

Lecture Notes in Computer Science

Edited by G. Goos, J. Hartmanis, and J. van Leeuwen

2216

Springer

Berlin

Heidelberg

New York

Barcelona

Hong Kong

London

Milan

Paris

Tokyo

Ehab S. Al-Shaer Giovanni Pacifici (Eds).

Management of Multimedia on the Internet

4th IFIP/IEEE International Conference on Management of
Multimedia Networks and Services, MMNS 2001
Chicago, IL, USA, October 29 – November 1, 2001
Proceedings



Springer

Series Editors

Gerhard Goos, Karlsruhe University, Germany
Juris Hartmanis, Cornell University, NY, USA
Jan van Leeuwen, Utrecht University, The Netherlands

Volume Editors

Ehab S. Al-Shaer
DePaul University, School of Computer Science
243 South Wabash Avenue, Chicago, IL 60604-2301, USA
E-mail: ehab@cs.depaul.edu

Giovanni Pacifici
IBM Research Division, Thomas J. Watson Research Center
P.O. Box 704, Yorktown Heights, NY 10598, USA
E-mail: giovanni@us.ibm.com

Cataloging-in-Publication Data applied for

Die Deutsche Bibliothek - CIP-Einheitsaufnahme

Management of multimedia on the Internet : proceedings / 4th IFIP/IEEE
International Conference on Management of Multimedia Networks and Services,
MMNS 2001, Chicago, IL, USA, October 29 - November 1, 2001. Ehab S. Al-Shaer
;
Giovanni Pacifici (ed.). - Berlin ; Heidelberg ; New York ; Barcelona ; Hong Kong ;
London ; Milan ; Paris ; Tokyo : Springer, 2001
(Lecture notes in computer science ; Vol. 2216)
ISBN 3-540-42786-4

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

ISSN 0302-9743

ISBN 3-540-42786-4 Springer-Verlag Berlin Heidelberg New York

This work is subject to copyright. All rights are reserved, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, re-use of illustrations, recitation, broadcasting, reproduction on microfilms or in any other way, and storage in data banks. Duplication of this publication or parts thereof is permitted only under the provisions of the German Copyright Law of September 9, 1965, in its current version, and permission for use must always be obtained from Springer-Verlag. Violations are liable for prosecution under the German Copyright Law.

Springer-Verlag Berlin Heidelberg New York
a member of BertelsmannSpringer Science+Business Media GmbH

<http://www.springer.de>

©2001 IFIP International Federation for Information Processing, Hofstrasse 3, A-2361 Laxenburg, Austria
Printed in Germany

Typesetting: Camera-ready by author, data conversion by PTP-Berlin, Stefan Sossna
Printed on acid-free paper SPIN 10840876 06/3142 5 4 3 2 1 0

Preface

In recent years we have witnessed the explosion of multimedia traffic on the Internet. The availability of high-bandwidth connections together with the recent advances in high-quality video and audio compression techniques have created a fertile ground for the growth of multimedia applications such as interactive video on-demand, collaborative distance learning, and remote medical diagnosis. Furthermore, the availability of low bit rate video and audio applications (e.g., H.263 and G.728) and the proliferation of pervasive devices create a new demand for wireless multimedia communication systems. After a decade or more of research and development in multimedia networking, the research community has learned a number of lessons. First, increasing the capacity of the “best effort” networks and services does not provide an effective and permanent solution for offering a guaranteed Quality of Service (QoS). Second, the integration of service and network management is a key element in providing end-to-end service management. Third, management techniques for Internet multimedia services must be scalable and adaptive to guarantee QoS and maintain fairness with optimal network resource.

The IFIP/IEEE International Conference on Management of Multimedia Networks and Services 2001 was the fourth in its series aimed at stimulating technical exchange in the merging field of management of multimedia networking. The IFIP/IEEE MMNS is the premier IEEE/IFIP conference and known for its high-quality papers from various research communities. The aim of this conference is to provide a forum for exploratory research and practical contributions from researchers all over the world. A total of 106 papers were submitted to the conference, from North America, Europe, South America, The Middle East, and The Far East, of which 23 were accepted as full papers and 6 were accepted as short/position papers. The program covers a variety of research topics in the area of management of multimedia networks, i.e., QoS management, multi-point and multicast services management, monitoring, network programmability for multimedia services, policy-based management for multimedia services, packet scheduling and dropping techniques, resource management in wireless multimedia, configuration management of edge and core for multimedia services, wireless and mobile network management, multimedia traffic management, multimedia content protection, deployment of multimedia services, multimedia service engineering, multimedia session management and middleware support for management.

In closing we would like to thank the members of the Program Committee and the army of reviewers that helped us put together this year’s program. We are also indebted to Raouf Boutaba and Guy Pujolle in the Advisory Committee, for generously providing their advice and assistance. Finally, we are indebted to Hazem Hamed and Bin Zhang, for their assistance in handling the electronic paper submission process.

August 2001

Ehab S. Al-Shaer
Giovanni Pacifici

Conference Co-chairs

Ehab Al-Shaer, DePaul University, USA
Giovanni Pacifici, IBM Research, USA

Tutorial Chair

Mohammed Atiquzzaman, University of Oklahoma, USA

Organization Committee Chair

Greg Brewster, DePaul University, USA

Local Publicity Chair

Curt White, DePaul University, USA

MBONE Broadcast Specialist

John Kristoff, DePaul University, USA

Media Specialist

Paul Sisul, DePaul University, USA

Advisory Committee

Raouf Boutaba, University of Waterloo, Canada
Guy Pujolle, Université Pierre et Marie Curie, France
Wolfgang Zimmer, GMD FIRST, Germany

Program Committee

Ahmed Abutaleb, Lucent Technologies, USA
 Nazim Agoulmine, University of Evry, France
 Salah Aidarous, NEC America, USA
 Kevin Almeroth, University of California, Santa Barbara, USA
 Nikos Anerousis, VoiceMate.com, USA
 Mohammed Atiquzzaman, University of Oklahoma, USA
 Michael Borella, 3Com, USA
 Mohamed Bettaz, Philadelphia University, Jordan
 Andrew Campbell, Columbia University, USA
 Jean Pierre Claudé, University of Versailles, France
 Mohammed Erradi, ENSIAS, Morocco
 Metin Feridun, IBM Research, Switzerland
 Dominique Gaiti, University of Troyes, France
 German Goldszmidt, IBM Research, USA
 Mohsen Guizani, University of West Florida, USA
 Abdelhakim Hafid, Telcordia, NJ, USA
 Masum Hasan, Cisco Systems, USA
 Go Hasegawa, Osaka University, Japan
 Ahmed Helmy, University of Southern California, USA
 Muhammad Jaseemuddin, Nortel Networks, Canada
 Joaquim Celestino Júnior, UECE, Brazil
 Ahmed Karmouch, University of Ottawa, Canada
 Lundy Lewis, APRISMA, USA
 Derong Liu, University of Illinois, Chicago, USA
 Manu Malek, Lucent Technologies, USA
 Allen Marshall, Queen's University, UK
 Ahmed Mehaoua, University of Versailles, France
 Jose M. Nogueira, Universidade Minas Gerais, Brazil
 Jong-Tae Park, Kyungpook National University, Korea
 Puneet Sharma, HP Labs, USA
 Yuval Shavitt, Tel Aviv University and Bell Labs, USA
 Chien-Chung Shen, University of Delaware, USA
 Rolf Stadler, Columbia University, USA
 Burkhard Stiller, ETH Zurich, Switzerland
 Ralf Steinmetz, GMD, Germany
 José Neuman de Souza, Universidade Federal do Ceara, Brazil
 Alaa Youssef, IBM Research, USA

Organization Committee

Mouayad Albaghdadi, Motorola, USA
 Anthony Chung, DePaul University, USA
 Hazem Hamed, DePaul University, USA
 Yongning Tang, DePaul University, USA
 Bin Zhang, DePaul University, USA

Reviewers

Ahmed Abutaleb, Lucent Technologies, USA
Nazim Agoulmine, University of Evry, France
Salah Aidarous, NEC America, USA
Mouayad Albaghdadi, Motorola, USA
Ehab Al-Shaer, DePaul University, USA
Irfan Ali, Motorola, USA
Kevin Almeroth, University of California, Santa Barbara, USA
Hesham Anan, Microsoft, USA
Nikos Anerousis, VoiceMate.com, USA
Mohammed Atiquzzaman, University of Oklahoma, USA
Mohamed Bettaz, Philadelphia University, Jordan
Michael Borella, 3Com, USA
Raouf Boutaba, University of Waterloo, Canada
Greg Brewster, DePaul University, USA
Andrew Campbell, Columbia University, USA
Joaquim Celestino Júnior, UECE, Brazil
Lai-Tee Cheok, Columbia University, USA
Jerry Chow, Nortel Networks, USA
Anthony Chung, DePaul University, USA
Jean Pierre Claudé, University of Versailles, France
Vasilios Darlagiannis, Darmstadt University of Technology, Germany
Mohamed El-Dariby, Carleton University, Canada
Abdulmotaleb El-Saddik, Darmstadt University of Technology, Germany
Mohammed Erradi, ENSIAS, Morocco
Metin Feridun, IBM Research, Switzerland
Bill Gage, Nortel Networks, Canada
Dominique Gaiti, University of Troyes, France
German Goldszmidt, IBM Research, USA
Mohsen Guizani, University of West Florida, USA
Abdelhakim Hafid, Telcordia, USA
Hazem Hamed, DePaul University, USA
Masum Hasan, Cisco Systems, USA
Go Hasegawa, Osaka University, Japan
Bassam Hashem, Nortel Networks, Canada
Ahmed Helmy, University of Southern California, USA
Stefan Hoermann, Darmstadt University of Technology, Germany
Sugih Jamin, University of Michigan, USA
Muhammad Jaseemuddin, Nortel Networks, Canada
Ahmed Karmouch, University of Ottawa, Canada
Sung-Ju Lee, Hewlett Packard Labs, USA
Lundy Lewis, APRISMA, USA
Hongyi Li, Nortel Networks, Canada
Ching-yung Lin, IBM TJ Watson Research Center, USA
Derong Liu, University of Illinois at Chicago, USA
Zhen Liu, IBM TJ Watson Research Center, USA
Dimitrios Makrakis, University of Ottawa, Canada
Manu Malek, Lucent Technologies, USA
Allen Marshall, Queen's University, UK

Ahmed Mehaoua, University of Versailles, France
Jonathan Munson, IBM TJ Watson Research Center, USA
Jose Nogueira, Universidade Minas Gerais, Brazil
Giovanni Pacifici, IBM Research, USA
Jong-Tae Park, Kyungpook National University, Korea
Guy Pujolle, Université Pierre et Marie Curie, France
Utz Roedig, Darmstadt University of Technology, Germany
Lopa Roychoudhuri, DePaul University, USA
Mohamed-Vall Salem, Université de Montréal, Canada
Puneet Sharma, HP Labs, USA
Yuval Shavitt, Tel Aviv University and Bell Labs, USA
Chien-Chung Shen, University of Delaware, USA
José Neuman de Souza, Universidade Federal do Ceara, Brazil
Rolf Stadler, Columbia University, USA
Ralf Steinmetz, GMD, Germany
Burkhard Stiller, ETH Zurich, Switzerland
Yongning Tang, DePaul University, USA
Olivier Verscheure, IBM Research, USA
Paul Ward, University of Waterloo, USA
Peter Westerink, IBM Research, USA
Tilman Wolf, Washington University, USA
Jason Yao, DePaul University, USA
Alaa Youssef, IBM Research, USA
Bin Zhang, DePaul University, USA
Yi Zhang, University of Illinois at Chicago, USA
Wolfgang Zimmer, GMD FIRST, Germany

Table of Contents

Management of Multimedia Traffic Streaming I

A Hysteresis Based Approach for Quality, Frame Rate, and Buffer Management for Video Streaming Using TCP	1
<i>Nagasuresh Seelam, Pankaj Sethi, and Wu-chi Feng, Ohio State University, USA</i>	
A Transmission Scheme for Streaming Variable Bit Rate Video over Internet	16
<i>Sunghoon Son, Electronics and Telecommunications Research Institute, Korea</i>	
MPEG-4 Video Transfer with TCP-Friendly Rate Control	29
<i>Naoki Wakamiya, Masaki Miyabayashi, Masayuki Murata, and Hideo Miyahara, Osaka University, Japan</i>	

Resource Management in Wireless Multimedia

IP Radio Resource Control System	43
<i>John Vicente, Intel Corporation, and Andrew T. Campbell, Columbia University, USA</i>	
Is It Worth Involving Several Cooperating Agents for Multimedia User's Admission in Cellular Networks?	57
<i>Youssef Iraqi and Raouf Boutaba, University of Waterloo, Canada</i>	
Analysis of Random Access Protocol under Bursty Traffic	71
<i>Jianbo Gao and Izhak Rubin, University of California, Los Angeles, USA</i>	

Management of Multimedia Traffic Streaming II

A Practical Model for VBR Video Traffic with Applications	85
<i>Tamás Borsos, Budapest University of Technology and Economics, Hungary</i>	
Enhancing Quality of MPEG Video through Partially Reliable Transport Service in Interactive Application	96
<i>Vincent Lecuire, Francis Lepage, Centre de Recherche en Automatique de Nancy, France, and Khalil Kammoun, University of Karlsruhe, Germany</i>	

Delivering of MPEG-4 Multimedia Content over Next Generation Internet 110
Toufik Ahmed, Guillaume Buridant, and Ahmed Mehaoua, University of Versailles, France

Video Skimming and Summarization Based on Principal Component Analysis 128
Dan Lelescu, Compression Science Inc., and Dan Schonfeld, University of Illinois at Chicago, USA

QoS Management on the Internet

Distributed Resource Management to Support Distributed Application-Specific Quality of Service 142
Gary Molenkamp, Michael Katchabaw, Hanan Lutfiyya, and Michael Bauer, The University of Western Ontario, Canada

Implementation of a Bandwidth Broker for Dynamic End-to-End Capacity Reservation over Multiple Diffserv Domains 160
Ibrahim Khalil and Torsten Braun, University of Berne, Switzerland

The Impact of Confidentiality on Quality of Service in Heterogeneous Voice over IP Networks 175
Johnathan M. Reason and David G. Messerschmitt, University of California, Berkeley, USA

Poster Session

Queue Length Based Fair Queueing in Core-Stateless Networks 193
Mingyu Zhai, Guanqun Gu, and Yuan Yuan, Southeast University, China

Study of TCP and UDP Flows in a Differentiated Services Network Using Two Markers System 198
Sung-Hyuck Lee, Seung-Joon Seok, Seung-Jin Lee, and Chul-Hee Kang, Korea University, Korea

Equation-Based Dynamic Shaping Algorithm 204
Halima Elbiaze, Tijani Chahed, Gérard Hébuterne and Tulin Atmaca Institut National des Telecommunications, France

A Novel Content-Based Video Streaming Algorithm for Fine Granular Scalable Coding 210
Li Zhao, Qi Wang, Yuwen He, Shiqiang Yang and Yuzhuo Zhong, Tsinghua University, China

Topological Synthesis of Mobile Backbone Networks for Managing Ad Hoc Wireless Networks	215
<i>Izhak Rubin and Patrick Vincent, University of California, Los Angeles, USA</i>	
QoS Monitoring System on IP Networks	222
<i>Marcelo Borges Ribeiro, Lisandro Zambenedetti Granville, Maria Janilce Bosquiroli Almeida, and Liane Margarida Rockenbach Tarouco, Federal University of Rio Grande do Sul, Brazil</i>	

Fault Management on the Internet

A Framework for Supporting Intelligent Fault and Performance Management for Communication Networks	227
<i>Hongjun Li and John S. Baras, University of Maryland, College Park, USA</i>	
Architecture of Generalized Network Service Anomaly and Fault Thresholds	241
<i>Zheng Zhang and Constantine Manikopoulos, New Jersey Institute of Technology, and Jay Jorgenson, City College of New York, USA</i>	
Providing Scalable Many-to-One Feedback in Multicast Reachability Monitoring Systems	256
<i>Kamil Saraç and Kevin C. Almeroth, University of California, Santa Barbara, USA</i>	
A Framework for Event Correlation in Communication Systems	271
<i>Mouayad Albaghdadi, Bruce Briley, Mohammed Petiwala, Mark Hamlen, Motorola Inc., Martha Evens, Illinois Institute of Technology, and Rafid Sukkar, Lucent Technologies, USA</i>	

Agents for Multimedia Management

Policy-Based Management for Multimedia Collaborative Services	285
<i>Hamid Harroud, Mouhsine Lakhdissi, Ahmed Karmouch, University of Ottawa, and Cliff Grossner, Nortel Networks, Canada</i>	
Agent-Enhanced Dynamic Service Level Agreement in Future Network Environment	299
<i>David Chieng, Alan Marshall, The Queen's University of Belfast, Ivan Ho, and Gerard Parr, University of Ulster at Coleraine, U.K.</i>	
WEBARM: Mobile Code Based Agent for Web Application Response Measurement – Software Implementations and Analysis	313
<i>Joseph V. Elarde and Gregory B. Brewster, DePaul University, USA</i>	

Multimedia Service Management

The Dynamics of Price, Revenue, and System Utilization	329
<i>Srinivasan Jagannathan and Kevin C. Almeroth, University of California, Santa Barbara, USA</i>	
Developing Pattern-Based Management Programs	345
<i>Koon-Seng Lim and Rolf Stadler, Columbia University, USA</i>	
Supermedia in Internet-Based Telerobotic Operations	359
<i>Imad Elhadj, Ning Xi, Michigan State University, USA, Wai Keung Fung, Yun hui Liu, Wen J. Li, Chinese University of Hong Kong, Hong Kong, Tomoyuki Kaga, and Toshio Fukuda, Nagoya University, Japan</i>	
Author Index	373

A Hysteresis Based Approach for Quality, Frame Rate, and Buffer Management for Video Streaming Using TCP

Nagasuresh Seelam, Pankaj Sethi, and Wu-chi Feng

The Ohio State University
Department of Computer and Information Science
Columbus, OH 43210
{seelam, psethi, wuchi}@cis.ohio-state.edu

Abstract. In today's Internet, the primary transport mechanism for video streams is the UDP protocol, either congestion sensitive or not. In this paper, we propose a mechanism that supports the high quality streaming and adaptation of stored video across best-effort networks using the TCP transport protocol. Our proposed approach has a number of useful features. First, it is built on top of TCP, which effectively separates the adaptation and streaming from the transport protocol. This makes the question of TCP-friendliness, the behavioral property of a flow that allows fair-sharing of bandwidth with other flows, much easier to answer. Second, it does not rely on any special handling or support from the network itself, although any additional support from the network itself will indirectly help increase the video quality. Finally, we show through experimentation that this approach provides a viable alternative for streaming media across best-effort networks.

1 Introduction

With the rapid advances in the "last mile" networking bandwidth, such as the digital subscriber loop (DSL) and cable modems, the ability to achieve higher quality video networking is becoming more feasible than ever. As a result of moving from 28.8kbps or 56kbps coded streams to megabits per second, streaming of higher quality MPEG video is now possible. As the bit-rates for the video streams increases, the bandwidth requirements for the stream become more diverse over time, requiring more intelligent buffering and rate adaptation algorithms to be put in place.

One of the key issues in creating highly scalable networks is the notion of TCP-friendliness, which measures how fairly transport-layer flows are sharing network bandwidth [7]. While various definitions of fairness have been proposed, the key idea is that all the flows share the bandwidth and that they avoid sending the network into congestion collapse. The manifestation of TCP-friendliness in video applications has resulted in techniques such as congestion-sensitive UDP-based flows [2, 8], TCP-flows without retransmission [1], or limited retransmission UDP-flows [11].

Another key design issue is *how* video streams can be adapted to fit within the available resources of the network. For video conferencing applications and live video applications, the adaptation typically occurs along either adapting the quality of the video, adapting the frame rate of the video, or using a mixture of the two [9, 14]. For the delivery of *stored* video streams, the adaptation really occurs over three parameters: quality, frame, rate, and buffering (greater than that needed to remove

delay jitter). All of these can be used in isolation or combined together, however, efficient techniques need to be developed for the algorithms to deliver the highest quality video over the network.

In this paper, we propose a system for the delivery of *stored* video streams across best-effort networks that has a number of unique properties. One of them is that it uses the TCP protocol for flow and congestion control. Thus, the question of TCP friendliness is not a problem. In this work, we will show how to effectively deliver stored video content over best-effort networks using TCP. To show the efficacy of this approach, we have conducted simulations of the algorithms and have implemented the algorithms for MPEG streaming applications. Our results show that we can effectively combine frame rate, frame quality, and buffering into a single coherent framework.

In the rest of the paper, we will first describe the background and related work necessary for the rest of the paper. In Section 3, we describe our proposed approach as well as an evaluation metric called the *effective frame rate*. Section 4 provides the experimental results of our research, including a simulation-based study of the algorithms and an exploration of the effective frame rate measure. Finally, a conclusion and directions for future work is provided.

Contributions of this work: We believe that there are two main contributions to this work. First, we believe that this work uniquely demonstrates the effectiveness of frame quality, frame rate, and buffering for streaming of stored video streams across best-effort networks. Second, we introduce an evaluation metric for streaming video protocols called the *effective frame rate* measure which provides better insight into streaming quality than using just the average frame rate and the variation in frame rate.

2 Background and Related Work

There are a large number of related research ideas that dove-tail to the techniques proposed here. In this section, we briefly highlight some of this work.

2.1 Content Distribution Networks and Proxy-Based Delivery Mechanisms

There are a number of different approaches to the wide-scale distribution of video information. They are distinguished primarily by the way that the data is managed between the server that created the video data and the clients.

Content Distribution Networks (CDNs) focus on distributing the video information, in whole, to caches and proxies that are close to the end clients that require the information. For companies that provide access to web data, such as Inktomi or Akamai, this involves creating large distributed data warehouses throughout the world and using high-capacity, peer-to-peer networks to make sure the distribution infrastructure is up to date. The video data that is “streamed” is streamed from the cache to the end host.

Proxy-based dissemination architectures focus on actively managing streaming video content through the proxy. Such techniques include efforts from the University of Southern California (RAP) [3], the University of Massachusetts (Proxy Prefix Caching) [13], and the University of Minnesota (Video Staging) [17]. Unlike the

CDNs, these proxy-based mechanisms partially cache video data as it is streamed from the server to the proxy and through to the client. Additional requests to retrieve a video stream from the proxy will result in missing data being filled in at the proxy.

For our work, we assume that the proxies are far enough from the client to achieve a reasonable hit rate to the video data. As an example, the proxy could serve an entire campus (such as Ohio State). The streaming mechanism that we have developed can be used to deliver data from the data warehouses in the CDNs to the clients or can be augmented to include the techniques that have been developed for proxy-based video distribution networks.

2.2 Video Streaming Protocols

There are a large number of streaming protocols that have been developed for point-to-point video distribution. For sake of brevity, we will purposely omit discussion of techniques developed for the distribution of video over multicast networks such as [15]. For point-to-point video streaming, there are a tremendous number of streaming algorithms including the Windows Media Player, RealPlayer, the Continuous Media Toolkit (CMT) [12], Vosaic [2], the OGI Media Player[16], and the priority-based streaming algorithm [4].

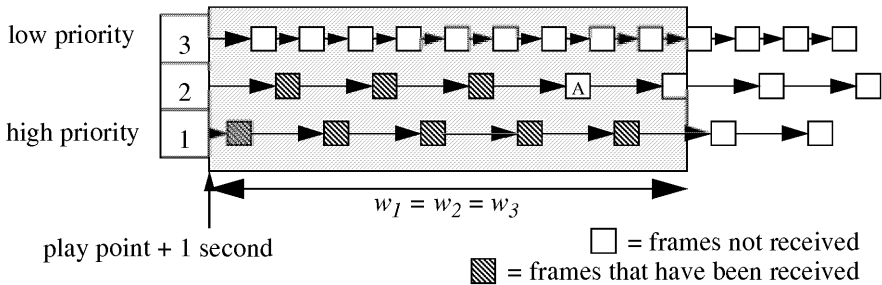


Fig. 1. This figure shows an example of the priority-based streaming algorithm. The window keeps track of which frames are buffered in the client. Using this window, the server always attempts to transmit the highest priority frame within the window that has not yet been sent. In this example, the first unsent frame in priority level 2 is transmitted next (labelled A)

These algorithms are for the most part network congestion sensitive, with some being arguably more TCP-friendly than others. In addition, most of these streaming algorithms focus on local adaptation of video sequences to network bandwidth availability and have either no or limited retransmissions of missing data.

The work proposed here uses buffer management uniquely by controlling the frame rate and quality based on a hysteresis loop which leads to a stabilized frame rate while using TCP as its transport protocol. We briefly describe the streaming mechanism here as it will be needed in the next section. The proposed algorithm prioritizes all the frames within a given video. This assignment can be assigned to gracefully degrade the frame rate as bandwidth becomes scarce. It does so by assigning frames to various priorities within a window (just ahead of the playpoint). This window is used to “look

ahead” into the movie at least a minute into the future. Using this window, it then delivers all frames in the window at the highest priority before delivering the lower priority frames within the window. As a result it is able to gracefully degrade the quality of video during times of congestion. An example of the windowing mechanism is shown in Figure 1.

3 Proposed Adaptive Streaming Mechanism for TCP Delivery of Stored Content

In this section, we propose a mechanism for the delivery of stored video content across best-effort networks. Before describing the algorithm, we first describe some basics of our approach including (i) some of the underlying assumptions that we make, (ii) a discussion of the trade-off between quality, frame rate, and buffering, and (iii) a metric for the evaluation of streaming media.

3.1 Basics

3.1.1 Underlying Assumptions

There are a number of unique aspects of our approach that we believe are important for the delivery of stored video content over best-effort networks. These characteristics are described below (some more contentious than others):

- **Moving towards more bursty video sources** - As the bandwidth over the network increases, the ability to stream higher-bit-rate MPEG streams becomes possible. This increase in bandwidth will create streams of greater bandwidth diversity than we currently see on the Internet.
- **Buffering at the Receiver** - We believe that buffering limitations in the client end systems are becoming a non-issue. We believe this because disk and memory capacity is outpacing the growth in bandwidth available to a single flow (not in aggregate). Thus, we can focus on getting the video data into the buffer without worry of buffer overflow.
- **TCP transport layer** - We believe that stored video streaming should happen over the TCP protocol. This is perhaps the most contentious part, but there are several reasons we believe this. First, TCP is a widely deployed transport protocol. This allows applications that are built on this architecture to be deployed rapidly over a large scale. Second, it allows us to resolve the TCP-friendliness issue rather easily: *it is TCP-friendly*. Third, we do want flow/congestion control *as well as retransmissions*. Flow/congestion control are fairly obvious. The reason we want retransmissions for stored video is that we are planning the delivery of the stored sources well in advance (even for B-frames), thus, when we transmit a frame we want to receive it in whole at the receiver.

3.1.2 Quality and Frame Rate Adaptation

For the delivery of stored video content, there are number of options available for adaptation over best-effort networks. Using these, we can control the quality of the video stream, the frame rate of the video stream, and the ordering in which the video data is streamed across the network. In fact, each of these can be adapted in isolation

or combined together in the delivery of video content. It is our belief that while any quality video can be used at any frame rate, we believe that there are *operating points* that will be used to correlate frame rate and frame quality.

Operating points define interesting sets of combinations of frame rate and quality. The streaming algorithms we develop are constrained to move from pre-determined operating points to other pre-determined operating points. Switching between operating points (as the bandwidth availability changes over the network) is guided by the principle of *equal priority* to frame-rate and quality. In other words, we choose not to compromise one parameter with respect to the other. Therefore any degradation/improvement in the streaming status would trigger comparable degradation/improvement in both the parameters. This is a fairly common approach as found in the Quasar streaming model as well as industrial models such as the Real Networks SureStream mechanism [10]. For frame rate adaptation, we allow the frame rate to change within a small set of values (e.g., 14-17 frames per second) for a fixed video quality level. Any further deviation from this frame rate will trigger the system to move to a different quality stream.

As an example of our philosophy, the operating point of (30 frames per second, low video quality) is not a reasonable operating point since we are severely sacrificing quality to achieve a high frame rate. Instead streaming at (15 frames per second, medium video quality) is a better choice in that it attaches equal importance to frame-rate and quality. For the rest of this work, we assume that multiple copies of the video stream are available at the encoded bit-rates, similar to the delivery mechanisms in the RealNetworks SureStream and the Oregon Graduate Institute streaming player. One could imagine with appropriate computing power that a transcoder could be implemented to do the same thing on-the-fly.

3.1.3 Effective Frame Rate

Our proposed video streaming algorithm attempts to maximize the frame rate displayed to the client, while minimizing the variability or the “jerkiness” in the display frame rate. While we believe that it is ultimately up to the end user to decide how aggressive or conservative the streaming algorithm is (while still being TCP compliant!), we believe it is important to define a metric that adequately measures this objective function in order to evaluate the performance of the algorithm.

In the past, researchers have often measured the average and variation in the frame rate delivered to the client as metrics for performance. While there is typically a strong correlation between standard deviation and jerkiness, this may not necessarily hold true. For example, consider a stream displayed at 10 frames per second for half the duration of the video, and at 20 frames per second for the remaining half. Additionally, consider a stream displayed alternately at 10 and 20 frames per second throughout the length of the video. The average frame rate and standard deviation in both cases is the same, but the perceived “jerkiness” in the former is much less than that in the latter.

As another example, we have graphed the frame rate delivered for a video sequence using the *priority based streaming algorithm* from reference [4] in Figure 2 (a). In Figure 2 (b) we have merely sorted the frame rate points making up figure (a). Again, it is clearly evident that the user would perceive much more “jerkiness” with the frame rate delivered in figure (a) than with (b). However, the standard deviation in both cases is the same (4.62 frames).

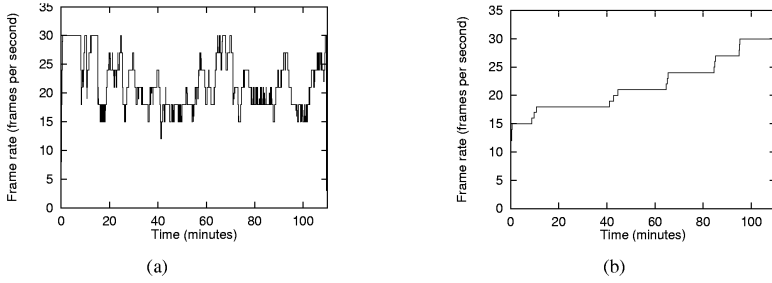


Fig. 2. (a) shows an example of the frame rate delivered by the priority-based streaming algorithm. (b) shows the same frame rate points but sorted in ascending order. Both figures exhibit the same frame rate and frame rate variation.

We propose a metric that accounts for the jerkiness in the display frame rate more appropriately. We define *Effective Frame Rate (EFR)* as follows

$$EFR = FPS_{avg} - \frac{W}{n-1} \sum_{i=2}^n |frame_{rate}_i - frame_{rate}_{i-1}|^P$$

where FPS_{avg} is the average frame rate, $frame_{rate}_i$ is the number of frames displayed in the i^{th} second of the video clip, and W and P weighting factors that we will describe shortly. The idea behind *EFR* is that it first measures the average frame rate delivered to the user and then *penalizes* the *EFR* for variations in the frame rate delivered.

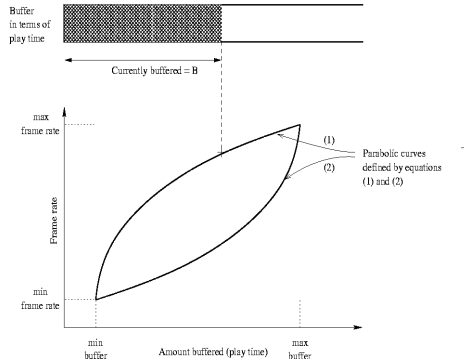


Fig. 3. Pictorial representation of the hysteresis scheme. Two parabolas shown determine how frame-rate should change when buffer level changes. Parabola P1 has the property that the frame-rate drops rapidly with buffer when it is close to the min buffer mark. On the other hand when the buffer level is close to the max buffer mark and dropping, the frame-rate does not drop as rapidly, thereby taking advantage of buffer. Parabola P2 has the property that when buffer level is close to the min buffer mark and rising, the frame-rate does not increase as rapidly, thereby building the buffer. On the other hand when the buffer level is close to the max buffer mark and rising the frame-rate increases at a higher rate thereby greedily utilizing excess bandwidth.

The penalty in the calculation of the frame rate is determined by two parameters in the metric, W and P . These two terms determine the nature of the penalty to be applied when the frame rate changes. The super linear term (P) is added to penalize

large changes in the frame rate more than small changes which in some cases may even be imperceptible. The linear function(W) is used to weigh the penalty for changes in frame rate. These two terms can be set to user preferences to allow one to evaluate the algorithms based upon their own idea of what makes a good streaming algorithm.

3.2 Hysteresis Based Video Adaptation

In this section, we describe our proposed approach. Our model first assigns each of the frames in the video stream to a priority level for streaming. As a simple case, one might assign all the I-frames in an MPEG sequence to the highest priority level, all P-frames to the second highest priority, and all B-frames to the lowest priority. Thus, no B-frame will be delivered before it's P-frame has been delivered. Also note that because we are using TCP as the transport protocol, we are guaranteed that the B-frame will have its reference frames in at the client. While we have described a very simple mapping here, our system is able to assign arbitrary priorities to frames depending upon the user's preferences. For example, one might map every alternating B-frame to a higher priority than the rest of the B-frames. In this case, during times of low bandwidth availability, every other B-frame will be delivered, spacing out the loss of frames. Finally, we create static priorities for all operating points for the movie that is stored.

3.3 Proposed Approach

Once we have the priority assignments, the next goal is to determine a plan for the delivery of the stored video. We observe that buffer occupancy is a measure of the frame rate and quality that can be supported. A shrinking buffer indicates either a decrease in average bandwidth and/or increase in frame sizes in the video. Similarly expanding buffer indicates increase in average bandwidth and/or decreasing frame sizes in the video.

The proposed approach is based on a hysteresis model. The streaming itself consists of two phases. First one is the prefetch phase wherein sufficient video is prefetched. In the second phase, the remaining video's play-time is divided into fixed length intervals. At the beginning of each time interval the streaming parameters, frame-rate and quality, are dynamically determined. For this a hysteresis based buffer monitoring mechanism is used.

The hysteresis model is pictorially represented in Figure 3. The algorithm tries to maintain the frame-rate constant when the buffer size lies between the two parabolas. Initially we use the parabola P1 to determine the frame-rate. Subsequently for each interval we check the buffer level to determine the next set of streaming parameters. For increasing buffer levels we use parabola P2 and for decreasing buffer levels parabola P1 is used. The parabolas P1 and P2 are defined as follows

$$f = f_{min} + (f_{max} - f_{min}) \cdot \sqrt{(B - b_{min} / b_{max} - b_{min})} \quad (P1)$$

$$f = f_{min} + (f_{max} - f_{min}) \cdot (B - b_{min} / b_{max} - b_{min})^2 \quad (P2)$$

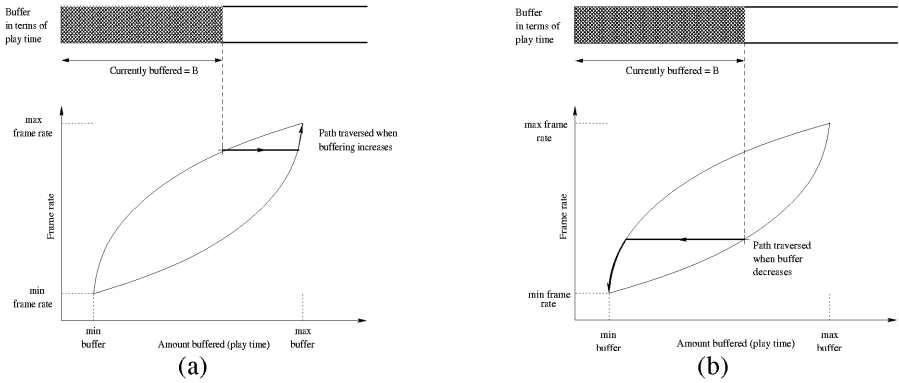


Fig. 4. Using parabola P1: For increments in buffer level the frame-rate is held constant at the previous value until the buffer occupancy hits parabola P2. Then P2 is used to recalculate frame-rates from this point when-ever buffer level increases. Using parabola P2: For drops in buffer level the frame-rate is held constant at the previous value until the buffer occupancy hits parabola P1. Then P1 is used to recalculate frame-rates from this point whenever buffer level decreases.

where

f is the new operating point. The new operating point depends on the following parameters.

B is the current buffer occupancy.

b_{\min} is the minimum buffer occupancy allowed. This is same as the lower threshold of the hysteresis loop. It's typical value might be half a minute of video.

b_{\max} is the maximum buffer occupancy allowed. This is usually the maximum amount of buffer available to the application. This is same as the upper threshold of the hysteresis loop. It's typical value is three to 5 minutes of video.

f_{\max} is the maximum frame rate. Typically this is set to maximum which is 30 frames per second.

f_{\min} is the minimum frame rate permitted. This is a user configurable parameter and indicates minimum frame rate which is considered reasonable by the user. Algorithm tries to maintain atleast this frame rate.

Assume we are currently using parabola P1 to make the streaming decisions. Figure 4(a) shows this scenario. Depending on the direction in which buffer occupancy changes, there can be two possible cases for the next interval.

- If the buffer level decreases then we use the current parabola-in-use (P1) to trace it downward.
- If buffer level does not decrease then we continue with the current frame-rate for the future intervals as long as the buffer level does not hit the parabola P2.

When the buffer occupancy hits parabola P2, this becomes the active parabola and the next streaming frame-rate parameter is decided based on this curve. Figure 4 (b)

shows this scenario. Here again, depending on the changes in buffer occupancy, there can be two possible cases for the next interval.

- If the buffer level increases we use the current parabola-in-use (P2) to climb upwards.
- If the buffer level does not increase then we hold to the current frame-rate as long as the buffer level does not hit parabola P1.

The motivation behind this is to achieve a constant frame-rate and quality video as long as the buffer occupancy remains between upper and lower thresholds of the hysteresis loop determined by parabolas P1 and P2. Frame rate is changed only when the buffer occupancy improves or degrades substantially. The hysteresis mechanism takes care of bursts of frames with large variation in sizes in the video by not changing the frame rate and the quality immediately. The maximum buffer limit and the minimum frame-rate are the user defined parameters. A higher maximum buffer limit allows more amount of video to be buffered thereby allowing a higher degree of smoothing of frame rate. The algorithm aims at building the buffer in advance to deal with a potential burst of larger sized frames in the future. The algorithm does not build too much of buffer. When the buffer occupancy increases beyond a threshold it increases the frame rate and/or quality of the video. The algorithm also provides cushion to the variations in bandwidth, immunity to short gaps in bandwidth and variations in the round trip times of the underlying transmission route by appropriately setting the lower and upper thresholds for the buffer occupancy.

4 Experimentation

In this section, we describe some of the performance results of our streaming algorithm compared with simple streaming algorithms that perform rate changes based on per-second adaptability and the priority-based streaming algorithm which uses windows that are in the size of minutes. We will also explore the use of the *effective frame rate* metric for measuring a streaming algorithm's performance. Before we describe our experimental results, however, we will describe our experimental simulation environment and the video data and network traffic traces that we captured.

4.1 Experimental Environment

We have captured and compressed two full-length movies (*Matrix* and *E.T. - the Extra Terrestrial*) into the MPEG compressed video format for use in the simulations¹. For the movie *Matrix*, we used Future Tel's PrimeWare hardware digital video compression board. This board captured the analog output of a DVD player and compressed the movies into MPEG-2 video format. Using this board, we captured four streams at varying qualities for the simulations. For the movie *E.T. - the Extra Terrestrial*, we already had an MPEG-1 video stream compressed into three different

¹ The video and network traces will be made available via the web site: <http://www.cis.ohio-state.edu/~wuchi/Videos>.

quality levels and are using those streams here. The statistics for the video trace data are shown in Table 1.

Table 1. This figure shows the statistics for the video traces that were used in the experimentation of the various algorithms

Movie	Ave. Bit Rate (Mbps)	Frames	Length (Minutes)	Ave. Frame Size	Ave. I-frame Size	Ave. P-frame Size	Ave. B-frame Size
<i>Matrix</i>	3.5	244301	129	13562	38341	16175	10039
<i>Matrix</i>	5	244301	129	19335	48686	24275	14424
<i>Matrix</i>	6.5	244301	129	24724	139733	28439	11736
<i>Matrix</i>	8	244301	129	30049	139733	39706	15217
<i>E.T.</i>	0.233	204679	113	965	4408	695	172
<i>E.T.</i>	0.755	204679	113	3126	12883	2435	860
<i>E.T.</i>	1.559	204679	113	6459	24591	6152	2004

To provide deterministic results of the various algorithms, we captured 2 two-hour long TCP traces. One was captured from the University of Michigan to Ohio State and the other was captured locally between two machines on different sub-networks. Thus, the comparisons that we draw in this paper are based upon the TCP traces. For actual streaming algorithms that just use UDP with or without flow control and with or without retransmissions may be slightly different than those presented here. We also note that comparing the performance of streaming algorithms that receive partial data (as in a UDP stream) is extremely difficult as it is hard to compute what the relative impact of receiving half a frame of video is for any streaming algorithm.

For comparison purposes, we use two streaming algorithms for comparison. The first, which we will refer to as the *naive* algorithm, simply calculates the bandwidth transmitted over the last several seconds to calculate the closest operating point for which to stream. This algorithm is very similar to the algorithms found in the CMT toolkit out of Berkeley or the Oregon Graduate Institute player. The *Priority-Based Streaming algorithm* is an extension to the basic *naive* algorithms that uses a much larger lookahead window. Here, the lookahead was set to two-minutes, meaning that all high-priority (I-frames) must be received before the algorithm starts delivering lower priority-frames (P and B). Thus, we expect the priority-based algorithm to provide substantially smoother operation than the *naive* algorithm. These two algorithms are explained in more detail in reference [6]. Finally, we note that while we are using simulation-based experimentation, all algorithms were not allowed to “look” at the future bandwidth availability from the network trace.

4.1.1 Experimental Results

In the rest of this section, we present several simulation results that demonstrate the effectiveness of the hysteresis-based streaming algorithm and that show how the *effective frame rate* metric that we propose work. Figure 5 shows the frame rate achieved by the naive, priority-based, and hysteresis-based streaming algorithms for the movie *Matrix* and the bandwidth trace shown in Figure 5 (a).

We see here that because the bandwidth availability drastically changes over time and the compressed video streams' network requirements vary significantly over time, the naive algorithm (as shown in Figure 5 (b)) has a very hard time sustaining a smooth frame rate for any significant period of time. This is because planning does not occur on anything greater than second-level in time making the frame rate delivered highly sensitive to the bandwidth requirements and network availability. We also note that the frame rate delivered by the naive algorithms is actually somewhat smooth over very small time scales but because we are displaying over 100 minutes in the figure, it appears significantly more bursty. In comparison, the priority-based streaming algorithm with a window of one-minute attempts to very conservatively change the frame rate over time by making sure a minute's worth of video at higher priority layers has been delivered before transmitting the lower priority frames. As shown in Figure 5 (c), the frame rate does not vary as drastically over time but is still somewhat bursty on a medium-term time-scale. We also see that the priority-based streaming algorithm is more conservative than the other algorithms in that it very slowly increases the frame rate of the video stream even when a higher frame-rate is possible. The main reason that the frame rate is still somewhat bursty is that the bandwidth requirements and bandwidth availability are changing over time, making it extremely difficult to track the available bit-rate that needs to be sent. Finally, the hysteresis-based streaming algorithm (as shown in Figure 5 (d)) does a much better job of smoothing out the frame rate delivered to the client. Buffering requirements are bounded by upper and lower thresholds and they also smooth out the changes in network bandwidth while keeping the bandwidth demand from the application somewhat more constant.

