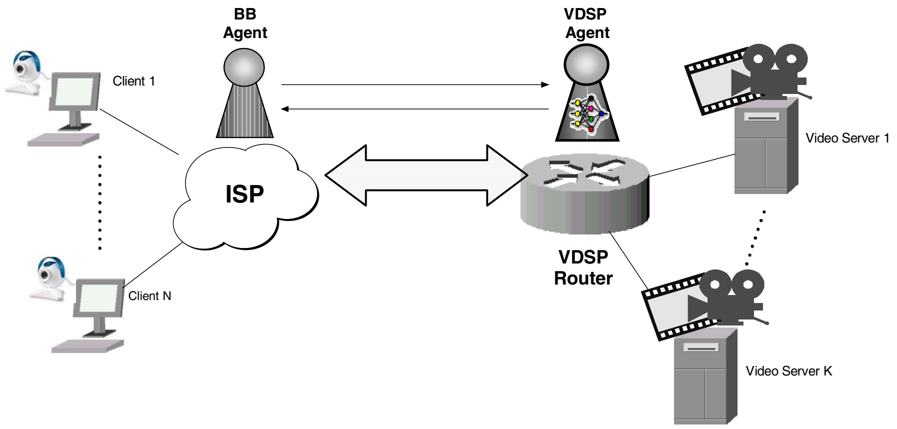


Fig. 4. VoD Model

It is known that in the event of video traffic a larger H-value reflects a larger amount of movement in the video sequence [9]. Note in Table I that all sequences have H-values which are higher than 0.73, therefore, the existence of long-range dependencies can be assumed.

Table 1. Hurst parameter of the Encoded Sequences

Sequence	Parameter H
race	0.99
soccer	0.91
lambs	0.89
terminator	0.89
mtv	0.89
simpsons	0.89
talk	0.89
dino	0.88
atp	0.88
mr.bean	0.85
Asterix	0.81
news	0.79
starwars	0.74

5 The Neural Agent Estimator

In this work, we have used a feed-forward network architecture with a back-propagation momentum training algorithm to estimate the parameter H of MPEG streams. The back-propagation algorithm was considered provided that it is the most successful algorithm for the design of multilayer feedforward networks. The number of neurons in the input-output layers was defined according to the structure of the problem. The output variable is the parameter *H*, i.e., the neural network presents one neuron on the output layer. There is not established procedure of choice for the optimum number of neuron. Then, experiments were tried with 2,5,10, 15 and 20 neurons and the better results were those obtained with 15 hidden neurons. Hence, the neural network has 10 input neurons, 15 hidden neurons and 1 output neuron. Fig. 5 shows the neural topology used and the results were derived by using the JavaNN Simulator [10].

The NN used 50 patterns for each MPEG real sequence. Each pattern has 10 video frames (input) and its respective Hurst parameter (output). Below is the summary of the neural agent behaviour:

i. Train the VDSP agent with existing movies;
ii. Install agent in the access router (border router);
iii. During execution, the agent estimate the Hurst parameter.

The calculation of the Hurst parameter will benefit in the future the agent to estimate the forecast traffic and decide whether or not to request or reduce bandwidth.

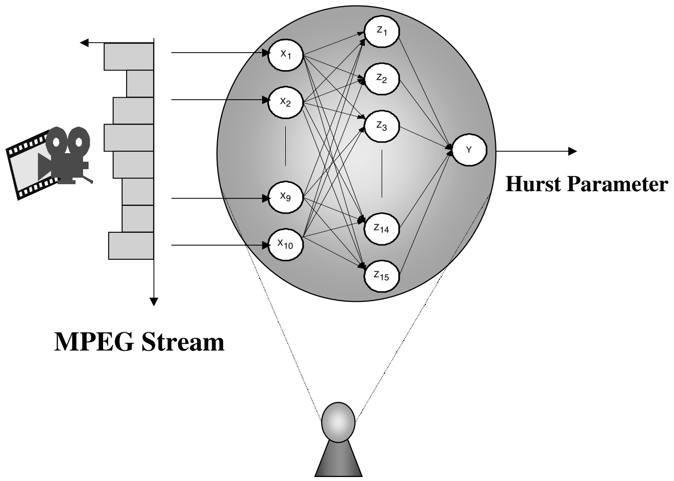
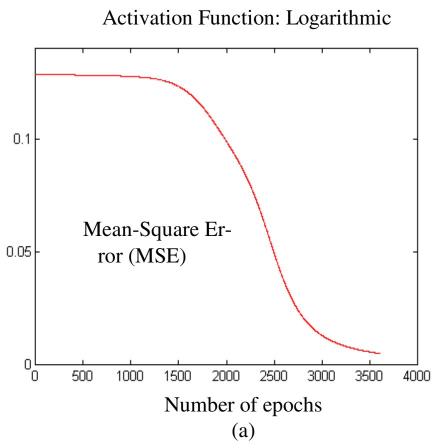