Applying our *effective frame rate* measurement to the algorithms, we have listed the various EFR measurements for a variety of W and P values in Table 2. By doing so, we can describe how the metric works with an example streaming session shown in Figure 2. For the parameters ($W=0$, $P=1$), no penalty is incurred by the algorithms for bursty frame behavior. As a result, the metric is a measure of how many frames were delivered. We see that under these parameters, the naive algorithm does the best. This is somewhat intuitive as the naive algorithm is very greedy. When there are small frames that are currently being transmitted, the algorithm fetches as many of them as it can, while the other approaches attempt to prefetch some of the larger frames in the future knowing that they need to prepare for these regions. We also see in Figure 2 that the priority-based algorithm is extremely conservative in its frame rate delivery to the client, with a consistently lower average frame rate. Under ($W=1$, $P=1$), the algorithms are penalized linearly for having bursty frame rates over time. The penalty here is the average amount of rate change between consecutive second intervals. It is important to note that simply achieving a constant frame rate is not the goal as it will undoubtedly result in a very low average frame rate in the first term of the EFR. Finally, we see that as the weight penalty increases the hysteresis-based approach does even better.

Figure 6 shows the delivery of the *Matrix* using the second bandwidth trace where the bandwidth availability is greater than in the previous experiment. We see here that the naive algorithm is able to achieve the full 30 frames per second delivery quite a number of times. We also see the same type of response as previously from the various algorithms. That is the priority-based algorithm is somewhat conservative,

while the hysteresis-based approach is able to smooth out the frame rate over considerable time-scales. The EFR measurement for the algorithms shown in Table 3.

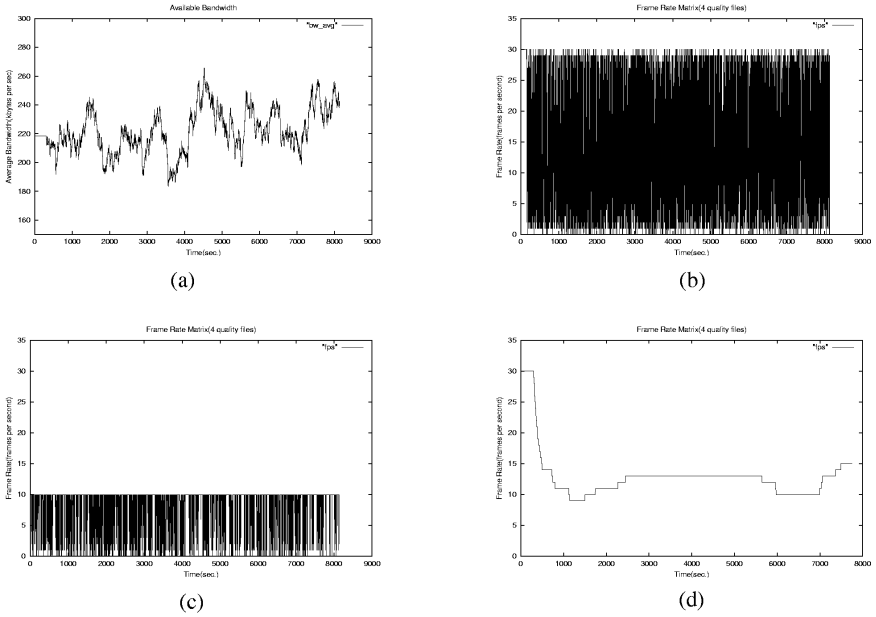


Fig. 5. This figure shows the frames rates obtained from the various algorithm for the movie *Matrix* using the bandwidth trace used in (a). Figures (b) through (d) show the frame rates of the naive algorithm, the simple priority-based algorithm, and the hysteresis-based approach for the entire duration of the movie. The variability in (d) is significantly lower then the other two algorithms.

Table 2. The effective frame rate for *Matrix* for frame rates shown in Figure 5.

Configuration	Naive	Priority-Based	Proposed approach
W = 0,P = 1	13.74	7.6	12.97
W = 1,P = 1	5.15	5.27	12.96
W = 3, P = 1	-12.03	.62	12.95
W = 1, P = 1.5	-18.22	1.84	12.96
W = 1, P = 2	-113.47	-7.42	12.96

As a final example, to show what happens when we change the movie to *E.T.* we have graphed the results in Figure 7. Finally, the EFR figures for the movie *E.T.* (shown in Figure 7) are shown in Table 4.

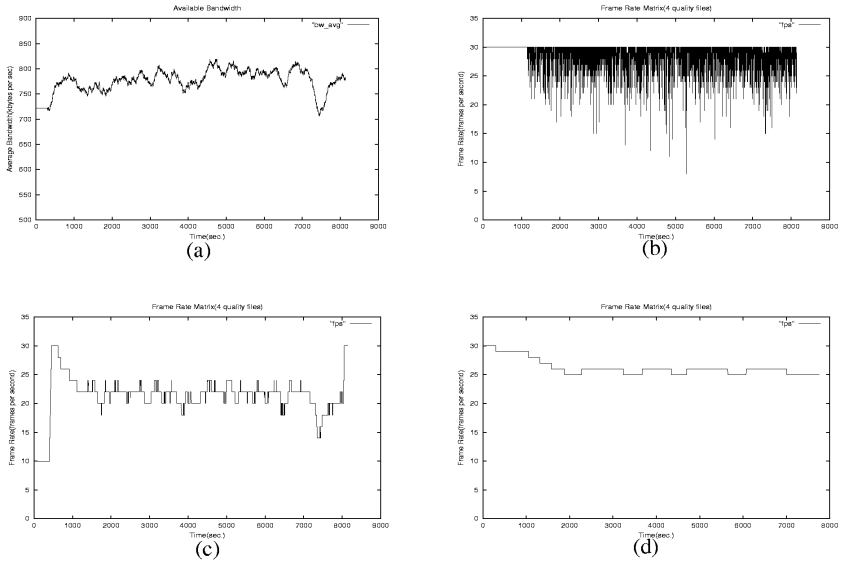


Fig. 6. This figure shows the frames rates obtained from the various algorithm for the movie *Matrix* using the bandwidth trace used in (a). The figures (b) through (d) show the frame rates of the naive algorithm, the simple priority-based algorithm, and the hysteresis-based approach. The variability in (d) is significantly lower then the other two algorithms

5 Conclusion

In this paper, we have introduced a hysteresis-based streaming algorithm that uniquely uses buffer management in the streaming of stored video data over best-effort networks. As we have shown, this approach is much more effective in smoothing out the variability in frame rate delivered to the user. We have also introduced an *effective frame rate* metric to evaluate the effectiveness of various stored video streaming algorithms. The key concept here is that instead of measuring just the average frame rate and the variability in frame rate delivered, it calculates an

Table 3. The effective frame rate for *Matrix* for frame rates shown in Figure 6.

Configuration	Naive	Priority-Based	Proposed approach
$W = 0, P = 1$	28.75	21.24	26.24
$W = 1, P = 1$	27.55	21.17	26.23
$W = 3, P = 1$	25.15	21.01	26.23
$W = 1, P = 1.5$	26.48	21.14	26.23
$W = 1, P = 2$	23.84	21.11	26.23

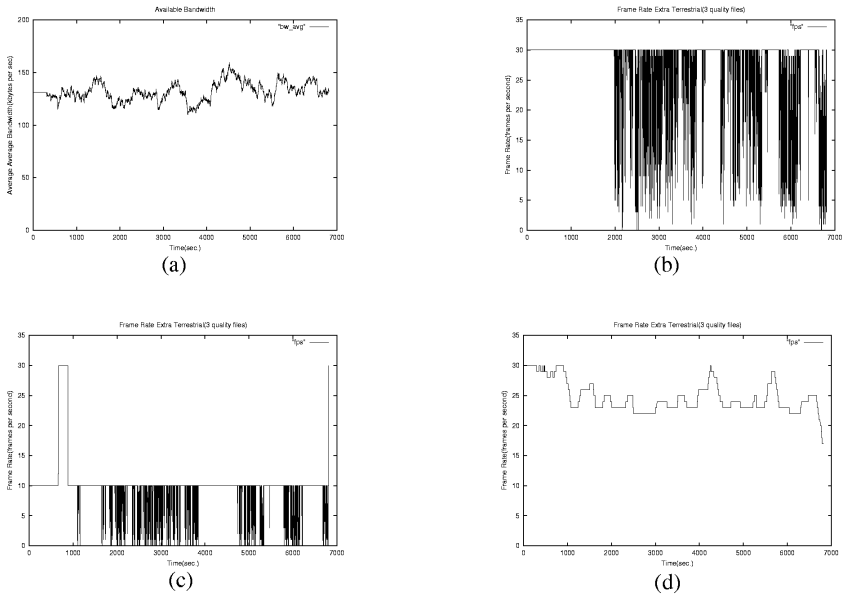


Fig. 7. This figure shows the frames rates obtained from the various algorithm for the movie *Extra Terrestrial* using the bandwidth trace used in (a). The figures (b) through (d) show the frame rates of the naive algorithm, the simple priority-based algorithm, and the hysteresis-based approach. The variability in hysteresis-based approach is significantly lower then the other two algorithms.

average frame rate and then penalizes the average frame rate depending on how quickly the frame rate changes at small time scales. Finally, we have captured several MPEG-1 and MPEG-2 video streams as well as two network bandwidth traces for simulation, all of which will be made available via the Web.

Table 4. The effective frame rate for *Extra Terrestrial* frame rates shown in Figure 7

Configuration	Naive	Priority-Based	Proposed approach
$W = 0, P = 1$	26.54	9.42	24.75
$W = 1, P = 1$	24.24	8.29	24.74
$W = 3, P = 1$	19.65	6.02	24.71
$W = 1, P = 1.5$	19.12	6.62	24.74
$W = 1, P = 2$	0.42	2.11	24.74

We are currently working towards incorporating semantic information into this streaming model. That is, all frames are created equal in this approach. One can imagine that if all the semantic information from the stream can be automatically extracted that some scenes within the video sequence might have higher priority than others.

References

1. Shanwei Cen, Jonathan Walpole, Calton Pu, "Flow and Congestion Control for Internet Media Streaming Applications", in *Proceedings of SPIE Multimedia Computing and Networking 1998*, pp 250-264.
2. Zhigang Chen, S.M. Tan, R. Campbell, Y. Li, "Real Time Video and Audio in the World Wide Web", in the *Fourth International World Wide Web Conference*, Boston, Massachusetts, December 1995.
3. E. Ekici, R. Rajaie, M. Handley, D. Estrin, "RAP: An End-to-end Rate-based Congestion Control Mechanism for Realtime Streams in the Internet," in *Proceedings of INFOCOM 99*.
4. W. Feng, M. Liu, B. Krishnaswami, and A. Prabhudev, "A Priority-Based Technique for the Best-effort Delivery of Stored Video", in *Proc. of the SPIE Multimedia Computing and Networking*, January 1999.
5. Wu-chi Feng, S. Sechrest, "Smoothing and Buffering for Delivery of Pre-recorded Compressed Video", in *Proceedings of IS&T/SPIE Multimedia Computing and Networking*, Feb. 1995, pp. 234-242.
6. Wu-chi Feng, S. Sechrest, "Critical Bandwidth Allocation for the Delivery of Compressed Pre-recorded Video", *Computer Communications*, Vol. 18, No. 10, Oct. 1995, pp. 709-717.
7. J. Mahdavi and S. Floyd. TCP-friendly unicast rate-based flow control. Note sent to end2end-interest mailing list, Jan 1997
8. Steve McCanne, V. Jacobson, "VIC: A Flexible Framework for Packet Video", *Proceedings of ACM Multimedia 1995*, November 1995.
9. P. Nee, K. Jeffay, and Gunner Danneels, "The Performance of Two-Dimensional Media Scaling for Internet Videoconferencing", in *Intl. Workshop on Network and Operating System Support for Digital Audio and Video*, May 1997.
10. RealNetworks - <http://www.real.com>
11. I. Rhee, "Error control techniques for interactive low-bit rate video transmission over the Internet", in *Proc. SIGCOMM, 1998*
12. L.A. Rowe, K. Patel, B.C. Smith, K. Liu, "MPEG Video in Software: Representation, Transmission and Playback", in *Proc. of IS&T/SPIE 1994 Int'l Symp. on Elec. Imaging: Science and Technology*, San Jose, CA, February 1994.
13. S. Sen, J. Rexford, and D. Towsley, "Proxy prefix caching for multimedia streams", in *Proceedings of INFOCOM 99*, April 1999.
14. M. Talley and K. Jeffay, "Two-Dimensional Scaling Techniques For Adaptive, Rate-Based Transmission Control of Live Audio and Video Streams", in *Proceedings of the Second ACM International Conference on Multimedia*, San Francisco, CA, October 1994.
15. Brett J. Vickers, Célio Albuquerque and Tatsuya Suda, "Source-adaptive multilayered multicast algorithms for real-time video distribution", *IEEE/ACM Transactions on Networking*, December 2000
16. Jonathan Walpole, Rainer Koster, Shanwei Cen, et. al, "A Player for Adaptive MPEG Video Streaming Over The Internet", in *Proc. 26th Applied Imagery Pattern Recognition Workshop*, Washington DC, October 15-17, 1997.
17. Zhi-Li Zhang, Yuewei Wang, David H. C. Du and Dongli Shu, "Video staging: a proxy-server-based approach to end-to-end video delivery over wide-area networks", *IEEE/ACM Transactions on Networking*, Aug. 2000

A Transmission Scheme for Streaming Variable Bit Rate Video over Internet

Sunghoon Son

Electronics and Telecommunications Research Institute,
305-350 Taejon, Korea
sonsh@etri.re.kr

Abstract. In this paper, we consider transmission of variable bit rate (VBR) stored video for distributed streaming service. In streaming service over Internet, video server usually employs network congestion control protocol in order to adapt the dynamic change of network traffic. This protocol often results in the discontinuity of playback at client's side. We propose a novel transmission scheme to overcome this problem. Under the proposed scheme, video server prepares basic transmission schedule of video by off-line processing of VBR video. Using this information, server can control the amount of data transmitted, which enables client to avoid the discontinuity in playback, even if the allocation of network bandwidth changes during service.

1 Introduction

Recent advances in hardware technologies have made possible many distributed continuous media applications. Video server is one of the most essential component in these applications. A video server stores a large amount of video data on a large array of high-capacity storage devices. On receiving a playback request from a remote client, the video server services the client by retrieving the requested video data and then transmitting it to the client's site.

Video server must carefully transmit video data due to the real-time requirement of video playback. Network bandwidth used in streaming is one of the most important resources that video server manage. Since video streams are usually encoded using variable bit rate (VBR) encoding scheme such as MPEG standard [1], the amount of data needed in unit time for playback varies in time. This makes the problem much more complicated.

The usage of network bandwidth is typically affected by the transport protocol that video server uses. A distinguishing feature of media streaming protocols is its feedback control mechanism between server and client to dynamically adjust transmission rate according to network traffic [2],[3]. Under the current Internet environment, an open standard such as Real-time Transport Protocol (RTP) is commonly used protocol for the real-time transmission of video data. RTP is an IP-based protocol providing support for the transport of real-time data such as video [4]. RTP is designed to work in conjunction with the auxiliary control protocol Real-time Transport Control Protocol (RTCP) to get feedback on the

quality of data transmission and information from participants in the sessions. Using RTP/RTCP, video server can change transmission rate dynamically based on the quality feedback from clients. Therefore a transmission scheme employed by video server should be aware of the operation of RTP/RTCP and dynamic QoS control mechanism based on the protocol.

In this paper, we consider the problem of continuous transmission of VBR video with the possible change of transmission rate during service. Especially we propose a new transmission scheme of VBR video considering feedback control mechanism for streaming over Internet. The proposed scheme first generates a continuous transmission schedule for a given video using fixed amount of network bandwidth reservation, which smoothes the bursty VBR video to reduce the rate variability. Then, another transmission technique is applied to the smoothed transmission schedule to adapt to the change of bandwidth allocation. All the procedures can be processed off-line. Some experimental results are given to demonstrate the performance of the proposed scheme.

The rest of this paper is organized as follows. In section 2, we introduce the notion of the continuous transmission and provide the review of the various VBR video transmission schemes. In section 3, we propose a new transmission scheme and its admission control algorithm that can adapt to the adjustment of transmission rate. Performance study is carried out in section 4. Conclusion is given in section 5.

2 Continuous Transmission for VBR Video

A video server transmits video frames periodically in *round* due to the periodic nature of video data [5]. The operation of client, the playback of video, is also periodic. A predefined number of video frames should be available at client's buffer during each round in order to guarantee the continuous playback. A video server often employs *transmission scheduler* to determine when and how much data should be sent to the client. Given a fixed amount of network bandwidth, a video server can accommodate only limited number of video streams concurrently. Before admitting a new client, a transmission scheduler must use an admission control procedure to verify whether there is enough network bandwidth available for the new client.

In case of stored VBR video, a priori information, such as a sequence of frame sizes, may be utilized to transmit VBR video frames efficiently. Using this information, the transmission scheduler can generate more efficient *transmission schedule*. Given a VBR video, transmission scheduler must determine the amount of video data to be transmitted during each round with no buffer starvation or buffer overflow at client site. A transmission scheduler must meet the following constraints for *continuous transmission schedule* it generates (See Table 1 for the definition of these quantities):

$$C(t) \leq P(t), 0 \leq t \leq n \quad (1)$$

$$P(t) - C(t) \leq B_{max}, 0 \leq t \leq n \quad (2)$$

Table 1. Notations

Notation	Definition
n	The length of video in time
t	Time index
t_d	Initial playback startup delay
$c(t)$	Frame size at t , i.e. the amount of data consumed (displayed) by client at t
$C(t)$	The cumulative amount of data consumed by client during $[0, t]$, $C(t) = \sum_{i=1}^t c(i)$
$p(t)$	The amount of data transmitted by server at t
$P(t)$	The cumulative amount of data transmitted by server during $[0, t]$, $P(t) = \sum_{i=1}^t p(i)$
$b(t)$	Client's buffer level at t
B_{max}	Client's maximum buffer size
r	Network bandwidth allocated to the stream

$$P(t+1) - P(t) \leq r, 0 \leq t \leq (n-1) \quad (3)$$

(1) is the continuous transmission constraint, which means that the server always transmits more data than the client consumes for lossless and starvation-free playback. (2) is the client's buffer constraint, which means that the server transmits carefully not to overflow the limited client's buffer. And, (3) is the network bandwidth constraint, which means that the fixed amount of network bandwidth is allocated to each stream. Fig. 1 depicts various continuous transmission schedules that meet the above three constraints. Any non-decreasing lines between VBR trace (the thick solid line) and client's buffer level (the thick dotted line) can become continuous transmission schedules.¹

A lot of schemes have been proposed for the continuous transmission of VBR video. They mainly focused on the utilization of network bandwidth, minimization of rate variability, reducing buffer requirement, and so on. [6] demonstrates the fundamental limits and tradeoffs of providing a deterministic guarantee to VBR video traffic of live video traffic, which does not consider the stored property. In [7], an optimal smoothing scheme was proposed. The scheme is optimal in that it minimizes the client's buffer size and the rate variability of the VBR data. The authors tried to obtain statistical multiplexing gain too, but the statistical multiplexing gain from the optimally smoothed traffic is the lower bound of statistical multiplexing gain of the original traffic. The scheme proposed in [8] employs Constant Bit Rate (CBR) transmission of VBR data. Since the required bandwidth is fixed, multiplexing and admission control are very simple, which results in the reduced server load and no cell loss due to overload. However, this scheme tends to transmit a large amount of data unnecessarily because it always

¹ Although the operation of a video server is based on discrete time cycle, we will hereafter use continuous time notation to clearly convey the idea

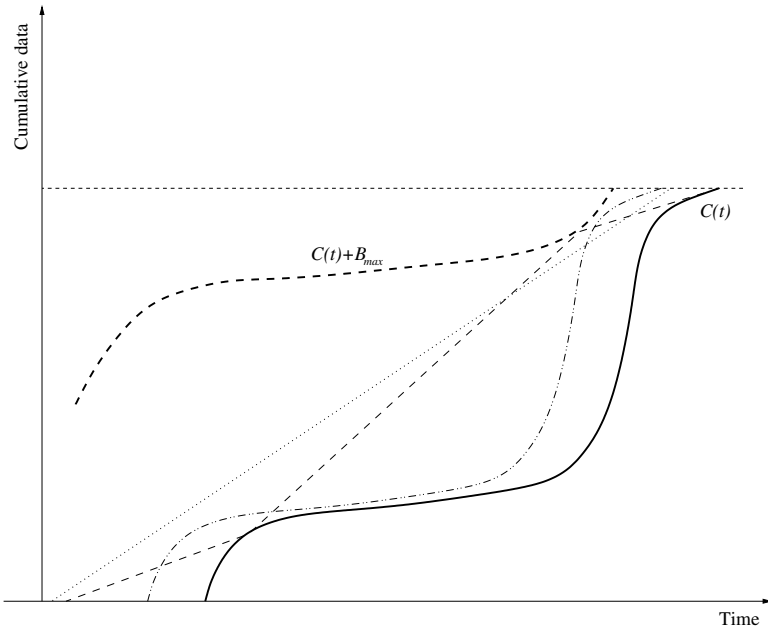


Fig. 1. Continuous transmission of VBR video and various transmission schedules

uses the entire bandwidth allocated to a stream, resulting in a large client buffer required.

3 Continuous Transmission under Bandwidth Adjustment of CBR Channel

In this section, we consider the continuous transmission of VBR video with the possible change of transmission rate during service. The scheme first generates a continuous transmission schedule for a given video trace based on the fixed amount of network bandwidth reservation, which smoothes the bursty VBR video to reduce the rate variability. Next, another transmission technique is applied to the transmission schedule to adapt to the change of bandwidth allocation. All the procedures described in this section can be processed off-line.

We assume that the video server uses CBR channel to transmit VBR video. We further assume that, when network traffic congestion occurs, the network bandwidth allocation of each stream may change during transmission due to the bandwidth renegotiation between client and server. We also assume that the client have limited buffer space which is usually less than a few mega bytes.

3.1 The Basic Transmission Schedule

In this section, we introduce a transmission technique using CBR channel, which forms the basis of the transmission scheme considering bandwidth adjustment. The main idea is to transmit some large frames in advance when small frames are transmitted. This inevitably requires extra buffer space at client site.

In order to minimize the required buffer space at client for a given transmission rate, scan a VBR video's sequence of bit-rate backward from the end. Given a video trace, $c(t)$, and network bandwidth, r , the client's buffer level, $b(t)$, at each frame time can be calculated by

$$\begin{aligned} b(n) &= 0 \\ \delta(t) &= c(t) - r + b(t+1), \quad t = (n-1), (n-2), \dots, 1 \\ b(t) &= \begin{cases} \delta(t), & \text{if } \delta(t) \geq 0 \\ 0, & \text{otherwise} \end{cases} \end{aligned} \quad (4)$$

By starting at the last of the video frame, calculating backward using (4), we can increase the client's buffer level by the exact amount needed.

Fig. 2 illustrates an example of the basic transmission schedule obtained by the above procedure (the thick line in the figure). The intuition behind the figure can be explained as follows. Given a video's cumulative curve, $C(t)$, assume a straight line L with the slope r . As we move L right to left along the x-axis (time axis), L and $C(t)$ can adjoin at several points. For example, in the figure, L and $C(t)$ adjoin at t_3 and L intersects $C(t)$ at t_2 at the same time. In this case, the transmission during the interval $[t_2, t_3]$ should be at the rate r . The reason is that there exists a point t in $[t_2, t_3]$ such that the (average) bit rate in $[t, t_3]$ is higher than the bandwidth r . To avoid the discontinuity expected in $[t, t_3]$, some video frames should be transmitted in advance during the interval $[t_2, t]$. As L moves, we can identify such intervals easily. For the other type of intervals, like $[t_1, t_2]$, the transmission should be at the video frame's rate that is naturally less than the bandwidth allocation r .

The above procedure divides the whole transmission interval into two types of intervals. The one is the *variable-rate transmission interval*, during which transmission should be at the video frame's original rate (the interval (a) in Fig. 2). The video frame's rates in this interval is always smaller than the reserved transmission bandwidth r . The other is the *fixed-rate transmission interval*, during which transmission should be at the rate r that is the fixed network bandwidth reserved (the interval (b) in Fig. 2). As a result, the whole transmission consists of alternating sequence of variable-rate intervals and fixed-rate intervals.

The client's buffer level at time t is $b(t) = P(t) - C(t)$, and the client's maximum buffer requirement is given by $B_{max} = MAX\{b(t)\}$. Therefore, the worst case startup delay is

$$t_d = \frac{MAX\{b(t)\}}{r}, \quad 0 \leq t \leq n$$

Under the basic transmission, there exists a certain relation between transmission rate and client's buffer size. This is a unique characteristic of a particular

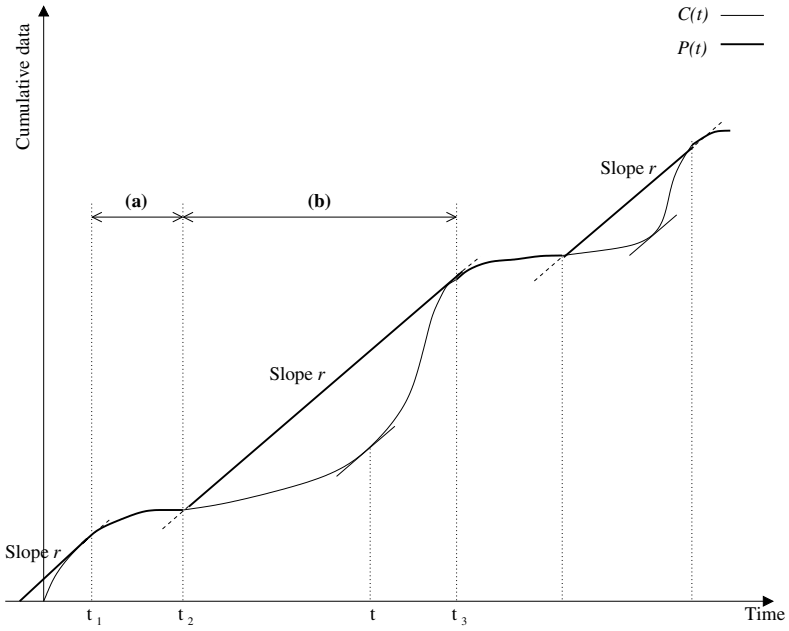


Fig. 2. Basic transmission schedule

video. Fig. 3 depicts a curve of real video’s trace (which is used again for performance evaluation in Section 4) This curve is a useful meta data about VBR video, which can be utilized by video server for admission control of service requests. It can be utilized for the initial allocation of transmission rate too.

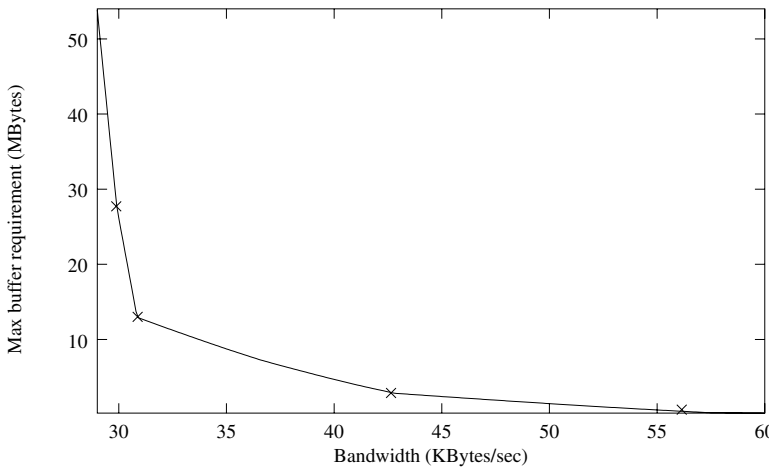


Fig. 3. Transmission rate vs. maximum client’s buffer requirement

3.2 Continuous Transmission under Rate Adjustment

Due to dynamic change of network traffic, media streaming over Internet often requires a congestion control mechanism. This significantly affects the determination of the bandwidth allocation and the calculation of transmission schedule. Here, we assume that a video server uses RTP/RTCP protocol as underlying congestion control mechanism.

RTP/RTCP typically operates as follows. A video server packetizes video into a series of RTP packets and sends it to client with timestamp and sequence number attached. Then, the client periodically sends RTCP packets, called *receiver report*, back to the server. Each receiver report contains reception quality feedback, including the highest packets number received, the number of packets lost, inter-arrival jitter, and timestamp to calculate the round-trip delay. Using an RTCP receiver report packet, the video server can estimate the current network state seen by the client and adjust transmission rate based on the estimation. An example of the mechanism can be found in [12]. In our scheme, we only consider the case of decrease in transmission rate. The case of increase, which is very rare in real world, is trivial and not addressed here. We also assume that there exists a lower bound on the change of transmission rate.

Fig. 4 shows a transmission schedule of a stream with initial transmission rate is r . Consider a fixed-rate transmission interval beginning at t_r . Assume that a network congestion occurs at t_{renew} , and the transmission rate decreases to r' ($r' < r$) after congestion control. As shown by the figure, this will result in the discontinuity of playback at $t_{discont}$. The reason of the discontinuity is that the beginning of transmission with rate r at t_r is too late for continuous playback in case of the bandwidth decrease, although t_r is the latest point at which the continuous transmission is possible with the minimum buffer requirement. This holds for any transmission start time between $[t_{r'}, t_r]$, which is indicated by the shaded area and a thick dashed line in the figure. As shown by Fig. 4, to avoid the discontinuity in spite of bandwidth decrease can only be achieved by starting the transmission at $t_{r'}$ or earlier. ($t_{r'}$ is the start time with the bandwidth r') However, this *early transmission start time* approach to avoid discontinuity may cause another problem. Assume that the transmission begins at time $t_{r'}$ with the transmission rate r . This early transmission cause a buffer overflow at client side, which also results in the discontinuity of playback.

As a result, we have two constraints on transmission schedule, which should be met in order to guarantee continuous, overflow-free transmission under the bandwidth renegotiation. The one is that a transmission schedule should always be *above* the straight line with the slope r' (which is the minimum bandwidth expected after the bandwidth renegotiation) in order to avoid discontinuity under the bandwidth renegotiation. The other is that there should be some brief pause (or decrease in the rate of transmission) so that the transmission schedule is always *below* the dashed line in order to avoid the buffer overflow at the client's side.

Fig. 5 depicts a possible transmission schedule that meets the both constraints. In the figure, the dashed line indicates client's maximum buffer require-

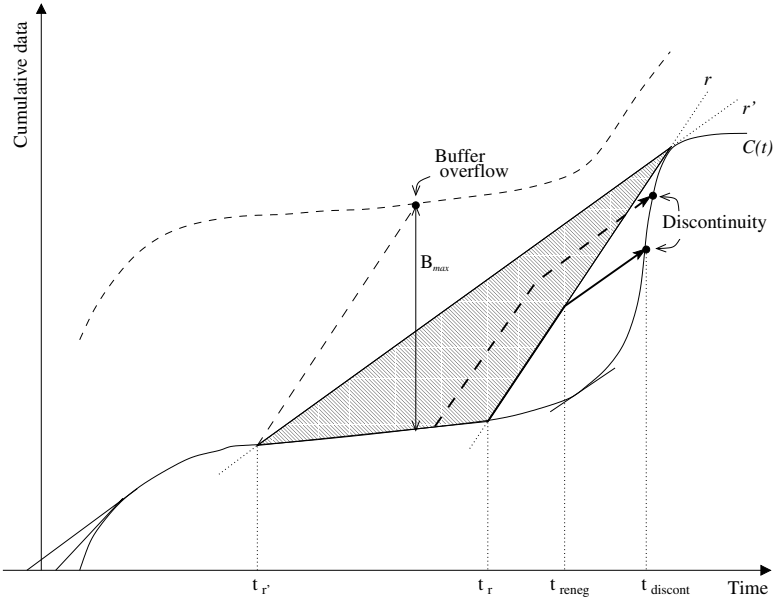


Fig. 4. Discontinuity due to the decrease in the bandwidth allocation

ments. Given two transmission rate r and r' with $r' < r$, the transmission scheme operates according to the basic transmission's intervals of whose bandwidth allocation is r' , although it still utilizes the initial transmission rate r . The details of the operations are as follows. For variable-rate intervals of basic schedule, like $[t_1, t_2]$, server begins transmission at the rate of r . The transmission stops at t_p where the cumulative amount of data transmitted becomes equal to the cumulative amount of video frames. The transmission pauses until the end of the variable-rate interval, t_2 . For fixed-rate intervals, like $[t_2, t_3]$, server also begins at the rate of r , it stops at $t_{p'}$ where client's buffer becomes full. It resumes at t_r when the cumulative amount of data transmitted is equal to that of the basic transmission. It stops at $t_{p''}$ where the cumulative amount of data transmitted becomes equal to that of video frames at the end of the fixed-rate transmission interval, t_3 . The thick line in the figure represents the newly suggested transmission schedule that can both avoid discontinuity and prevent overflow of client buffer.

3.3 Calculation of Transmission Schedule

The following is an algorithm to calculate the amount of data to be transmitted to client at each frame time under network bandwidth renegotiation. The algorithm first obtains the information about the intervals of the basic transmission schedule using the algorithm (line 1). This information includes type of interval (variable-rate interval or fixed rate interval), starting and end frames of interval

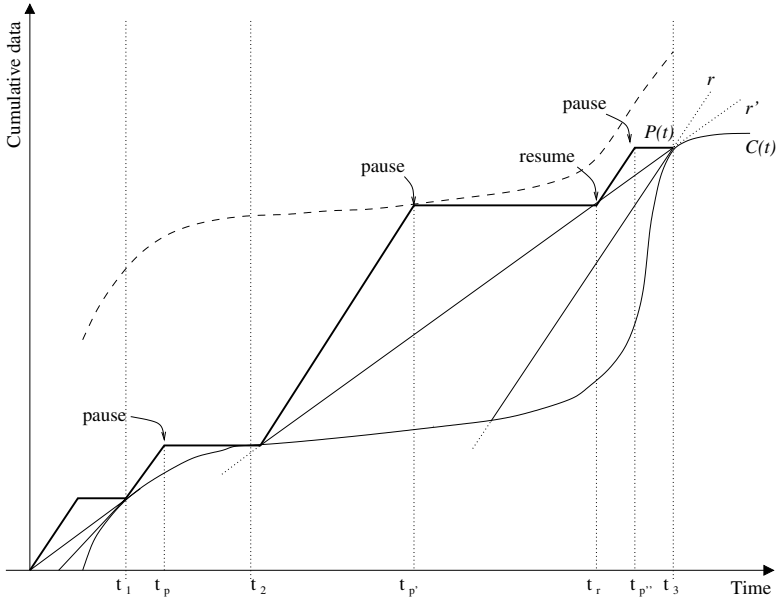


Fig. 5. New transmission schedule considering bandwidth renegotiation

t_s and t_e , and the cumulative amount of video frames at these frame times $C(t_s)$ and $C(t_e)$. Then, the algorithm outputs $p(t)$ for each interval according to the method described in Fig. 5. The complexity of the algorithm is $O(m+n)$, where n is the number of video frames and m is the number of basic transmission schedule's intervals. This algorithm may be integrated with the transmission scheduler of video server. It can be utilized to off-line processing of video for more efficient transmission.

Transmitting a VBR stream considering rate adjustment

input: sequence of frame sizes, $c(t)$
 initial transmission rate, r
 minimum transmission rate expected, r'
 client's maximum buffer requirement, B_{\max}
 output: transmission schedule, $p(t)$

```

1 Identify fixed-rate & variable-rate intervals using  $r'$ ;
2 for (each interval)
3    $t_s$  = starting frame time of interval;
4    $t_e$  = ending frame time of interval;
5    $t = t_s$ ;
6   if (variable-rate interval) then
7      $p\_sum = C(t_s)$ ;
8      $c\_sum = C(t_s)$ ;

```

```

9      repeat
10     if (p_sum < C(t_e)) then
11       p(t) = r;
12     else
13       p(t) = 0;
14     end if
15     p_sum = p_sum + p(t);
16     c_sum = c_sum + c(t);
17     t = t + 1;
18   until (t >= t_e)
19 else /* fixed-rate interval */
20   p_sum = C(t_s);
21   p2_sum = C(t_s);
22   c_sum = C(t_s);
23   paused = FALSE;
24   repeat
25     if (not paused) then
26       p(t) = r;
27       p_sum = p_sum + p(t);
28       p2_sum = p2_sum + r';
29       c_sum = c_sum + c(t);
30       if (p_sum >= c_sum+B_max || p_sum >= C(t_e)) then
31         paused = TRUE;
32       end if
33     else
34       p(t) = 0;
35       p2_sum = p2_sum + r';
36       c_sum = c_sum + c(t);
37       if (p_sum <= p2_sum) then
38         paused = FALSE;
39       end if
40     end if
41     t = t + 1;
42   until (t >= t_e)
43   end if
44 end for

```

4 Performance Analysis

In this section, some experimental results are presented to evaluate the performance of the proposed transmission scheme. In these experiments, we use the Starwars trace as our workload.²

Fig. 6 shows the distribution of variable-rate transmission intervals under the basic transmission scheme in section 3.1. As the reserved network bandwidth

² <ftp://ftp.bellcore.com/pub/vbr.video.trace>

increases, the number of frame times belonging to the variable-rate intervals dramatically increases. It means that the most of frame times belongs to variable-rate intervals during which the transmission is at the rate of video frame's rate. In other words, the basic transmission wastes most of network bandwidth, and the rate variability increases as the bandwidth allocation increases.

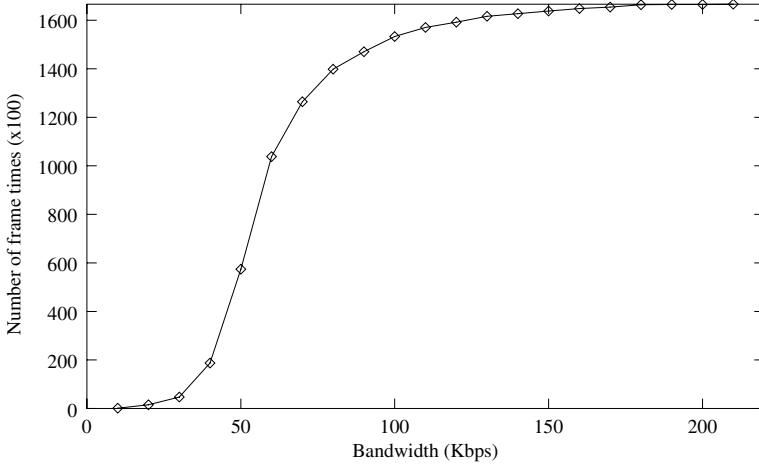


Fig. 6. The distribution of variable-rate transmission intervals

We compare the rate variability of the proposed transmission scheme with the optimal smoothing technique [7]. Table 2 shows the maximum/minimum transmission rates according to various client's buffer sizes. In our scheme, the transmission rate is constant except the case of brief pause in transmission (i.e. the minimum rate is 0, which is omitted in the table). Our scheme seems to show slightly higher variability than the optimal one, but the difference between both schemes is closer as the client's buffer size increase. Moreover, the length of pausing intervals decrease as the client's buffer size increase, which reduces rate variability.

Table 2. Comparison of rate variability

	optimal smoothing	proposed scheme
Client buffer - 2 Mbytes	43769/12938	50033
4Mbytes	43769/19384	48230
8Mbytes	43769/24743	45928

Fig. 7 compares the bandwidth utilization of the proposed transmission scheme with the basic one. As mentioned in section 3.2, in order to prevent the

discontinuity of playback in spite of the decrease in bandwidth allocation, the scheme begin transmission much earlier than the basic scheme. And it have no variable-rate intervals. As a result, the scheme have better bandwidth utilization than the basic scheme.

On the contrary, the scheme have larger buffer requirement than the basic one (Fig. 8). The increase in the buffer usage is unavoiable, but the scheme guatantees that it does not exceed the maximum client's buffer space.

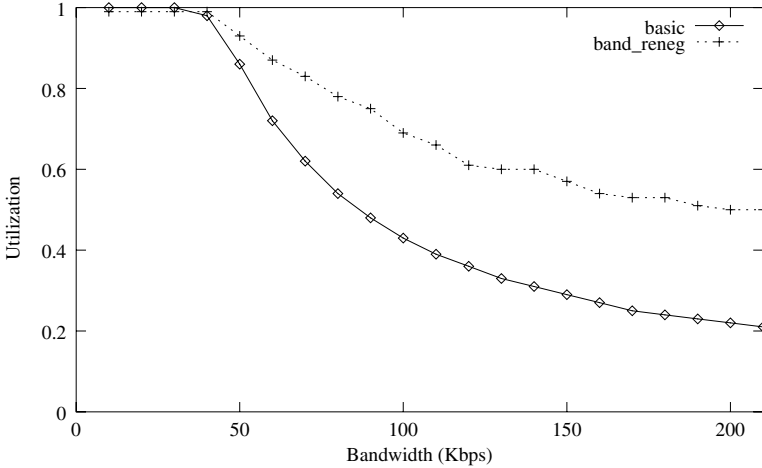


Fig. 7. The comparison of bandwidth utilization

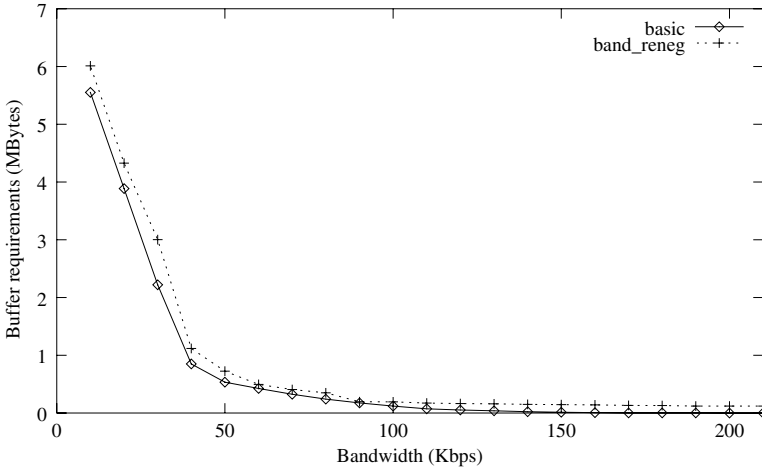


Fig. 8. The comparison of buffer requirement

5 Conclusion

In this paper, we propose a new transmission scheme of VBR video considering feedback control mechanism for streaming over Internet. Given fixed-size client's buffer, the proposed scheme first calculates the basic transmission schedule. This can be done off-line using a priori information such as a sequence of frame sizes. Then, the scheme revises the transmission schedule in order to prepare for the rate adjustment during the transmission. Some experimental results are given to demonstrate the performance of the proposed scheme. The scheme shows higher rate variability than the optimal smoothing scheme

References

1. D. Le Gall, "MPEG: A video compression standard for multimedia applications", *Communications of the ACM*, Vol. 24, No. 4, pp. 59-63 1991.
2. T. T. J.-C. Bolt and I. Wakeman, "Scalable feedback control for multicast video distribution in the internet", In *Proceedings of ACM SIGCOMM '94*, pp. 58-67, September 1994.
3. H. Zhang and E. W. Knightly, "Red-vbr: a renegotiation-based approach to support delay-sensitive vbr video", *ACM Multimedia Systems*, vol. 5, pp. 164-176, May 1997.
4. H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson, "RTP: a transport protocol for real-time applications", *RFC1889*, 01/25/96.
5. D. J. Gemmell, H. M. Vin, D. D. Kandlur, P. V. Rangan, and L. A. Rowe, "Multimedia storage servers: a tutorial and survey", In *IEEE Computer*, Vol. 28, No. 5, pp. 40-49, May 1995.
6. E. W. Knightly, D. E. Wrege, J. Liebeherr, and H. Zhang, "Fundamental limits and tradeoffs of providing deterministic guarantees to vbr video traffic", In *Proceedings of ACM SIGMETRICS '95*, pp. 98-107, May 1995.
7. J. Salehi, Z.-L. Zhang, J. Kurose, and D. Towsley, "Supporting stored video: reducing rate variability and end-to-end resource requirements through optimal smoothing", In *Proceedings of ACM SIGMETRICS '96*, pp. 222-231, May 1996.
8. J. M. McManus and K. W. Ross, "Video on demand over atm: constant-rate transmission and transport", In *Proceedings of IEEE INFOCOM '96*, pp. 1335-1362, March 1996.
9. S. Paek and S. Chang, "Video server retrieval scheduling for variable bit rate scalable Video", In *Proceedings of IEEE Int'l Conference on Multimedia Computing and Systems*, pp. 108-112, June 1996.
10. N. Shroff and M. Schwartz, "Video modelling within networks using deterministic smoothing at the source", In *Proceedings of IEEE INFOCOM '94*, pp. 342-349, June 1994.
11. D. Ferrari and D. Verma, "A scheme for real-time channel establishment in wide-area networks", *IEEE Journal on Selected Area in Communications*, Vol. 8, pp. 368-379, April 1990.
12. I. Busse, B. Deffner, and H. Schulzrinne, "Dynamic qos control of multimedia applications based on RTP", *Computer Communications*, Vol. 19, pp. 49-58, 1996.

MPEG-4 Video Transfer with TCP-friendly Rate Control

Naoki Wakamiya, Masaki Miyabayashi, Masayuki Murata, and Hideo Miyahara

Graduate School of Engineering Science, Osaka University

1-3 Machikaneyama, Toyonaka, Osaka 560-8531, Japan

{wakamiya, miyabays, murata, miyahara}@ics.es.osaka-u.ac.jp

Abstract. It is widely known that network bandwidth is easily monopolized by distributed multimedia applications due to their greedy UDP traffic. In this paper, we propose TCP-friendly MPEG-4 video transfer methods which enable real-time video applications to fairly share the bandwidth with conventional TCP data applications. We consider how video applications should regulate video quality to adjust video rate to the desired sending rate which is determined by TCP-friendly rate control algorithm. Carelessly applying TCP-friendly rate variation to the video application would seriously degrade the application-level QoS. For example, the control interval should be long enough to avoid the fluctuation of video quality caused by too frequent rate control. However, popular TCP-friendly rate control algorithms recommend that a non-TCP session regulates its sending rate more than once a RTT. Through simulation experiments, it is shown that high-quality and stable video transfer can be accomplished by our proposed methods.

1 Introduction

Since the current Internet does not provide QoS (Quality of Service) guarantee mechanisms, each application chooses the preferable transport protocol to achieve required performance. For example, traditional data applications such as `http`, `ftp`, `telnet` employ TCP which accomplishes loss-free data transfer by means of window-based flow control and retransmission mechanisms. On the other hand, loss-tolerant real-time multimedia applications such as video conferencing or video streaming prefer UDP to avoid unacceptable delay introduced by packet retransmissions. UDP is considered selfish and ill-behaving because TCP throttles its transmission rate against the network congestion whereas UDP does not have such control mechanisms. As the use of real-time multimedia applications increases, a considerable amount of “greedy” UDP traffic would dominate network bandwidth. As a result, the available bandwidth to TCP connections is oppressed and their performance extremely deteriorates.

In order that both TCP and UDP sessions fairly co-exist in the Internet, it is meaningful to consider the fairness among protocols. In recent years, several researches have been devoted into investigation on the “TCP-friendly” rate control [1,2,3,4,5,6,7,8,9, 10]. “TCP-friendly” is defined as “a non-TCP connection should receive the same share of bandwidth as a TCP connection if they traverse the same path” [5]. A TCP-friendly system regulates its data sending rate according to the network condition, typically expressed in terms of the round-trip-time (RTT) and the packet loss probability, to achieve the same throughput that a TCP connection would acquire on the same path. In particular, TCP-Friendly Rate Control (TFRC) proposed in [9,10] has the feature of adjusting a

transmission rate so smoothly while coping with network congestion. Therefore, TFRC has been receiving attention as the effective rate control mechanism for realizing multimedia communications fairly sharing the network bandwidth with TCP data sessions. It is meaningful to consider TCP-friendly video transfer because many researchers are engaged in investigations on packet scheduling algorithms on a router with which ill-behaving, that is, non-TCP flows are penalized aiming to provide QoS-guaranteed service in the Internet.

In our previous works [7,8], we have been devoted into investigation of the applicability of TCP-friendly rate control to real-time MPEG-2 video communications. We proposed effective control mechanisms, called MPEG-TFRC, with consideration of several factors which affect the efficiency of rate control, such as length of control interval, algorithms to adjust video rate to the target rate. For example, although it is recommended that a TFRC system regulates sending rate more than once a RTT, it is unrealistic to control video quality so frequently, in some cases, at the rate higher than video frame rate. We verified the effectiveness of our proposed mechanisms through simulation experiments on video trace data. Further, we implemented the mechanisms on a actual video communication system and performed several experiments to investigate the applicability and practicality of our mechanisms and showed that MPEG-2 video application could fairly share link bandwidth with TCP connections without introducing serious video quality degradation and fluctuation. However, our mechanisms cannot be applied to wide area networks which consist of variety of networks such as PSTN, ISDN, wireless networks, optical fiber networks, because we only consider MPEG-2 video streams ranging from 1.5 Mbps to 24 Mbps.

In this paper, we focus on MPEG-4 video systems which have highly efficient coding algorithms and error resilient capabilities, and investigate TCP-friendly video rate control mechanisms. Specifically, we employ Fine Granular Scalability (FGS) [11,12,13] as a video coding algorithm to accomplish highly efficient and scalable rate control. The rate adjustment based on regulating the level of quantization in our previous work on MPEG-TFRC is also applicable to MPEG-4 video systems. However, to successfully adapt to the TCP-friendly rate, we should know the relationship among the quantizer scale and the resultant video rate. In addition, the rate control is not flexible enough because the quantizer scale is a discrete value. First we describe the basic characteristics of FGS algorithm, then we consider mechanisms for adjusting quality and rate of FGS video streams according to the TFRC mechanism. Through simulation experiments, we show that our proposed methods can provide high-quality, stable and TCP-friendly video transfer.

The paper is organized as follows. In Section 2, we briefly introduce MPEG-4 video coding technique and FGS algorithm, then evaluate the basic characteristics of FGS. In Section 3, we propose several methods of FGS video transfer which accomplish TCP-friendly and high quality real-time video communication. Finally, we summarize our paper and outline our future work in Section 4.

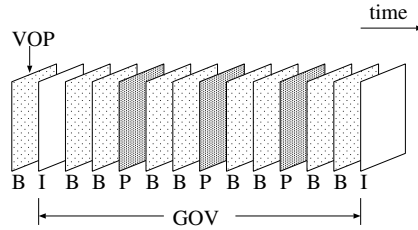


Fig. 1. An example of MPEG-4 video structure

2 Fine Granular Scalability Coding Algorithm

In this paper, we consider real-time video applications in narrow bandwidth networks and employ compressed video streams coded by MPEG-4, specifically, FGS algorithm, which is excellent in adaptation to the bandwidth variation, compression efficiency and error tolerance among MPEG-4 video-coding standards.

2.1 MPEG-4 Video Coding Technique

An MPEG-4 video stream (VOS: Visual Object Sequence) consists of one or more visual objects (VO). Each VO sequence consists of several layer streams (VOL: Video Object Layer). A layer video stream is composed from a sequence of VOP (Video Object Plane), each of which corresponds to a frame. MPEG-4 accomplishes high compression ratio by applying optimum coding algorithm suited to each VO. A VO corresponds to whole of a rectangle frame as in MPEG-1 and MPEG-2, a specific region of a frame, or a natural object such as a human, an animal, a building and so on. MPEG-4 can handle both rectangular and arbitrary shape VO, but all VOs are first divided into macroblocks of 16×16 pixels, which consists of four 8×8 blocks of luminance and two 8×8 blocks of chrominance (called 4:2:0 format). Coding operation is performed on a macroblock basis. In this paper, we only consider traditional rectangle VOs as in MPEG-1 and MPEG-2. However, our method is applicable to arbitrary shape VO only if there exist techniques to regulate video rate.

An MPEG-4 video stream consists of three types of VOPs as shown in Fig 1. VOP is the basic unit of image data and is equivalent to the frame or picture of MPEG-1 and MPEG-2. I-VOP is a self-contained intra-coded picture and coded using information only from itself. P-VOP is predictively coded using motion compensation algorithm referring to the previously coded VOP. B-VOP is a bidirectionally predictive-coded VOP using the differences between both the previous and next VOPs. An I-VOP is directly and indirectly referred to by all following P and B-VOPs until another frame is coded as an I-VOP. It is recommended to have I-VOPs regularly in a MPEG-4 video stream since motion compensation-based compression efficiency decreases as the distance from a referring frame to a referred picture becomes longer. In addition, by inserting I-VOPs as “refreshing points”, we can achieve error resilience in the video transfer. Since an entire frame can be completely reconstructed from an successfully received I-VOP, error

propagation can be interrupted when video quality is degraded in the preceding VOPs due to accidental packet losses. A sequence of VOPs beginning from an I-VOP is called GOV (Group Of VOP) and defined by two parameters, number of P-VOPs between two I-VOPs and number of B-VOPs between two P-VOPs. In an example of Fig. 1, they are 3 and 2, respectively. Using GOV structure is highly recommended for regularity of video traffic, error resilience and accessibility to video contents.

2.2 FGS Video Coding Algorithm

To cope with TCP-friendly rate control mechanisms, video applications should adjust video traffic rate to the desired rate by controlling the amount of video data. Since the amount directly corresponds to the video quality, rate control can be accomplished by regulating video quality. One way of video rate regulation is changing coding parameters such as frame rate, frame size and degree of quantization. Those are related to temporal, spatial and SNR resolution of compressed video data, respectively. In our previous works on TCP-friendly MPEG-2 video transfer [7,8], we regulate the MPEG-2 video rate by choosing appropriate quantizer scale, i.e., SNR resolution, according to the desired target rate determined by a TCP-friendly mechanism. Through experiments, it is shown that high-quality and TCP-friendly MPEG-2 video transfer can be performed with our proposed method. We take into account the relationship among quantizer scale, video rate and perceived video quality obtained in our research work on QoS mapping method for MPEG-2 video [14]. This is a good starting point of determining the control parameter appropriate for video rate adjustment with consideration of the perceived video quality. However, we found that the parameter changing is somewhat a coarse control and generated video traffic does not necessarily fit to the desired TCP-friendly rate. This is because we have only stepwise variation of possible video rate due to discrete set of parameters. Furthermore, the relationship among coding parameters and resultant video rate differs among video streams.

One might think of scalable or layered video coding algorithms standardized in MPEG-2 and MPEG-4, i.e., temporal, spatial and SNR scalability. The layered coding is certainly another way of video rate adjustment, but not powerful enough. Even if we combine two or more scalabilities, the number of achievable rate is at most several tens [15].

In this paper, expecting higher flexible and scalable rate adjustment capability, we employ Fine Granular Scalability (FGS) video coding algorithm [11,12,13] considered as a compression method suitable for video streaming applications and being introduced into MPEG-4 standards. Figure 2 illustrates the basic structure of FGS video stream. FGS is also categorized into layered coding algorithm and an FGS video stream consists of two layers, Base Layer (BL) and Enhancement Layer (EL). The BL is generated using motion compensation and DCT (discrete cosine transform)-based conventional MPEG-4 coding algorithm and provides minimum video quality. The EL is generated from the BL data and the original frame. The embedded DCT method is employed for coding EL to obtain fine-granular scalable compression. By combining BL and EL, one can enjoy higher quality video presentation.

The video quality depends on both the encoding parameters (quantizer scale, etc.) and the amount of supplemental EL data added. Even if only little EL data is used in

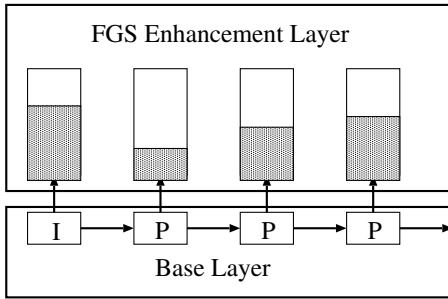


Fig. 2. An example of FGS video structure

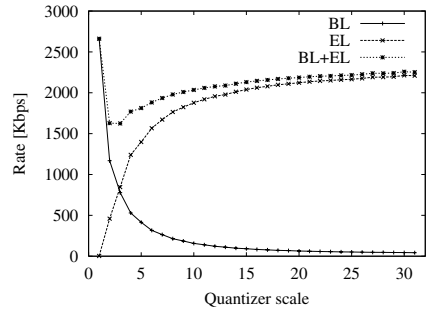


Fig. 3. Relationship among quantizer scale and average rate

decoding a VOP, the perceived video quality becomes higher. Thus, it seems effective to send as much EL data as possible in addition to the BL data, according to the target rate which is generally higher than BL rate. Losses of the BL data have a significant influence on perceived video quality because BL is indispensable for decoding VOP. On the other hand, although compression efficiency is not high since EL is coded without motion compensation technique, the EL data have the outstanding error tolerance because of the scalable coding algorithm and the locality of error propagation where loss of EL data only affects the VOP.

2.3 Basic Characteristics of FGS

In this section, we evaluate the basic characteristics of FGS, in terms of variations of rate and quality. We use two MPEG-4 test sequences, “coastguard” and “akiyo”. They are QCIF large (176×144 pixels) and consist of 300 frames. They are coded at 30 frames-per-second (30 fps) with GOV structure of one I-VOP and 14 P-VOPs. No B-VOP is used avoiding inherent coding delay. We choose the quantizer scale from 1 to 31 to investigate the effect of coding parameter on the coded video. We employed both sequences in all experiments. However, due to the space limitation, we only show results of “coastguard”, but the results with “akiyo” are consistent with those shown in this paper.

In Fig. 3, we depict the relationship among quantizer scale and average video rate of BL, EL and BL+EL. The video quality variation in terms of SNR (Signal to Noise Ratio) against the quantizer scale is shown in Fig. 4. It is shown that the BL rate decreases as the quantizer scale increases, and the video quality also decreases. The quality degradation is supplemented by the EL data whose total amount grows as the minimum video quality attained by the BL data deteriorates. In a case of the smallest quantizer scale, i.e., 1, no EL data is generated and the video quality is the highest because no DCT coefficient is quantized.

By observing variations of average rate and video quality of BL+EL data, we find that, as the quantizer scale becomes large, more bandwidth is required to obtain the same video quality as the lower quantizer scale. Figure 5 shows the relationship among the

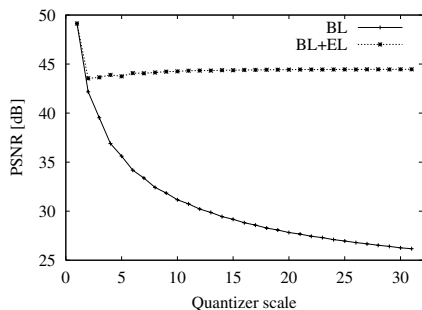


Fig. 4. Relationship among quantizer scale and video quality

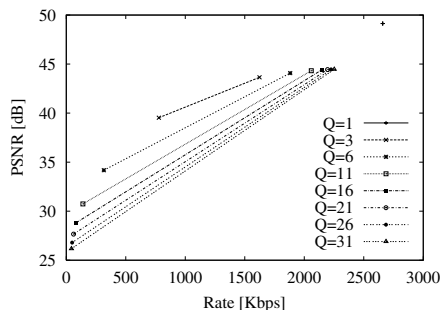


Fig. 5. Relationship between average rate and video quality

average video rate and the average video quality for several quantizer scales Q . Each line corresponds to a quantizer scale and the leftmost point of the line stands for the case of decoding only the BL data, while the rightmost point does the case of combining with all EL data. Thus, each line shows a range of possible video rate with the quantizer scale. When we plot some additional points on lines, they fluctuate a little but the relationship among lines still hold. If the desired data sending rate is below the minimum possible rate, the sender should buffer and smooth the data. On the other hand, the sender cannot satisfy the desired rate when it is beyond the maximum possible rate. The figure also shows that the smaller quantizer scale leads to the higher quality video presentation when the average rate is the same. Thus, we can expect the high-quality video transfer with the limited bandwidth when we appropriately choose the quantizer scale as small as possible.

However, a video stream coded with small quantizer scale intending to achieve high quality has limited capability of rate adjustment, which can be seen in Fig. 5 as a shorter line. For example, the video stream with quantizer scale $Q = 3$ only meets the bandwidth from 778 Kbps to 1,624 Kbps while the possible rate with $Q = 31$ ranges from 43 Kbps to 2,252 Kbps. If the coded video rate exceeds the target rate determined by TCP-friendly rate control mechanism, a video sender must regulate sending rate by buffering some part of video data making a sacrifice of smoothness of video presentation and interactivity of video application. Moreover, even if the target rate is in the range of possible rate, one must be prepared for the serious degradation of perceived video quality. Since the Internet is a best-effort network and no end-to-end QoS provisioning can be expected, packet losses cannot be avoided in video transfer especially when the video application employs UDP shunning delay introduced by retransmission. As shown in Fig. 3, a proportion of BL in entire video data increases as the quantizer scale decreases. Consequently, a possibility that BL data will be lost becomes higher and, as a result, the perceived quality of decoded video at a receiver considerably deteriorates.

Thus, an appropriate strategy might be to have a quantizer scale as large as possible, considering its error resilience and capability of rate adjustment. However, even if all

data are successfully transmitted and received by a receiver without error and loss, the perceived video quality is smaller than that of video data with smaller quantizer scale.

In the following sections, we investigate how we should employ the FGS coding algorithm and adjust video rate when TCP-friendly rate control mechanism is applied to video application. Some preliminary results useful for investigating appropriate quantizer scale selection algorithm are also shown.

3 FGS Video Transfer with TCP-friendly Rate Control

In this section, we first briefly introduce TFRC (TCP-Friendly Rate Control) [9,10], which accomplishes fair-share of bandwidth among TCP and non-TCP connections. Then, we propose several rate control methods to adjust FGS video rate to the desired sending rate determined by TFRC. Through simulations, we evaluate the effectiveness and practicality of our proposed methods.

3.1 TCP-friendly Rate Control

TFRC is the rate regulation algorithm to have a non-TCP connection behave similarly to, but more stable than a TCP connection which traverses the same path. It means that a TFRC connection reacts to network condition, typically congestion indicated by packet losses. For this purpose, a TFRC sender estimates the network condition by exchanging control packets between end systems to collect feedback informations. The sender transmits one or more control packets in 1 RTT. On receiving the control packet the receiver returns a feedback information required for calculating RTT and packet loss probability p . The sender then derives the estimated throughput of a TCP connection which competes for bandwidth on the path that the TFRC connection traverses. The estimated TCP throughput r_{TCP} is given as:

$$r_{TCP} \approx \frac{MTU}{RTT \sqrt{\frac{2p}{3}} + T_0 (3 \sqrt{\frac{3p}{8}}) p (1 + 32p^2)}$$

where T_0 stands for retransmission timeout [3]. Finally, the TFRC sender adjusts its data sending rate to the estimated TCP throughput r_{TCP} by means of, for example, video quality regulation. From now on, we call the estimated TCP throughput r_{TCP} , which determines the target rate of the application-level rate regulation, as ‘‘TFRC rate’’.

3.2 FGS Video Transfer on TFRC Connection

If an application successfully adjusts its sending rate to the TFRC rate, TCP-friendly data transfer can be accomplished. However, TFRC itself does not consider the influence of the TCP-friendly rate control on the application-level performance. For example, the TFRC sender changes its sending rate at least once a RTT. Such a frequent rate control obviously affects the perceived video quality when a video application regulates amount of coded video data by controlling video quality according to the target rate. Thus, to accomplish TCP-friendly video transfer with consideration of the application-level performance, i.e., video quality, we should consider the following issues.

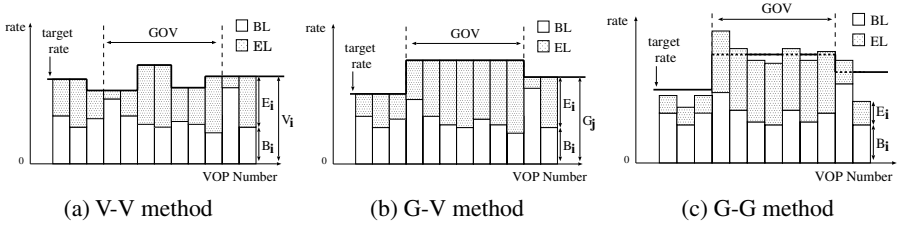


Fig. 6. Variants of video rate adjustment

1. control interval

The FGS video rate can be regulated by discarding a portion of the EL data. In this paper, considering the FGS video structure shown in Fig. 2, we propose VOP-based method (V method) and GOV-based method (G method). In the case of the V method, the target rate V_i of a VOP $_i$ is defined as the TFRC rate at the beginning of VOP $_i$. Analogously, the target rate G_j of a GOV $_j$ is defined as the TFRC rate at the beginning of GOV $_j$ in the G-method. Those are illustrated in Fig. 6 where (a) corresponds to the V method whereas (b) and (c) show the G method case. In Figs. 7 and 8, which correspond to the V and G method respectively, we also show variations of the target rate derived from trace data of simulated TFRC connections.

2. video rate adjustment

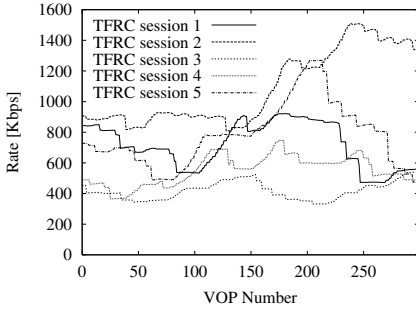
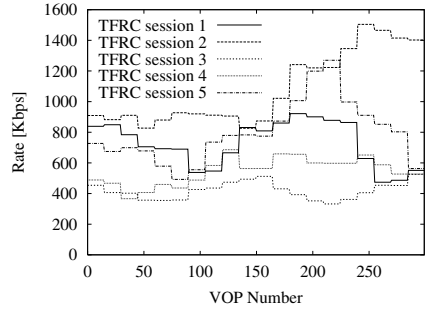
Adjustment of the FGS video rate to the target rate is performed by discarding a portion of the EL data. There are the alternatives of rate adjustment methods, VOP-based and GOV-based. For the VOP-based adjustment, we further propose two methods, i.e., V-V and G-V methods. In the case of the V-V method (Fig. 6(a)), the target rate V_i of the VOP $_i$ is first determined by the V method from the TFRC rate, and the rate (or amount) of the additional EL data E_i is obtained by subtracting the BL data rate B_i from the target rate V_i . On the other hand, the G-V method (Fig. 6(b)) first determines the target rate G_j of a GOV $_j$ by means of the G method, then applies the identical rate to all VOPs in the GOV ($V_i = G_j, \text{VOP}_i \in \text{GOV}_j$). Then, the video rate adjustment is performed VOP by VOP as $E_i = G_j - B_i$ for a VOP $_i \in \text{GOV}_j$ in the G-V method. Finally, in the GOV-based rate adjustment method, called as a G-G method (Fig. 6(c)), the video rate averaged over GOV $_j$ satisfies the target rate G_j . The rate of the EL data added to each VOP in the GOV is given as $E_i = (NG_j - \sum_{\text{VOP}_k \in \text{GOV}_j} B_k) / N$ where N stands for the number of VOPs in a GOV and identical among all VOPs in the GOV. The G-G method is proposed to achieve the smooth variation of video quality by equalizing the amount of supplemental EL data among VOPs, but the instantaneous video rate may exceeds the target rate.

3. BL rate violation

Even if the quantizer scale is carefully determined considering the network condition, the BL rate occasionally exceeds the available bandwidth for the video application. Since the BL data are crucial for video decoding, they are always sent out but an excess is managed by reducing the EL rate of the following VOPs or GOVs. In

Table 1. FGS video rate control methods

	control interval	rate adjustment	excess canceler
V-V <i>early</i>	VOP-based	VOP-based	<i>early</i>
V-V <i>smooth</i>	VOP-based	VOP-based	<i>smooth</i>
G-V <i>early</i>	GOV-based	VOP-based	<i>early</i>
G-V <i>smooth</i>	GOV-based	VOP-based	<i>smooth</i>
G-G <i>smooth</i>	GOV-based	GOV-based	<i>smooth</i>

**Fig. 7.** VOP rate variation**Fig. 8.** GOV rate variation

the *smooth* method, the excess is divided and equally assigned to the rest of VOPs in the GOV, thus averaged rate over several VoPs matches the target rate. On the other hand, to cancel the excess as fast as possible, the *early* method assign much excess to a VOP right after the the mischievous VoP, thus only a few VOPs are affected.

In table 1, we summarize possible rate control methods obtained by combining above mentioned methods. We should note here that there is not the G-G *early* method because the amount of EL data added to each VOP in the GOV must be identical in the “G-G” method.

3.3 Simulation Results

In this section, we compare six control methods proposed in the preceding section through simulation experiments. In comparison, we also consider another method called MP4-TFRC. The MP4-TFRC control method adjusts video rate by changing the quantizer scale as in our previous works on MPEG-2 [7,8].

In Figs. 7 and 8, we show the variation of target rate V_i and G_j of five simultaneous video sessions with TFRC rate control. These figures are obtained by applying V and G methods (see item “control interval” in Sec. 3.2) to trace data of the TFRC connections generated by a network simulator ns-2 [16]. A simulated network consists of two nodes and one 10Mbps bottleneck link of 15 msec delay connecting them. Each node has

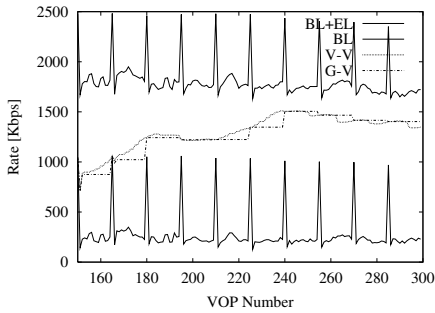


Fig. 9. Video rate variation (V-V vs. G-V)

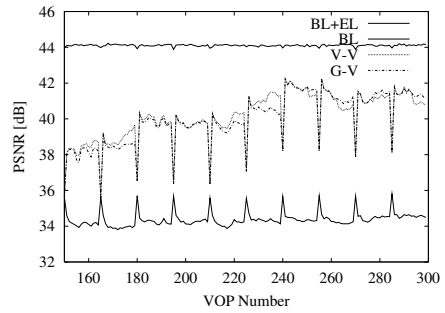


Fig. 10. Video quality variation (V-V vs. G-V)

thirty end systems via 150Mbps access links of 5 msec delay. The end systems on one node behave as senders and the others are receivers. Ten TFRC connections, ten TCP connections and ten UDP connections compete for the bottleneck bandwidth. In the following experiments, frame rate of coded video is 30 fps and the number of pictures in a GOV is 30. Figures shown in this section correspond to one of five video sessions for the sake of readability.

In Figs. 9 and 10, we show simulation results of the V-V and G-V methods when the FGS video data are generated from the test sequence “coastguard” by employing the quantizer scale of 2. The quantizer scale is determined to keep the BL rate below the minimum target rate during the session (TFRC session 2 in Figs. 7 and 8) in order to see the ideal performance of the video rate adjustment. Thus, the excess canceler is irrelevant. In those figures, “BL” and “BL+EL” correspond to the result of transmitting and decoding the BL data only and the BL with entire EL data, respectively.

In both methods, rate controls are successful and the FGS video rate follows the target. Although the target rate of each VOP differs among methods, the video rates averaged over longer interval, e.g., the duration of the session, are almost the same and the TCP-friendly video transfer are performed. The V-V and G-V methods differ in the control interval. The former adjusts the FGS rate VOP by VOP and the latter does GOV by GOV. As a result, the video rate of the G-V method changes much from GOV to GOV whereas that of the V-V method gradually increases or decreases according to the TFRC rate variation. The affect of rate variation can be seen in the video quality in Fig. 10.

The video quality in terms of SNR differs among GOVs in the G-V method, but is more stable than the V-V method regarding the difference among VOPs which belong to the same GOV. The reason that we find wedge-shaped and periodical quality degradation in the figure comes from the VOP-based rate adjustment. In the VOP-based adjustment, the resultant FGS rate becomes equal to the target rate by adding EL data $E_i = V_i - B_i$ or $E_i = G_j - B_i$. This means that the amount of supplemental EL data differs among VOPs even if we employ the G method where the target rates of VOPs in a GOV are identical. As a result, I-VOPs whose BL size is larger have less EL data than the other types of VOPs and experience lower quality. The interval between two wedges in the video quality variation corresponds to the distance between two successive I-VOPs.

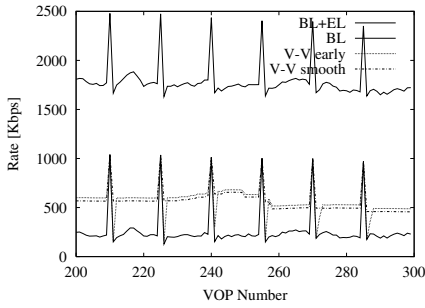


Fig. 11. Video rate (V-V *early* vs. *smooth*)

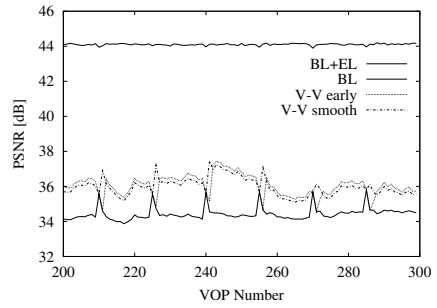


Fig. 12. Video quality (V-V *early* vs. *smooth*)

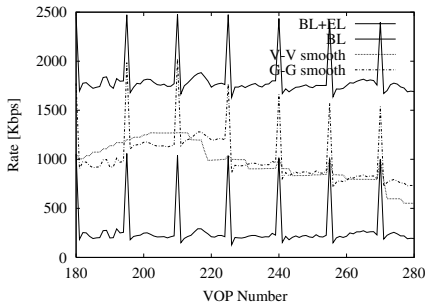


Fig. 13. Video rate (V-V *smooth* vs. G-G *smooth*)

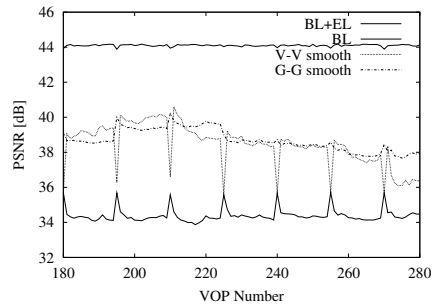


Fig. 14. Video quality (V-V *smooth* vs. G-G *smooth*)

From these observations, we can conclude that the V-V method achieves more preferable results than the G-V method in regarding variations of the FGS video rate and the video quality.

Next, we compare the excess cancelers of the V-V method. In Figs. 11 and 12, simulation results of the V-V *early* and V-V *smooth* methods on the TFRC session 4 are shown. In these figures, we employ the quantizer scale of 3 to make the BL rate higher than the target rate on relatively low TFRC rate. As a result, the BL rate in I-VOPs (VOPs 210, 225, 240, 255, 270 and 285 in the example) exceeds the target rate V_i . In such cases, the V-V *early* method reduces the amount of EL data added to a few VOPs right after the I-VOP and faces serious but instantaneous quality degradation. On the other hand, the V-V *smooth* method which fairly share excess among the rest of VOPs in the GOV, the degree of quality degradation is almost the same among VOPs in the GOV. However, the FGS video rate stays lower than the target rate during the GOV in the V-V *smooth* method whereas the V-V *early* method soon recovers from the rate decline. In addition, the V-V *early* method achieves higher video quality than the V-V *smooth* method in most of the time.

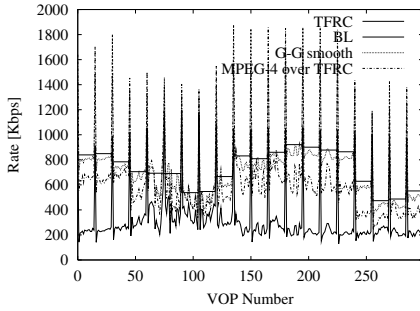


Fig. 15. Video rate (G-G *smooth* vs. MP4-TFRC)

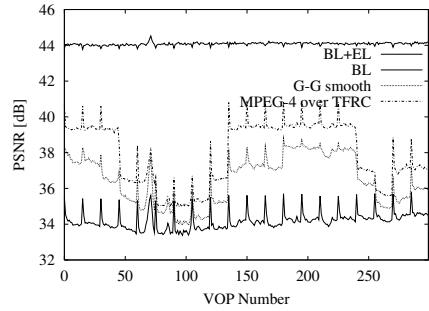


Fig. 16. Video quality (G-G *smooth* vs. MP4-TFRC)

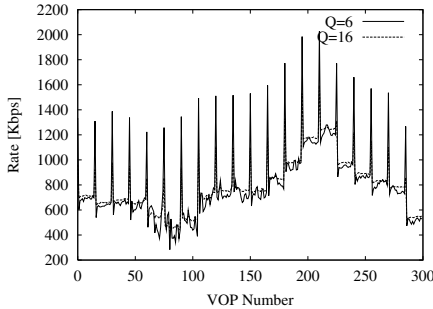


Fig. 17. Video rate under lossy condition

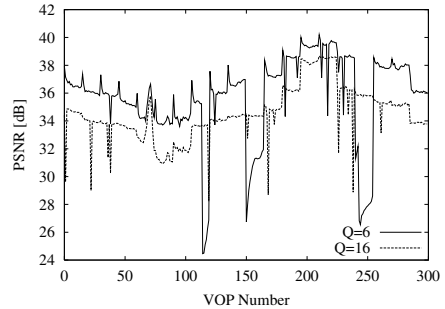


Fig. 18. Video quality under lossy condition

In Figs. 13 and 14, we compare the V-V *smooth* and G-G *smooth* methods to investigate the effect of video rate adjustment on the TFRC session 5. The G-G *smooth* method equalizes the amount of EL data among VOPs in a GOV to achieve the stable video quality. The FGS video rate follows the target rate in the V-V *smooth* method where the amount of EL data to add is determined VOP by VOP. The step-wise variation of the VoP-based target rate is due to the TFRC rate variation. On the other hand, the variation of FGS video rate in the G-G *smooth* method resembles that of BL rate because the identical amount of EL data is added to each VOP. In addition, the variation of video quality in the G-G *smooth* method is more gradual than that of the V-V *smooth* method. However, the instantaneous video rate of the G-G *smooth* method is not necessarily TCP-friendly and may introduce the smoothing delay required to make the data sending rate TCP-friendly. Furthermore, without an appropriate estimator of BL rate variation, the G-G *smooth* method introduces one GOV-time delay because all of the BL rate B_i of VOPs in the GOV must be known in advance to determine the amount of EL data to add to each VOP. Thus, the G-G *smooth* method is preferable when the video application emphasizes video quality while the V-V *smooth* is faithful to the TFRC rate.

To see superiority of FGS algorithm to DCT-based MPEG-4, we compare the G-G *smooth* method with the quantizer scale-based rate control method, i.e., MP4-TFRC. The MP4-TFRC is similar to the G-G *smooth* method except that the video rate is regulated by choosing an appropriate quantizer scale to fit the video rate to the target rate G_i according to the relationship among quantizer scale and resultant BL rate (Fig. 3). Results on the TFRC session 1 are shown in Figs. 15 and 16. Figure 15 indicates that the capability of rate control of the MP4-TFRC is poor and TCP-friendly video transfer cannot be expected. This is because that the quantizer scale-based rate control is coarse and there is often no appropriate quantizer scale with which the resultant video rate matches the target rate. Even if the quantizer scale is appropriately chosen, the resultant video rate has a highly bursty nature because of the GOV-based rate adjustment. Moreover, the coarse control leads to the sudden and drastic quality variation as shown in Fig. 16 which is triggered by increasing or decreasing the quantizer scale.

So far, we do not take into account packet loss to see the ideal performance of the TCP-friendly MPEG-4 video transfer. As mentioned in Sec. 2, packet loss affects the video quality. Figures 17 and 18 compares the variation of video traffic sent from the sender on the TFRC session 5 and video quality affected by packet loss for various quantizer scales, 3 and 6. Video data are segmented into packets of 1 KBytes long and packets are randomly discarded at a 10^{-3} probability. As shown in those figures, the video rate does not differ much among the quantizer scale because we apply the rate control method G-G *smooth* to the video data. The video quality is higher when the quantizer scale is smaller as long as there is no packet loss. However, once one or more packets are lost, the video quality considerably deteriorate in the video with a smaller quantizer scale because the proportion of BL data to the entire video traffic is large. Thus, we should carefully determine the quantizer scale taking account of not only the available bandwidth but the packet loss probability.

4 Conclusion

In this paper, we proposed and evaluated several TCP-friendly FGS video transfer. Through simulation experiments, we showed that the G-G *smooth* method, which determines the target rate every GOV and adds the identical amount of EL data to each VOP in the GOV, is preferable for video rate control in order to achieve the high and stable video quality.

However, there still remains some research issues. When the video application employs TFRC as the transport protocol, the video data injected into the transport layer are smoothed to fit to the TFRC rate, but such a smoothing delay is not considered in this paper. We showed only preliminary results on evaluation of influence of packet loss on the video quality. As mentioned in the previous section, the quantizer scale should be determined based on the network condition. We are currently considering a quantizer scale selection algorithm with which the video sender dynamically changes the coding parameter.

Acknowledgments. This work was partly supported by Special Coordination Funds for promoting Science and Technology and a Grant-in-Aid for Encouragement of Young

Scientists 12750322 from the Ministry of Education, Culture, Sports, Science and Technology, Japan, Research for the Future Program of Japan Society for the Promotion of Science under the Project “Integrated Network Architecture for Advanced Multimedia Application Systems”, and Telecommunication Advancement Organization of Japan under the Project “Global Experimental Networks for Information Society Project.”

References

1. Mathis, M., Semke, J., Mahdavi, J., Ott, T.: The macroscopic behavior of the TCP congestion avoidance algorithm. *ACM SIGCOMM Computer Communication Review* **27** (1997) 67–82
2. Bolot, J.C., Turletti, T.: Experience with control mechanisms for packet video in the Internet. *ACM SIGCOMM Computer Communication Review* **28** (1998) 4–15
3. Padhye, J., Firoiu, V., Towsley, D., Kurose, J.: Modeling TCP throughput: A simple model and its empirical validation. In: *Proceedings of ACM SIGCOMM’98*. Volume 28. (1998) 303–314
4. Rejaie, R., Handley, M., Estrin, D.: RAP: An end-to-end rate-based congestion control mechanism for realtime streams in the Internet. In: *Proceedings of IEEE INFOCOM’99*. (1999)
5. Padhye, J., Kurose, J., Towsley, D., Koodli, R.: A model based TCP-friendly rate control protocol. In: *Proceedings of NOSSDAV’99*. (1999)
6. Bansal, D., Balakrishnan, H.: TCP-friendly congestion control for real-time streaming applications. MIT Technical Report MIT-LCS-TR-806 (2000)
7. Wakamiya, N., Murata, M., Miyahara, H.: On TCP-friendly video transfer with consideration on application-level QoS. In: *Proceedings of IEEE ICME 2000*. (2000)
8. Miyabayashi, M., Wakamiya, N., Murata, M., Miyahara, H.: MPEG-TFRCP: Video transfer with TCP-friendly rate control protocol. In: *Proceedings of IEEE ICC 2001*. (2001) 137–141
9. Widmer, J.: Equation-based congestion control. Diploma Thesis, University of Mannheim (2000)
10. Floyd, S., Handley, M., Padhye, J., Widmer, J.: Equation-based congestion control for unicast applications: the extended version. Technical Report TR-00-003, International Computer Science Institute (2000)
11. Radha, H., Chen, Y.: Fine-granular-scalable video for packet networks. In: *Proceedings of Packet Video’99*. (1999)
12. van der Schaar, M., Radha, H., Dufour, C.: Scalable MPEG-4 video coding with graceful packet-loss resilience over bandwidth-varying networks. In: *Proceedings of IEEE ICME 2000*. (2000)
13. Radha, H., van der Schaar, M., Chen, Y.: The MPEG-4 fine-grained scalable video coding method for multimedia streaming over IP. *IEEE Transactions on Multimedia* **3** (2001) 53–68
14. Fukuda, K., Wakamiya, N., Murata, M., Miyahara, H.: QoS mapping between user’s preference and bandwidth control for video transport. In: *Proceedings of IFIP IWQoS’97*. (1997) 291–302
15. Fukuda, K., Wakamiya, N., Murata, M., Miyahara, H.: Real-time video multicast with hybrid hierarchical video coding in heterogeneous network and client environments. *Proceedings of IFIP/IEEE MMNS’98* (1998)
16. The VINT Project: UCB/LBNL/VINT network simulator - ns (version 2) (1996) available at <http://www.isi.edu/nsnam/ns/>.

IP Radio Resource Control System

John Vicente^{1,2} and Andrew T. Campbell¹

¹Center for Telecommunications Research, Columbia University
campbell@comet.columbia.edu

²Intel Corporation
john.vicente@intel.com

Abstract. With the need for mobility and wireless communications now being motivated by the commercial sector, more attention must be given towards IP QOS in the wireless infrastructure, if existing or emerging services are to be commercialized over it. In this work, we investigate and propose a system approach to balance time-varying, space-varying (usage) wireless and mobile channel conditions against IP-based service differentiation and end-end service requirements. The proposed solution leverages network programmability [1], multiple timescales, feedback-based mechanisms across flows, mobile clients and a centralized RAN or LAN control point to configure QOS policies and coordinate state information between congestion control, bandwidth control and reliability mechanisms.

1 Introduction

The current convergence of wireless networks and the Internet is forging new developments in communications and networking services unlike any other time in the history of the telecommunications industry. In the area of quality of service (QOS), while the recent IETF efforts have made significant steps forward in QOS, the progress has been mainly driven from the perspective of a wired infrastructure. With the need for mobility and wireless communications now being motivated by the commercial sector, the same attention must be given towards QOS in the wireless infrastructure, if existing or emerging services are to be commercialized over it. However, the current Wireless Internet is a best-effort paradigm, and the traditional layered services fail to deliver reliable, beyond best-efforts services over an efficient, managed wireless infrastructure. Yet, these are the necessary requirements that information technology managers and service providers seek to ensure that their business customer needs will be met.

To achieve these objectives, several key philosophical barriers must be overcome or at least moderated. First, the end-end principles and in-network control must converge under the wireless, mobile paradigm. The applications ability to adapt to positive or negative wireless conditions, by either leveraging in-network services or

binding alternative end-system services, will give it greater flexibility under a highly mobile and wireless environment. Second, a wireless evolution will require tighter layer integration and automation of application control and bandwidth management. The application's congestion control flexibility will depend on its ability to detect or respond on a much faster time-scale (e.g., to support fast handoff), thus requiring the network to cooperate with the flow's congestion control loop. Alternatively, the necessity to manage the wireless channel bandwidth will depend on the clocking rate and control mechanisms being used by the application to control the incoming rate of the flow. In this paper, we propose using a rate-based congestion control scheme as described in [2], as a more compatible scheme (*vis-à-vis* TCP windowing) with wireless traffic patterns and varying channel conditions. Third, we view existing layered or component services (e.g., TCP congestion control, 802.11 DCF, DiffServ) having overlapping or addressing uncoordinated functions including congestion management, service differentiation, bandwidth management and reliability. Fourth, this paper proposes that programmable networking [1] is needed to support this shift towards a Wireless Internet, thereby enabling new, dynamic services beyond the traditional barriers of current fixed services and QOS proposals. Finally, we propose that such a framework be legacy compatible in as much as is necessary not to burden the control system from efficient operation.

In what follows in Section 2, we present a systems approach to the problem of QOS, wireless resource control and flexible application adaptation in response to fluctuating conditions exhibited in the wireless environment. In Section 3, we present a programmable, open control system to support wireless IP resource control, along with a description of the components which make-up the proposed framework. Next in Section 4, we summarize additional work as follow-up to the current proposal.

2 A Systems Approach

In this work, we investigate and propose a system approach to balance wireless and mobile channel conditions against IP-based service differentiation and end-end service requirements. The underlying aspects of our proposal are summarized as follows:

- Tightly-couple resource management and control to achieve a stable, feedback-based control system, combining both end-end and in-network control mechanisms,
- Leverage middleware (e.g., network APIs) services to “glue” components (legacy or otherwise) and policies in order to systemize the control framework,
- Maintain flow, local and global control separation, while dispatching or exposing specific control policies or state information between them,
- Discriminate control policy from time-varying channel conditions, fluctuating channel usage, or both.

Figure 1 depicts a simple, conceptual model of our system goal to support flow adaptation in concert with local and global wireless channel resource management, and using both control and management signaling services to maintain (i.e., reliable, stable and performing) proper end-to-end flow delivery, optimal channel access and more efficient channel usage. By control, we are referring collectively to such services as QoS, congestion control, reliability and bandwidth control. By management, we are referring to monitoring mechanisms to support appropriate feedback to optimize control or policy-based decisions. In addition, we propose that control policies be based on conditions specific to the application flow, the local wireless device and the global channel. On a flow level, these conditions are at the scope of the application or connection transport level. The local device refers to the client mobile host, which may have time-varying conditions (e.g., fading, overlapped cells) imposed on its aggregate flows within its local vicinity, but may not necessarily reflect global conditions affecting all mobile hosts within the span of the wireless LAN or RAN. Alternatively, global conditions (e.g., channel traffic load) may affect the entire channel, and thus, all wireless devices within the same local area network.