Fig. 5. Neural agent structure

6 Results and Discussion

The learning phase included simulations with logarithmic and sigmoid activation functions. Fig. 6 (a) and (b) show that the second case is better than the first given that it needs a smaller quantity of epochs.



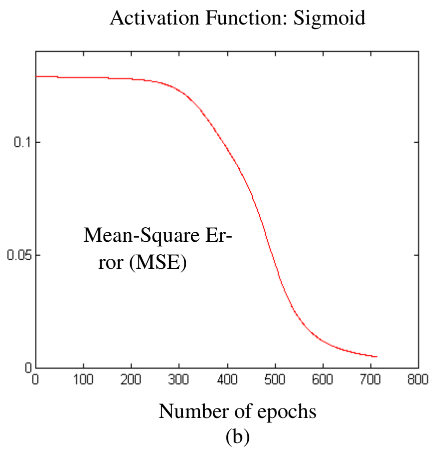


Fig. 6. Training error with (a) logistic and (b) sigmoid function

In the case of MPEG individual streams, the neural estimator error for different values of the parameter H is showed in Table 2 and in Fig. 7. Note that if the parameter is closer to 1 it means that the estimation error is smaller.

Table 2. H-estimation of individual streams

Sequence #	Parameter H	NN Estimation	Error
1.race	0.99	0.96	0.03
2.soccer	0.91	0.79	0.12
3. lambs	0.89	0.70	0.19
4. terminator	0.89	0.71	0.18
5. mtv	0.89	0.69	0.20
6. simpsons	0.89	0.70	0.19
7. talk	0.89	0.71	0.18
8. dino	0.88	0.70	0.18
9. atp	0.88	0.68	0.20
10. mr.bean	0.85	0.65	0.20
11. asterix	0.81	0.6	0.21
12. news	0.79	0.6	0.19
13. starwars	0.74	0.55	0.19

Note that in Fig. 7 that for values of the Hurst parameter close to 1 (LRD), the neural network estimator produces accurate results. However, the error increases and remains about in the same value for the rest of the sequence (~20%).
Afterwards, we mixed some sequences and estimated the Hurst parameter by the R/S and neural estimators, respectively (Table 3 and Fig. 8 show the results).

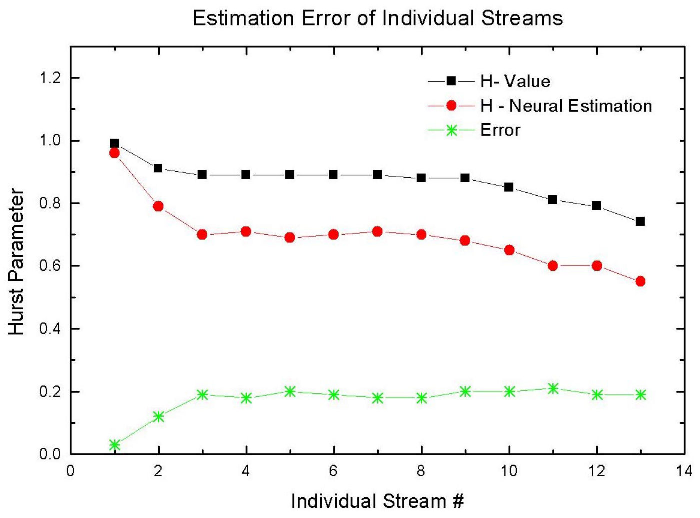


Fig. 7. Neural estimator error of the 13 individual streams

Table 3. H-estimation of mixed streams (6 sequences)

Sequence #	R/S Estimation	NN Estimation	Difference (R/S – NN)
1. race + news + starwars	0.981	0.976	0.005
2.soccer + asterix	0.90	0.89	0.01
3.lambs + mr.bean	0.87	0.86	0.01
4. terminator + atp	0.88	0.895	-0.015
5.mtv + dino	0.88	0.87	0.01
6.simpsons + talk	0.885	0.875	0.01

Although error increase, the neural networks only demanded one hundreds frames for each MPEG sequence while each total trace has 40000 frames. The drawback of the neural networks is the delay in learning which impact, however, decreases for large traces.