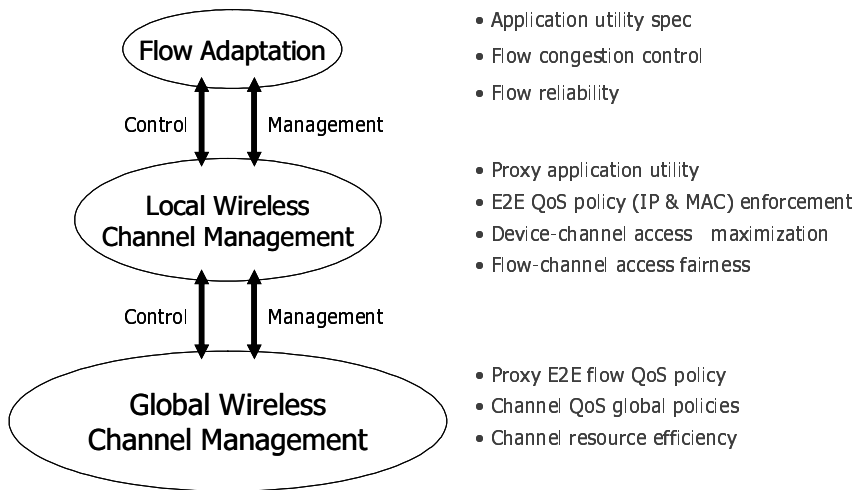


Fig. 1. Conceptual Model

Our approach does not suggest these three (i.e., state machines) as being independently managed or controlled. Instead, they must be cooperative in such a way, that each operates autonomously, however policies and states for their operation are exposed or exchanged for the intended stability and balance of the system operation. To achieve this, we consider an open network framework, where different components cooperate through exposed interfaces for binding purposes, dynamic configuration or state management. Furthermore, we can have greater control on the stability and efficiency of the system by enforcing policy controls at different timescales as needed to react, maintain or be proactive as warranted by the wireless device, the application flow or the wireless channel.

To implement this system, we propose putting more cooperative bandwidth control into the IP stack and following along the same architectural lines of the UMTS Radio Resource Control component described in [3]. However, we recommend using the existing IP stacks with QoS extensions as defined by [4],[5],[6],[7] and providing more programmable integration across the layered boundaries. Furthermore, we argue for a rate-based congestion control scheme [2] as a more transport compatible scheme with wireless traffic patterns and varying channel conditions. We propose a middleware layer (e.g., CORBA) to automate cooperative control between a central wireless IP radio resource manager (RRM) handling RAN/WLAN global resource management and multiple, distributed RRM Agents at the mobile clients (and base stations). These agents would be working on behalf of the RRM to distribute global policies, while also managing local resource management and reacting to local channel conditions. The RRM Agent can be seen as mediation control point between the global channel (layer 2 and 3) policies, local (layer 2) channel policies and application flow adaptation at the transport-level (layer 4) and above. Figure 2 illustrates the proposed approach to IP-based wireless QoS and resource control.

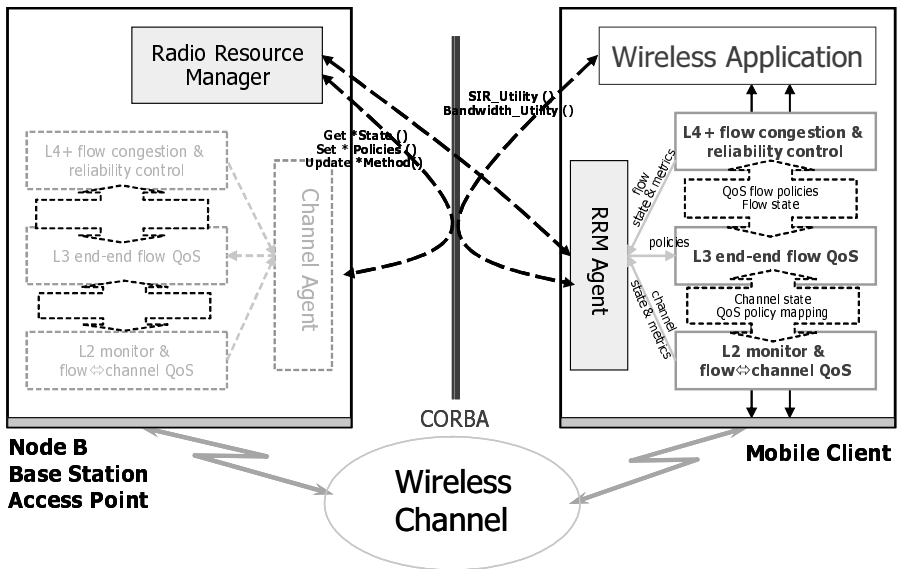


Fig. 2. IP Radio Resource Control Framework

3 Wireless IP Resource Control System

Figure 2 depicts a general model from which alternative wireless radio infrastructure can be supported via the Internet Protocol and supporting, enhanced IP-based network services. We do not propose an alternative network architecture (e.g., UMTS) nor new

infrastructure elements (e.g., RNC), but instead, we envision a resource control framework, which is consistent with an evolving Wireless Internet. More specifically, we propose a framework that supports alternative radio link technologies, maintains the ubiquitous nature of IP, and is built on the layered transport and application services, which preserve the fundamental nature of the Internet's design. However, our framework supports the use of middleware services to enable a more *dynamic*, *distributed* and *automated* resource control strategy. Thus, the IP stack is exposed to support a more integrated QoS and resource control system, while preserving its layered boundaries. This open integration is needed for multiple reasons:

- Integrating the resource control hierarchy (i.e., end-to-end, global channel, local device, flow)
- Coordinating resource policies and states across multiple layers
- Binding alternative service mechanisms or algorithmic solutions
- Automating the resource control (provisioning and QoS maintenance) system

3.1 System Components

The motivation for our current work is based on [8], where we proposed a host-based network traffic control system for administering manual or automated QoS provisions based on algorithmically adjusting LAN resource and flow QoS policies. In [8], we use a programmable classifier and scheduler based on a traffic control API [9],[10] to dynamically, configure or enforce IP-based QoS policies (e.g., marking, shaping, metering, dropping, priority, etc) on-the-fly based on per flow and LAN usage feedback. In the current work, we believe the radio access network or wireless LAN environment is more suited for such a solution due to its highly dynamic nature. The wireless channel is a more autonomous system; not bearing the partitioned aspects of LAN switched or VLAN environments, which we perceived as an architectural

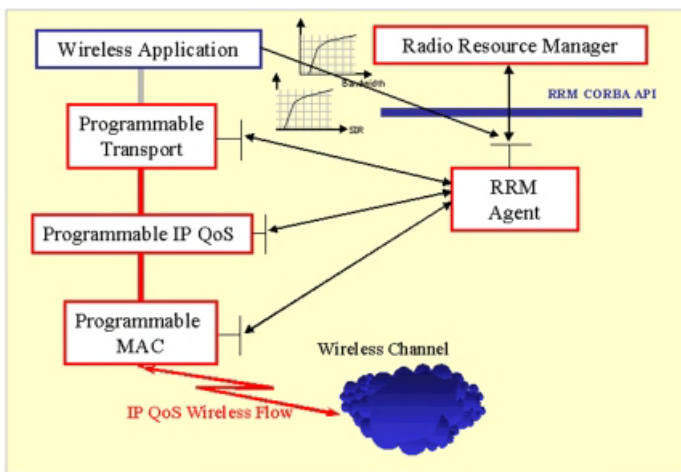


Fig. 3. Radio Resource Control Component Services

limitation for the realization of the work in [8]. Thus, we extend the work in [8] by recognizing the need to feedback time-varying conditions and not just congestion or flow-level performance, and further, by distinguishing local feedback conditions from global conditions in making proper resource and QoS policy decisions. We also extend the previous work by enabling more dynamic features across multiple layers and providing support for utility-based, adaptation (see [11],[12],[13]). This latter requirement allows the wireless application to adapt to either local SINR or global channel congestion feedback. Figure 3 illustrates the resource control components in a distributed object structure.

Our work is not intended to address complete end-to-end SLA management, but assumes that the framework can be ‘plugged-in’ to a variety of policy-based or broker management systems, e.g., using COPS [14] for provisioning policy communications or CORBA for binding layered provisioning services. This contribution is geared towards an infrastructure-based wireless networking environment, however, we believe the proposed framework can be decentralized and peered to support an ad hoc wireless network environment as well. In either case, it is the spirit of this work to provide an ‘auto-pilot’ solution for automating resource control with dynamic provisioning in the face of a highly dynamic, mobile and time-varying wireless environment within a *localized* context. Finally, by maintaining the legacy aspects of the IP architecture, providing architectural separation from the underlying physical radio architecture, and using an API-based framework, it allows us to build upon the static nature of IP protocols and supporting layered services without introducing protocol or layer complexity to the same. Table 1 describes the system components supporting the proposed IP control framework. In what follows, we describe the design functionality of each of the components.

Table 1. System Components

System Component	Timescale	Functions	Programmable services
Radio Resource Manager	>seconds	<ul style="list-style-type: none"> ▪ Gateway E2E QoS services ▪ RAN resource control & management ▪ RAN usage monitoring 	<ul style="list-style-type: none"> ▪ RRM agent services ▪ Radio bandwidth monitoring
RRM Agent	~ msec – seconds	<ul style="list-style-type: none"> ▪ Local device control & management ▪ RRM Proxy 	<ul style="list-style-type: none"> ▪ RRM proxy services to L2, L3, L4 components ▪ Manage bandwidth & SIR Utility
Programmable Transport	~10msecs	<ul style="list-style-type: none"> ▪ Flow adaptation (congestion control, loss & delay packet monitoring) 	<ul style="list-style-type: none"> ▪ Congestion control algorithms ▪ Flow policy & state management
Programmable IP QoS	~10msecs	<ul style="list-style-type: none"> ▪ E2E QoS policy enforcement ▪ Programmable QoS 	<ul style="list-style-type: none"> ▪ Enforce bandwidth utility API ▪ E2E QoS policies ▪ Binding QoS algorithms/services ▪ Flow rate policing
Programmable MAC	1msec – 100msecs	<ul style="list-style-type: none"> ▪ Mapping of E2E QoS enforcement ▪ Local device adaptation ▪ Local SINR monitor 	<ul style="list-style-type: none"> ▪ Enforce channel utility API ▪ QoS L3 → L2 policy mapping ▪ Local SINR monitoring ▪ Binding QoS algorithms/services

Radio Resource Manager (RRM). The Radio Resource Manager is a central resource controller operating at a radio network access point, base station or wireless LAN router. It can also act as a gateway to provision end-to-end QOS (e.g., from an ISP WAN) by allocating or managing local resources or mapping policies within the radio network. The RRM primarily oversees RAN resource control and management by negotiating wireless channel bandwidth requirements supporting its wireless clients while maximizing global channel bandwidth efficiency. RAN usage monitoring is critical to the automated resource control system by providing the feedback mechanisms at the highest level of RAN management. The RRM can employ alternative channel resource management schemes but uses distributed method invocations to dynamically adjust provisioning policies or program underlying layered component services resident at one or more wireless clients via the RRM Agent API.

RRM Agent. The RRM Agent is a distributed extension of the central RRM. The Agent is a mediation point balancing global channel resource policy, local channel policy, and flow-level adaptation. It communicates with client applications accepting application utility (utility-based SINR and bandwidth) functional specifications. Using RRM APIs, it communicates these utility specifications to the central RRM to support global channel optimization algorithms operating at the RRM. The Agent supports service programming (e.g., buffer management algorithm) or dynamic provisioning (e.g., marking, rate shaping policies), directly or via of the RRM. It accepts global policies delivered by the RRM and makes the necessary local enforcement decisions. Local channel resource control allows the link layer QOS service to adjust its scheduling and queuing mechanisms, temporarily preempting IP QOS policies in order to adjust access differentiation against time-varying conditions.

Inter-layer policy & state synchronization. An important aspect of the RRM Agent is to coordinate policy and state between protocol stack layers to maintain constant synchronization or resynchronization induced either by global policy changes, local policy changes or flow adaptation. Such coordination may happen directly using header information (e.g., IP options), inter-layer header mapping (e.g., IP DSCP to 802.1p mapping) or via the RRM Agent through method invocation and parameter passing. The latter can be, for example, used for requesting/responding with monitoring state or QOS enforcement policies.

Programmable Transport. While the Internet Transport layer (TCP/UDP) has become one of the underpinnings in the Internet's end-to-end design philosophy, it has received much debate in terms of its flexibility to support real-time services, wireless infrastructure and mobility. Recent work (e.g., RTCP and [2]) have proposed alternative transport services necessary to match the requirements of emerging services. It is the general argument in this paper, that perhaps a rate-based congestion

¹ Infrastructure mode. It is quite possible to decentralize the RRM function as a distributed system operating across nodes in an ad hoc network.

mechanism [2] and flow measurement (e.g., latency, reliability) approach that supports faster congestion and drop type detection may be more appropriate for a wireless and mobile infrastructure. Furthermore, a programmable transport, where alternative transport layer control schemes can be employed or configured in real-time to provide greater flexibility to the multiple, varying conditions exhibited by the wireless channel. This is perhaps further complicated by a transport connection over several LAN/WAN networks consisting of both wireline and wireless underlying link transports. While we perceive that a programmable transport may create connection inconsistency across network nodes, a service mechanism, perhaps offered by the programmable transport may allow nodes to reconfigure the transport layer dynamically to synchronize transport services during connection setup.

Programmable IP QOS. In [8], we demonstrated that a flexible network layer QOS mechanism could allow automated provisioning and reconfiguration through threshold-triggered remote method invocation. Using centralized (multi-threaded) resource management algorithms, alternative IP flow QOS policies (e.g., token bucket parameters, packet marking, shaping, discard policies) could be enforced remotely over a common API exposed by client QOS agents.

In this paper, we propose a similar IP-based provisioning service at wireless devices, however we extend the previous work [8] to a greater context, allowing alternative IP QOS bindings and algorithmic choices enabling greater design and provisioning freedom to the IP radio QOS programmer or administrator. Moreover, through coordination with the RRM Agent acting as a local proxy, the IP QOS service can be (per application flow) configured through global policies algorithmically determined through bandwidth utility curves [11],[12] and managed by the centralized RRM. In this scenario, the RRM Agent obviates operational complexity between the various components, yet exposes necessary local methods and attributes allowing the layered components to coordinate with the Agent and communicate policies and state.

Programmable MAC. While noted progress has been made in the area of programmable wireless MACs [11] [12], [15], we adopt the notion of SINR utility curves [16] and apply this model towards differentiated access mechanisms supporting a programmable QOS link layer. Once again, through coordination with the RRM Agent, differentiated access can be (per application flow) dynamically (faster timescale) reconfigured through local policies algorithmically determined through SINR utility curves managed by the RRM Agent. In this scenario, the local policies may preempt the IP QOS global policies in order to manage time-varying or fairness issues caused by a fading or degraded local conditions on one or more flows. These policies may conflict with the end-end or global channel policies enforced at the higher layer. However, the higher layer QOS policies may become active once local channel conditions are improved. Figure 4 illustrates a current proposal [17] from the IEEE 802.11 [18] QOS Working Group on an Enhanced DCF supporting link layer QOS. We assume that IP network layer QOS policies are appropriately mapped down to MAC layer policies through pre-configured CoS/QOS mapping (e.g.,

DiffServ/802.1p header field mapping). QOS traffic classes are then assigned to appropriate EDCF queues along with associated per class contention window and QOS interframe space parameters. Thus, bandwidth, access, and latency differentiation is made possible at the local link level.

Finally, as in the network and transport layers, alternative MAC level QOS bindings or algorithmic choices should be allowed to be reprogrammed into the MAC layer. In the above example, each of the EDCF components may be reprogrammed or configured dynamically by the RRM Agent as needed.

- Prioritized output queues (queue[i])
- Legacy DCF finite state machine per queue (queue[i])
 - CWmin differentiated per TC (CWmin[i]), controllable by EAP
 - DIFS differentiated per TC (QIFS[i]), controllable by EAP
 - Queue state machines count backoff slots in parallel
 - Low-priority queues defer to higher-priority queues

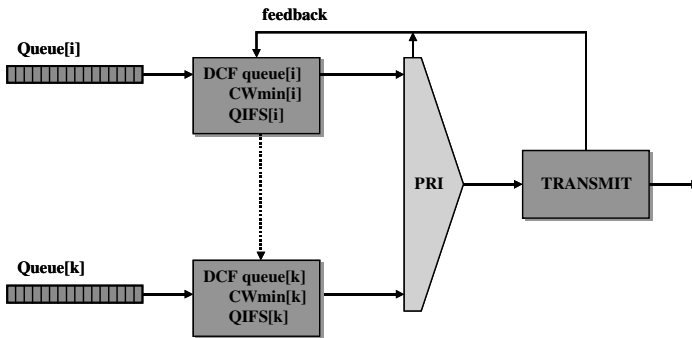


Fig. 4. Enhanced DCF, IEEE 802.11e IEEE 802.11-01/132r1 Proposal

3.2 Flow Adaptation and Automated Resource Control

The ideal Wireless Internet scenario has a user operating a real-time, multimedia videoconferencing session over a wide-area Wireless Internet connection. The wireless client is highly mobile within a local campus or dense city limits and is sharing a media rich presentation with multiple parties, dispersed in different connectivity scenarios (e.g., home, hotel, or corporate office). The user is experiencing a high degree of QOS supporting the multimedia session and presentation delivery, regardless of his movement or locality. While such a scenario seems far from current bandwidth and QOS paradigms, it is a motivation for a solution supporting wireless IP QOS and dynamic resource control. Multiple effects including time-varying channel conditions, local or remote congestion conditions, and end-to-end QOS requirements must be matched with a dynamic, automated resource control framework. Along with this, a flexible, adaptive application capable of offsetting the limitations of the network to bring to harmony reliability, latency and throughput degradations while hiding the user from the underlying complexity, degraded QOS, and mobility effects is

also essential. To enable our solution, we promote the use of middleware enabled IP network services as necessary extension to the traditional IP architecture.

Resource Control & Management Hierarchy. Fundamental to our approach is the need to integrate with an end-to-end QOS strategy. We support this at the IP layer through the Differentiated Services model presently under development by the IETF community; however, our framework must be complementary with other schemes (i.e., IntServ/RSVP) or new IP (or higher layer) QOS models as the Internet community creates them. Under a wide area Internet provisioning model, we utilize the role of the RRM to manage or proxy the end-to-end provisioning model against the resource requirements within the global wireless channel. The RRM can support an admission control scheme, such as defined in [19],[20] using policy-based management, or may integrate the requested provision in a more programmable fashion, where the provisioning request is invoked via a remote method, allowing the RRM, operating as a standalone admission control thread, to respond dynamically to the rapidly fluctuating conditions and policies of the environment. If the admission process is successful, the RRM must propagate new provisioning policies to specific RRM Agents, which must also invoke local methods to finalize admission and ‘connect’ the admitted session or flow. Otherwise, the RRM will decline admission and inform the outside party without the knowledge or participation of local RRM Agents. This separation is a necessary design feature of the distributed framework. In this paper, all provisioning aspects of the global wireless channel and the components within it are assumed autonomous and automated within the wireless resource controlled and managed domain.

Wireless channel policy control & monitoring. As discussed in the previous section, the role of the RRM is to manage the end-to-end provisioning model against the resource requirements within the global wireless channel. Additionally, RRM monitors the channel for utilization² and keeps congestion within policy threshold limits pre-configured or adjusted throughout channel operation. There may exist multiple thresholds specific to the global channel utilization, specific QOS categories or alternative timescales, allowing the RRM to make granular policy decisions. This flexibility was also employed and demonstrated in [8]. By way of RRM APIs, the central RRM can accept application-specific bandwidth utility functions from RRM Agents on behalf of local applications and normalizes³ these functions to specific provisioning policies (e.g., DiffServ PHBs specs, RSVP QOS_Spec), which are provisioned locally or requested outside of the RRM provisioning realm (i.e., WLAN).

² It is assumed that the RRM Agents will appropriately enforce and/or police flow QOS provisions, making it redundant that the RRM must do the same.

³ It is anticipated that such bandwidth utility functions may not be pervasive, however, they may be quite helpful to the RRM in its flexibility to accommodate adaptive applications. Alternatively, qualitative or quantitative specifications may be more likely, and thus, support for utility function normalization to simpler specification may be necessary.

The scope of the bandwidth utility is global in nature and does not address adaptation to time-varying conditions related to local fading, overlapping cells, etc.

It is possible that the RRM may uniformly inform Agents of global channel usage conditions allowing the RRM Agents to tune policies at the transport (or application⁴) layer or local link layer components (e.g., MAC scheduler). This would support proactive control on a local device level.

Local policy control & monitoring. The RRM Agent's scope is centered on its local environment. These Agents will reside on both wireless clients and wireless controller stations (e.g., base station or router). A critical function of the RRM Agent is to assess the time-varying conditions around the local device and manage this against the requirements of the running applications' flows and global channel effects. A SINR monitoring function is supported by the link layer to continuously fast monitor the time-varying conditions surrounding the wireless client as programmed by the RRM Agent. The fast timescale feedback, allows the Agent to adjust policies on a local level, allowing the client to compete more aggressively for the wireless channel. These updated policies can be specific to particular flows to ensure fairness, uniformly across all running flows on the client, or weighted (or prioritized) according to specific QOS specifications. Similar to the global bandwidth utility functions (discussed in 3.2.1.2), the RRM Agent exposes APIs allowing it to accept application-specific SINR utility functions [16] from local applications directly and translates these functions into specific, link layer provisioning policies. The scope of the SINR utility functions allow applications to adapt to local conditions at the device on a faster timescale.

State and policy information can be made available between layers by the RRM Agent to synchronize transport (per flow) congestion control and reliability states, network layer QOS policies and flow policing state, global usage and local SINR state. This capability is essential to the operation of the RRM Agent, allowing it dynamically control various layered components of the wireless client by leveraging multi-layer state or policies; extending its local intelligence (i.e., across application, transport, network, link layers) or influence on application adaptation.

Distributed State Machine Operation. In [8], we demonstrated that a flexible network layer QOS mechanism could allow dynamic provisioning reconfiguration either through manually (GUI-driven) invoked commands, or automatically through threshold triggered policy enforcement algorithms. These centralized and multi-threaded (i.e., manual and pre-configured closed-loop threads) algorithms ran on a central, logical administrator handling the complete measurement-based control system, with the resulting policies dispatched to QOS agents operating at remote clients to enforce flow-level QOS policies. Our previous solution allowed the central administrator to operate in either a manual, user-driven provisioning mode or in an automated provisioning mode. However, under the wireless environment, we partition the control system across the resource control hierarchy. This is necessary to achieve

⁴ Informing the applications to consider employing alternative bandwidth savings or aggressive mechanisms or to dial down or dial up bandwidth utilities.

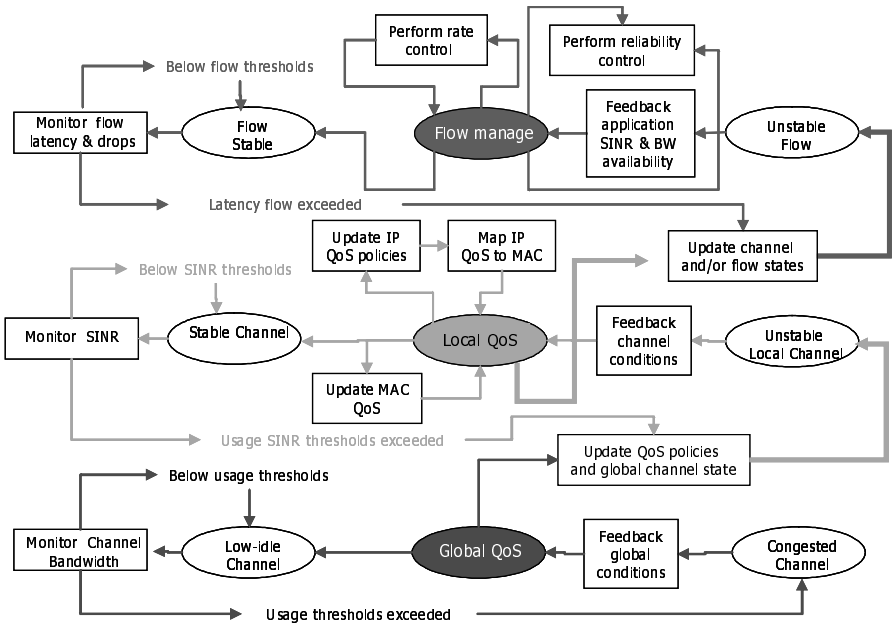


Fig. 5. Distributed State Machine

autonomous control on a global level, a local level and flow level; each of which is directed at a different set of objectives (e.g., global usage efficiency, local access maximization and fairness, and end-to-end flow control and adaptation), but overlapped on their influence on the wireless channel resource.

As illustrated in Figure 5, three autonomous levels of feedback-based control are supported in the distributed state machine. At each level, a stable and unstable state exists, while an operational state is centered between them to represent the control state. Also at each level, a monitoring procedure checks against stability thresholds to determine the possibility of instability and the need to enter into a control state, invoking alternative algorithms, which manage the particular level of concern. At the global and local level, policy changes will cause the state machine to enter into an unstable local channel state or unstable flow state, respectively. Multiple instances of the state machine procedure will run; one per wireless client and one for each of the flows running within the wireless environment. The flow procedure is essentially part of the normal transport process supporting both congestion and reliability control for each session flow. However, we expose it here as a necessary integration aspect of our framework. Also shown at the local and flow level is a procedure to update the application on specific bandwidth availabilities and SINR state, respectively.

4 Further Work

While the proposed framework is still preliminary, we believe it is, nevertheless, quite feasible and appropriate for IP-based RANs or wireless LANs. However, a number of open issues and areas require further investigation. The specific optimization algorithms, which can be run at the RRM and RRM Agent are clearly of open research. Inter-layer control interactions and their dynamics are also of open research and analysis. Since the various control and policy threads are happening on multiple timescales, we would need to explore their dynamics, both in terms of reaching appropriate stability and monitoring feedback synchronization. The programmability aspects also pose operational uncertainty when coupled with the traditional strict layering in the IP-based stacks. We intend to approach our investigations using simulation techniques.

In closing, with the convergence towards a Wireless Internet, it is our belief that automation of wireless resource and QOS control is necessary to match the highly dynamic nature of wireless and mobile environments. This paper has argued for an IP-based radio resource control system as the means to achieve this objective.

References

1. Campbell et al, "Mobiware Project", <http://www.comet.columbia.edu/mobiware/>.
2. Sinha P., Nandagopal, T., Venkitaraman N., Sivakumar R., Bharghavan V., "WTCP: A Reliable Transport Protocol for Wireless Wide-Area Networks", Wireless Networks, 1999
3. Holma, H., Toskala A., "WCDMA for UMTS: Radio Access for Third Generation Mobile Communications," John Wiley & Sons, ISBN 0-471-72051-8
4. Bernet Y., Blake S., Binder J., Carlson M., Keshav S., Davies B., Ohlman B., Verma D., Wang Z, Weiss W., "A Framework for Differentiated Services," Internet-Draft draft-ietf-diffserv-framework-01.txt, work in progress, Feb. 1999.
5. Bernet Y., "The Complementary Roles of RSVP and Differentiated Services in the Full-Service QOS Network," IEEE Communications Magazine, Vol. 38, No. 2, pp. 154-162, Feb. 2000.
6. Braden R., Zhang L., Berson S., Herzog S., Jamin S., "Resource ReSerVation Protocol (RSVP)," Version 1 Functional Specification, RFC 2205, Sept. 1997.
7. Braden R., Clark D., Shenker S., "Integrated Services in the Internet Architecture: an Overview," RFC 1633, June 1994.
8. Vicente J. et al, "Host-based, Network Traffic Control System", Intel Technical Report (patent pending).
9. Ayyagari, A., Bernet Y., Moore T., "IEEE 802.11 Quality of Service –White Paper", IEEE 028 document, February, 2000.
10. Microsoft, "Quality of Service Technical Overview", <http://www.microsoft.com/windows2000/library/howitworks/communications/trafficmgt/qos.asp>, Sept. 1999.
11. Bianchi G., Campbell A.T., Liao, R.R.F., "On Utility-Fair Adaptive Services in Wireless Networks", IEEE 1998.

12. Barry M., Campbell A.T., Veres A., "Distributed Control Algorithms for Service Differentiation", Proc. IEEE INFOCOM'2001, Anchorage, Alaska, 2001.
13. Lee K., "Adaptive Network Support for Mobile Multimedia", MOBICOM 95, Berkeley, CA, 1995.
14. Boyle J., Cohen R., Durham D., Herzog S., Rajan R., Sastry A., "The COPS (Common Open Policy Service) Protocol," Internet Draft draft-ietf-rap-cops-07.txt. Aug. 1999.
15. Gomez J., Campbell, A.T., Morikawa, H. "A Systems Approach to Prediction, Compensation and Adaptation in Wireless Networks", WOWMOM 98, Dallas, Texas, 1998.
16. Liu X., Chong, E. K. P., Shroff, N.B., "Transmission Scheduling for Efficient Wireless Utilization", IEEE INFOCOM 2001.
17. Cervello G., Choi S., Qiao D., "Channel Model Proposal v2.0 for 802.11e MAC Simulations", IEEE document, April, 2001.
18. IEEE, "Wireless LAN Media Access Control (MAC) and Physical Layer (PHY) Specifications", IEEE Standard 802.11, January 1999.
19. Rajan R., Verma D., Kamat S., Felstaine E., Herzog S., "A Policy Framework for Integrated and Differentiated Services in the Internet," IEEE Network Magazine, Sept./Oct. 1999.
20. Yavatkar R., Hoffman D., Bernet Y., Baker F., Speer M., "SBM (Subnet Bandwidth Manager): A protocol for RSVP-based Admission Control over IEEE 802-style networks, draft-ietf-issll-is802-sbm-10.txt," IETF Internet-Draft, January 2000.
21. Disabato C. M., Wireless LANs, The Burton Group, March 2001.

Is It Worth Involving Several Cooperating Agents for Multimedia User's Admission in Cellular Networks?

Youssef Iraqi and Raouf Boutaba

University of Waterloo, DECE, Waterloo, Ont. N2L 3G1, Canada,
{iraqi, rboutaba}@bbcr.uwaterloo.ca

Abstract. In this paper, we enhance our multi-agent system for resource management developed for cellular mobile networks. The new scheme allow agents to dynamically adapt to changes in the network load in order to maintain a target call dropping probability. We investigate the impact of the number of neighboring agents involved in a call admission decision in addition to the agent receiving the call request. The neighboring agents provide significant information about their ability to support the new mobile user in the future. This distributed process allows the original agent to make a more clear-sighted admission decision for the new user. Simulations are presented with a detailed analysis of a comparison between two schemes involving different numbers of agents.

1 Introduction

Cellular mobile networks have to continue supporting their mobile users after they leave their original cells. This rises a new challenge to resource management algorithms. For instance a call admission (CA) process should not only take into consideration the available resources in the original cell but also in neighboring cells as well.

Mobile users are in a growing demand for multimedia applications, and the next generation wireless networks are designed to support such bandwidth greedy applications. The (wireless) bandwidth allocated to a user will not be fixed for the lifetime of the connection as in traditional cellular networks, rather the base station will allocate bandwidth dynamically to users. Many evolving standards for Wireless Broadband Systems, UMTS and IMT2000 have proposed solutions to support such capability.

In [4] we have proposed a Multi-Agent system for call admission management designed for wireless mobile multimedia networks with dynamic bandwidth allocation. The call admission process involves the cell that receives the call admission request and a cluster of neighboring cells. The agents share important resource information so the new admitted user will not be dropped due to hand-offs. Consequently, the network will provide a low call dropping probability while maintaining a high resource utilization.

In this paper, we propose an enhancement of the Multi-Agent system and

propose a mechanism for dynamic adaptation to obtain a target call dropping probability (CDP). We investigate the impact of the number of involved agents in the CA process on the achieved performance, in terms of average bandwidth utilization and call dropping probability. Other parameters such as call blocking probability are also investigated.

The paper is organized as follows. In section 2, we describe the Multi-Agent architecture considered in this paper. Section 3 defines the dynamic mobile probabilities used by our Multi-Agent system and presents the call admission process performed locally by agents in our system. It also introduces the overall admission process and agent's cooperation. Section 4 describes the algorithm used to dynamically achieve a target call dropping probability. Section 5 discusses the conducted simulation parameters and presents a detailed analysis of the obtained results. Finally, section 6 concludes the paper.

2 The Multi-agent Architecture

We consider a wireless/mobile network with a cellular infrastructure that can support mobile terminals running applications which demand a wide range of resources. Users can freely roam the network and experience a large number of handoffs during a typical connection. We assume that users have a dynamic bandwidth requirement. The wireless network must provide the requested level of service even if the user moves to an adjacent cell. A handoff could fail due to insufficient bandwidth in the new cell, and in such case, the connection is dropped.

To reduce the call dropping probability, we have proposed in [4] a multi-agent system that allows neighboring cells to participate in the decision of a new user admission. Each cell or base station has an agent running on it. The agent keeps track of the cell's resources and shares information with neighboring agents to better support mobile users. Each involved agent in an admission request will give its local decision according to its available resources and information from other agents and finally the agent at the cell where the request was issued will decide if the new request is accepted or not. By doing so, the new admitted connection will have more chances to survive after experiencing handoffs.

We use the notion of a cluster similar to the shadow cluster concept [5]. The idea is that every connection exerts an influence upon neighboring base stations. As the mobile terminal travels to other cells, the region of influence also moves. The cells influenced by a connection are said to constitute a cluster (see figure 2). Each user¹ in the network, with an active connection has a cluster associated to it. The agents in the cluster are chosen by the agent at the cell where the user resides. The number of agents of a user's cluster depend on factors such as user's current call holding time, user's QoS requirements, terminal trajectory and velocity.

¹ in the rest of the paper the term "user" and "connection" are used interchangeably

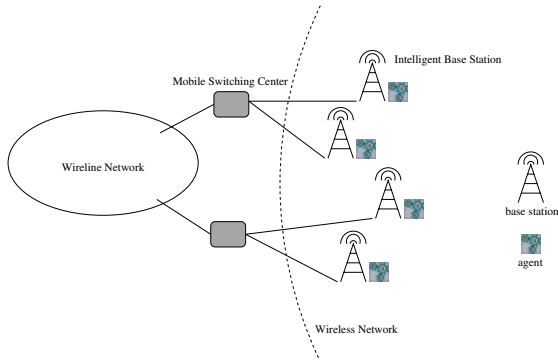


Fig. 1. A wireless Network and the Multi-Agent system

3 The Call Admission Process

3.1 Dynamic Mobile Probabilities

We consider a wireless network where the time is divided in equal intervals at $t = t_0, t_1, \dots, t_m$. Let j denote a base station (and the corresponding agent) in the network, and x a mobile terminal with an active wireless connection. Let $K(x)$ denote the set of agents that form the cluster for the active mobile terminal x . We denote $P_{x,j,k}(t) = [P_{x,j,k}(t_0), P_{x,j,k}(t_1), \dots, P_{x,j,k}(t_{m_x})]$ the probability that mobile terminal x , currently in cell j , to be active in cell k , and therefore under the control of agent k , at times $t_0, t_1, t_2, \dots, t_{m_x}$. $P_{x,j,k}(t)$ represents the projected probabilities that mobile terminal x will remain active in the future and at a particular location. It is referred to as the Dynamic Mobile Probability (DMP) in the following. The parameter m_x represents how far in the future the predicted probabilities are computed.

Those probabilities may be function of several parameters such as: handoff probability, the distribution of call length for a mobile terminal x , cell size, user mobility profile, etc.

For each user x in the network, the agent that is responsible for, decides the size of the cluster $K(x)$, which is the set of agents involved in the admission process, and sends the DMPs to all members in $K(x)$. The agent must specify whether the user is a new one (in which case the agent is waiting for responses from the members of $K(x)$) or not. As the user roams the network, the agents belonging to his cluster change. Only those agents that belong to a user's cluster will receive the DMPs and hence will have to deal with bandwidth reservation for the user. Agents that are no more part of the user's cluster will release any reserved bandwidth made for that user.

A method for computing dynamic mobile probabilities taking into consideration mobile terminal direction, velocity and statistical mobility data, is presented in [1]. Other schemes to compute these probabilities are presented in [2] [3]. To

compute these probabilities, one can also use mobiles' path/direction information readily available from certain applications, such as the route guidance system of the Intelligent Transportation Systems with the Global Positioning System (GPS).

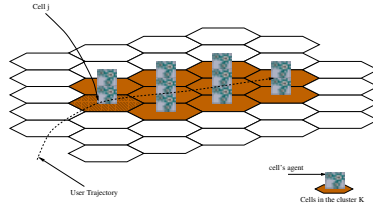


Fig. 2. Example of a user's cluster

3.2 Local Call Admission Decision

At each time t_0 each agent, in a cluster $K(x)$ involved in our CA process for user x , makes a local CA decision for different times in the future (t_0, t_1, \dots, t_{m_x}). Based on these CA decisions, we call Elementary Responses, the agent makes a final decision which represents its local response to the admission of user x in the network. Elementary responses are time dependent. The computation of these responses is different depending on the user location and type. The user can be either a local new user or a new user that has a non null probability to be in this cell in the near future.

The network tries first to continue supporting old users and uses the DMPs to check if a cell can accommodate a new user who will possibly come to the cell in the future. The cell's agent can apply any local call admission algorithm to compute the elementary responses. We write $r_k(x, t)$ the elementary response of agent k for user x for time t . We assume that $r_k(x, t)$ can take one of two values: -1 meaning that agent k can not accommodate user x at time t ; and $+1$ otherwise. A detail description of how to compute the elementary responses is presented in [4].

Since the elementary responses for future foreign users are computed according to local information about the future, they should not be assigned the same confidence degree. Indeed, responses corresponding to the near future are more likely to be more accurate than those of the far future.

We write $C_k(x, t)$ the confidence of agent k in its elementary response $r_k(x, t)$. Agent k has to compute (or simply choose) the confidence degree $C_k(x, t)$, typically between 0% and 100%.

The confidence degrees depend of many parameters. It is clear that the time in the future for which the response is computed has great impact on the confidence of that response. The available bandwidth when computing the elementary response also affects the confidence.

To compute the confidence degrees we propose a formula that uses the percentage of available bandwidth when computing the elementary response as an indication of the confidence the agent may have in this elementary response. The confidence degrees are computed using:

$$Confidence = e^{(1-p)} * p^n \quad (1)$$

where p is a real number between 0 and 1 representing the percentage of available bandwidth at the time of computing the elementary response. And $n \geq 1$ is a parameter that is chosen experimentally to obtain the best efficiency of the call admission routine.

If for user x , agent k has an elementary response $r_k(x, t)$ for each t from t_0 to t_{m_x} , those elementary responses are weighted with the corresponding DMPs $P_{x,j,k}(t_0)$ to $P_{x,j,k}(t_{m_x})$, to compute the final response. The final response from agent k to agent j concerning user x is then :

$$R_k(x) = \frac{\sum_{t=t_0}^{t=t_{m_x}} r_k(x, t) \times P_{x,j,k}(t) \times C_k(x, t)}{\sum_{t=t_0}^{t=t_{m_x}} P_{x,j,k}(t)} \quad (2)$$

where $C_k(x, t)$ is the confidence of agent k in the elementary response $r_k(x, t)$. To normalize the final response, each elementary response is also divided by the sum over time t of the DMPs in cell k . Finally, agent k sends the response $R_k(x)$ to the corresponding agent j . Note that $R_k(x)$ is a real number between -1 and 1 .

3.3 Distributed Call Admission Process

Here the decision takes into consideration the responses from all the agents in the user's cluster. The admission process concerns only new users seeking admission to the network, not already accepted ones. We assume that agent j has already decided the cluster $K(x)$ and that agent j has already assigned to each agent k in the cluster $K(x)$ a weight $W_k(x)$. Each weight represents the importance of the contribution of the associated agent to the global decision process. Usually, the more an agent is involved in supporting the user, the higher is its weight value. Weights $W_k(x)$ depend on the DMPs. We propose the following formula to compute the weights $W_k(x)$:

$$W_k(x) = \frac{\sum_{t=t_0}^{t=t_{m_x}} P_{x,j,k}(t)}{\sum_{k' \in K(x)} \sum_{t=t_0}^{t=t_{m_x}} P_{x,j,k'}(t)} \quad (3)$$

Relevance. In this paper, we introduce a new parameter that we call spatial relevance or simply relevance of an agent. To explain the idea of relevance, let's take the following example: consider a linear highway covered by 10 square cells as in figure 3. Assume that a new user, following the trajectory shown in figure 3, is requesting admission in cell number 0 and that the CAC process involves

5 cells. Responses from agents number 1, 2, 3 and 4 are relevant only if agent number 0 can accommodate the user. Similarly, responses from agents 2, 3 and 4 are relevant only if agent 1 can accommodate the new user when it hands off from cell 0. And the same principle applies to the other agents. This is because a response from an agent is irrelevant if the user can not be supported until that agent. We write $\Phi_k(x)$ the relevance of agent k for user x .

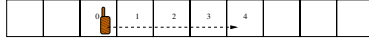


Fig. 3. An example of a highway covered by 10 cells

The relevance depends only on the topology of the considered cellular network. For the linear highway example of figure 3, we propose the following relevance formula:

$$\Phi_0(x) = 1 \text{ and } \Phi_k(x) = \prod_{l=1}^k \frac{(1 + R_{l-1}(x))}{2} \quad (4)$$

Note that for each $k \in K(x)$ we have $0 \leq \Phi_k(x) \leq 1$. Note also that in eq. 4, agent j (the agent receiving the admission request) has the index 0 and that the other agents are indexed in an increasing order according to the user direction as in figure 3.

The agent computes the sum of $R_k(x) \times W_k(x) \times \Phi_k(x)$ over k . The final decision of the call admission process for user x is based on:

$$D(x) = \frac{\sum_{k \in K(x)} R_k(x) \times W_k(x) \times \Phi_k(x)}{\sum_{k' \in K(x)} W_{k'}(x) \times \Phi_{k'}(x)} \quad (5)$$

Note that $-1 \leq D(x) \leq 1$. If $D(x)$ is higher than a certain threshold, we call acceptance threshold, the user x is accepted; the user is rejected otherwise. The more higher is $D(x)$ the more likely the user connection will survive in the event of a handoff.

Agent's Cooperation. At each time t , an agent j decides if it can support new users. It decides locally if it can support old users as they have higher priority than new users. This is because, from a user point of view, receiving a busy signal is more bearable than having a forced termination. Agent j also sends the DMPs to other agents and informs them about its new users requesting admission to the network (step 2 in figures 4, 5). Only those new users who can be supported locally are included. New users that can not be accommodated locally are immediately rejected.

At the same time, agent j receives DMPs from other agents and is informed about their new users requests. Using equation 2, agent j decides if it can support

theirs new users in the future and sends the responses to those agents (step 3 in figures 4, 5). When agent j receives responses from the other agents concerning its own new users, it performs the following for each of these users (step 4 in figures 4, 5): If it can not accommodate the call, the call is rejected. If the agent can accommodate the call, then the CA decision depends on the value of $D(x)$ as computed in eq. 5. A detailed description of the algorithm is presented in [4].

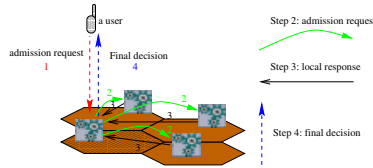


Fig. 4. Agent’s cooperation for the admission of a user

Figure 4 shows the different steps of agent’s cooperation when processing an admission request. Figure 5 depicts the admission process diagram at the agent receiving the admission request and at an agent belonging to the cluster. Because the admission request is time sensitive the agent waiting for responses from the agents in the cluster will wait until a predefined timer has expire then he will assume a negative response from all agents that could not respond in time.