7 Conclusion

The present work investigates the effectiveness of a neural network H-estimator for VoD traffic monitoring. Neural networks, even demanding a significant time fortraining, represent an accurate and fast estimation of the parameter *H*. The presented approach aims to use this neural network as the intelligent part of an agent that allows a

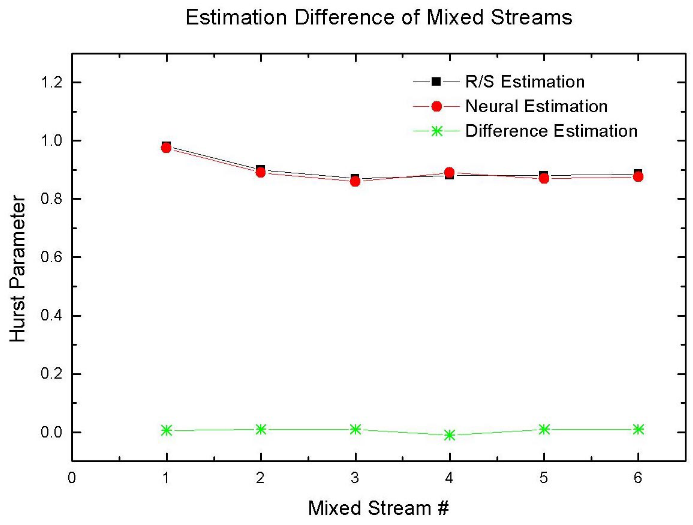


Fig. 8. R/S and neural H-estimations of the 6 mixed streams

VDSP to predict its traffic. The global architecture will permit in a competitive telecommunication market to an ISP to provide a new type of on-demand bandwidth provisioning. The numerical results showed that the neural estimation presents an error approximately up to 20% for the MPEG individual streams. However, concerning to the MPEG mixed streams, the difference between neural and R/S estimators is almost zero. Such results are considered interesting given that in VoD real scenario flow is aggregated.

References

1. Taqqu M., Teverovsky V., Willinger W.: Estimators for Long-Range Dependence: an Empirical Study. Fractals, vol 3, No 4 (1995) 785–788
2. Blake S., Black D., Carlson M., Davies E., Wang Z., Weiss W.: RFC 2475: An Architecture for Differentiated Services. Network Working Group, IETF, December 1998. (www.ieft.org)
3. Braden R., Clark D., Shenker S.: RFC 1633: Integrated Services in the Internet Architecture: An Overview. (July 1994). fttp://ftp.isi.edu/in-notes/rfc1633.ps
4. Leland W., Taqqu M., Willinger W., Wilson D.: On the Self-Similar Nature of Ethernet Traffic (Extended Version). IEEE/ACM Transaction on Networking, vol 2, no 1, (February 1994) 1–15
5. Chieng D.: A Mobile Agent Brokering Environment for The Future Open Network Marketplace. Seventh International Conference On Intelligence in Services and Networks (IS&N2000), Athens, Greece, Feb 2000

6. Agoulmine N., Dragan D., Gringel T., Hall J., Rosa E., Tschichholz M.: Trouble Management for Multimedia Services in Multi-Provider Environments. International Journal on Network and Service Management, Wiley publisher, Vol n°8, (January 2000) 99–123
7. Hect-Nielsen R.: Neurocomputing. Addison-Wesley Publishing Company (1990)
8. Fausset L.: Fundamentals of Neural Networks. Prentice-Hall International, New Jersey, (1994)
9. Beran J., Sherman R., Taqqu M., Willinger W.: Variable-Bit-Rate Video Traffic and Long-Range Dependence. IEEE Transactions on communications, (1994)
10. Stuttgart U., *JNNS – Java Neural Network Simulator* – User Manual, Version 1.0 beta http://www-ra.informatik.uni-tuebingen.de/downloads/JavaNNS/