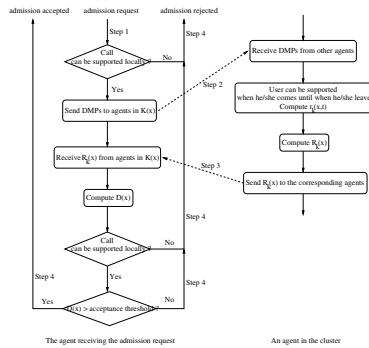


Fig. 5. Admission process diagram

It is worth noting that as the user roams, the corresponding dynamic mobile probabilities in certain cells will decrease; and as a consequence the bandwidth allocation made in these cells will also be decreased and eventually released. This is because a cell makes new reservations for each time slot. If the cell receives

new DMPs for a user, the reservations are updated with the new requirements. And if the cell does not receive the DMPs for a user that had some bandwidth reservation in the past time slot, the cell assumes that it does not make part of the user's cluster any more and hence releases the reserved bandwidth.

4 Maintaining a Target Call Dropping Probability

In this section we explain how our algorithm varies the value of the acceptance threshold to maintain a target CDP value. We assume that each Mobile Switch Center (MSC), controlling a set of agents in the network, modifies the acceptance threshold of the agents it controls in order to maintain a target CDP. The following is the pseudo-code of the algorithm for adjusting the acceptance threshold, we will refer to as algorithm 1 in the remaining of the paper.

Algorithm 1: Pseudo-code for adjusting the acceptance threshold T_{acc}

```

 $w = \lceil \frac{1}{target\ CDP} \rceil$ ;  $wobs = w$ ;  $nA = 0$ ;  $nD = 0$ 
if a user is accepted
  {  $nA++$ ;
  if ( $nA \geq wobs$ )
    {
    if ( $nD == wobs/w$ ) { $wobs = w$ ;  $nA = 0$ ;  $nD = 0$ ;}
    else { $wobs++ = w$ ; if ( $T_{acc} > -1.0$ )  $T_{acc}- = 0.01$ ;}
    }
  }
if a user is dropped
   $nD++$ ;
  if ( $nD > wobs/w$ ) { $wobs++ = w$ ; if ( $T_{acc} < 0.95$ )  $T_{acc}+ = 0.01$ ;}

```

The MSC begins by selecting a reference observation window w according to the target CDP as follows: $w = \lceil \frac{1}{target\ CDP} \rceil$. Note that we do not include the case where the target CDP is equal to zero, since this one is almost impossible to achieve and not practical from the provider point of view.

The variable representing the observation window $wobs$ is set to w , and the number of accepted users nA as well as the number of dropped users nD are set to zero.

If a new user is accepted in the system then nA is incremented by one. If we have observed at least $wobs$ accepted users ($nA \geq wobs$) then, if the number of users dropped is equal to the maximum allowed dropping value, we set $wobs$ to w and set nA and nD to zero and restart from the beginning. If the number of users dropped is less than the maximum allowed, then we increase $wobs$ and decrease the acceptance threshold. This means that we will allow more users to be admitted in the system.

In case a user is dropped then nD is incremented by one. If the number of dropped users exceeds the maximum allowed value, then we increase $wobs$ and increase the acceptance threshold. This means that we increase our observation

window and will allow less users to be admitted in the system.

Note that the proposed algorithm aims to achieve exactly the target CDP. This can easily be modified to let the actual CDP lay between a maximum and a minimum allowed values. Note also that the maximum allowed acceptance threshold is set to 0.95 in algorithm 1 in order for the network to accept a minimum number of users even if a congestion occurs.

5 Performance Evaluation

We evaluate here our Multi-Agent system with different numbers of agents involved in the CA process. We compare the performance of the scheme in the two following scenarios:

1. two agents are involved in the CAC process. This scheme will be referred to as **SC1**.
2. five agents are involved in the CAC process. This scheme will be referred to as **SC2**.

5.1 Simulation Parameters

For simplicity, we evaluate the performance of our Multi-Agent system for mobile terminals which are traveling along a highway as in figure 3. This is a simplest environment representing a one-dimensional cellular system. In our simulation study we have the following simulation parameters and assumptions:

1. The time is quantized in intervals $\tau = 10s$
2. The whole cellular system is composed of 10 linearly-arranged cells, laid at 1-km intervals. Cells are numbered from 1 to 10.
3. Cell 1 and 10 are connected so that the whole cellular system forms a ring architecture. This allows to avoid the uneven traffic load that will be experienced by cell 1 and 10 otherwise.
4. Each cell has a fixed capacity of 100 bandwidth units except cells 3, 4 and 5 which have 50, 30 and 50 bandwidth units respectively. This is to create a local congestion that will remain for a long period. An example of such case is a temporary increase in the interference level which prevents the cells from using all their capacity.
5. Connection requests are generated in each cell according to Poisson process. A newly generated mobile terminal can appear anywhere in the cell with equal probability.
6. Mobile terminals can have speeds of: 70, 90, or 105 km/h. The probability of each speed is 1/3, and mobile terminals can travel in either of two directions with equal probability.
7. We consider three possible types of traffic: voice, data, and video. The probabilities associated with these types are 0.3, 0.4 and 0.3 respectively. The number of bandwidth units (BUs) required by each connection type is: voice = 1, data = 5, video = 10. Note that fixed bandwidth amounts are allocated to users for the sake of simplicity.

8. Connection lifetimes are exponentially-distributed with a mean value equal to 180 seconds.
9. We simulate a total of 10 hours of real-time highway traffic, with a constant cell load equal to 720 new calls/h/cell.
10. The DMPs are computed as in [1].
11. The confidence degree is computed using eq. 1 with $n = 3$.
12. The relevance is computed using eq. 4.
13. All users with a specific type of service have the same acceptance threshold. Algorithm 1 is used to adjust the acceptance threshold T_{acc} of all 10 agents and the target CDP is 10%. We assume that all 10 agents are under the control of one Mobile Switching Center. The accepted thresholds for voice, data and video users are set to $1.7 * T_{acc}$, $1.2 * T_{acc}$ and T_{acc} respectively. This is to achieve fairness between voice, data and video users. Indeed, if we use the same acceptance threshold for all users irrespective to their class of service, very few video users will be admitted to the network. This is because video users require more capacity than the other users, and hence it is more difficult to obtain high responses ($D(x)$).

The following additional simulation parameters are used for the **SC1** scheme:

- m_x is fixed for all users and is equal to 18. This means that the DMPs are computed for 18 steps in the future.
- The size of the cluster $K(x)$ is fixed for all users and is equal to 2. This means that one agent in the direction of the user and the agent of the cell where the user resides form the cluster.

For **SC2** scheme, the following additional simulation parameters are assumed:

- m_x is fixed for all users and is equal to 25. This means that the DMPs are computed for 25 steps in the future.
- The size of the cluster $K(x)$ is fixed for all users and is equal to 5. This means that four agents in the direction of the user and the agent of the cell where the user resides form the cluster.

5.2 Simulation Results

In our simulations, a user x requesting a new connection is accepted into a cell only if the final decision $D(x)$ is above the acceptance threshold corresponding to the user class of service (voice, data or video). Figure 6 depicts the call dropping percentage achieved when using scheme **SC2**. The call dropping percentage represents the ratio of dropped users to the number of admitted users in the system. This is the aggregate call dropping percentage including all types of service. We can notice that algorithm 1 allows the actual CDP to approach the target CDP by varying the value of the acceptance threshold T_{acc} .

In figure 7, we compare the percentage of refused calls, given the offered load, when using scheme **SC1** and **SC2**. We can notice that **SC2** refuses less users than **SC1**. Indeed, **SC2** accepts about 8% more users than **SC1**. At a first sight,

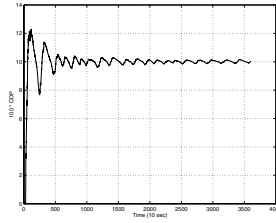


Fig. 6. Call dropping percentage

this result may seem abnormal. Indeed, scheme **SC2** involves five agents in the CAC decision process (3 more agents than **SC1**), and thus it is more difficult for a new user to be admitted by **SC2** than **SC1**. However, as we will see later in this section, **SC2** has the ability to avoid admitting those users who are most likely to be dropped and can use the saved bandwidth to accept more users who can most likely be supported.

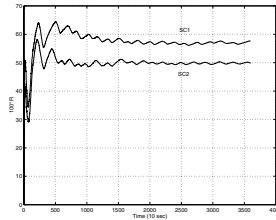


Fig. 7. Percentage of refused calls

Figure 8 shows that **SC2** not only accepts more users than **SC1** but also allows for a better resource utilization. In fact, **SC2** uses almost 10 bandwidth units more than **SC1**. It is worth noting that the low resource utilization experienced by the two schemes is due to the number of video users in the system and to the assumption that the whole system is controlled by one MSC. The latter assumption means that when a part of the network experience a congestion, the whole network is affected by refusing more users (since the MSC increases the acceptance threshold for all the agents in the network). Although the simulated one MSC configuration is not likely to happen in the real-world, simulation results show the potential benefit of using scheme **SC2** compared to scheme **SC1**.

To further compare the two considered schemes, we compute the individual dropping percentage among the three considered classes of service, namely voice, data and video. The simulation results are shown in figure 9.

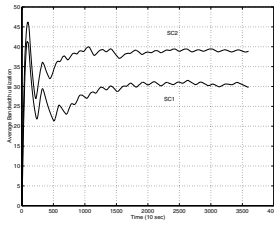


Fig. 8. Average bandwidth utilization

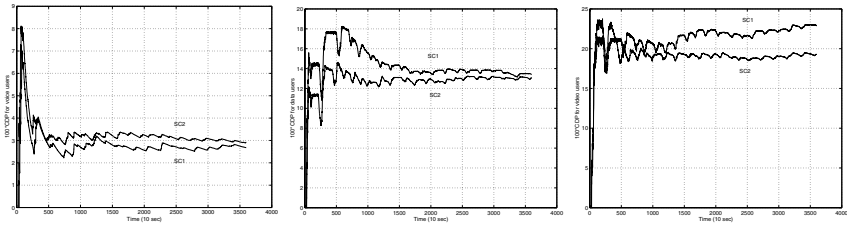


Fig. 9. Percentage of dropped voice, data and video users

In this figure, we can observe that the two schemes, **SC1** and **SC2**, achieve almost the same dropping percentage for voice and data users respectively, with a slightly better performance of **SC2** in case of data users. However, **SC2** drops almost 4% video users less than **SC1**.

As the percentage of dropped users depicted in figure 9 is computed according to the number of accepted users in each class of service, the comparison will not be fair if we do not observe the number of admitted users within each class of service for the two schemes. Figure 10 shows the percentage of refused calls within each class of service, and figure 11 plotted the number of accepted users within each class of service when using the two schemes.

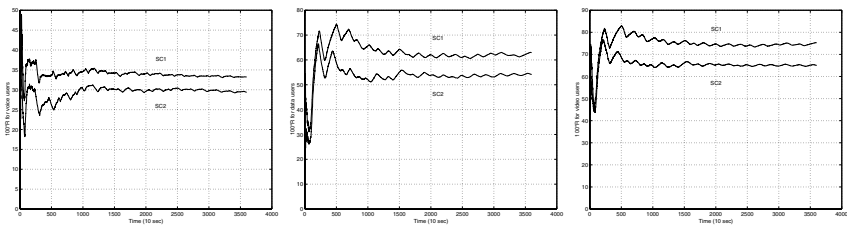


Fig. 10. Percentage of refused voice, data and video calls

According to figure 10, **SC2** refuses less users than **SC1** irrespective of users classes of service. This means that **SC2** accepts more users while achieving the same CDP in case of voice and data users, and that it allows more video users to be admitted to the network while achieving a lower CDP compared to **SC1**. According to figure 11, **SC2** accepts about 1500 video users more than **SC1** for the 10 real-time hours considered.

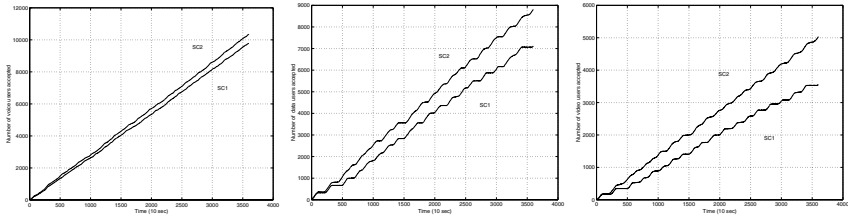


Fig. 11. Number of accepted voice, data and video calls

The bad performance achieved by **SC1** is explained by the fact that this scheme can not differentiate between those users who can be supported and those who can not. Its short sight prevents the scheme from being informed about a far congestion. Thus, the only way for **SC1** to reduce the CDP to the target value is to accept less users in the network, which results in a poor resource utilization.

On the other hand, since **SC2** involves more agents in the CA process than **SC1**, the scheme is able to distinguish between those users who can be supported and those who are most likely to be dropped due to some congestion. This has the two following benefits: (1) the scheme can accept more users without sacrificing the CDP; (2) the bandwidth saved from not allowing some “bad” users to be admitted in the network, can be used to admit more “good” users.

We have conducted several other simulations with different offered loads and different simulation parameters (such as different mean holding time). The main observation worth highlighting here is that the two schemes **SC1** and **SC2** achieve almost the same performance in case of no congestion or in case of a uniformly distributed congestion. The latter case is less important since it can be solved off-line by increasing the network capacity. We have observed in the simulations presented in this paper, **SC2** achieves a better performance in case of a local congestion. The fact that the two schemes achieve the same results in case of a non congested network or in case of a uniformly distributed congestion is foreseeable. This is mainly because the responses from the three additional agents in **SC2** (agents 2, 3 and 4 in figure 3) only confirm what the two involved agents in **SC1** (agents 0, 1 in figure 3) have decided.

Of course, **SC2** does not have only advantages. As **SC2** involves more agents in the CAC decision process, it induces more communications between base stations and also requires more processing power than **SC1**. These resources are

less critical compared to the wireless network bandwidth. A good compromise is to use **SC1** when the network is not congested and use **SC2** when a congestion is detected. The process of selecting the good scheme is out of the scope of this paper and is subject to future work.

6 Conclusion

We have described a Multi-Agent system for resource management suitable for wireless multimedia networks. The proposed system operates in a distributed fashion by involving, in a call admission decision, not only the agent receiving the admission request, but also a determined number of neighboring agents. We also proposed an algorithm that dynamically adjusts users acceptance threshold to achieve a target call dropping probability. We also presented an analysis of the comparison between two call admission schemes involving a different number of agents in the decision process. We have observed that it is worth involving more agents in the CA decision in case of local congestion. This allows the scheme to take a more clear-sighted admission decision for new users, hence, achieving better resource management and quality of service.

In this paper, we have compared the performance of our Multi-Agent system for resource management when involving two and five agents in the admission decision respectively. However, our system can involve any number of agents. We have demonstrated that in some cases it is worth involving several agents in the admission process. The choice of the number of agents to involve and when this should happen is an important issue that will be addressed in futur work.

References

1. D. A. Levine, I. F. Akyildz and M. Naghshineh, 'A Resource Estimation and Call Admission Algorithm for Wireless Multimedia Networks Using the Shadow Cluster Concept,' *IEEE/ACM Transactions on Networking*, vol. 5, no. 1, Feb. 1997.
2. Sunghyun Choi and Kang G. Shin, 'Predictive and Adaptive Bandwidth Reservation for Hand-Offs in QoS-Sensitive Cellular Networks,' in *Proc. ACM SIGCOMM'98*, pp. 155-166, Vancouver, British Columbia, Sep. 2-4, 1998.
3. Songwu Lu and Vaduvur Bharghavan, 'Adaptive Resource Management Algorithms for Indoor Mobile Computing Environments,' in *Proc. ACM SIGCOMM'96*, pp. 231-242, Aug. 1996.
4. Y. Iraqi and R. Boutaba, 'A Multi-Agent System for Resource Management in Wireless Mobile Multimedia Networks,' in *Proc. IFIP/IEEE DSOM 2000*, Dec. 2000.
5. D. A. Levine, I. F. Akyildz and M. Naghshineh, 'The shadow cluster concept for resource allocation and call admission in ATM-based wireless networks,' in *Proc. ACM MOBICOM'95*, pp. 142-150, Nov. 1995.

Analysis of Random Access Protocol under Bursty Traffic

Jianbo Gao and Izhak Rubin

Electrical Engineering Department, University of California,
Los Angeles, CA 90095
{jbgao, rubin}@ee.ucla.edu

Abstract. Aloha-type random-access protocols have been employed as access control protocols in wireline and wireless, stationary and mobile, multiple-access communications networks. They are frequently employed by the control and signaling subsystem of demand-assigned multiple access protocols for regulating the sharing of a control channel. The latter is used for the transport of reservation packets requesting the allocation of channel resources to an active terminal. Such a random access channel is used, among others, by cellular wireless networks, by two-way CATV networks (such as those based on the DOCSIS protocol recommendation), and by demand-assigned satellite networks. The correct design and sizing of the random access operated control/signaling channel is a critical element in determining the performance of these networks. Excessive delays in the transport of signaling messages (induced by too many collisions and retransmissions) lead to unacceptable session connection set-up times. Consequently, in this paper, we investigate the performance behavior of a random-access protocol when loaded by bursty traffic processes. The latter exhibit long range dependence (LRD). The LRD traffic flows are modeled here as multiplicative multifractal processes. The random access protocol is modeled as an Aloha channel with blocking. Parameters are defined to characterize the protocol behavior. We demonstrate that the burstiness feature of the traffic processes, rather than their LRD character, is the essential element determining the performance behavior of the protocol. When the loading traffic process is not very bursty, we show that the performance of the random-access channel can be better than that exhibited under Poisson traffic loading; otherwise, performance degradation is noted. We demonstrate the impact of the selection of the protocol operational parameters in determining the effective performance behavior of the random-access protocol.

1 Introduction

In the currently operating mobile, satellite, and other communications networks, Aloha-based random multiple access schemes have been extensively used especially for scenarios where the activity factor of sources is low and low packet delay is a primary requirement. Random access schemes have also been extensively used for regulating access into control (or order-wire type) channels for

the transmission of reservation packets, as is the case for cellular wireless networks. It is essential to design such networks to yield acceptable packet delay performance to ensure that users are allocated network resources in a timely manner to satisfy their required quality of service (QoS) performance level.

The input traffic process for an Aloha-based channel is typically modeled as a Poisson process. Formulas have been developed to describe the channel's throughput and delay performance and are used for network dimensioning purposes. Recently, it has been observed that measured network traffic data often possess the long-range-dependent (LRD) property. Such traffic processes include LAN traffic [1], WAN traffic [2], variable-bit-rate (VBR) video traffic [3], and WWW traffic [4]. Furthermore it has been shown that the LRD property has a profound impact on the performance of a network. For example, Norros [5] has shown that the queue length tail distribution under the fractional Brownian motion traffic model follows an Weibull distribution instead of an exponential distribution for a queueing system under Poisson or Markovian traffic input. Erramilli et al. [6] have shown that the mean delay time for a queueing system driven by some actual LAN traffic is much worse than a queueing system driven by certain traffic processes obtained by shuffling the order of packet arrivals of the measured traffic process in such a way that the marginal distribution of the constructed traffic is the same as that of the measured traffic while the long-range-correlations are lost. Gao and Rubin [7] have shown that a Poisson model often underestimates buffer size or delay time statistics by several orders of magnitude compared to a queueing system driven by actual traffic trace data.

With LRD traffic measured in many data networks, one expects the aggregated traffic loading an Aloha-based channel to also possess such properties. Furthermore, one expects such LRD properties to have a significant impact on the performance of Aloha-based channels as well. This issue has been recently studied by Aracil and Munoz [8] and by Harpantidou and Paterakis [9]. By employing LRD real Telnet packet arrival processes as input traffic, Aracil and Munoz have found that the Aloha channel actually performs much better than that loaded by a Poisson traffic. A similar result has been obtained by Harpantidou and Paterakis, by employing a simple LRD traffic model with Pareto-distributed interarrival times as their input traffic. These results are quite the opposite of those presented by Norros [5], Erramilli et al. [6], and Gao and Rubin [7]. This situation, and the ensuing importance of the model in many applications, motivate us to carefully study the performance of Aloha channel loaded by LRD traffic.

A key issue in traffic engineering in general and LRD traffic study in particular is the characterization of the burstiness of network traffic. An LRD traffic process is characterized by the Hurst parameter $1/2 < H < 1$ [1,3]. H characterizes the persistence of correlations in a traffic process and is often interpreted as an effective indicator of the burstiness of traffic [1]. Recently we have found that although an LRD traffic process is sometimes very bursty, a burstier LRD traffic is not necessarily associated with a larger value for the Hurst parameter [10]. We show here that what really matters for the performance of an Aloha

channel is not the LRD nature, but the burstiness level, of the traffic. For this purpose, we employ a multiplicative multifractal traffic process [7,10-16] as an LRD traffic model, since multiplicative multifractal traffic processes have very well defined burstiness indicators. In this paper, we show that when the LRD traffic is not very bursty, then the performance of the random-access channel can be much better than that produced under Poisson traffic loading; otherwise, a distinct performance degradation takes place. In light of these results, we can conclude that neither the LRD Telnet arrival processes used by Aracil and Munoz [8] nor the Pareto traffic processes used by Harpantidou and Paterakis [9] are very bursty. In fact, by re-examining the latter paper, we have found that in constructing the Pareto traffic model, Harpantidou and Paterakis have deliberately selected the parameters for the Pareto traffic model in such a way that the modeled traffic becomes less and less bursty when the channel loading gets heavier.

The second purpose of this paper is to determine the desirable range for the selection of key parameter values for the random-access algorithm. This is performed to tune-up the algorithm to operate effectively under multifractal traffic, under a wide range of burstiness level conditions.

The rest of the paper is organized as follows. In Sec. 2, we describe the Aloha system adopted here and the multiplicative multifractal arrival traffic model. In Sec. 3, we first study the performance of the Aloha system under Poisson traffic, for purposes of comparing it with the same system loaded by multifractal traffic. We also examine the dependence of the performance of the system on different parameters. We then study the performance of the random-access scheme under multifractal traffic loading, assuming a wide range of burstiness levels. Finally, we draw conclusions in Sec. 4.

2 The Multi-access Scheme and the Arrival Traffic Model

2.1 Slotted Aloha System with Blocking

The system studied here is the slotted Aloha multiple-access communication channel driven by a large (theoretically infinite) number of identical users. The aggregated traffic, with mean arrival rate λ , is modeled as a multiplicative multifractal process, which will be described shortly. Each terminal handles at most a single packet at a time. All transmitted packets have the same length. A packet transmission requires a single time slot. If only one user transmits a packet in a given time slot, the packet is assumed to be correctly received by the receiver. If two or more terminals transmit packets in the same time slot, then a destructive collision occurs. For simplicity of the implementation of the system, we assume that what is detectable in a given time slot is whether the packets transmitted during the slot collide or not. The number of colliding packets is not a given observable.

The system works as follows (see the schematic of Fig. 1). A ready terminal transmits its packet at the start of a time slot. The system detects how many

time slots experienced collisions during the most recent W time slots. If, at a given time slot, this number of collisions, $N_{CS}(W)$, exceeds a threshold value, TH , then with probability P_b , each ready terminal is blocked during this time slot. The blocked terminal discards its packet. Non-blocked ready terminals are permitted to transmit their packets at this time slot. The latter packets are checked to confirm whether their transmissions resulted in a collision. If no collision is incurred, the transmission is declared successful. Otherwise, a collision has occurred. The collided packets are scheduled for retransmission at a later slot that is selected by using a uniform redistribution over a subsequent window of length L (slots).

Next we describe the traffic model for the packet arrival process.

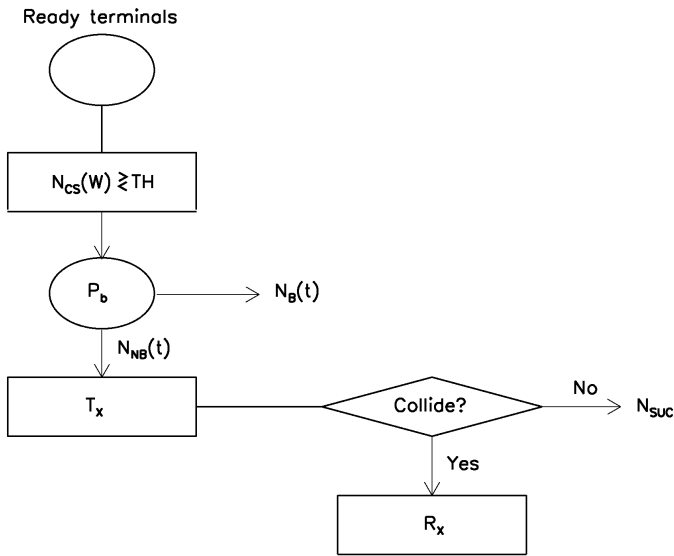


Fig. 1. Schematic of the random multi-access scheme.

2.2 The Multiplicative Multifractal Arrival Traffic Model

We consider the interarrival times for the aggregated traffic representing the overall arrivals of packets to the system. Note that for the Poisson traffic model, the interarrival times are assumed to be exponentially distributed. Here we employ a multiplicative multifractal traffic model, as developed by us [7, 10-16]. The model is described as follows.

Construction rule: Consider a unit interval. Associate it with a unit mass. Divide the unit interval into two (say, left and right) segments of equal length.

Also partition the mass into two fractions, r and $1-r$, and assign them to the left and right segments, respectively. The parameter r is in general a random variable, governed by a probability density function $P(r)$, $0 \leq r \leq 1$, which is symmetric about $r = 1/2$. The fraction r is called the multiplier, and $P(r)$ is called the multiplier function. Each new subinterval and its associated weight (or mass) are further divided into two parts following the same rule (Fig. 2). The weights calculated at stage N are employed to model the interarrival time series of the aggregated traffic.

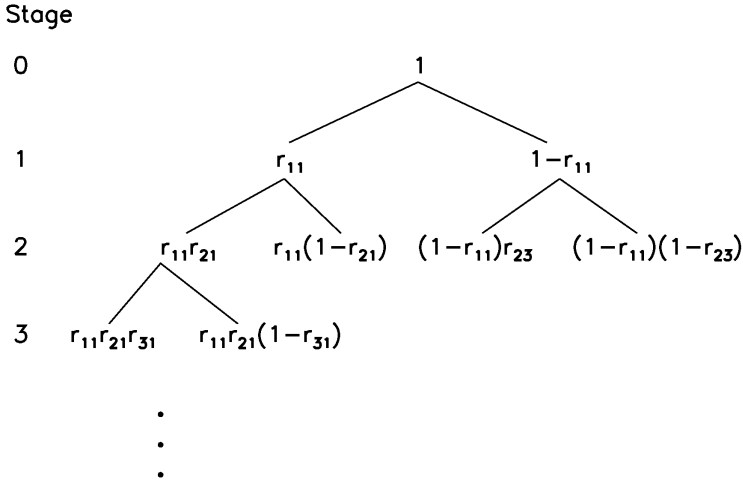


Fig. 2. Schematic of the construction rule

Among the most interesting properties of multiplicative multifractal processes are [10,14]:

- $M_q(\epsilon) = E(\sum_{n=1}^{2^N} (w_n(N))^q) \sim \epsilon^{\tau(q)}$, where $w_n(N)$ is a weight at stage N , $\epsilon = 2^{-N}$, and $\tau(q) = -\ln(2\mu_q)/\ln 2$. This property says that multiplicative processes are multifractals.
- The weights at stage $N \gg 1$ have a log-normal distribution. We note that among the four different types of distributions, normal, lognormal, exponential, and Weibull distributions, the lognormal distribution is found to be the best in describing certain features of WWW traffic such as page size, page request frequency, and user's think time [17].
- Ideal multiplicative multifractals have the LTD property, with the Hurst parameter given by $1/2 < H \leq -\frac{1}{2} \log_2 \mu_2 < 1$.

Since $P(r)$ is symmetric about $r = 1/2$, the mean value for the weights at stage N is 2^{-N} . In order to obtain an aggregated traffic with mean arrival rate λ , we can simply choose the length of the time slot to be $2^{-N}\lambda$.

For simplicity, we choose the multiplier distribution to be Gaussian:

$$P(r) \sim e^{-\alpha(r-1/2)^2}. \quad (1)$$

Parameter α determines the burstiness of the traffic: the larger the value of α , the less bursty the traffic is [11]. Typically, we generate a multifractal process by iterated computation till stage $N = 20$, so that the number of weights at our disposal is about one million.

Since the concept of the burstiness of the traffic plays a key role in this study, we first quantitatively determine how bursty a multifractal traffic process (with given α) is. For this purpose, we consider a single server FIFO queueing system with an infinite buffer. We then drive the queueing system by the multiplicative multifractal traffic and compute the queue tail size distribution. We choose the complementary queue length corresponding to the 99.99-percentile tail value as our effective measure for the burstiness of the traffic. Such a procedure can of course also be carried out for Poisson traffic. We can then normalize the 99.99 percentile of the complementary queue length for the multifractal traffic by that for Poisson traffic. Fig. 3 shows the result. We note several interesting features

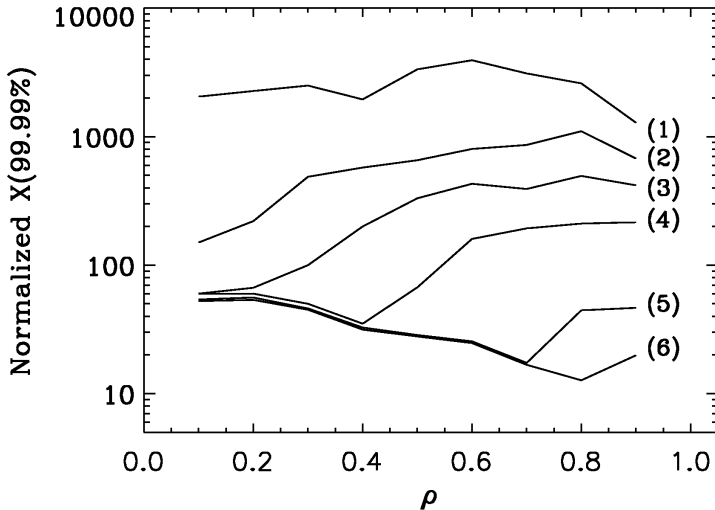


Fig. 3. The ratio of the 99.99 percentile of the complementary queue length for a FIFO queueing system with multifractal traffic and that with Poisson traffic. The six curves, denoted as (1) to (6), from top to bottom, correspond to $\alpha = 10, 30, 50, 100, 400,$ and 1000 .

revealed by the figure: (i) The multifractal traffic is more bursty when α is smaller; (ii) In terms of absolute burstiness, all the multifractal traffic processes studied here are much more bursty than the Poisson traffic; (iii) In terms of the

rate of change of the burstiness, Poisson traffic can induce a higher such rates of change when compared to not very bursty multifractal traffic processes. This phenomenon is elucidated by the downward-trending portion of curves (4)-(6) in Fig. 3.

3 Performance of the Aloha System

In this section, we study the performance of the Aloha channel under different parameter values. We employ five parameters: the burstiness parameter α ; the window size W ; the threshold value TH for determining if blocking takes place; the blocking probability P_b ; and the window size L for redistributing the transmissions of the collided packets. For simplicity, we fix $L = 10$ for the illustrative cases presented in the following. To better understand the performance of the Aloha system under multifractal traffic, we first exhibit the performance of the system under Poisson traffic. We shall only compare the best performance for the channel under Poisson traffic and that under multifractal traffic. Hence, the parameter set for the system under Poisson traffic may be different from that for the system under multifractal traffic.

3.1 Performance of the System under Poisson Traffic

Under the assumption that the arrival traffic is Poisson and the total aggregated traffic, which represents the superposition of new and retransmitted message process, is also Poisson, we state the well-known throughput formula:

$$s = Ge^{-G} , \quad (2)$$

where s is the normalized throughput and G denotes the total channel loading rate.

Assuming a window size of $W = 20$, our simulations indicate that a good selection for the threshold level is $TH = 10$. Fig. 4 shows the throughput vs. channel loading (G) performance curves under three selected values for the blocking probabilities, $P_b = 0.1, 0.25, \text{ and } 0.5$. The thin solid curve is generated by using Eq. (2). We observe that all three dash-dot curves lie below the thin solid curve, indicating that the throughput of the system under Poisson arrival traffic is actually slightly lower than that predicted by Eq. (2), under the selected set of parameters (noting also that L is limited to a range of 10 slots). We note that when P_b is small, such as 0.1, the throughput of the system becomes highly degraded as the loading rate increases. When P_b is increased to a level such as to 0.25, the realized throughput level is distinctly improved. However, when P_b is further increased to a blocking level of 0.50, the total channel loading ends at 0.7 rather than continuing to 2.0, due to heavy blocking. Since in this case, the maximum throughput level achieved is only slightly lower than that attained for $P_b = 0.25$, hence, this parameter value is also considered usable. As P_b is further increased, the maximum throughput level gradually continues to decrease.

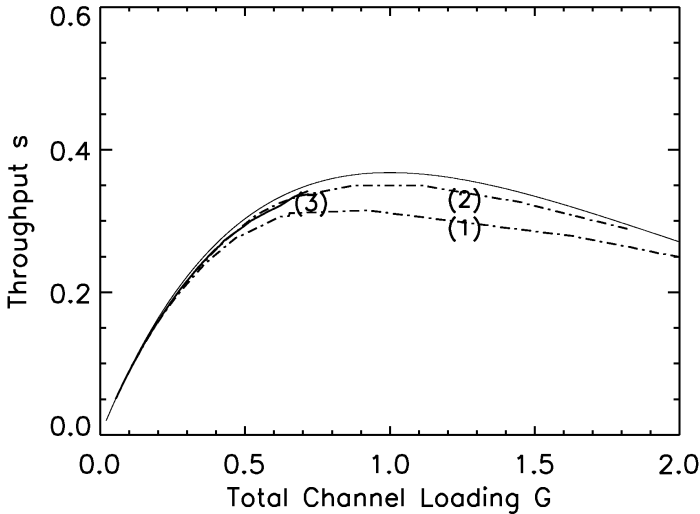


Fig. 4. The throughput s vs. the total channel loading G curves under Poisson traffic. The top most solid thin line is generated from Eq. (2). The other three curves, denoted as (1), (2) (both dash-dot curves), and (3) (solid thick curve), correspond to $P_b = 0.1$, 0.25, and 0.5, respectively.

Packet delay time statistics also serve as important performance measures for an Aloha system. To assess the packet's delay performance, we examine the number of retransmissions incurred by a packet until it is successfully transmitted. In Fig. 5(a-c), we show the behavior of the 90-percentile of the number of retransmissions, $N_{Ret}(90\%)$, vs. the total channel loading G , under Poisson traffic loading, for parameter values corresponding to the three curves exhibited in Fig. 4. It is interesting to note that the delay performance under $P_b = 0.1$ is worse than that observed for $P_b = 0.25$ in terms of both the realized throughput level (Fig. 4(a,b)) and the attained delay performance, as expressed by $N_{Ret}(90\%)$. For $P_b = 0.50$, however, an improved delay performance is observed, as expected, due to the high blocking level. Under the assumed conditions, we note that designs (2) and (3) of Figs. 4-5 yield good performance behavior.

Note from Fig. 5(d), in comparison with Fig. 5(b), that as the threshold level is increased (from $TH = 10$ to $TH = 12$), message delay performance somewhat degrades. Similar throughput performance has been noted by us for the two cases (not shown for (d)).

3.2 Performance of the System under Multifractal Traffic

First we fix $\alpha = 100$ and study the performance of the system under different parameter values for TH , W , and P_b . We find by numerical simulations that a good selection of (TH, W) is $(7, 20)$. It turns out that for not too high channel

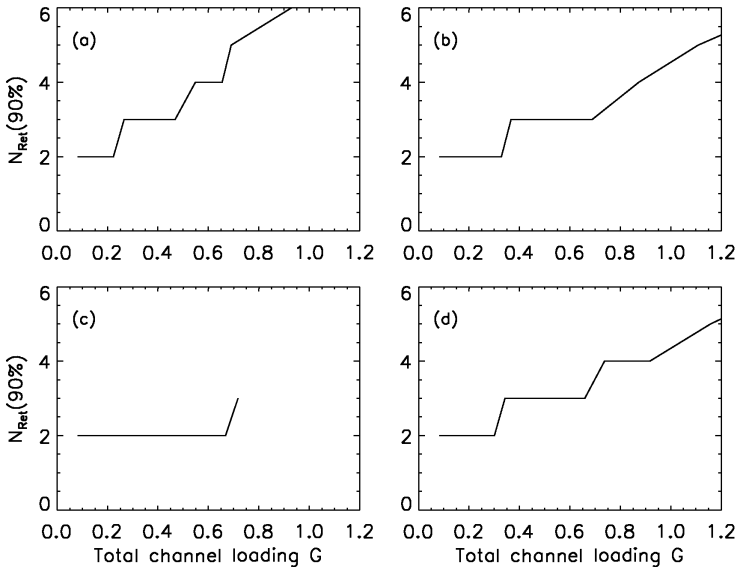


Fig. 5. The 90 percentile of the number of retransmissions vs. the total channel loading G curves for Poisson traffic: (a) $P_b = 0.1$, and $(TH, W) = (10, 20)$; (b) $P_b = 0.25$, and $(TH, W) = (10, 20)$; (c) $P_b = 0.5$, and $(TH, W) = (10, 20)$; and (d) $P_b = 0.25$, and $(TH, W) = (12, 20)$.

loading conditions, this parameter combination also works very well for the other studied α parameter values. Since an Aloha channel is designed to work for not too high loading conditions, we thus fix (TH, W) to be $(7, 20)$ in the rest of the paper.

Next, we study the dependence of the performance of the system on the blocking P_b parameter. Fig. 6 shows the throughput s vs. the total channel loading G for four different P_b values. For comparison, the curve generated from Eq. (2) is also plotted as a dashed curve. We observe that for low loading conditions, the throughput level is larger when the system is loaded by multifractal traffic (than that loaded by Poisson traffic), irrespective of the blocking probability P_b value. Under high loading conditions, the performance of the system loaded by the LRD traffic process remains much better than that experienced under Poisson traffic loading, if suitable P_b values (0.4 and 0.6, corresponding to curves (2) and (3) in Fig. 6) are used.

It is interesting to note that a good selection of a P_b level under multifractal traffic loading with $\alpha = 100$ is higher than that used under Poisson traffic loading. When the multifractal traffic process becomes more bursty (corresponding to smaller α values), we find that the optimal P_b level has to be increased gradually.

Fig. 7 shows $N_{Ret}(90\%)$ vs. total channel loading curves for the four P_b values studied in Fig. 6. We observe that typically when the blocking level P_b is

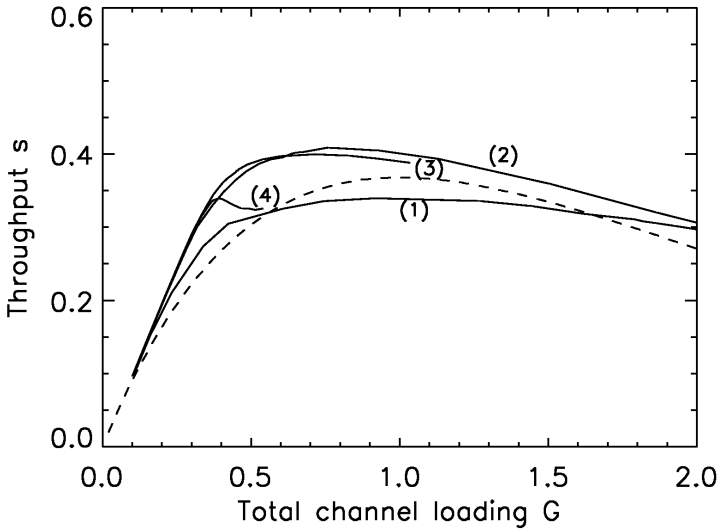


Fig. 6. The throughput s vs. total channel loading G curves for the Aloha system under multifractal traffic with $\alpha = 100$: (1) $P_b = 0.2$, (2) $P_b = 0.4$, (3) $P_b = 0.6$, and (4) $P_b = 0.8$.

increased, the delay performance is improved. In a satellite network, for example, $N_{Ret}(90\%)$ is many times required to not exceed 2 (or a similarly small value, due to the high retransmission delay involved) for not too high loading conditions. Under such a criterion, a value of $P_b = 0.6$ is most preferred. Re-examining Fig. 6, we find that this P_b value also produces good throughput behavior.

Now that we have demonstrated the performance of the system under different values for the TH , W and P_b parameters, we study the performance of the system under multifractal traffic loading processes characterized by different burstiness levels, as expressed by their different corresponding α parameters. Fig. 8 shows system throughput s vs. total channel loading G performance curves for five different α levels. It is clearly noted that the attained system throughput behavior critically depends on the selected value for the loading process burstiness parameter α value. When the multifractal traffic is not very bursty ($\alpha \geq 100$), the throughput performance of the system under multifractal traffic loading is noted to be much better than that attained under Poisson traffic loading. However, when the loading LRD traffic process becomes highly bursty, as is the case when we set $\alpha = 30$ or 10 , the throughput performance exhibited under multifractal traffic is much worse than that attained under Poisson traffic loading.

The delay performance for the system when loaded by multifractal traffic processes, corresponding to the conditions used for Fig. 8 (except for the case corresponding to $\alpha = 100$, which is shown in Fig. 7(c)) is shown in Fig. 9. We

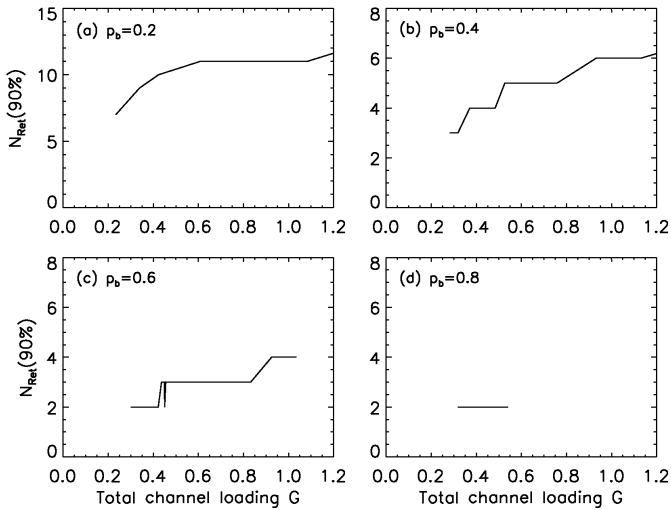


Fig. 7. The 90 percentile of the number of retransmissions vs. total channel loading G curves for the system under multifractal traffic with $\alpha = 100$: (a) $P_b = 0.2$, (b) $P_b = 0.4$, (c) $P_b = 0.6$, and (d) $P_b = 0.8$.

observe that for not too high loading conditions, $N_{Ret}(90\%)$ is less than or equal to 2 (the delay statistics for $\alpha = 400$ for low loading conditions is not sufficient for the computation of $N_{Ret}(90\%)$). In other words, such a delay requirement for $\alpha = 400$ is automatically ensured).

Finally, we examine the behavior of the system's attained total blocking probability levels when the system is loaded by a multifractal traffic process, under a wide range of burstiness levels. Fig. 10 shows the corresponding blocking performance results. We observe that the blocking probability is larger when the loading process is burstier, with the blocking level attained for $\alpha = 400$ being similar to that exhibited under Poisson traffic loading. Comparing curves (5) shown in Fig. 8 and Fig. 10 with corresponding curves presented by Aracil and Munoz [8], we conclude that the traffic used in [8] is much less bursty than a multifractal traffic process with $\alpha = 400$.

4 Conclusions

In this paper, we have studied the performance of a random-access scheme when loaded by an LRD traffic loading process. The LRD traffic process is modeled as a multiplicative multifractal process. The random access scheme is the prototypical Aloha channel with blocking. Five key parameters are introduced to characterize the scheme. We show that the key performance features of the random-access protocols are determined by the burstiness level of the loading traffic process and not solely by its LRD character. We demonstrate that when the LRD loading

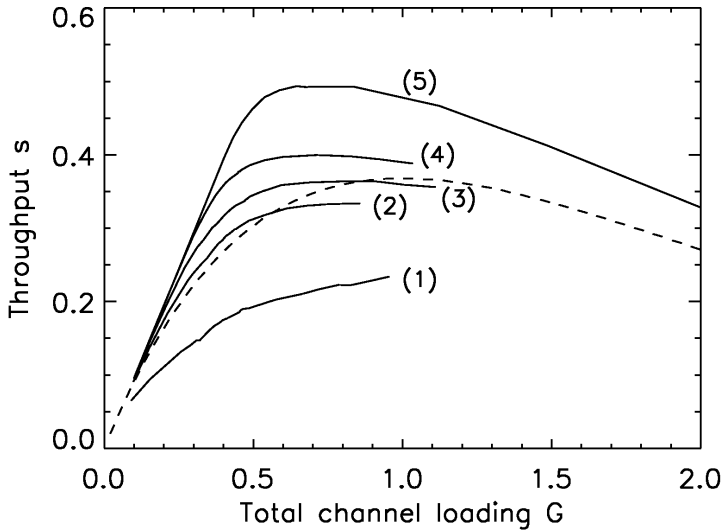


Fig. 8. Throughput s vs. total channel loading G curves for multifractal traffic. The 5 curves, from bottom to top, denoted as (1) to (5), correspond to $\alpha = 10, 30, 50, 100,$ and 400 . The dashed curve is generated from Eq. (2).

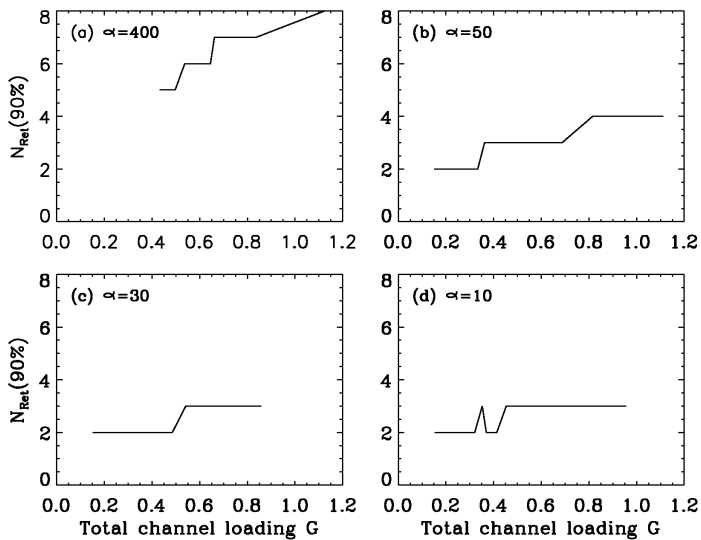


Fig. 9. The 90 percentile of the number of retransmissions vs. total channel loading G curves under multifractal traffic with different α values.

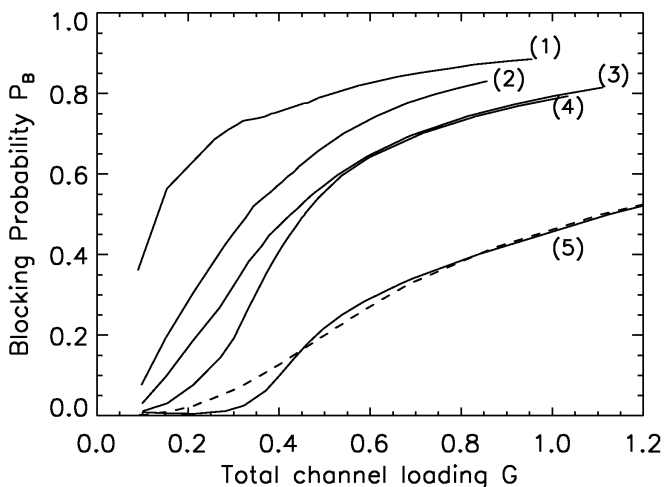


Fig. 10. Blocking probability P_B vs. total channel loading G curves. The 5 solid curves, from top to bottom, denoted as (1) to (5), correspond to multifractal traffic processes with $\alpha = 10, 30, 50, 100,$ and 400 . The dashed curve is for Poisson traffic loading.

process is not very bursty, the multiple-access protocol can yield better performance than that exhibited under Poisson loading conditions. In turn, significant performance degradations are observed when the system is loaded by LRD processes characterized by higher levels of burstiness. We further show that the parameters of the protocol must be tuned-up carefully to yield efficient performance behavior.

References

- [1] W.E. Leland, M.S. Taqqu, W. Willinger, and D.V. Wilson, On the self-similar nature of Ethernet traffic (extended version). *IEEE/ACM Trans. on Networking*, **2**, 1–15 (1994).
- [2] V. Paxson and S. Floyd, Wide Area Traffic—The failure of Poisson modeling. *IEEE/ACM Trans. on Networking*, **3** 226–244 (1995).
- [3] J. Beran, R. Sherman, M.S. Taqqu, and W. Willinger, Long-range-dependence in variable-bit-rate video traffic. *IEEE Trans. on Commun.*, **43** 1566–1579 (1995).
- [4] M.E. Crovella and A. Bestavros, Self-similarity in World Wide Web Traffic: Evidence and Possible Causes. *IEEE/ACM Trans. on Networking*, **5**, 835–846 (1997).
- [5] I. Norros, A storage model with self-similar input. *Queueing Systems*, **16**, 387–396 (1994).
- [6] A. Erramilli, O. Narayan, and W. Willinger, Experimental queueing analysis with long-range dependent packet traffic. *IEEE/ACM Trans. on Networking*, **4**, 209–223 (1996).

- [7] J.B. Gao and I. Rubin, Multiplicative Multifractal Modeling of Long-Range-Dependent (LRD) Traffic in Computer Communications Networks. *World Congress of Nonlinear Analysts*, July, 2000, Catania, Sicily, Italy. Also *J. Nonlinear Analysis*, in press.
- [8] J. Aracil and L. Munoz, Performance of Aloha channels under self-similar input. *Electronics Lett.*, **33**, 659-660 (1997).
- [9] Z. Harpantidou and M. Paterakis, Random multiple access of broadcast channels with pareto distributed packet interarrival times. *IEEE Personal Commun.*, **5**, 48-55 (1998).
- [10] J.B. Gao and I. Rubin, Multifractal modeling of counting processes of Long-Range-Dependent network Traffic. *Computer Communications*, in press (2001).
- [11] J.B. Gao and I. Rubin, Multiplicative Multifractal Modeling of Long-Range-Dependent Traffic. *Proceedings ICC'99*, Vancouver, Canada.
- [12] J.B. Gao and I. Rubin, Multifractal modeling of counting processes of Long-Range Dependent network traffic. *Proceedings SCS Advanced Simulation Technologies Conference*, San Diego, CA, 1999.
- [13] J.B. Gao and I. Rubin, Superposition of Multiplicative Multifractal Traffic Streams. *ICC2000*, June, 2000, New Orleans, Louisiana.
- [14] J.B. Gao and I. Rubin, Statistical Properties of Multiplicative Multifractal Processes in Modeling Telecommunications Traffic Streams. *Electronics Lett.* **36**, 101-102 (2000).
- [15] J.B. Gao and I. Rubin, Multifractal analysis and modeling of VBR video traffic. *Electronics Lett.* **36**, 278-279 (2000).
- [16] J.B. Gao and I. Rubin, Superposition of multiplicative multifractal traffic processes, *Electronics Lett.*, **36**: 761-762 (2000).
- [17] M. Molina, P. Castelli, and G. Foddis, Web traffic modeling exploiting TCP connections' temporal clustering through HTML-REDUCE. *IEEE Network*, vol.14, (no.3):46-55 (2000).

A Practical Model for VBR Video Traffic with Applications

Tamás Borsos

Traffic Analysis and Network Performance Laboratory,
Ericsson Hungary, Laborc u. 1, 1037 Budapest
Tamas.Borsos@eth.ericsson.se

&

Budapest University of Technology and Economics,
Dept. of Computer Science and Information Theory,
Budapest, Hungary

Abstract. Video traffic compressed with variable bit rate coding scheme is known to possess high variations and multiple time scale characteristics. This property makes parsimonious video modeling a difficult task. A possible way of describing this traffic is via self-similar models, which also produce high variations on many time scales. However, these are general traffic models and do not represent many important characteristics of video. In this paper we show that video traffic has well-separable time scales. Based on this result, a new model is presented, which is capable of capturing the main properties of VBR video. The concept is scene-oriented, while a larger time scale - called epoch - is introduced. The main contribution of this paper is that the presence of multiple time scales seem to be the real reason for the slowly decaying autocorrelation function rather than heavy tailed level durations. Finally, the application of the model is shortly discussed for dimensioning, admission control and simulation purposes.

1 Introduction

The emerging multi-service architectures speed up the widespread adoption of video applications in communication networks. Streaming video – in which viewers can begin watching content almost as soon as it starts downloading – is now developing as a mainstream technology on the Internet. Wireless networks like GPRS and UMTS will offer mobile users a convenient way to access these services. However, the efficient support of high quality video services over packet switched wireline or wireless channels is still a challenging task.

Modern effective coding standards use hybrid coding (e.g., MPEG, H.263, H.263+) which comprises lossy intraframe coding and motion compensation to exploit both spatial and temporal redundancy. In this paper MPEG-4 traces are analyzed, since - due to the similarity of the coders - the results can be generalized straightforward to MPEG-2 or H.263 traffic and they also serve as a basis for understanding the bandwidth requirement of compressed video streams.

There are different coding schemes that can be used depending on the actual application (e.g., constant bit rate, variable bit rate, adaptive rate coding). This paper is concerned with the variable bit rate (VBR) scheme, which provides the best compression for given image quality. Although, from the point of view of performance analysis, it is not easily tractable due to its high variability on many time scales.

There are a number of different approaches that can be found in the literature. In the discrete autoregressive DAR(1) model [10] a finite-state Markov chain is used to generate the sequence of the number of bits associated with each video frame. This process stays at a constant level for geometrically distributed period, then steps onto another level independently of the current state. The transition matrix of this Markov chain is composed of identical rows equal to the marginal frame size distribution. The use Markov Modulated Process [17] is similar to the previous approach, but it is extended by state-dependent transitions and varying bit rate during the scenes. Clearly, this model requires quantized average scene bit rates. In addition, the number of quantization levels can not be large because all of the elements of the transition matrix must be estimated from the trace. Another type of models [6], [11] capture the marginals and the autocorrelation function (ACF) of a video trace. In this case, video traffic is modeled as a self-similar process. The frame (or group of pictures) size distribution can be arbitrary (in [6] it is a mixture of Gamma and Pareto distributions), while the correlation structure is caught through the Hurst parameter. Thus, beside the marginal distribution, only the Hurst parameter needs to be estimated, which determines the decay rate of the autocorrelation function. The shifting level process (SLP) (also called space-time renewal process) is a very simple model [5], [16], [7], [8]; it consists of constant bit rate levels, which may represent the scenes. Both the mean and the length of these levels are independent, identically distributed random variables. By choosing the length distribution appropriately, any ACF structure can be captured [12]. Thus, the process is completely described by two distributions. In [14] the compound ACF behaviour is captured by a special family of functions, the stretched exponential, instead of polynomial (long range dependence) or exponential (Markovian). The video model with such a correlation structure is based on $M/G/\infty$ input process, where the actual bit rate is proportional to the number of customers being in the $M/G/\infty$ system. By varying G many forms of time dependence can be displayed, while it affects the bit rate marginal distribution only through its mean. The marginal will be Poisson distribution, which is further transformed into the appropriate frame size distribution.

The temporal behaviour of a VBR coded video has always been the most debated question in the literature. Its bandwidth requirement is often regarded as a long range dependent (LRD) process because of its slowly decaying autocorrelation function. Thus, traditional short range dependent Markovian models may fail to predict QoS parameters in certain circumstances [6]. In [18] the use of LRD models is questioned in the context of realistic traffic engineering, i.e., using short network buffers.

A simple Markovian model is indeed able to capture only one time scale. The scene level characteristics of video is a well known property, which also has an obvious physical meaning (i.e., scene cuts in a movie). However, the Markov chain driven scene levels are still not descriptive enough for the whole video trace. Therefore, self-similar models are used, which are able to capture infinite number of time-scales. In that model the marginals and autocorrelation functions are properly matched to the video trace. This approach is not necessarily effective due to the following reasons:

- there is no physical meaning of the parameters and the model provides no support about how to change them for different video traces and coding standards;
- without the actual trace there is no way of estimating the parameters;
- very different video traces may have similar models;
- when estimating the ACF the traces were assumed to be stationary which may not be true [15];
- second order statistics may be irrelevant from the queueing point of view [12,2].

In the light of these criteria, we propose a model in which the time scales of VBR video are explicitly represented, having a physical meaning for each of them. The model is scene-oriented, but a larger time scale – called *epoch* – is introduced. This result suggests that the large tail of the ACF is not necessarily due to the heavy tailed level durations (e.g. scenes), but seems to be the sign of *large time scale level shifts*.

The paper is organized as follows. In Section 2 the structure of the examined MPEG video traffic is described. Section 3 contains the scene level analysis, while in Section 4 the epoch concept is introduced and validated. In Section 5 we look at three application areas of this model, i.e., dimensioning, admission control and simulations. Section 6 concludes the paper.

2 MPEG Video Structure

The modeling approach in this paper is intended to describe the common properties of all of the widely used *hybrid coding* standards, rather than a specific coding standard. Hybrid coding, like MPEG or H.263, comprises *lossy intraframe coding* and *motion compensation* to exploit both spatial and temporal redundancy. The subject of the actual investigation is a set of VBR MPEG-4 traces taken from [4]. Similar results were obtained for the traces of Rose [17].

A standard MPEG encoder generates three types of compressed frames. I-frames are compressed using intra frame information only. P-frames are coded similarly to I-frames but with the addition of motion compensation with respect to the previous I- or P-frame. B-frames are similar to P-frames except that the prediction is bidirectional. Typically I-frames are the largest followed by P-frames, while B-frames require the lowest bandwidth. After coding, the frames are arranged in a deterministic order, which is called group of pictures (GOP,

e.g., 'IBBPBBPBBPBB'). The GOP pattern is not specified by the standard and coders may use different patterns for subsequent GOPs. However, since many sequences are being coded with regular GOP patterns (often to simplify the codec design), developing a traffic model for such sequences has its merits.

There are two main levels of modeling: *frame level* and *GOP level*. The former attempts to catch the size of each frame, while the latter takes a GOP as a whole and is not interested in the individual frame sizes. In order to eliminate the deterministic alternation of frame types, this paper is concerned with the GOP level, thus making the model independent of the MPEG coding scheme. The model can then be further refined to frame level.

3 Scene Level Analysis

It has been observed by many authors (e.g., [10], [12]) that the fluctuation of frame sizes exhibit different behaviour at different time scales. Within small locality, I-frames have relatively small fluctuation about some average level, which itself varies at larger time scale. These *level shifts* are usually attributed to *scene changes* [5]. The scene can be defined as a portion of video without sudden changes in view. Such behaviour can be captured using a mean-nonstationary time series or some kind of modulated process. These models are called *scene-based models*. Incorporating the 'scenic' component in the traffic model gives its predictions a physical meaning.

A simple implementation of such a model is the Shifting Level Process (SLP), which consists of constant bit rate levels. The bit rate Y_i and the duration T_i of these levels are independent, identically distributed random variables. The GOP size is clearly distributed according to Y . It has also been shown [12] that the ACF ρ_k is exactly equal to the integrated tail of the renewal distribution function F_T . That is

$$\rho_k = 1 - \frac{1}{m} \sum_{u=0}^k (1 - F_T(u)),$$

where m is the mean scene length. One can see that, despite the simple structure of the SLP, it provides an arbitrarily exact match of the first- and second-order statistics. Thus, it is able to match subexponential ACF and to capture long range dependence. The model can be extended to involve bit rate variations during scenes. This modification has little impact on the tail characteristics of the ACF. According to this reasoning the slow decay of the ACF is due to the heavy tailed scene length distribution. However, our investigations show that the scene durations follow Weibullian distribution with a mean of 2-3 seconds.

In order to obtain scene statistics one has to find scene boundaries in a trace. To make the estimation of the distribution robust, two different methods were used for this purpose. The first one was performed with change point detection using the trace data. In this case, the well known intra-coded Star Wars movie was used, since this coding scheme provides more reliable basis for scene change detection. In the second case, the scene identification was performed on the basis

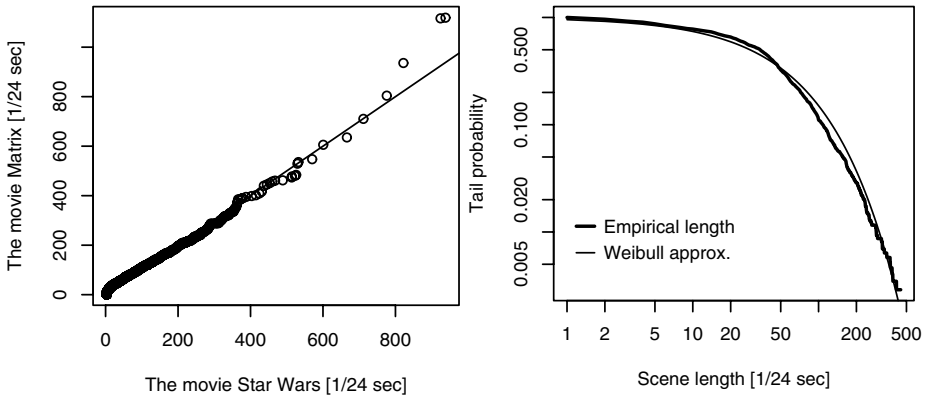


Fig. 1. Q-Q plot for the two different types of scene length estimates and the scene length distribution with a Weibullian approximation on a log-log scale

of a real MPEG source (the movie Matrix) instead of its trace. The frames were decoded and the difference of the luminance bitmaps of every two consecutive frames was stored. This is a visual identification and is supposed to be more reliable than the frame size based method. Figure 1 shows that the character of the two empirical distributions are surprisingly similar, despite that two different movies were processed by two different methods. This validates the obtained result. The tail of the scene length distribution can also be seen in Figure 1 with its Weibullian approximation.

The fact that scene bit rates are not independent can be easily checked: by shuffling the scenes in a video trace the time dependence structure is destroyed. Grasse *et al.* [7] suggest that the Y_i and T_i processes should be modeled by fractionally differenced white noise passed through an ARMA(1,1) filter. This solution, however, results in a general traffic model beyond the scene level, with little physical meaning of the filter parameters.

4 Epoch Time Scale

After extensive statistical and graphical tests [1] concludes that video traffic exhibits long range dependence, which is very important from the point of view of queueing behaviour. When calculating the empirical autocorrelation function of several traces, one may see that the tail behaviour is really far from exponential.

Figure 2 shows the empirical ACF of the Star Wars trace. It can be seen [6] that the initial part of the curve can be accurately matched to an exponentially decaying function, but only up to 5-10 sec lag. Beyond that it decreases slower than exponentially, up to c.a. 40 sec lag. It then adopts a quite erratic behaviour decaying toward zero but extremely slowly. This composite shape of the ACF (i.e., exponential start and slow decay later) is a property of many other video

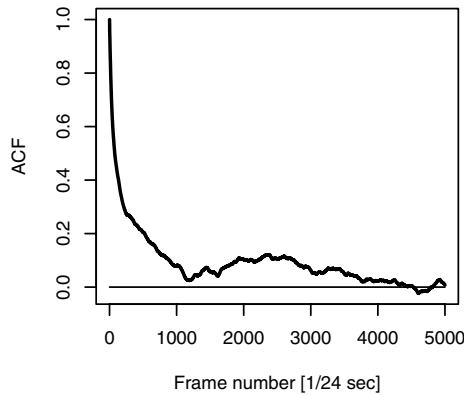


Fig. 2. Empirical ACF of the Star Wars trace (intra frame coded)

traces, as well, suggesting the property of long range dependence. This long time dependency is often explained by the heavy tail of scene lengths (e.g., Pareto distribution). Although our statistical analysis reinforces the subexponentiality of scene length duration, it does not seem the main cause of the large tail of the autocorrelation function, since the tail of the ACF is not simply slowly decaying but remains high (~ 0.4) for large time lags, beyond the scene time scale.

In our concept the complex behaviour of the ACF is *due to the presence of multiple time scales instead of heavy tailed level durations*. Analyzing the VBR video traces one can observe that the level shifts, which are the characteristics of the scenic behaviour, appear on a larger time scale, as well (Figure 3). These long term level shifts constitute the modulating process of the scene level process. These larger time scale levels – we called them *epochs* – are introduced, which

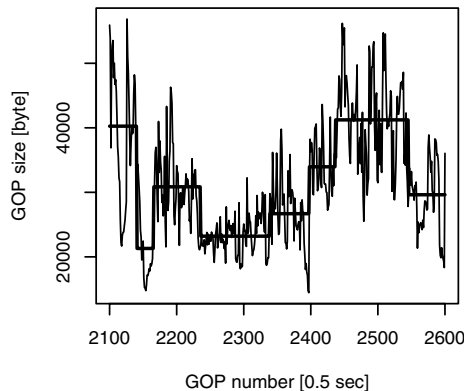


Fig. 3. Level shifts on large time scale in the GOP trace of the movie MrBean

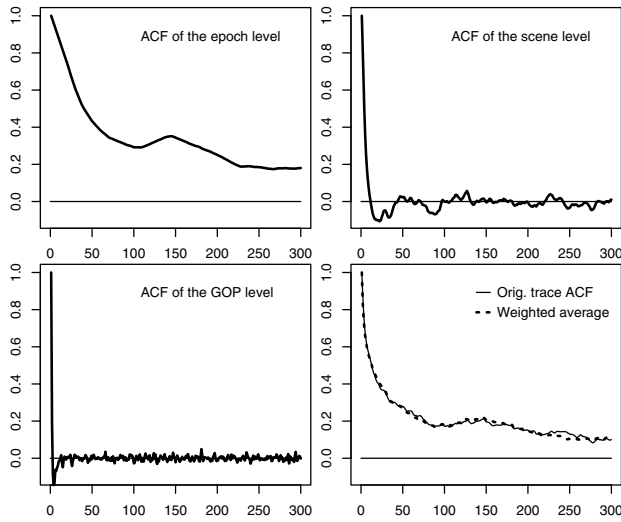


Fig. 4. The ACF of the epoch, scene, GOP levels and the ACF of the trace (GOP numbers, up to 150 sec)

serve as an explanation for the large tail of the ACF. Therefore, our model has 3 level characteristics (the index i refers to the GOP number):

- GOP size variation during a scene N_i ;
- level shifts due to scene changes S_i ;
- level shifts due to epochs (a group of scenes) E_i .

S_i and N_i are zero-mean processes. The GOP trace T_i is then simply the sum of the three level processes: $T_i = E_i + S_i + N_i$. Note that this concept contains no specific information about the size or duration distributions, or about the correlation structure of the individual level processes. Moreover, *long range dependency is not a requirement to produce the desired ACF function*.

The hypothesis that N_i , S_i and E_i are independent is not completely true, but it proved to be a reasonable modeling assumption. The weighted average of their ACFs gives the ACF of the whole trace, as can be seen in Figure 4. This structure explains the quickly decaying start (due to S_i and N_i) and the slowly decaying tail (due to E_i), which has also been observed by several authors [6]. The stochastic behaviour beyond some time lag may be the result of non-stationarity, i.e., the epoch process varies too slowly to be evaluated from a "short" 1 hour trace.

The concept of defining epochs in a movie has some practical rationale, since consecutive scenes may be filmed in a similar environment and background. Thus, this modeling approach is based on the real traffic behaviour, and it is probably easier to generalize to other coding schemes. On the other hand, level shifts on the large time scale mean sudden jumps in bandwidth requirement

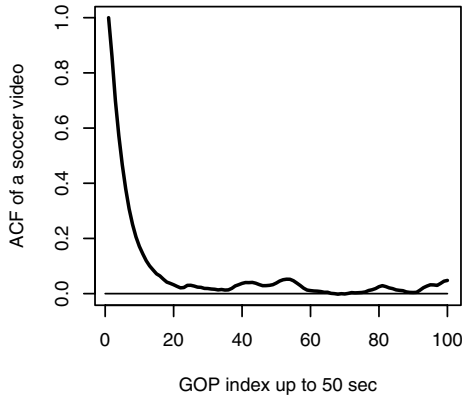


Fig. 5. The ACF of a "stationary" soccer video

instead of slow changes. This can be a significant property in adaptive environments, like, e.g., TCP background traffic, WCDMA or measurement based admission control. Finally, the three-level model is supported by the fact that movie traces always possess large ACF tail (up to even 400 sec lag), while those TV broadcasts that contain "stationary" environment, like tennis or football, have significant tail only up to ~ 20 sec lag, as can be seen in Figure 5. This phenomenon indicates that the epoch time scale in this video type is missing. This means that the distribution of the epoch mean is already meaningful and can be used as a video traffic descriptor. (One can also see that the negative correlation values of Figure 4 have also disappeared in Figure 5, since these were the result of making empirical averages for the level processes.)

The next step is characterizing the three level processes. As mentioned above, the marginal distribution of the epoch level E_i is the main descriptor of a VBR video. On the other hand, the temporal behaviour of E_i is of no importance at all in most of the applications. In particular, it is not interesting from the queueing point of view, since this time scale is too large to be smoothed in a buffer of reasonable size [9].

The scene level process S_i can already be smoothed in case of streaming video applications (e.g., video on demand or Internet streaming applications), where a delay of 10-20 sec is allowed. We have not developed a parsimonious model for this process yet, but a short example shows that despite the Weibullian scene length distribution, a Markovian model may be an appropriate description. The process S_i is quantized to 50 levels and a 50 state Markov chain is fitted to S_i (note that it is not for modeling purpose, only for demonstration). The queue tail distribution of the original trace S_i and a randomly generated trace S_i^* is illustrated in Figure 6 for different service rates on logarithmic scale.

The GOP level process N_i is similar to S_i but appears on a smaller time scale. It should be accounted for in case of low-delay video applications.

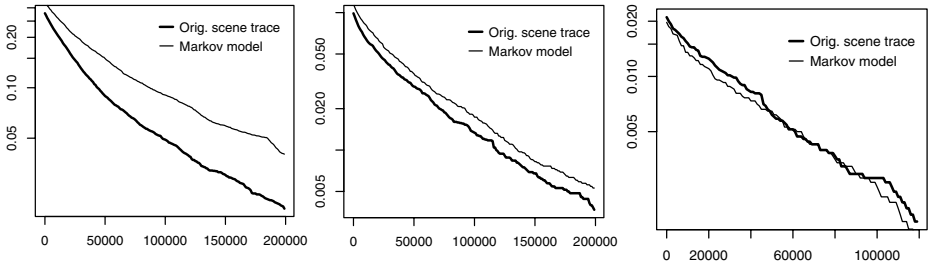


Fig. 6. The queue tail distribution of S_i and a Markovian equivalent S_i^* for different service rates (increasing from left to right) on logarithmic scale. S_i^* is usually an upper bound.

5 Applications

Dimensioning for streaming video. In this application the GOP time scale process N_i is not necessary to appear in the model, since its average is approximately zero on 4-5 sec interval, while the smoothing buffer is typically larger. The epoch process E_i affects the bandwidth requirement only through its marginal distribution. After some analysis of the 20 available video traces the Gamma distribution seems to be a good conservative approximation (see Figure 7). For the estimation of the traffic description parameters a large video database should be used. The required capacity can be calculated separately for the epoch level processes (with bufferless multiplexing) and the scene level processes (effective bandwidth with a given buffer [13]).

Admission control for streaming video. Based on the above reasoning a robust admission control method can be constructed [3]. While a simple measurement based admission control (MBAC) is not a reliable method for VBR video traffic due to its long term variations, a compound, measurement based and parametric admission control is already suitable. While the bursts coming from the scene time scale (and GOP) bit rate variations can be buffered and thus, its temporal behaviour must be accounted for by the measurement based scheme, the large time scale variation affects the bandwidth requirement only through its marginal distribution. The two time scales can be separated in real-time by applying moving average. Thus, the algorithm is measurement based on small time scales and parametric on the large time scale.

Video traffic simulations. With the proposed model it is possible to generate video traces for simulations that have similar characteristics to the real video traffic. The main impact of the model on packet level simulations is that the large time scale variation actually appears as level shifts. This causes a sudden jump in the bandwidth requirement occasionally. These level jumps may have serious effects on the performance of adaptive background applications, like TCP or adaptive video. In these applications, besides the bandwidth oscillation, it also causes bursty packet losses.

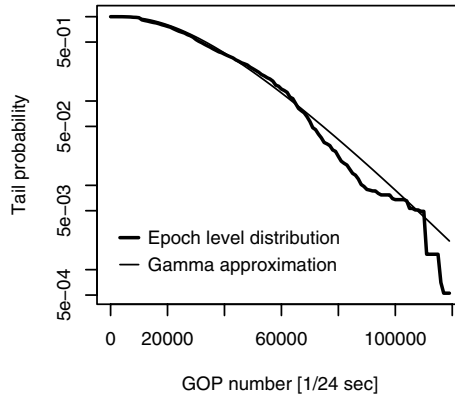


Fig. 7. The marginal distribution of epochs with its Gamma approximation

6 Conclusion

VBR video traffic is known for its high correlation on large time scales. In this paper we argued for multiple time scale video models instead of long range dependent models. We described a modeling approach which explicitly captures the time scales of video traffic. The model is scene-oriented, but large time scale level shifts are introduced, called epochs. This time scale process has a physical meaning and can be used for parsimonious characterization of video traffic. The model also explains the compound structure of a typical video traffic auto-correlation function without the use of long range dependent models. Although the model has a composite structure, certain time scales are irrelevant in the performance analysis, thus, it remains parsimonious for a given application. In particular, the time scales beyond the epoch levels and their temporal behaviour is of practically no importance. Examples were also shown that the model can be used in a wide range of applications.

Acknowledgment. The author wishes to thank Prof. László Györfi and András György for useful comments. He would also like to thank the Telecommunication Networks Group at the Technical University of Berlin for making their video traces publicly available.

References

1. J. Beran, R. Sherman, M. S. Taquq, and W. Willinger. Long-range dependence in variable-bit-rate video traffic. *IEEE Trans. Communications*, 43:1566–1579, 1995.
2. T. Borsos, L. Györfi, and A. György. On the second order characterization in traffic modeling. Proc. 9th IFIP Working Conference on Performance Modelling and Evaluation of ATM and IP Networks, pages 41–49, July 2001.
3. T. Borsos and A. György. Robust admission control for video traffic. Preprint.

4. F. H. P. Fitzek and M. Reisslein. MPEG-4 and H.263 video traces for network performance evaluation. Technical Report TKN-00-06, Technical University of Berlin, Telecommunication Networks Group, 2000.
5. M. R. Frater, J. F. Arnold, and P. Tan. A new statistical model for traffic generated by VBR coders for television on the broadband ISDN. *IEEE Trans. Circuits and Systems for Video Technology*, 4:521–526, 1994.
6. M. W. Garrett and W. Willinger. Analysis, modeling and generation of self-similar VBR video traffic. In *Proc. ACM SIGCOMM '94*, pages 269–280, Aug. 1994.
7. M. Grasse, M. R. Frater, and J. F. Arnold. Implications of non-stationarity of MPEG2 video traffic. *COST257TD(97)10*, 1997.
8. M. Grasse, M. R. Frater, and J. F. Arnold. Origins of long-range dependence in variable bit rate video traffic. In *Proc. ITC 15*, pages 1379–1388, June 1997.
9. M. Grossglauser and J. C. Bolot. On the relevance of long-range dependence in network traffic. *Computer Communications Review*, 26:15–24, 1996.
10. D. P. Heyman and T. V. Lakshman. Source models for VBR broadcast-video traffic. *IEEE/ACM Trans Networking*, 4:40–48, 1994.
11. C. Huang, M. Devetsikiotis, and I. Lambadaris A. R. Kaye. Modeling and simulation of self-similar variable bit rate compressed video: a unified approach. In *Proc. ACM SIGCOMM '95*, pages 114–125, Aug. 1995.
12. P. Jelenkovic, A. A. Lazar, and N. Semret. The effect of multiple time scales and subexponentiality in MPEG video streams on queueing behavior. *IEEE Journal on Selected Areas in Communications*, 15:1052–1071, 1997.
13. F. P. Kelly. *Stochastic Networks: Theory and Applications*, chapter Notes on effective bandwidths, pages 141–168. Oxford University Press, 1996.
14. M. Krunz and A. Makowski. Modeling video traffic using M/G/infinite input processes: a compromise between Markovian and LRD models. *IEEE Journal on Selected Areas in Communications*, 16:733–748, 1998.
15. J. T. Lewis, N. O'Connell, R. Russel, and F. Toomey. Predicting QoS for traffic with long range fluctuations. Technical Report DIAS-STP-94-31, Dublin Institute for Advanced Studies, Applied Probability Group, 1994.
16. B. Melamed, D. Raychaudhuri, B. Sengupta, and J. Zdepski. TES-Based video source modeling for performance evaluation of integrated networks. *IEEE Trans. Communications*, 42:2773–2777, 1994.
17. O. Rose. Simple and efficient models for variable bit rate MPEG video traffic. Technical Report TR No. 120, Institute of Computer Science, University of Wurzburg, 1995.
18. B. K. Ryu and A. Elwalid. The importance of long-range dependence of VBR video traffic in ATM traffic engineering: myths and realities. *ACM Computer Communication Review*, 26:3–14, 1996.

Enhancing Quality of MPEG Video through Partially Reliable Transport Service in Interactive Application

Vincent Lecuire¹, Francis Lepage¹, and Khalil Kammoun²

¹ Centre de Recherche en Automatique de Nancy (CNRS ESA 7039),
BP 239, F-54506 Vandoeuvre-lès-Nancy, France
{Vincent.Lecuire, Francis.Lepage}@cran.uhp-nancy.fr

² University of Karlsruhe, Kaiserstr. 12,
D-76128 Karlsruhe, Germany

Abstract. Real-time video communication over a packet switching network is subject to losses and errors. In few networking environment, a partially reliable transport service based on error control by retransmission is a possible option. We propose a retransmission-based correction scheme which is driven by the time remaining for the data delivery and also by the importance of artifacts the data loss produces in the reconstructed video. This scheme is applied to continuous MPEG video packet streams. Experimental results show that the picture quality of MPEG video is degraded more gracefully in the presence of network errors, and those with a low cost on network bandwidth. In addition, the best QoS policy to use the partially reliable transport service is highlighted.

1 Introduction

Real-time compressed video stream over a packet switching network is very sensitive to losses and errors. Network errors require the use of error correction and concealment techniques to provide the video quality necessary for the video based applications to be widely accepted and used. In general, error correction by retransmission is considered as not useful because video data must be played back continuously with a short delay time. In addition, when errors are due to network congestion, the retransmission of lost packets is not the correct reaction because it aggravates the situation and jams the network. Nevertheless, when time driven retransmission can be performed with a high probability of success, a retransmission-based approach becomes a viable option for error control on real-time stream, as is shown in [5] for audio, and in [20] for video. In comparison with forward error correction (FEC) which is very costly in extra bandwidth [18, 19], correction by retransmission can be very attractive under the condition that the number of retransmissions is well controlled.

In this paper, we propose a novel retransmission-based error control scheme in order to provide a partially reliable transport service suitable for interactive video application. Retransmission is driven by the time remaining for the data delivery, as well as

the importance of artifacts the data loss produces in the reconstructed video. A such transport service is attractive for error control on video coded with motion compensation techniques where losses in certain parts of the data stream are more annoying when viewed than losses in other parts of the data stream. Also, our scheme is applied to the MPEG coded video. In the following, section 2 presents the special features of our retransmission-based scheme in framework of an unicast transport protocol. Section 3 briefly describes the MPEG standard and evaluates the impact of losses on the coded MPEG video packet stream. The most and the least important parts of coded video packet stream are highlighted for partially reliable transport service requirements. In section 4, the experimental results are shown and discussed with an accompanying analysis. This shows the error control policy which is the best in the ratio between the picture quality improvement and the network bandwidth cost. Finally, section 5 concludes with a discussion of the possible extensions for the proposed service and future research developments.

2 An Error Control Scheme for Partially Reliable Transport Service

In this section, we present a partially reliable transport service, so that application's delay and error requirements are best respected. Few error control schemes for a such transport service have been yet proposed in [5, 7, 16, 6, 8]. However, the novel error control scheme that we propose is attractive in interactive multimedia application because application can specify the packet's QoS on the fly of the packetized stream.

2.1 Motivations

Our error control scheme, TSR, is based on retransmission which is driven by the time remaining for the data delivery and also by the packet's QoS class. This scheme is original due to QoS driven retransmission, which distinguishes two kinds of data packets :

- *valuable* packets (v-packets) : The loss of these packets is judged as serious for the application, and the retransmission should be tried if the time remaining until its scheduled playout time is more than one network round trip time.
- *ordinary* packets (o-packets) : The loss of these packets is considered as tolerable for the application, and the data transfer should proceed on as if nothing happens.

For requirements of interactive multimedia applications, we assume an error control that is characterized by two main features :

- Firstly, we give the responsibility of the error correction to the receiver side. In this case, the receiver returns to sender a negative acknowledgment (NAK) in order to request the retransmission of lost v-packets. The NAK approach is the best for the bandwidth cost to be added by the error correction scheme in comparison to cumulative acknowledgment (ACK) approach : With the ACK approach in effect, when an ACK message is lost, the sender will execute many useless retransmissions.

- Secondly, the QoS class associated with a data packet is attributed by the application to the sender side. It is absolutely necessary to prepare an on-line, and fine-grained QoS specification of all data units, which are marked on the fly of the packet stream building by the application. This feature is well suitable with the ALF (Application Level Framing) architectural concept [4].

The set of these two features conduces to a tricky challenge : When a packet is lost by the network, the receiver must to be able, both to detect this loss and also to identify its QoS class in order to launch a correct reaction. We treat this problem by using a technique for lasting coloration of data packets.

2.2 Technique for Lasting Coloration of Packets

This technique was presented in detail in [13]. Its principle consists in packet tagging, depending on the QoS class of the contained data, and with a lasting property in order to be recoverable without the packet itself. Informally, we refer to this tagging as the lasting coloration given to the packet.

2.2.1 Tagging the Packet Coloration at Sender Side

The coloration is tagged directly in the data packet by a packet numbering using a pair of counters (i, j). The counter i is used to trace the evolution of the v-packet sequence ; the counter j serves to trace the evolution of the o-packet sequence. However, this sequence number is not used for the usual sequence number because the roles are different. The packet numbering technique is shown in Fig. 1 with any QoS policy for partially reliable service.

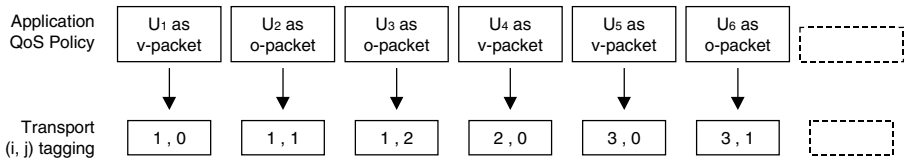


Fig. 1. A sample of (i, j) numbering for coloration of packets. The QoS class which is affected to a data packet is carried on the counter j. When j is equal to 0, it is a valuable data packet (v-packet). When j is greater than 0, it is an ordinary data packet (o-packet).

The sequence number (i, j) evolves according to the following rules:

- i and j are cyclical ordinal counters, with $0 \leq i < k$ and $0 \leq j < p$, where (k, p) defines the dimension of the numbering range (by example $k = p = 2^{32}$).
- The associated packet color is carried on the counter j, $j=0$ indicating a v-packet while $j \neq 0$ an o-packet.
- i and j are initialized both to 0 at the connection establishment.
- When the application requests to transmit a v-packet, i is incremented and j is reseted in the general case. Assuming this one follows (i, j) packet, it is numbered (i+1, 0) if $i+1 < k$, or (0, 0) if $i+1 = k$.

- When the application requests to transmit an o-packet, j is incremented in the general case. Assuming this one follows (i, j) packet, it is numbered $(i, j+1)$ if $j+1 < p$. But it is numbered $(i+1, 0)$ if $j+1 = p$ and $i+1 < k$, or $(0,0)$ if $j+1 = p$ and $i+1 = k$. The coloration of the packet is forced to the v-packet by the protocol in these both last cases. More the k and p values will be big, more a such situation will be rare.

2.2.2 Identifying the Coloration at Receiver Side

On the receiver side, the coloration of the lost packet should be identified to launch a correct reaction. A proper functioning requires the detection of all v-packet losses in order to test this correction. This is the case with the proposed numbering technique. In the following, we assume the last packet received correctly is numbered $(1,0)$, as that is shown in Fig. 2. The next expected packet would have a number that depends on its color : the number $(1,1)$ for a o-packet, or $(2,0)$ for an v-packet. In one case the packet $(1,1)$ is delivered is normal. In a second case of the packet $(2,0)$ also, but an ambiguity could not be resolved by the receiver if the loss of o-packet series beginning with $(1,1)$ occured. Finally, the arrival of a packet having a number different from $(1,1)$ and $(2,0)$ presents a third case, where the packet loss can be detected. A comparison between the two last packet numbers allows the determination of the lost packet color : If the counter i has not changed, the loss has affected only o-packets ; in the other case, where the counter i has changed, the loss affects v-packets.

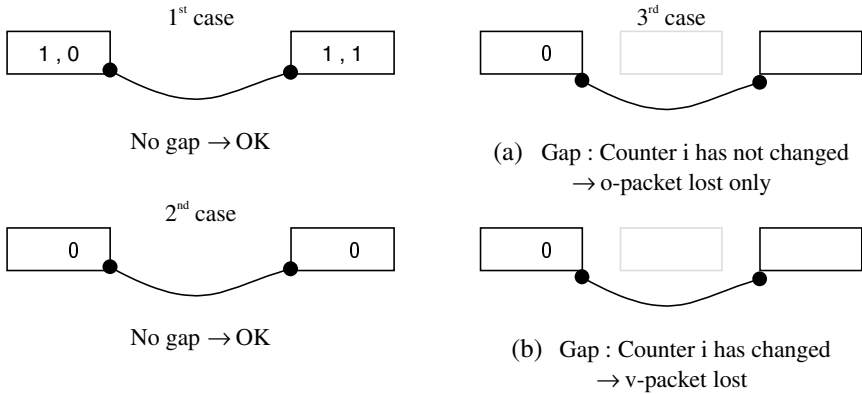


Fig. 2. Identification of its lasting coloration when packet is lost.

This original lasting coloration tagged directly in the packet guarantees the detection by the receiver of any v-packet loss, and allows the detection of most o-packet losses. This lasting coloration given to each packet, indicates how it needs to be controlled for its loss. This is a deterministic error correction scheme.

2.3 Technique for Time Driven Retransmission

In time driven retransmission part, the scheme is classic, based on the playout delay initialization, the packet timestamp, and the network delay and jitter estimation. The playout delay must to be adjusted in order to compensate for variable network delays,

and also to give extra time to attempt some correction. As in [17], the playout delay is defined to be the difference between the playout time for the receiver and the generation time for the sender. Various studies concluded that for natural hearing, interactive real-time applications can tolerate a latency of about 250 ms for only video stream [9], and of about 100 ms for audio/video stream [12]. Received packets are first queued into a smoothing buffer until their scheduled playout. When a loss of packet is detected on the side of the receiver, its retransmission can be requested if the delay till its scheduled playout is more than one round trip time (RTT).

2.4 Global Protocol Building

We have implemented a connection oriented transport protocol which is based on TSR scheme. This transport protocol (TSR-TP) has been designed as a light-weight protocol working on the top of UDP/IP. It is characterized as below :

- A three-way handshake procedure is used to establish a TSR-TP connection. In this step, network delay is estimated for playout delay auto-adjustement.
- NAK strategy is used to request packet retransmission applying selective repeat policy. The NAK message contains information including the part i in (i, j) number of the last v -packet that has been received in sequence and such a 64-bits value representing the table of the 64 next v -packets. One bit is set to 1-value in order to indicate that the corresponding v -packet in the table must be retransmitted. Inversely, one bit is reset to 0-value when the corresponding v -packet has been received successfully by the receiver.
- In a similar way to RTCP, report is periodically generated by the receiver to convey feedback on quality of data delivery and information of sender. The report message contains information including the highest packet number received, the number of packets lost, and timestamp. In the experiments described below, the delay between a report and the following is configured to 250 ms (i.e. 2 per GOP).

TSR-TP provides a partially-reliable transport service, i.e less-than-fully reliable and more-than-no reliable data transfer. That is akin to the concept of partial order connection (POC) presented in [1]. In this viewpoint, TSR-TP can be considered as an example of POC.

3 MPEG over Best-Effort Network

This section shows that the MPEG coding is well adapted to the partially reliable transport service which has been presented in previous section when this video stream is transmitted over best-effort network.