MMNS Posters

1. **An approach for Implicit and Explicit Activation of Service Level Agreements**
   Dimitrios Kagklis, Cristos Tsakiris, Nicolas Liampotis, Efstathios Sykas
   Telecommunications Lab.Dept of Electrical & Computer Engineering National Technical University of Athens
   Zografou Campus, Heroon Polytechniou 9, 157 73 Athens, Greece
   Email: kaglis,chrtsak,nliam,sykas@telecom.ntua.gr

2. **A Multi-Domain Service Session Accounting & Charging Architecture**
   M. van Le[1], B.J.F van Beijnum[1], G.B. Huitema[2]
   1 Department of Computer Science, University of Twente, Netherlands
   2 Department of Management and Organization University of Groningen, Netherlands
   Email: le,beijnum@cs.utwente.nl; g.b.huitema@bdk.rug.nl

3. **New formal approach for QoS management in Distributed Multimedia Applications**
   Sophie Laplace, Marc Dalmau, Philippe Roose
   Laboratoire d'Informatique de l'Université de Pau et des Pays de l'Adour
   Avenue de l'Université, B.P. 1155, 64013 Pau Cedex, France
   Email: laplace, dalmau, roose@iutbayonne.univ-pau.fr

4. **Management of Security in TCP/IP Hosts Using Dedicated Monitoring Applications**
   Rui Cardoso, Mario Freire
   Networks and Multimedia Group, Institute of Telecommunications at Coimbra,
   Department of Informatics, University of Beira Interior,
   Rua Marques D'Avila e Bolama,
   P-6200-001 Covilhã, Portugal
   Email: rcardoso, mario@di.ubi.pt

5. **Mobility/Roaming Parameters and constraints in a Service Level Agreement**
   Rima Tfaily Souayed[1], Dominique Gaïti[1,2] and Guy Pujolle[2]
   1.Université de technologies de Troyes (UTT) - LM2S 12 rue Marie Curie, BP 2060 - 10010 Troyes Cedex, France
   2. Université Pierre et Marie Curie, 8 rue du Capitaine Scott, 75015 Paris, France
   Email: rima.tfaily, dominique.gaiti@utt.fr, guy.pujolle@lip6.fr

6. **Event Management Middleware Services for Proactive Management**
Dugki Min[1], Eunmi Choi[2]
1 School of Computer Science and Engineering, Konkuk. University,
Hwayang-dong, Kwangjin-gu, Seoul, 133-701, Korea
2 School of Computer Science and Electronic Engineering,
Handong Global University,
Heunghae-eub, Puk-ku, Pohang, Kyungbuk 791-940, Korea
Email: dkmin@konkuk.ac.kr, emchoi@handong.edu

7. **A Vendor and Consumers Dream: Web based Service Provision**
Stephen Dawson, Sakir Sezer
School of Electrical and Electronic Engineering
Queen's University Belfast
Ashby Building, Stranmillis Road
Belfast, BT9 5AH, N. Ireland, U.K.
Email: Stephen.Dawson@ee.qub.ac.uk

8. **Dynamic Reconfiguration of IP Domain Middleware Stacks to Support Multicast Multimedia Distribution in a Heterogeneous Environment**
Kevin Curran, Gerard Parr
Telecommunications & Distributed Systems Research Group
Northern Ireland Knowledge Engineering Laboratory
University of Ulster, Magee Campus, Northern Ireland, BT48 7JL, UK
Email: kj.curran@ulst.ac.uk

9. **Proactive Network Diagnosis Model Using Fuzzy Logic in Mobile Agents and Remote Objects**
Arnoldo Nunes da Silva[1], Carlos André Guimarães Ferraz[2],
Geber Lisboa Ramalho[2], José Neuman de Souza[3]
1 National Network of Research - Federal University of Ceará Campus do PICI,
Bl. 901 – ZIP: 60455-760 – Fortaleza – Ce – Brazil
2 Center of Informatics - Federal University of Pernambuco P.O.BOX: 7851 -
ZIP: 50732-970 – Recife – Pe – Brazil
3 Department of Computer Science - Federal University of Ceará Campus do PICI,
Bl. 910 ZIP: 60455-760 – Fortaleza – Ce – Brasil
Email : arnoldo@ufc.br, cagf,glr@cin.ufpe.br, neuman@ufc.br

Author Index