3.1 MPEG Video Coding

The MPEG-1 (Motion Picture Experts Group) video encoding standard [11] was designed to support video encoding at bit rates of approximately 1.5 Mbps. The image

size is based on the QIF-format (352x288 pixels). The quality of the video achieved with this standard is rough, but is good enough for person-to-person audio/video interaction or video-aided remote control. MPEG-2 was developed to meet the demands for high quality video coding. It is similar to MPEG-1. The MPEG compression algorithm relies on two basic techniques : block-based motion compensation (MC) for the reduction of the temporal redundancy and Discrete Cosine Transform (DCT) based compression for the reduction of spatial redundancy [14].

Three types of compressed frames are generated by an MPEG encoder : Intra-coded (I), Predictive (P), and Bi-directional (B) frames. The use of these three frame types allows MPEG to be robust (I-pictures provide error propagation reset points) and efficient (B- and P-pictures allow a good overall compression ratio). An I-frame is encoded independently from other frames based on DCT coding. A P-frame is encoded with reference to a past I- or P-frame, and is used as a reference point to the next P-frame. A B-frame is an interpolated frame that requires both a past and a future reference frame (I or P), and results in the highest amount of compression. B frames are never used as reference frames.

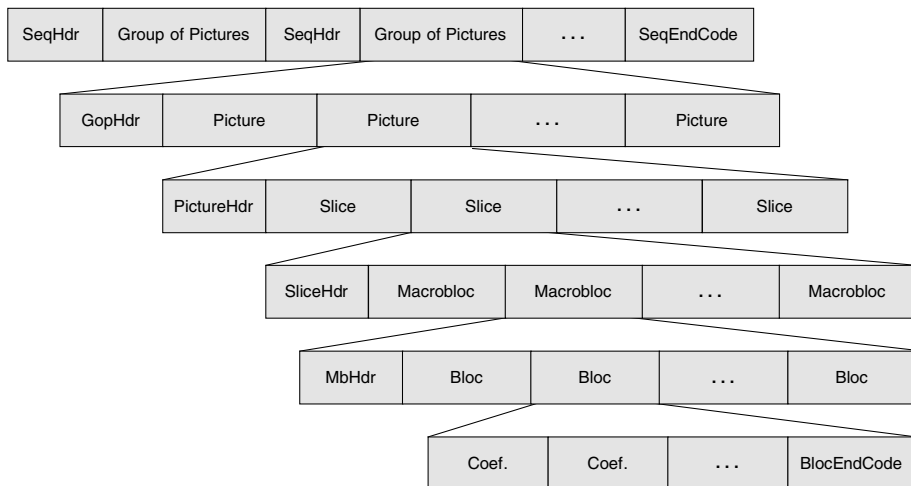


Fig. 3. An MPEG video stream is highly hierarchically structured, with six layers : Block, macroblock, slice, picture, GOP, and sequence. Each layer supports a definite function : Either a signal-processing function (Discrete Cosine Transform, Motion Compensation) or a logical function (Resynchronization, Random access point).

The MPEG video bitstream is hierarchically structured as illustrated in Fig. 3. The smallest entity defined by the standard is the *block*, which is an area of 8x8 pixels of luminance or chrominance. A *macroblock* contains four blocks of luminance samples and two, four or eight blocks of chrominance samples, depending on the chrominance format. A variable number of macroblocks is encapsulated in an entity called *slice*. A *picture* is composed of a variable number of slices. The entity called *group of pictures* (GOP) defines the relative frequency of occurrence of I, P, and B frames in the *sequence*. The application can fix the encoding pattern by specifying the value for both

of M , the distance between successive P frames, and N , the distance between successive I frames. A I-frame (one per GOP) is always placed at the beginning of the GOP and followed by a number of P-frames, which are interspersed between B-frames. Each frame contains exactly one picture.

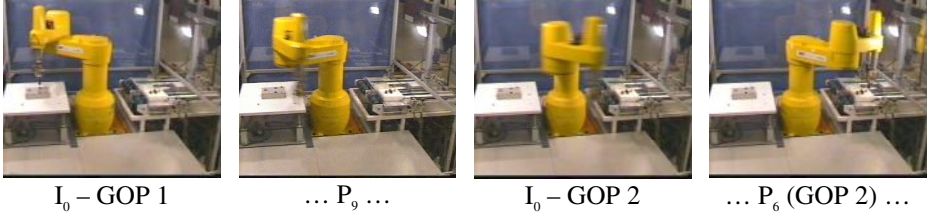


Fig. 4. The video *robot* sample used for experimentations. Encoding parameters value $N=12$, $M=3$, and $S=18$. The GOP is structured as I_0 -B1-B2-P3-B4-B5-P6-B7-B8-P9-B10-B11.

3.2 The Video Sample Used for Experimentations

All the experiments are executed in this paper on an two-minute long MPEG-1 elementary video stream with data constant bit rate encoding of 1 Mbps, for a total size of 16,8 Mbyte. The video sample shows a manufacturing robot in operation, as shown in Fig. 4. The scene was shot in a single fixed plan. The table 1 provides the statistics on the MPEG *robot* sample.

Table 1. Statistics for video *robot* sample.

Frame Type	Frames		Bytes	
	nr.	%	nr.	%
I	251	8.4	3430184	19.5
B	750	25.0	5945588	33.8
P	2000	66.6	8223618	46.7
Total	3001		17599390	

3.3 Network Loss Effects on MPEG Video Data

In an MPEG video stream, packet loss by the network reduces quality depending strongly on the type of the lost information [2]. Losses of syntactic data, such as headers and system information, affect the quality differently than losses of semantic data such as pure video information (e.g. motion vectors, DCT coefficients, etc.). Their effects on video quality are studied in detail in [3]. The MPEG sequence header information is very important in configuring system and computation environment. The picture header contains crucial information to initiate and configure the decoder for picture stream decoding. The GOP header provides a resynchronization point when errors occur. However, the GOP header is not so important as the sequence header and the picture header : Indeed if the GOP header is lost, there is no effect on the video quality.

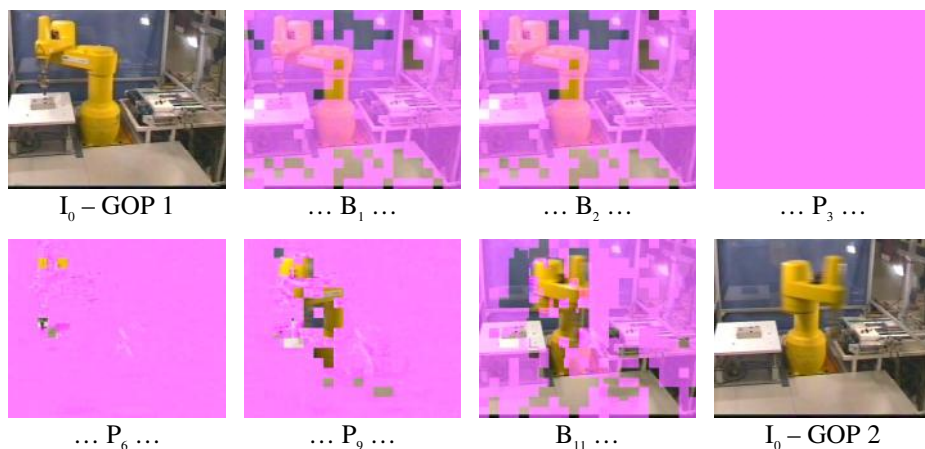


Fig. 5. Impact of the loss of the picture header on P_3 . It seriously disrupts the decoding process and leads to the loss of the original picture. Since P_3 is used as a reference picture, it affects also the predicted pictures, which reference the P_3 's macroblocks.

Furthermore, quality reduction depends on the location of the lost semantic data due, not only to the predictive structure of MPEG video coded streams, but also to the visual relevance of the data. Data loss spreads within a single picture up to the next resynchronization point (e.g. picture or slice headers) mainly due to the use of differential coding, run-length coding and variable length coding. This is referred to as spatial propagation and may damage any type of picture. When loss occurs in a reference picture (intra-coded or predictive frame), the damaged macroblocks will affect the non intra-coded macroblocks in subsequent frame(s), which reference the errored macroblocks. This is known as temporal propagation and is due to inter-frame predictions. Fig. 5 shows an example of network loss effect on the *robot* sample.

3.4 Packetization Scheme to Transport MPEG Video Stream

A well-designed packetization scheme may ease the influence of the packet loss on the picture quality. Some recommendations are suggested in [10] about the packetization to transport video or audio flow over transport protocols supported by RTP. For example, each header must be completely constrained within the packet. In addition, the packetization scheme must take into account the Maximum Transfer Unit (MTU) of the network to bind the maximum packet size. Packets larger than MTU size may be divided into fragments. If any of the fragments is lost during the network transmission, then the entire packet is lost. It is therefore expected that packets larger than MTU size will have a higher packet loss rate than packets smaller than the MTU size. Our MPEG frames are fragmented, respecting the following rules :

- Individual slices from I- and P- frames were put in their own packet. This requirement insures that the beginning of the next slice, after a packet loss, can be found without the receiver having to scan the packet contents for headers.

- Since B-frame slices are small, and their loss is less noticed, it would be inefficient to pack B-frame slices individually. Also, as far as it is possible in relation to the MTU of the network, we placed several slices from B- frames into a single packet.
- Picture headers were packed at the beginning of the packet enclosing the first slice of the frame they predated.
- Sequence headers and GOP headers were included in the same packet as the picture header of the I- frame which they preceded.

The packetization scheme takes advantage of the MPEG video stream structure to reduce dependencies among packets and to maximize the amount of decodable data at the receiver end in case of packet losses. It contributes to improve the robustness of the MPEG video.

3.5 Packet Classification by QoS Order

Due to the nature of MPEG video data streams, network losses in certain packets are more annoying when viewed than losses in other packets of the data stream. When the packetization scheme is used to transport the MPEG encoded bitstream, a packet error sensitivity list can be established to provide useful information for error control purposes. This list, which is given in table 2, shows a priority list in order to control the losses by QoS driven retransmission.

Table 2. Packet loss sensitivity list.

Rank	Pck Type	Enclosed Data	Error Propagation	
			Spatial	Temporal
1	A	Sequence Header	all pictures	multiple GOPs
2	B	I- Picture Header P- Picture Header	all slices	all pictures in GOP multiple pict. in GOP
3	C	I- Slice	one slice	all pictures in GOP
4	D	P- Slice	one slice	multiple pict. in GOP
5	E	B- Picture Header B- Slice	all slices one or more slices	one picture

A good policy in order to drive a partially reliable transport service is to try not too much correction (i.e. only for the loss concerning the most important packets in term of error sensitivity). The table 3 provides the packet statistics on the MPEG *robot* sample when the packetization scheme is applied.

4 Experimental Results and Discussion

In this section, the partially reliable transport service is evaluated for MPEG video stream. Several application QoS policies are tested. Results demonstrate a good ratio between the gain of video quality and the cost of network bandwidth.

Table 3. Statistics for packetized video sample.

Packet Type	Packets		Packet Size			
	nr.	%	avg.	min	max	total
A	251	1.0	614	584	628	153992
B	750	3.0	326	180	488	244816
C	4267	17.2	768	260	1436	3276192
D	13101	52.7	435	64	896	5700772
E	6504	26.1	1264	256	1368	8223618

4.1 Description of the Experiments for Performance Evaluation

For data gathering, we have carried out many experiments. The architecture of the environment is depicted in Fig. 6 : The sender and the receiver use TSR-TP, as described in the section 2. On the sender side, the program controls the target bit rate for transmission of the video sample, 1 Mbps. Packets exchange between both sender and receiver systems go across the lossy router which introduces random packet losses and artificial delays, in a same way as well-known *dummysnet*.

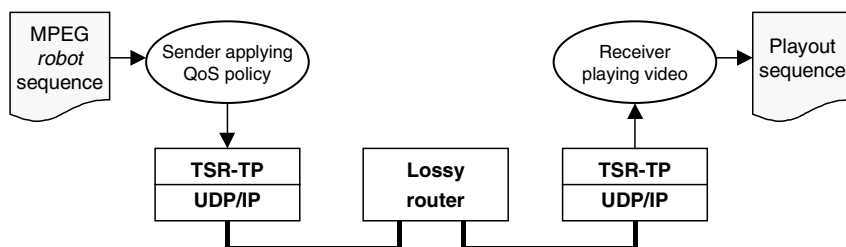


Fig. 6. The experimental environment is based on three computerized systems : The sender, the receiver, and the lossy router.

This experiment has been conducted with the *robot* sample for variable network loss rate. The average network delay and maximal jitter introduced by lossy router are respectively configured to 25 ms and 5 ms. In this network condition, we assume that the POC can attempt only one retransmission when the playout delay is initialized to 100 ms. In order to observe the performance of the POC using the proposed selective packet correction, the sequence has been transmitted with five different application QoS policies :

- PACK : The video stream is only packetized. The QoS policy is to specify all packets as o-packets. Then none packet is corrected by transport protocol when loss occurs. In this case, the POC provides a full-unreliable transport service.
- SPC1 : Here, the correction must be tried for the most annoying losses – i.e. those which cause an important degradation in the video quality. The QoS policy is to specify packets of type A and B as v-packets, and the others as o-packets. On the *robot* sample, the POC provides a partially reliable transport service, and 4 % of packets are under the error control.
- SCP2 : Here, the application is more demanding than previously. The QoS policy is to specify packets of type A, B and C as v-packets, for 21 % of packets being under the error control.
- SPC3 : With error concealment technique, MPEG replace a damaged slice in P-picture with the spatially corresponding slice of the first P- picture of the GOP. For this reason, first P- picture in the GOP is more important than other P- picture. So, the QoS policy is to specify packets of type A, B, C as v-packets, more the ones of type D which enclose slices of the first P- picture of each GOP. In the case of this POC, 39 % of packets are under the error control.
- SPC4 : The QoS policy to be evaluated is to specify packets of type A, B, C and D as v-packets. In the case of this POC, 74 % of packets are under the error control.

The experimental result is based both on objective QoS parameters extracted from real-time capture video traces as well as on subjective QoS viewed on the video client. For each experiment run, we computed the average PSNR for all pictures in the video sequence. In addition to assessing the subjective QoS perceived by the end user, 17 users watch the video signal decoded on the video client in all the traffic scenarios. The video material is assessed according to ITU BT.500-7 methodology. With this methodology, the user grades the video material in a 5-value impairment scale : Imperceptible (5), perceptible but not annoying (4), slightly annoying (3), annoying (2), and very annoying (1).

4.2 Gain of Picture Quality vs. Cost of Bandwidth Quantity

The gain is measured with a different transmission realization as a function of the packet loss rate, as shown in Fig. 7. It can be seen that the partially reliable service at the transport level enhances the packetization at the application level, achieving graceful degradation as the packet loss probability increases. With the SPC1 policy, the gain of PSNR is low. However, by the correction of I- and P- picture header SPC1 policy allows to rebuild more pictures at the receiver side (2755 vs 2558 with the *robot* sample of 3001 pictures when the packet loss rate is equal to 12 %). The loss of a picture is not noticed in the results because in this case, the PSNR is calculated with previous picture. In the cases of SPC2, SPC3 and SPC4 policies, the improvement on the average PSNR is more important. The results of the subjective tests confirm the general idea of an improvement in the video quality.

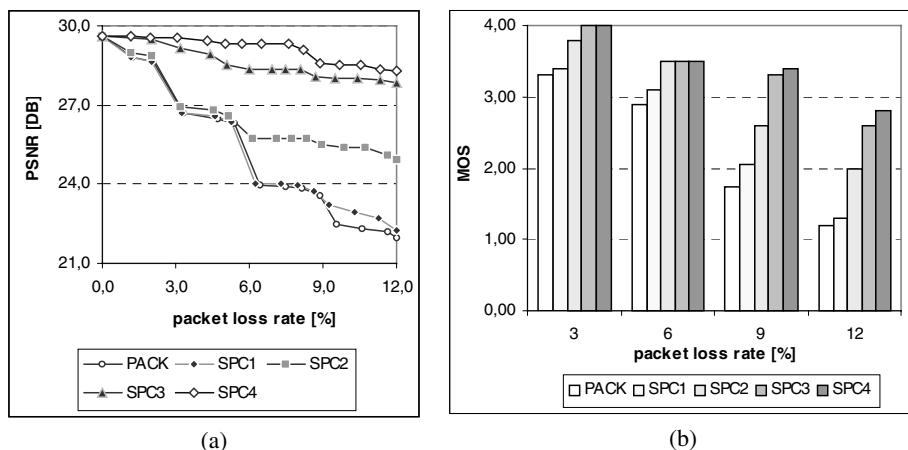


Fig. 7. Gain of picture quality. Part (a) shows the results of the objective tests. The largest gain is equal to 0.30 dB for SPC1, 2.98 dB for SPC2, to 5.58 dB for SPC3, and to 6.36 dB for SPC4 when the network loses 12% of packets. Part (b) shows the results of the subjective tests. The policies under test contribute to improvement respectively until 0.1 point for SPC1, 0.8 point for SPC2, 1.4 point for SPC3, and 1.6 point for SPC4 in the score given by human viewers.

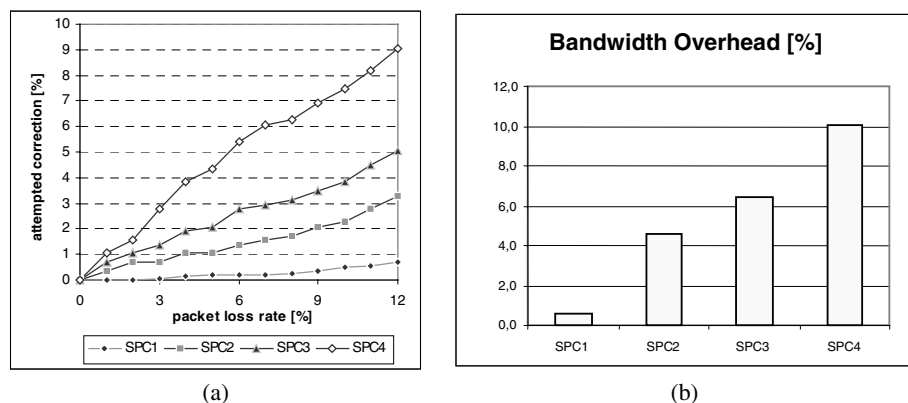


Fig. 8. Cost of bandwidth quantity. Part (a) shows the percentage of retransmissions which are attempted depending on packet loss rate and the QoS policy applied by application. Part (b) shows the extra bytes imposed by retransmissions. When packet loss rate is equal to 12 %, the overhead is to 0.5 % for SPC1, to 4.6 % for SPC2, to 6.4 % for SPC3, and to 10.1 % for SPC4.

The partially reliable transport service which is based on TSR scheme, provides a significant gain of picture quality by well-selected error correction by retransmission. In return, it imposes overhead on network bandwidth which must be evaluated. The results are shown in Fig. 8. We can see that the TSR scheme is not very costly to the network bandwidth in comparison to the use of FEC-based scheme such as *Priority Encoding Transmission* [15]. Although the performance gains entailed are notable, this FEC-based approach introduces a traffic overhead of about 25 %.

4.3 Toward a Proposal of the Best QoS Policy

In order to achieve the comparison between the tested QoS policies for the TSR-based partially reliable transport of interactive MPEG video, our purpose is to establish if it is better to define many of packets as v-packets (i.e. to address the picture quality) than define many of packets as o-packets (i.e. to address the network availability). We have adopted as criteria the ratio between the cost of the retransmissions and the gain of picture quality. On this criteria for comparison, the results show clearly that the SPC3 QoS policy is the best, as depicted in Fig. 9.

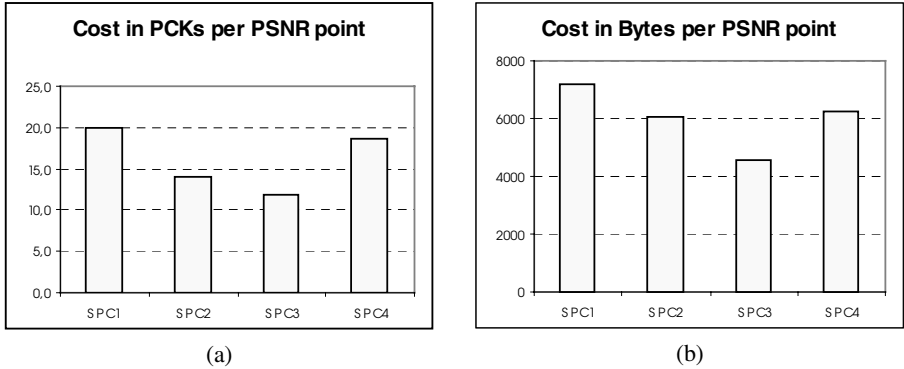


Fig. 9. Cost per PSNR point. Part (a) shows that TSR-TP generates an average 12 extra packets in order to gain 1 point of PSNR when SPC3 is applied. It is better than with SPC1 (20 extra packets), with SPC2 (14 extra packets), and with SPC4 (19 extra packets). Part (b) conduces to similar results. In order to gain 1 point of PSNR, TSR-TP generates an average 4537 extra bytes when SPC3 is applied. It is better than with SPC1 (7180 extra bytes), SPC2 (6043 extra bytes), and SPC4 (6277 extra bytes).

5 Conclusion

In this paper, we have proposed a partially reliable transport service based on time and QoS driven retransmission scheme in order to achieve a deterministic error control. We have shown that this scheme is very suitable when all data packets in the transport packet stream have not the same importance for the QoS perceived by the end user, as this is the case in MPEG video packetized stream. When the use of a such partially reliable service is possible, this approach is attractive because it imposes little overhead on network resources, as that has been resulted from our performance evaluation executed by experimentations. This experiment has highlighted the best QoS policy for MPEG video based application, on the criteria of ratio between the picture quality improvement and the network bandwidth cost. Finally, a perspective of this work is to show that TSR-TP is suitable on MPEG-4 as well as JPEG2000 coded video streams.

References

1. Amer, P., Chassot, C., Connolly, C., Conrad, P., Diaz, M.: Partial Order Transport Service for Multimedia Applications. *IEEE/ACM Transactions on networking*, vol. 2, no. 5 (1994)
2. Boyce, J.M., Gaglianella, R.: Packet Loss Effects on MPEG Video Sent Over the Public Internet. *Proc. of ACM Int. Multimedia Conference*, Bristol, England (1998)
3. Chiu, D., Cai, L., Mc Cutcheon, M., Ito, M., Neufeld, G.: Transport of MPEG-2 Video in a Routed IP Network : Transport stream errors and their effects on video quality. *Proc. of 6th workshop on Interactive Distributed Multimedia System*, Toulouse, France (1999)
4. Clark, D., Tennenhouse, D.: Architectural Considerations for a New Generation of Protocols. *Proceedings of the ACM SIGCOMM'90*, Philadelphia, USA (1990)
5. Dempsey, B.J., Liebeherr, J., Weaver, A.C.: On retransmission-based Error Control for Continuous Media Traffic in Packet-Switching Networks. *Computer Networks and ISDN Systems*, vol. 28, no. 5 (1996)
6. Dwyer, D., Ha, S., Li, J.R., Bharghavan, V.: An Adaptive Transport Protocol for Multimedia Communication. *Proc. of IEEE Int. Conference on Multimedia Computing and Systems*, Austin (1998)
7. Gong, F., Parulkar, G.: An Application-Oriented Error Control Scheme for High-Speed Networks. *IEEE/ACM Transactions on Networking*, vol. 4, no. 5 (1996)
8. Grinnemo, K., Brunstrom, A.: Evaluation of the QoS offered by PRTP-ECN - A TCP-Compliant Partially Reliable Transport Protocol. *Proc. of 9th Int. Workshop on Quality of Service IWQoS 2001*, Karlsruhe, Germany (2001)
9. Hehmann, D.B., Salmony, M.G., Stüttgen, H.J.: Transport services for multimedia applications on broadband networks. *Computer Communication*, vol. 13, no. 4 (1990)
10. Hoffman, et al: RFC 2250 : RTP Format for MPEG1/MPEG2 Video. *Internet Draft* (1998)
11. ISO/IEC 11172-2: Coding of moving pictures and associated audio for digital storage media at up to about 1.5 Mbit/s.
12. Karlsson, G.: Asynchronous transfer of video. *IEEE Computer Magazine*, vol. 34, no. 8 (1996)
13. Lecuire, V., Lepage, F.: Contrôle déterministe de la fiabilité par un marquage direct sur les paquets. *Electronic Journal on Networks and Distributed Processing*, vol. 9 (2000)
14. Le Gall, D.: MPEG : A video compression standard for multimedia applications. *Communications of the ACM*, vol. 34, no. 4 (1991)
15. Leicher, C.: Hierarchical Encoding of MPEG Sequences Using Priority Encoding Transmission (PET). *Tech. Report TR-94-058*, ICSI, Berkeley (1994)
16. Marasli, M., Amer, P., Conrad, P.: Retransmission-based Partial-Reliable Transport Service : An Analytic Model. *Proc. of the IEEE Infocom'96*, San Francisco (1996)
17. Moon, S.B., Kurose, J., Towsley, D.: Packet Audio Playout Delay Adjustment : Performance Bounds and Algorithms. *ACM/Springer Multimedia Systems*, vol. 6, no. 1 (1998)
18. Pillai, R.R., Prabhakaran, B.P., Qiang, Q.: A Forward Error Recovery Technique For MPEG-II Video Transport. *Proc. of 7th ACM Int. Multimedia Conference*, Orlando (1999)
19. Rasheed, Y., Leon-Garcia, A.: AAL 1 with FEC for the Transport of CBR MPEG2 Video over ATM Networks. *Proc. of IEEE Infocom'96*, San Francisco (1996)
20. Rojas-Cardenas, L., Dairaine, L., Senac, P., Diaz, M.: Error recovery mechanisms based on retransmissions for video coded with motion compensation techniques. *Proc. of IEEE Packet Video Workshop*, New York (1999)

Delivering of MPEG-4 Multimedia Content over Next Generation Internet

Toufik Ahmed, Guillaume Buridant, and Ahmed Mehaoua

University of Versailles - CNRS-PRISM Laboratory
45, av. des Etats Unis 78035 – Versailles - France
Tel: +33 1 39 25 40 56
Fax: +33 1 39 25 40 57
{tad, bug, mea}@prism.uvsq.fr

Abstract. IP and MPEG-4 digital video are two key technologies that will be increasingly combined in the very near future for the deployment of next generation Internet multimedia services. This article discusses technical issues related to the transport and the QoS control of MPEG-4 multimedia applications, such as interactive video-on-demand, over Internet. In particular, it describes the design, implementation and performance evaluation of an experimental Java-based MPEG-4 interactive VOD system over IP DiffServ networks. We demonstrate that Assured Forwarding Per Hop Behavior is a performing candidate for conveying real-time video communications in conjunction with video-aware packet marking and scheduling mechanisms and advanced MPEG-4 DMIF signaling protocol. In this issue, we also propose a Video packet Marking Algorithm for IP Diffserv routers, called DVMA that reduces video packet loss probability and end-to-end transfer delay during network congestion.

1 Introduction

The rollout of Internet opens up a new frontier for the digital video broadcasting industry. Two worlds that were barely connected are on the verge of being merged: namely, real-time video and Internet. Thanks to the transition of the video medium from analogue to digital, and the rise of powerful video compression standards, such as MPEG-2 and MPEG-4, it is now possible to combine video, audio, and data within the same signal and transmit it over an integrated packet switching network infrastructure, such as Internet.

This combination leads to powerful new multimedia applications, with limitless possibilities of great commercial potential. For example, computers can be turned into traditional TV receivers and the digital set-top boxes can host applications such as interactive TV, e-commerce, and customized programming.

This article discusses technical issues related to the delivery of MPEG-4 multimedia content, such as interactive video-on-demand, over next generation Internet. The article is organized as follows. Section 2 is devoted to the description of

IP QoS networking and MPEG-4 video technologies that contribute to this issue. Section 3 discusses the design and the implementation of the proposed interactive video distribution service. Interaction between MPEG-4 framework and IP DiffServ architecture is also investigated. In section 4, a QoS control mechanism named DVMA, for *IP DiffServ video packet Marking Algorithm*, is presented and integrated in our architecture. Performance evaluation and analysis is carried out in section 5 and 6 respectively. Finally, we conclude and discuss future work in Section 7.

2 Internet and Digital Video Convergence

2.1 Internet Multimedia Protocols: RTP, RTCP, and RTSP

Unlike ordinary data services, multimedia services are sensitive to delay, jitter, and loss, which in turn are affected by the volume and pattern of traffic load in the network. A Transport Control Protocol (TCP)-based stream will respond to losses or excessive delays by triggering a retransmission and exerting window flow control, both of which will have negative effects on the quality of the multimedia service.

The IETF *Audio/Video Transport* (AVT) working group has partially addressed this issue by proposing the *Real Time Transport Protocol* (RTP), the associated *Real Time Transport Control Protocol* (RTCP) and the *Real Time Streaming Protocol*. RTP is a lightweight transport protocol based on the concept of application-level framing [1]. Since the reliable transmission mechanism offered by TCP incurs considerable overhead by retransmission, RTP does not rely on TCP. Instead, applications are put on top of UDP. How to cope with the lost packets is up to the applications. Following the application-level framing principle, RTP functionality is usually integrated into the application rather than being implemented as a separate protocol entity. RTP provides basic packet format definitions to support real-time communication but does not define control mechanisms or algorithms. The packet formats provide information required for audio and video data transfer, such as the incoming video data packet sequence. Continuous media such as non-compressed PCM audio can be synchronized by the use of sequence numbers. Non-continuous data such as MPEG can be resynchronised by using the time stamp fields [17].

RTCP is the control protocol for RTP, and provides mechanisms for data distribution monitoring, cross media synchronization, and sender identification [1]. The sender transmits a multimedia data stream by using RTP packets. The receiver periodically sends RTCP packets that contain information about the received RTP packets. The information includes statistics such as the highest sequence number received, inter-arrival jitter, or packet loss rate.

RTSP [28] is used for initiating and controlling delivery of stored and live multimedia content to both unicast and multicast destinations. RTSP borrows time concept from MPEG-2 DSM-CC, but unlike DSM-CC, it does not depend on an entire set of supporting protocols [29]. RTSP is transport-independent, and can use either TCP or UDP as the transport protocol. The state machine makes RTSP suitable for

remote digital editing and retrieval. RTSP is also text-based protocol, and therefore easy to parse. RTSP reuses the HTTP concept, but unlike HTTP, RTSP is a stateful protocol.

2.2 Internet QoS Architectures: INTSERV and DIFFSERV

The provision of quality of service within the Internet has been the subject of significant research and deployment efforts recently. Two approaches have drained the attention of the IETF: namely the *IntServ* and the *DiffServ* architectures.

The *Integrated Service* (IntServ) model is motivated by the ability for applications to choose among multiple, controlled levels of delivery service for their data packet [5]. In the integrated service framework, many functions are used to provide QoS: the first function is control services such as Controlled-Load [6] and Guaranteed [7]. The second function may be provided in a number of ways, but is frequently implemented by a resource reservation setup protocol such as RSVP [8].

The *Differentiated Services* (DiffServ) model [9] uses a small, well-defined set of building blocks from which a variety of aggregate router behaviours may be designed to provide quality of service [10]. IP packets are tagged with distinct labels before entering an IP Diffserv domain and will receive particular forwarding treatment at each network node along the path. This set of routing function is called a *Per-Hop Behavior* (PHB). The PHB is characterized and invoked according to the Differentiated Service Code point (DSCP) value located in the packet's header. A small number of PHBs has been currently standardized by the IETF Diffserv working group. The most well known are Expedited Forwarding (EF) [11] and Assured Forwarding (AF) [12].

The key difference between Intserv and Diffserv is that while Intserv provides end-to-end QoS service on a per-flow basis, Diffserv is intended to provide better scalability through flow aggregation and class differentiation over a long timescale.

Therefore, we believe that IP Diffserv is the most suitable model for delivering interactive video content over Internet and we present, in the following lines, some related works in this field.

In [2], The authors present a content-based packet video forwarding mechanism where the QoS interaction between video applications and Diffserv network is taken into account. The interaction is performed through a dynamic mapping between the RPS (relative priority score) of each video packet and the differentiated forwarding mechanism.

In [3], the authors introduce a QoS management system for multimedia servers that benefits from the scaling properties of layered media streams. The presented system enable to map application QoS demands to available streams according to inter-stream QoS dependencies.

Authors in [4] present a bit-stream classification, prioritization, and transmission scheme designed for MPEG-4 video. Different type of data are re-assembled and assigned to different priority using IP Diffserv model.

2.3 MPEG-4 Multimedia Framework: BIFS and DMIF

MPEG-4 is an emerging digital multimedia standard with associated protocols for representing, manipulating and transporting natural and synthetic multimedia content (i.e. audio, video, data) over a broad range of communication infrastructures. The original characteristic of MPEG-4 is to provide an integrated object-oriented representation of multimedia content for the support of new ways of communication, access, and interaction with digital audiovisual data, and offering a common technical solution to various telecommunications, broadcast, and interactive services. MPEG-4 addressed a broad range of existing and emerging multimedia applications such as video on the Internet, multimedia broadcasting, content-based audiovisual database access, games, audiovisual home editing, advanced audiovisual communications and video over mobile networks.

This is achieved by a set of tools defined in several recommendations of the standard. First, the media compression schemes are defined in the Visual [14] and Audio [15] parts of the MPEG-4 framework. Every multimedia element to be compressed are presented as individual audiovisual objects. The combination of these elements is assured by the scene description language called *Binary Format for Scenes* (BIFS) defined in MPEG-4 Systems document [13]. The delivery of the media is defined in the *Delivery Multimedia Integrated Framework* (DMIF) [16], which is the part 6 of MPEG-4 specification. DMIF is actually the control plane of MPEG-4 *Delivery layer*, which allows applications to transparently access, retrieve and view multimedia streams whether the source of the stream is located on a remote or local end-system.

MPEG-4 terminal architecture is composed of three layers (see Figure 1): the *Compression Layer*, the *Sync Layer* and the *Delivery Layer* [13].

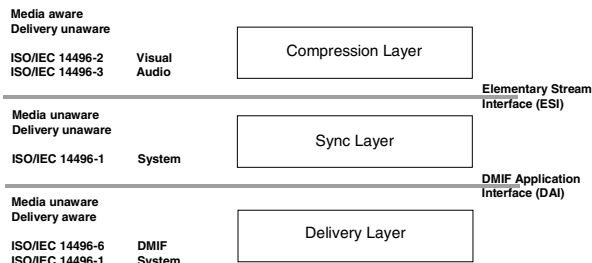


Fig. 1. MPEG-4 Framework

Compression layer generates representation of content data called *Elementary Streams* (ES), the hierarchical relations; locations and properties of ESs in a representation are described by dynamic set of *Object Descriptors* (OD). ESs are the basic abstraction for any data source. ODs are themselves conveyed through one or more ESs. MPEG-4 scene is described by *Scene Description* (SD) information that addresses the organization of audiovisual objects within a scene, in terms of both spatial and

temporal location. SD is performed using BIFS, which is a VRML-based language. ESs are partitioned into *Access Units* (AU) which are defined by MPEG-4 general framework for the framing of semantic entities in ES. For examples, a valid MPEG-4 ES could be an MPEG-1 video, labeled with MPEG-4 system information in its headers. An AU would then be one video frame I, B or P. Those AUs will be labeled with priority information, timing information, and others. An AU is the smallest data entity to which timing information can be assigned.

On the *Sync layer*, AUs are conveyed into a sequence of packets called SL-PDUs. A SL-PDU consists of SL packet header and SL packet payload. The header provides means of continuity checking in case of data loss, carries the time stamps, and associated control information.

The SL-PDUs are subsequently transmitted to the *Delivery Layer* for multiplexing and generating a FlexMux stream. The FlexMux tool is one of the components of the MPEG-4 DMIF. FlexMux is a flexible and simple data multiplexer that accommodates interleaving of SL-PDUs. Two different modes of operation for FlexMux are defined. The first mode is called “Simple Mode” in which one SL-PDU is encapsulated into one FlexMux packet, and the second mode is called “MuxCode Mode” in which one or more SL-PDU are encapsulated into a single “FlexMux” packet. The interface between the *Sync layer* and the *Delivery layer* is referred to DAI (*DMIF Application Interface*) [16]. It offers content location independent procedures for establishing MPEG-4 sessions and access to transport channels such as RTP/UDP/IP. DMIF is primarily an integration framework. It provides default DMIF signalling protocol which corresponding to DNI (*DMIF Network Interface*).

3 An Experimental Real-Time and Interactive Video on Demand Service over IP

3.1 Proposal of a Video Packet Marking Algorithm for IP DiffServ Routers

MPEG-4 uses hierarchical coding for resolving scalability and heterogeneity issues. Different coding modes exist: *Spacial scalability* allows the decoder to treat a subset of streams produced by the coder to rebuild and display textures, images and objects video with a reduced space resolution. *Temporal scalability* allows the decoder to treat a subset of streams produced by the coder to rebuild and display a video with reduced temporal resolution. With *SNR Scalability* (Signal to Noise Ratio), the coder transmits the difference between the original image and the preceding one. This allows improving subjective quality of the final image to get maximum quality [21].

We use the IP packet marker Algorithm described in [21] to tag the different MPEG-4 ESs (see Figure 2). The MPEG-4 video stream is composed of tree hierarchical layer: *Base layer stream* that offers a **minQoS** (Minimum QoS), *enhanced layer 1 stream* is used to improve minQoS and to offer a **MedQoS** (Medium QoS) and *enhanced layer 2 stream* is used to improve medQoS and to offer a **MaxQoS** (Maximum QoS).

In the Best Effort service, when network congestion occurs, packets are discarded with no distinction. Therefore, essential audiovisual data (i.e. control and decoding information, audio, ...) can encounter important drop probability while less important information. The result of this situation is that the decoder cannot reproduce properly the video sequences. The solution of this problem is to distinguish between important data and less relevant ones. This can be achieved by taking into account MPEG-4 video coding properties and marking the outgoing video packet at the video server side. In this situation, IP video packets are dropped according to their semantic

```

if stream is "audio stream" then (application of property 2)
  if coder rate is "low rate"
    then DSCP=AF Low Drop Prec
    //example AF11
  if coder rate is "medium rate"
    then DSCP=AF Medium Drop Prec
    //example AF12
  if coder rate is "high rate"
    then DSCP=AF High Drop Prec
    //example AF13

if stream is "video stream" (application of property 3)
  if "base layer video stream" (level 1 = minimum QoS)
    then DSCP = AF low Drop Prec
    //example AF21
  if "enhanced layer video stream 1" (level 2 = medium QoS)
    then DSCP = AF Medium Drop Prec
    //example AF22
  if "enhanced layer video stream 2" (level 3 = maximum QoS)
    then DSCP = AF Medium Drop Prec
    //example AF23

if stream is "objects descriptor, scene descriptor" (application of property 4 )
  then DSCP = EF
  //descriptions streams are significant, no loss
  //it's necessary that these streams will be available
  //as soon as possible in MPEG-4 player to interpret
  //correctly the scene

```

Fig. 2. DVMA – An IP DiffServ packet Video Marker Algorithm

relevance. If the packet is marked as low priority, then it will be dropped first in situation of congestion. In addition, when the video sequence is structured as a multi-layered video stream, multiple heterogeneous clients can receive simultaneously the video sequence by selecting the number of layer that they can handle.

3.2 Design and Implementation of the IP Diffserv Experimental Network

The latest Linux kernel (2.4) offers a wide variety of network traffic control features [18], [17], such as IP Diffserv support. Unfortunately, basic classification and IP packet DS-field manipulation required by the Diffserv model are not correctly addressed by Linux.

When a Linux-based IP gateway receives IP packets from its input device that must be forwarded to the output device, the traffic control determines if packets have to be queued or dropped. For example, if the queue length reaches a threshold or if the traffic exceeds some rate limit, the scheduling module decides which packet should be served first according to given priority. Finally, it can delay the sending of packets, for example to limit the rate of outbound traffic. Linux Box can operate as Edge or Core router. The Edge router can classify, police, mark aggregates and reshape traffic. The Core routers provide transit packet forwarding service between other core and edge routers. It manages traffic to avoid and cope with congestion.

The traffic control code in the Linux kernel consists of the following major conceptual components: (1) queuing disciplines, (2) classes (within a queuing discipline), (3) filters and (4) traffic policing. Figure 3. depicts the Linux traffic control components.

The queue determines the order in which packet will be served and send to the outgoing ports. Traffic can be divided into different classes according to specific rules. Each class maintains a queuing discipline to serve the packets, and it is associated with a priority. Actually there are 11 queuing disciplines implements in the Linux Kernel [18].

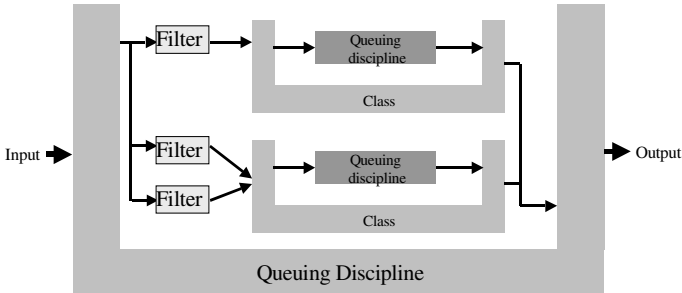


Fig. 3. Linux Traffic Control Components

Filters are used to discriminate the packet and to associate them with the appropriate class of traffic. The classification is based on certain properties of the packet. In the differentiated service the classification is based on the IP DSCP (DiffServ Code Point) field.

Traffic policing is used to control the amount of entering traffic in each class. It controls that a specific traffic does not exceed a predefined bound.

3.3 Design and Implementation of the Proposed MPEG-4 Interactive VOD System

We developed an MPEG-4 interactive video on demand system consisting of a video server and a multimedia client communicating over the previously described IP Diffserv network testbed as illustrated in Figure 4. The client and server code is based on *Java Media Framework* developed by Sun Microsystems [19]. We extended the client and the server functionalities by integrating DMIF signalling. Thus, the client can establish a session with the server using DMIF primitives to select, negotiate and retrieve MPEG-4 video sequences. The DMIF implementation is out of the scope of this paper but additional information can be found in [20].

The MPEG-4 Server consists of MPEG-4 pump, DMIF Instance for IP network, and tools for RTP/UDP/IP stack.

The server delivers ES packetized Stream to FluxMux tools which encapsulates ES packet in RTP packets. For our testbed we have used a video only presentation with three hierarchically stream, which are carried in three separated RTP session. Document [21] explains in more details the video encapsulation protocol used.

The MPEG-4 pump talks to a clients via DMIF and delivers RTP stream during the time of the session. The DMIF Instance is responsible for session setup request and release, it also provides QoS requirement for the application. This option is not yet supported in our implementation. The signaling channel for DMIF uses TCP protocol for reliability.

When a client requests a presentation from the server, he queries its local DMIF daemon to initiate a session with the remote video server. The local DMIF contact the remote DMIF to establish a signalling channel. The server answers client using the same channel with response code. When the session is setup, the client requests one or several stream channels for presentation, afterwards, the server pushes the audiovisual data in the RTP channel and marks the stream with the appropriate PHB using DVMA.

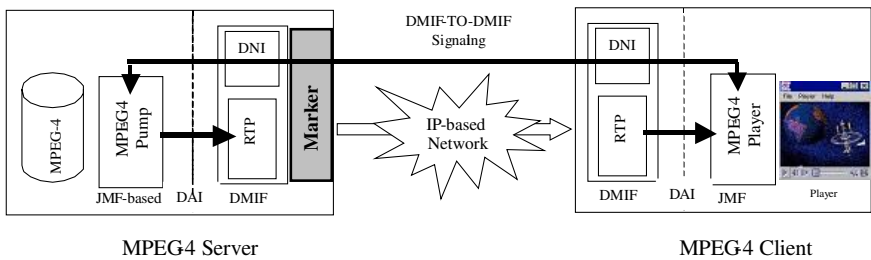


Fig. 4. A JAVA-based MPEG-4 VOD system with DMIF Session Signaling Implementation

The example below resumes the steps for activating a VoD session by the client:

1. The client application initiates the services by calling in it local DMIF a *DA_ServicesAttach* primitive.
2. The DAI determines weather a new Network Session is needed. If it is, it calls a *DN_SessionSetup*.
3. The server DMIF replays the client DMIF by attaching creating a new session.
4. The DMIF server activates the session in the video server.
5. Upon these steps, media channels are opened and media flows can be sent..

4 Performance Evaluation

4.1 Network and Traffic Models

Let is consider the IP network testbed depicted in Figure 5., where a server delivers MPEG-4 video content to several heterogeneous receivers. The receiver can be a simple wired PC or any wireless terminals such as a mobile GSM/GPRS/UMTS phone capable of rendering MPEG-4 video sequences. The network nodes are IP Diffserv compliant routers. We use Linux-based IP routers with Diffserv implementation.

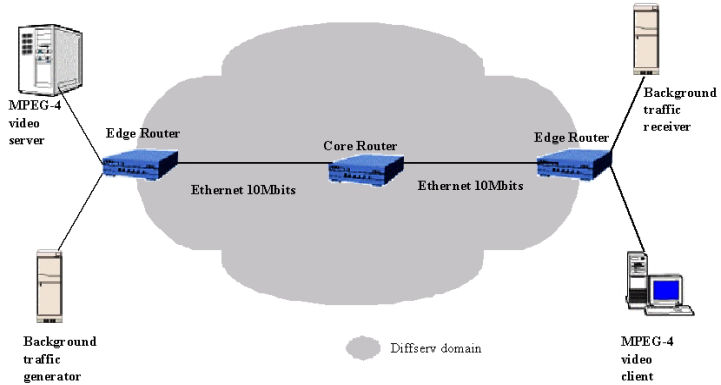


Fig. 5. IP DiffServ Network testbed

Our IP Diffserv network testbed is composed of a video server application that plays MPEG-4 video stream for many heterogeneous clients. The server and clients are connected through TCP/IP network. We exploit some test-scenario on the transmission process. We quantify QoS measurement and we compare between transmission process in Best Effort service and in Differentiated Service.

We've implemented an MPEG-4 video streaming system to test and evaluate the performance of the proposed marker algorithm. The testbed is illustrated in Figure 5. The testbed is composed of two edge routers and a core router running Linux with IP Diffserv components. All Ethernet links are 10 Mb/s.

4.2 Edge Router Configuration

Edge routers accept traffic into the network. They can characterize, police, and/or mark customer traffic between other edge or core routers.

Within the Diffserv domain, service association is performed according to the DSCP value in each packet's IP header. Therefore, the video application must mark the packets correctly to obtain a particular level of service within the Diffserv region.

It is not desirable to let the network (edge router) marking the incoming traffic as several algorithms do. The marking algorithms such as *Time Sliding Window Three Colour Marker* (TSWTCM) [19] and a *Two Rate Three Color Marker* (TRTCM) [23] cannot be used in the case of multimedia traffic. These algorithms make no distinction between essential audiovisual data and less important ones and thus cannot mark the packet correctly for future control within the network.

The configuration of the edge router is simple in our testbed. Our edge router limits the amount of EF traffic to 15% of the bandwidth capacity rate i.e. 1.5Mbit. We used a policing mechanism to limit the EF traffic, because EF is more require in term of latency time and losses. Furthermore, the router must make sure that the departure rate configured is greater than the arrival rate and the queuing delay is minimized. This is extensively sufficient since we use EF only for sensitive information such as OD and

BIFS signalling, that should be transported to the destination as early as possible, with no loss and with a minimum jitter. The EF flow is bounded and isolated.

For the Edge Router we used a simple *Class-Based Queuing* (CBQ) discipline to classify the incoming traffic.

4.3 Core Router Configuration

Core routers are configured to perform (1) packet classification based on DSCP, (2) packet scheduling, (3) queue management, (4) policing and (5) packet dropping.

We used CBQ as the packet scheduler, which is a classical assumption as proposed in [24]. For CBQ a single set of mechanisms is proposed to implement link-sharing and real-time services. In our implementation, CBQ is used to classify EF, AF, and BE traffic classes so each connection can get appropriate resources based on packet marking.

Our CBQ mechanisms include:

- a *classifier* to classify arriving packets to the appropriate class. This classification is based on DSCP field in the IP header,
- a *scheduler* to determine the order in which packets from the various classes will be sent. Linux Kernel implements several queuing disciplines (e.g. **RED** “Random Early Detection” or **GRED** “generalized RED”). The GRED queuing discipline is used to support multiple drop priorities as required by AF PHB. One physical GRED queue is composed of multiple **VQ** (Virtual Queue). GRED can operate in **RIO** (RED with In/Out bit) *mode* [25], with coupled average queue estimates from the virtual queues, or in *standard mode* where each virtual queue has its own independent average queue estimate as required by RED [26]. In our testbed, we used GRED as queuing discipline for the AF classes, since our marker algorithm takes into account these properties to give different level of QoS: *minQoS*, *MedQoS* and *MaxQoS*.

For the AF classes we allocated 1.5Mbit/s for each AF sub-classes namely AF1, AF2, AF3 and AF4, all of which are bounded. For the best effort traffic, we allocated a minimum of 3.5Mbit but this traffic is allowed to borrow any available bandwidth.

5 Performance Analysis

Figures 6 and 7 give statistical properties of the MPEG-4 video traffic generated by the video server to the client. The network is loaded by TCP and UDP background traffic. In our testbed, the background traffic is marked as best effort traffic.

The first set of performance measurements are on packet loss probability for each video elementary streams, in both IP best effort and Diffserv models. The second set of measures concern the end-to-end one-way delay encountered by video packet

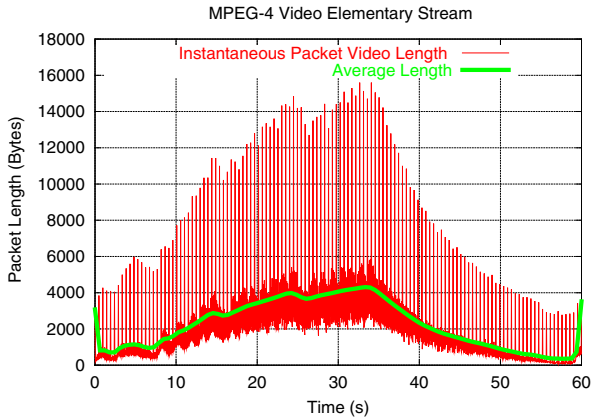


Fig. 6. Instantaneous packet length in MPEG-4 Video Elementary Stream

between the server and the destination. Two different network loads have been tested, i.e. 80% and 95% of the available bandwidth.

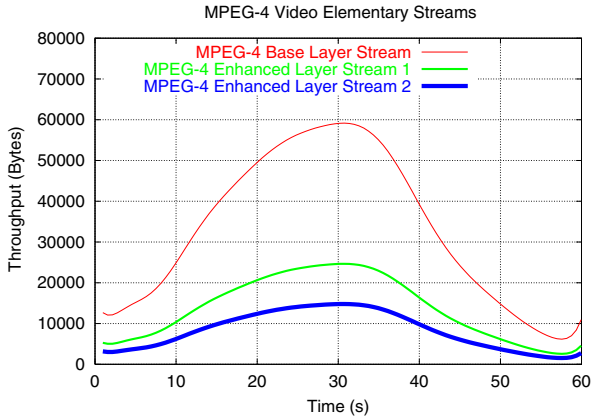


Fig. 7. MPEG-4 Video Layered Streams

5.1 IP Packet Losses

Figures 8 and 9 depict the variation of the video packet loss versus network load for IP Best Effort and IP Diffserv respectively. Individual MPEG-4 video layers encounter different packet loss probability. With IP Best Effort, the most essential video layer (i.e. base layer) obtains the highest loss with 60 % for a network load of about 80%. Using IP Diffserv and DVMA, the base layer faces the lowest packet drop probability with a maximum of about 0.2 %.

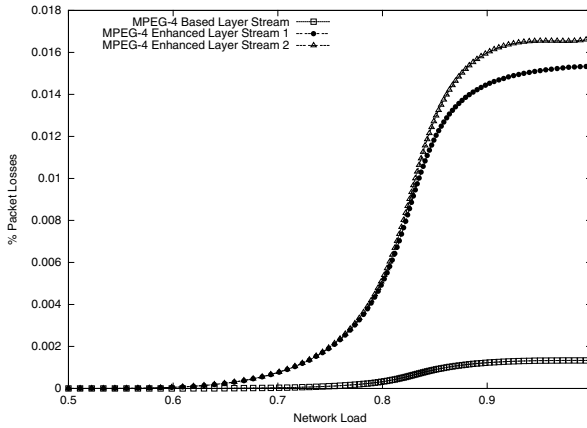


Fig. 8. MPEG-4 video packet Loss ratio vs Network load with IP Diffserv

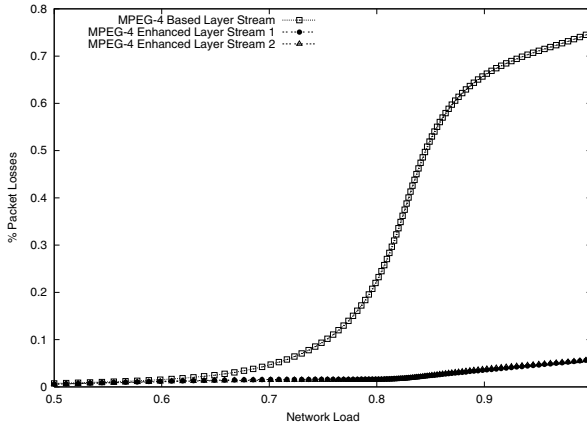


Fig. 9. MPEG-4 video packet Loss ratio vs Network load with IP Best Effort

Figure 10 shows the percentage of packets losses when the amount of background traffic is about 80% (6.5 Mbit/s) of the bandwidth. This leads to some losses essentially at time $t=30s$ i.e. when the video server sends at its peak rate (figure 9). The majority of the losses are concentrated within the base layer stream. This provides a degradation of the video quality at the client. Moreover, the receiver can't decode properly the other elementary video streams without the good reception of the base layer. Loss increases dramatically when the network load increases (Fig. 11).

The high losses of the base layer are due to his highest requirement of bandwidth. We can compare it with a MPEG-2 video stream where I pictures are bigger than P and B pictures. However, they are much more important. In this case, I packet's losses must be lower than the other packets types losses. When talking about MPEG-4, the

base layer stream must have a low packet loss when the enhanced layers streams 1 and 2 must have a respectively increasing drop probability.

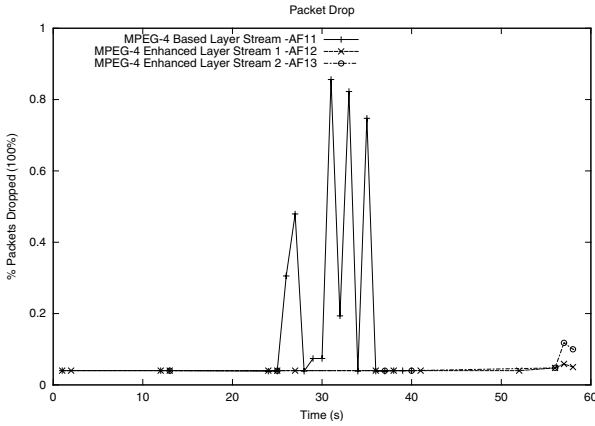


Fig. 10. Packet drops in Best Effort scenario. - Network Load 80% -

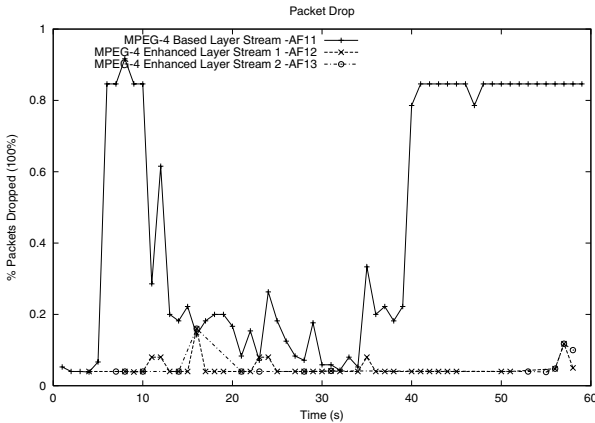


Fig. 11. Packet drops in Best Effort scenario. - Network Load > 95%

With IP Diffserv, we can see that the video packet losses are almost equals to 0, and we have ensured that the base layer has no losses. Figures 12 and 13 illustrate this fact.

5.2 End-to-End Transmission Delay

Figures 14 and 15 illustrate the end-to-end delay, which is an important component of the user-perceived QoS. Since, we are dealing with real time information, two much delayed audiovisual IP packets are simply discarded by the destination.

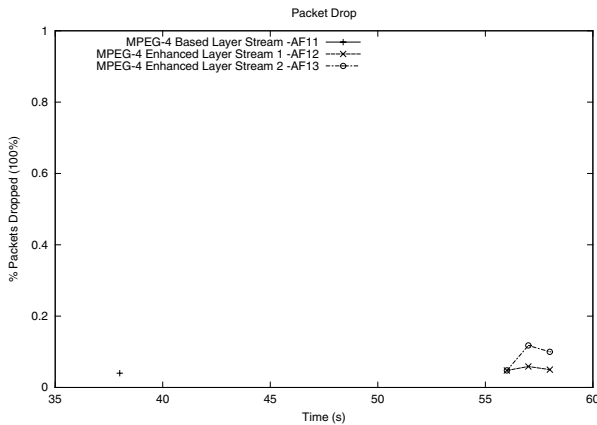


Fig. 12. Packet drops in IP Diffserv.- Network Load 80% -

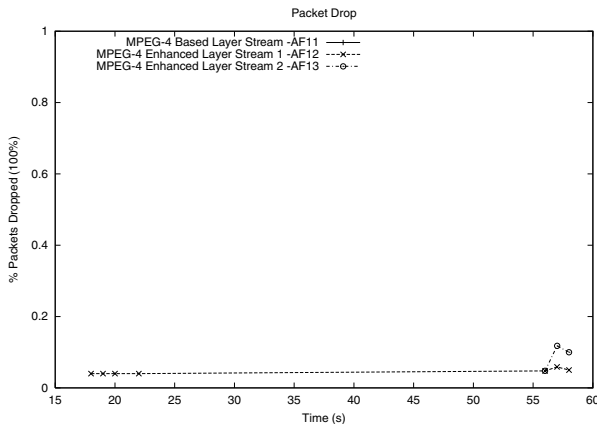


Fig. 13. Packet drops with IP Diffserv.- Network Load >95% -

We can notice that the video packet's losses are the similar regardless the network load, i.e. 80% or 95% of the bandwidth. It indicated that traffic load variation have no deep effect upon the video traffic. This is simply due to the static priority-based packet scheduling mechanism performed by the gateways. We also note that best effort traffic class (background traffic) can dynamically use any available bandwidth.

Figure 14 shows that during the peak rate, the delay dramatically increases. When the network load is about 95%, the one-way delay measurement is the highest, about 150 ms (Figure 15).

When using the Diffserv scenario, the packet delay is decreasing and doesn't really increase when the network load reaches 95%. In both Figures 16 and 17 we can also see that the highest delay is during the peak rate at the middle of the transmission process; i.e. 116 ms.

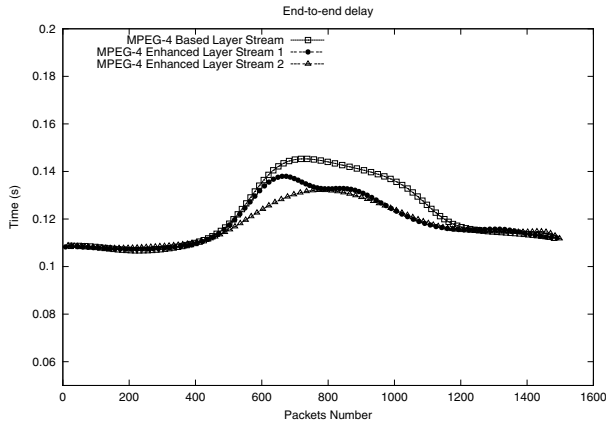


Fig. 14. End-to-end transfer delay with IP Best Effort- Network Load of 80% -

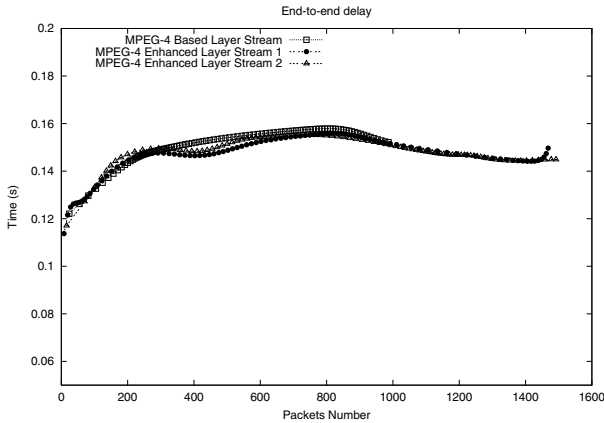


Fig. 15. End-to-end Transfer Delay with IP Best Effort - Network Load > 95% -

6 Conclusion and Future Work

In this article, a hierarchically encoded MPEG-4 interactive video on demand (VOD) service over IP Diffserv networks is proposed and evaluated using an experimental testbed. An IP packet marker, called DVMA, has been also designed and integrated in this QoS effective IP video delivery framework. Performance results have shown that two sensitive QoS parameters have been sensibly improved during network overload: video packet loss and end-to-end video packet transmission delay.

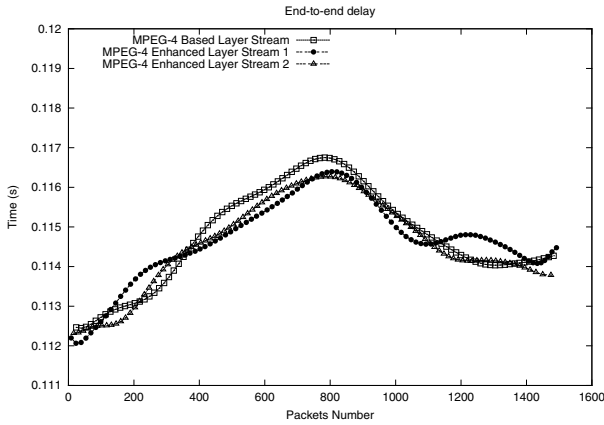


Fig. 16. End-to-end Transfer Delay with IP Diffserv - Network Load of 80% -

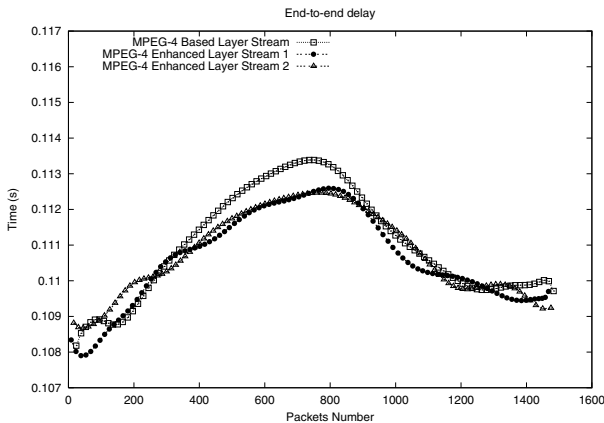


Fig. 17. End-to-end Transfer Delay with IP Diffserv - Network Load >95% -

The proposed marking algorithm better takes into account the characteristics and relevance of MPEG-4 sub streams (audio, video, BIFS, OD signalling...) and performs well with Assured Forwarding IP Diffserv PHB. Consequently, sensitive video streams will undergo a privileged processing by the routers using our proposed transport delivery service.

Further work will examine the interaction between video stream and background traffic. As mentioned, the background traffic in our testbed is marked as best effort traffic. This is not realistic in next generation Internet services. Additionally, domain's administrators should configure edges and cores routers according to predefined high-level resource management policies and Service Level Agreements (SLA). It is

commonly accepted that cooperative Bandwidth Brokers communicating with the Common Open Policy Service Protocol (COPS) will probably assist them [27]. In this perspective, we will investigate the performance of such architecture and will propose solutions for dynamic configurations of multimedia gateways.

References

1. H. Schulzrinne, S. Casner, R. Frederick, V. Jacobson: RFC1889 RTP - A Transport Protocol for Real-Time Applications. (January 1996)
2. Jitae Shin, JongWon Kim and C.-C. Jay Kuo: Content-Based Packet Video Forwarding Mechanism in Differentiated Service Networks. IEEE Packet Video Workshop 00 (May 2000)
3. M. Albrecht, M. Köster, P. Martini, M. Frank: End-to-end QoS Management for Delay-sensitive Scalable Multimedia Streams over DiffServ. Proc. of the 25th Annual Conference on Local Computer Networks, LCN'00, Tampa-FL. (November 2000) 314-323
4. Huai-Rong, Wenwu, and Ya-Qin Zhang: Scalable Object-Based Video Multicasting Over The Internet. International Conference on Image Processing, Vancouver, Canada. (September 2000)
5. J. Wroclawski: RFC2210 – The Use of RSVP with IETF Integrated Services. (September 1997)
6. J. Wroclawski MIT: RFC2211 – Specification of the Controlled-Load Network Element Service. (September 1997)
7. S. Shenker, C. Partridge, R. Guerin: RFC 2212- Specification of Guaranteed Quality of Service. (September 1997)
8. R. Braden, L. Zhang, Berson, S. Herzog, S. Jamin: RFC 2205- Resource ReSerVation Protocol (RSVP) -- Version 1 Functional Specification. (September 1997)
9. S. Blake, D. Black M. Carlson, E. Davies, Z. Wang, W. Weiss: RFC 2475- An Architecture for Differentiated Services (December 1998)
10. K. Nichols, S. Blake, F. Baker, D. Black: RFC 2474- Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers (December 1998)
11. V. Jacobson, K. Nichols, K. Poduri: RFC-2598 An Expedited Forwarding PHB. (June 1999)
12. J. Heinanen, F. Baker, W. Weiss, J. Wroclawski : RFC 2597- Assured Forwarding PHB Group. (June 1999)
13. ISO/IEC 14496-1: Coding of audio-visual objects – Part 1: Systems -final committee draft. (May 1998)
14. ISO/IEC 14496-2: Coding of audio-visual objects – Part 2: Visual -final committee draft. (May 1998)
15. ISO/IEC 14496-3: Coding of audio-visual objects – Part 3: Audio -final committee draft. (May 1998)
16. ISO/IEC 14496-3: Coding of audio-visual objects – Part 6: Delivery Multimedia Integration Framework (DMIF) final committee draft (May 1998)
17. Werner Almesberger: Differentiated Services on Linux Home Page <http://icawww1.epfl.ch/linux-diffserv/>
18. Werner Almesberger, Jamal Hadi Salim, Alexey Kuznetsov: Differentiated Services on Linux. Work in progress. (June 1999)
19. Sun Microsystems: Java Media Framework API <http://java.sun.com/products/java-media/jmf/index.html> (2000)

20. T. Ahmed. MPEG-4 DMIF Implementation in Java Technical report, Master of CS, University of Versailles, France. Available at : <http://www.prism.uvsq.fr/~tad/publication> (2000)
21. T. Ahmed, G. Buridant, A. Mehaoua: Encapsulation and Marking of MPEG-4 Video over IP Differentiated Services. In proceeding of Sixth IEEE ISCC 01. Hammamet Tunisia (July 2001) 346-352
22. W. Fang, Seddigh, B. Nandy: RFC2859 - A Time Sliding Window Three Colour Marker (TSWTCM). (June 2000)
23. J. Heinanen, R. Guerin: RFC2698 – A Two Rate Three Color Marker (TRTCM). (September 1999)
24. Floyd, S., Jacobson, V.: Link-sharing and Resource Management Models for Packet Networks. IEEE/ACM Transactions on Networking, Vol. 3, No. 4, pp., (August 1995) 365-386
25. David D. Clark and Wenjia Fang: Explicit Allocation of Best Effort Packet Delivery Service. ACM Transactions on Networking (August 1998) 362-373
26. Sally Floyd and Van Jacobson: Random early detection gateways for congestion avoidance, IEEE/ACM Transactions on Networking. Vol. 1, no. 4, (August 1993) 397-413
27. D. Durham, J. Boyle, R. Cohen, S. Herzog, R. Rajan, A. Sastry: RFC 2748 - The COPS (Common Open Policy Service) Protocol. (January 2000)
28. H. Schulzrinne, A. Rao, and R. Lanphier:): IETF RFC 2326 RTSP - Real-Time Streaming Protocol (April 1998)
29. H. Schulzrinne: A Comprehensive Multimedia Control Architecture for the Internet. In Proc. IEEE 7th Workshop on Network and Operating System Support for Digital Audio and Video (May 1997) 65–76.

Video Skimming and Summarization Based on Principal Component Analysis

Dan Lelescu¹ and Dan Schonfeld²

¹ Compression Science, Inc.,
901 Campisi Way, Campbell, Ca., U.S.A.
dan@compressionscience.com

² University of Illinois at Chicago,
851 S. Morgan St., Chicago, IL., U.S.A.
ds@eecs.uic.edu

Abstract. An increasing number of applications such as content-based multimedia retrieval in a distributed system and low-bitrate video communications, require the efficient processing and transmission of video information. In content-based video retrieval, video segmentation produces video shots characterized by a certain degree of visual cohesiveness. The number of relevant video shots returned by the system can be very large, thereby requiring significant transmission bandwidth. In this paper, we present a new algorithm for the representation of visual information contained in video segments. The approach is based on Principal Component Analysis and takes advantage of the characteristics of the data in video shots, and the optimal energy compaction properties of the transform. The algorithm can use additional information about video sequences provided by a video analysis and retrieval system, such as a visual change estimator, and a video object tracking module.

1 Introduction

In this paper, we present a video representation approach that utilizes a structuring of the video data and determines the optimal energy compaction of the resulting video entities in order to enable their efficient representation and compression. Although the emphasis of the paper is placed on the video retrieval context, the extension of the approach to the video compression context is also discussed.

Content-based multimedia processing and retrieval systems are created to offer the means to automate the selective access to information, as expressed in a query. Ideally, the retrieval system should find and return to the user only the information that is relevant for a particular query. The information obtained as a result of processing video shots, such as key frames, video objects, trajectories, can be used for immediate response to a query, or it can be indexed for later retrieval. The detail level at which the user initially receives visual information from the processing system as a result of a query is hierarchical in that the user may be given one or multiple key frames per relevant video shot, a text

description of the video shot contents, or the full-resolution shot. Text or key frames are limited in their capacity to provide dynamic information about a video shot. For a large video archive the number of video shots returned by the system may be significant. A full resolution video shot might have to be separately decoded and then re-encoded by itself for transmission. An alternative for providing the user with sufficient video information to allow for a decision about the full resolution video shot is to efficiently transmit a lower resolution representation of it.

In this paper, we discuss a method for the efficient representation of video shots. This approach can be applied hierarchically both in terms of resolution, and scene contents (i.e., full frame, or objects). For this purpose, we utilize the optimality properties of the Principal Component Analysis transformation in terms of energy compaction. Also, information provided by other components of the multimedia processing system, such as the video segmentation and video tracking modules, is used to facilitate the operation of the PCA-based algorithms at the corresponding hierarchical levels mentioned above. We propose an approach that utilizes the notion of visual activity in video sequences to facilitate important savings in the number of transform coefficients required to represent each video frame, while maintaining a reasonable level of distortion of the reconstructed video sequence. The information about the activity in a video segment is provided by a statistical estimator of visual changes in the video scene. The granularity of the video segmentation process can be adjusted to ensure a sufficient degree of visual consistency and locality in video shots. Given this new structuring of video information, a local PCA-based representation at the level of video shots becomes very efficient. This representation can be applied at the full frame level, or at the video object level. For the latter case, moving video objects can be extracted and tracked through the video sequence, for the purpose of enabling their PCA representation and separate encoding.

The PCA-based video representation approach presented can also represent a basis for developing video compression algorithms. The temporal segmentation of the original image sequence is achieved through the use of video segmentation, to obtain the video segments which are processed by the algorithm. The same video segmentation approach mentioned above can be applied for full or lower-resolution image sequences. The determined video segments can be encoded at various resolution levels, taking into account the increasing computation that comes with increasing image resolution. The hierarchical operation of the algorithm in terms of scene contents is also applicable through the separate encoding of video objects. This can be made possible by object recognition and tracking in the image sequence. Also, in this paper, we investigate the PCA-based transformation of the video data for the purpose of compressed representation. The issues of optimal quantization and entropy coding are beyond the scope of this presentation.

The paper is organized as follows. Section 2 reviews related work. In Section 3, algorithms for PCA are discussed. Section 4 presents the PCA-based repre-

sentation of video sequences. Section 5 contains simulation results. The paper is concluded in Section 6.

2 Related Work

Content-based multimedia processing, retrieval and transmission naturally shares topics with the low bit rate video compression area. Content-based video retrieval and indexing [1], [2], [3], [4], has the potential to automate to a large degree the multimedia retrieval process. The objective of such systems is to provide selective access to multimedia data, by finding query-relevant video segments and presenting them to the user. Once video segments have been found as relevant, various levels of information regarding the shots must be transmitted to the user. This information can include a text description of the shot, one keyframe per shot, visual storyboards consisting of multiple keyframes, time-compressed video shots, and low or full resolution versions of the video shot. Dynamic video information is the most preferable to be transmitted to the user.

Next, video coding techniques are briefly discussed. Waveform coding methods (i.e., transform and subband coding) encode data using the pixel structure of an image, whether at full frame level, or block level. Although these methods are well-established at high bit rate encoding, they have known limitations at low bit rates and introduce artifacts to which the human visual system is very sensitive. Block-based transform techniques have been included in video compression standards such as the MPEG family and the H.26x. Subband coding [5],[6], utilizes a decomposition of the image as produced by an analysis filter bank which is followed by down-sampling. The resulting subbands can represent the input to other stages of analysis/downsampling, producing a hierarchical, multi-resolution description of the image data.

Second generation (object level) coding techniques utilize the scene contents to improve the low bit rate coding performance and eliminate artifacts of the waveform coding techniques. However, contour and texture may require complex encoding operations and a significant bit rate allocation. These techniques attempt to provide better performance by considering the visual scene contents and its structure, as opposed to an artificial partition of an image into blocks. Object-based coding [7], [8], depends on the ability to recognize objects in the scene and represent them by shape, texture, and motion. The extraction of objects from the video scene is difficult and currently can be achieved satisfactorily if there are not many moving objects in the scene, the object motion is dominant and moderate, and camera motion is limited. The possibility that object/region based encoding may fail for particular types of video sequences or portions of a video sequence has prompted the creation of hybrid coders that use block based coding when the object based techniques become unfeasible.

The use of PCA for optimal dimension reduction and transform coding for still images has received renewed attention [9], [10]. In [9], the resource allocation using local linear models for non-linear PCA is discussed. In the local PCA model, the data is partitioned into regions and PCA is performed in each

region. The local PCA model is applied to image dimension reduction and transform coding. The allocation of PCA representation and coding to different image regions permits the adjustment of the dimension locally, while the average dimension is constrained to a particular value. In [10], the problem of finding an appropriate data partition for the application of the local PCA is investigated. In this paper, we use the PCA transform as the basis for encoding video data. We introduce a video sequence representation approach that can facilitate important savings in the number of transform coefficients required to represent each video frame. Using a visual activity-based structuring of video sequences and video object tracking in scenes, a PCA-based representation becomes suitable for use in encoding video segments, with additional computational methods to make such algorithms efficient.

3 Algorithms for Efficient Principal Component Analysis

Let \mathbf{P} be an $N_c \times L$ data matrix corresponding to a set of L data vectors with dimension $N_c \times 1$. In the following, we use deterministic estimates of statistical variables, by taking the matrix $\mathbf{C} = \mathbf{P}\mathbf{P}^T$ to be an estimate of the correlation matrix of the data. Let X_k denote a data vector (column) of matrix \mathbf{P} . The Principal Component Analysis (PCA) performs a partial KL transform by finding the largest $M < L$ eigenvalues and corresponding eigenvectors of \mathbf{C} . The new representation Y_k of an original data vector X_k is $Y_k = \Phi_M^T X_k$, where Φ_M is the eigenmatrix formed by selecting only the M eigenvectors corresponding to the largest M eigenvalues.

Assuming that $N_c \gg L$ is very large for the case of interest, the size of matrix \mathbf{C} is also large, which would result in computationally intensive operations using the direct PCA computation. As described in [11], an efficient approach for negotiating this problem is to consider the implicit correlation matrix $\tilde{\mathbf{C}} = \mathbf{P}^T \mathbf{P}$. The matrix $\tilde{\mathbf{C}}$ is of size $L \times L$, which is much smaller than the size of \mathbf{C} . The $M \leq L - 1$ largest eigenvalues λ_i , and corresponding eigenvectors e_i of \mathbf{C} can be *exactly* found from the $M \leq L - 1$ largest eigenvalues and eigenvectors of $\tilde{\mathbf{C}}$ as follows [11]: $\lambda_i = \tilde{\lambda}_i$, $e_i = \tilde{\lambda}_i^{-\frac{1}{2}} \mathbf{P} \tilde{e}_i$, where $\tilde{\lambda}_i$, \tilde{e}_i are the corresponding eigenvalues and eigenvectors of $\tilde{\mathbf{C}}$. The eigenvectors \tilde{e}_i of $\tilde{\mathbf{C}} = \mathbf{P}^T \mathbf{P}$ are given by the right singular vectors of \mathbf{P} , determined using the SVD.

An efficient iterative approach for computing *approximations* of the M largest eigenvalues and corresponding eigenvectors of \mathbf{C} was also proposed in [11]. It is assumed that the data vectors are processed sequentially. The algorithm is initialized by direct computation of the M most significant eigenvectors of an initial set of $(M+1)$ data vectors. In the following iterative procedure only the M eigenvectors corresponding to the largest eigenvalues are retained at every stage of the iteration. For every new input vector, the M eigenvectors computed in the previous step are refined. Let us denote by $\{x^{(i)}\}_{i=1\dots L}$ the L data vectors. Assume that only the M most significant eigenvectors $\{\psi_L^{(i)}\}_{i=1\dots M}$, obtained after all L data vectors have been processed, will be retained. Let \mathbf{A}_k be an $(M+1) \times (M+1)$ correlation matrix formed at iteration k . Its corresponding

eigenvectors and eigenvalues are $\{a_k^{(i)}\}_{i=1\dots M}$ and $\{\lambda_k^{(i)}\}_{i=1\dots M}$. The iterative PCA algorithm proceeds as follows:

Step 0:

Set $k = M + 1$.

Read the $(M+1)$ data vectors $\{x^{(i)}\}_{i=1\dots M+1}$.

Determine the matrix \mathbf{A}_k .

Calculate eigenvectors $\{a_k^{(i)}\}_{i=1\dots M}$ and $\{\lambda_k^{(i)}\}_{i=1\dots M}$ by direct calculation.

Compute the initial set of M eigenvectors $\{\psi_k^{(i)}\}_{i=1\dots M}$ as follows:

$$\psi_{M+1}^{(i)} = (a_{M+1}^{(i)})_1 x^{(1)} + (a_{M+1}^{(i)})_2 x^{(2)} + \dots + (a_{M+1}^{(i)})_{M+1} x^{(M+1)}. \quad (1)$$

where $(a_k^{(i)})_I$ is the I^{th} component of the i^{th} eigenvector $a_k^{(i)}$, at iteration k .

Step 1:

$k = k + 1$.

As the new vector $x^{(k)}$ is processed, re-compute the $(M+1) \times (M+1)$ matrix \mathbf{A}_k from $x^{(k)}$ and the previous principal vectors $\{\psi_k^{(i)}\}_{i=1\dots M}$ as follows ($(\mathbf{A}_k)_{i,j}$ represents the $(i,j)^{th}$ entry in matrix \mathbf{A}_k):

$$\begin{aligned} (\mathbf{A}_k)_{i,j} &= \frac{k-1}{k} (\lambda_{k-1}^{(i)} \lambda_{k-1}^{(j)})^{1/2} \delta_{ij}; i, j = 1 \dots M, \\ (\mathbf{A}_k)_{i,M+1} &= (\mathbf{A}_k)_{M+1,i} = \frac{1}{k} \psi_{k-1}^{(i)T} x^{(k)}; i = 1 \dots M, \\ (\mathbf{A}_k)_{M+1,M+1} &= \frac{1}{k} x^{(k)T} x^{(k)}. \end{aligned} \quad (2)$$

The updated principal eigenvectors $\{\psi_k^{(i)}\}_{i=1,M}$ and their corresponding eigenvalues $\{\lambda_k^{(i)}\}_{i=1\dots M}$ are then obtained by finding the eigenvalues and eigenvectors of matrix \mathbf{A}_k and using the following formula:

$$\psi_k^{(i)} = (a_k^{(i)})_1 \psi_{k-1}^{(1)} + (a_k^{(i)})_2 \psi_{k-1}^{(2)} + \dots + (a_k^{(i)})_M \psi_{k-1}^{(M)} + (a_k^{(i)})_{M+1} x^{(k)}. \quad (3)$$

Step 2:

Repeat Step 1 until $k = L$ (all vectors have been processed).

At $k = L$, normalize the M retained eigenvectors $\{\psi_L^{(i)}\}_{i=1\dots M}$ by $1/(\sqrt{L\lambda_L^{(i)}})$ to obtain the normalized principal vectors of the data set $\{x^{(k)}\}_{k=1\dots L}$.

4 PCA-Based Representation of Video Sequences

The video representation and compression algorithms presented in this section are intended for use in two contexts that require the efficient coding of video sequences. In the case of a distributed multimedia processing and retrieval system, the query-relevant video shots must be transmitted to the user's location. Within computational considerations, the proposed compression approach can represent the basis for encoding video data at very low bit rates. In both cases, the use of video segmentation, visual change estimation, and object tracking,

enables the operation of the PCA-based video representation algorithm. A video segmentation algorithm can be applied to compressed or raw video data. The object extraction and tracking information, however, is obtained differently depending on the context. For example, for motion-compensated video, objects can be extracted and tracked as presented in [3].

4.1 Video Shot Representation in a Multimedia Retrieval System

In the case of a distributed multimedia processing and retrieval system, the query-relevant video shots must be efficiently transmitted to the user. As mentioned before, the information presented to the user can be hierarchical in nature (a single key frame, multiple key frames, story boards (tree structures) of key frames, lower-resolution versions of the video shot, time-compressed video shots, or the full-resolution video shots).

We shall illustrate the operation and characteristics of the proposed video representation algorithm for the case of a video shot extracted from a compressed MPEG-2 video sequence. The preliminary operations of video segmentation and video object tracking are assumed to have already taken place. Thus, we have access to the video shot of interest in compressed form. Additionally, as produced by the other modules of the retrieval system operating on compressed bitstream ([3], [4]), there exists information regarding the visual changes in the video shot, and objects of interest in the shot. To introduce the algorithm, its functionality is presented for the case of low resolution, frame-level operation. For efficiency purposes, a low resolution version of the original video shot can be directly extracted from the bistream by considering only the DC coefficients of the blocks in each picture of the shot. One of the important characteristics of video shots is that they are relatively visually-coherent, that is, the visual information in the scene is expected to vary within certain limits.

For the case considered, the video segment to be encoded consists of a sequence of DC images. Each of these images are lexicographically ordered, resulting in the corresponding sequence of DC vectors. Let us denote these vectors by $\{X_k\}$, having dimensionality $N_c \times 1$. Let us also assume that the number of vectors in the video segment is L . We are interested in approximating the original vector space by a much smaller number M of eigenvectors. In order to use a PCA-based representation of the video segment, a number of options are available. The PCA-based representation algorithms can use the *training sample* approach, or the *iterative* approach, as described in Sect. 3.

For the training sample PCA approach (TS-PCA), a training subset τ of sample vectors taken from the entire vector set can be used. The M largest eigenvalues and the corresponding eigenvectors of τ can be found using the procedure outlined in Sect. 3. The retained eigenvectors can be utilized to represent the original DC vectors X_k in the video segment by the corresponding transformed vectors Y_k of dimensionality $M \times 1$:

$$Y_k = \Phi_M^T X_k. \quad (4)$$

where Φ_M is the corresponding eigenmatrix.

Additionally, two options are available for the selection of the training set τ . The first one is to select this set randomly from the vectors forming the video segment. The second option is to use information provided by modules of the retrieval system. Specifically, one by-product of the video segmentation approach presented in [4] is related to the fact that the activity or visual changes in a video shot are indicated by the test statistic g_k that is used for the detection of scene changes. The test statistic can be used as an estimator of activity in the video shot in order to improve the performance of the PCA representation, by means of selecting a training sample from the shot. The samples can be assigned in a non-uniform manner, by taking more samples in portions of high activity in the shot. The simulations conducted show that the performance of the TS-PCA algorithm is dependent on the degree of visual change that takes place in the video shot. Also, as expected, taking a training sample that spans the entire video segment results in improved performance compared to selecting the training set only from the first images in the shot. The alternative to using a training sample for the PCA video representation, is to use the iterative PCA algorithm (I-PCA) presented in Sect. 3. Because this approach uses all the data vectors in the set in determining their PCA representation, the corresponding algorithm is expected to provide an improvement in the quality of the representation. This is confirmed by the simulations presented in Sect. 5.

For the lossy reconstruction of the images in the video shot, the information needed consists of the eigenspace description as provided by the retained eigenvectors in matrix Φ_M , and the coordinates of each image in this space, represented by its corresponding vector Y_k . Formally, using the orthogonality of the transform, a reconstructed vector (image) is obtained as follows:

$$\hat{X}_k = \Phi_M Y_k. \quad (5)$$

Up until now, we have considered the case of applying the PCA-based representation algorithm for low-resolution video shots and at the frame level. Its functionality can be extended to a hierarchical level based on video scene contents. In general, tracking can provide us with information about the position and size a specific object through the frames of the shot. This information can be used to extract query-relevant object 'streams' in a video shot, by recording the images of the object as it appears in each frame of the shot. Furthermore, this enables the diversification of video representation and encoding at both full image and object levels. Simulations conducted show that the separate PCA-representation of moving objects in the scene results in superior quality of the reconstructed object images, compared to the exclusive frame-level PCA representation, for small to moderate affine changes of the object inside the tracking rectangle.

4.2 Video Compression

The functionality of the PCA-based video representation and compression can constitute a basis for developing algorithms for encoding raw video data. The

original video data can be assigned new structure through the use of video segmentation, change estimators, and video object tracking.

The video segmentation approach can be applied to images at a given resolution. For example, a subband transform can be used for generating a multi-resolution representation of the original images. As we know, low-resolution images are sufficient for the detection of scene changes in the video. The resulting video segments are represented using the PCA algorithm with a similar technique to that described in Sect. 3. This is shown in the block diagram in Fig. 1. From a practical point of view, two challenges arise. One is related to the fact that as the resolution of the images increases, the visible distortion introduced by the PCA representation can increase for video sequences with large motion in the scene. The locality of the data has to be ensured by selecting the granularity (threshold) of the video segmentation process. Also, for video frames at full resolution the dimensionality increases, and therefore the computation cost associated with the PCA algorithm increases as well. An alternative to the PCA

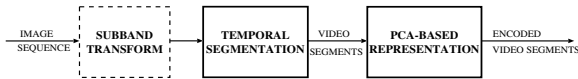


Fig. 1. Hierarchical video compression for different image resolutions.

representation of the actual image sequence is to encode the sequence of corresponding error images. If the image with index k in the sequence is denoted by X_k , then the error image E_k is obtained as $E_k = X_k - X_{k-1}$. The vectors corresponding to the sequence E_k are represented by the PCA algorithm discussed in this section. The most significant M eigenvectors of the residual sequence and the PCA representation of each residual image, along with the first image in the video sequence must be encoded and transmitted. Through the inverse transformation, the residual images are reconstructed, and using the first image for the initial step, all other frames of the video segment are obtained, i.e., $\hat{X}_k = \hat{X}_{k-1} + \hat{E}_k$. One additional area that is currently investigated consists of the use of a PCA-based approach in conjunction with motion compensation (MC). For example, the motion fields corresponding to frames from a video shot can be modeled and transmitted using a PCA-based technique.

The PCA-based video representation can also be applied hierarchically based on video scene contents, similarly to the case of compressed content-based processing. Tracking allows for the registration of the objects inside the tracking rectangle and contributes to the locality of video data at the object level. The objects are extracted and tracked inside a determined video segment, obtained through the prior video segmentation of the image sequence. The PCA transform can be applied to the sequence of object images. For video object tracking in video segments two options are available. Firstly, objects can be extracted and tracked through the frames of the raw video sequence. The images of the

objects as delimited by the tracking bounding box form the object streams that are processed by the PCA representation algorithm, as shown in Fig. 2.

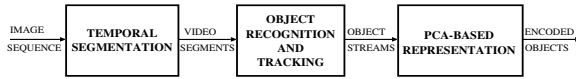


Fig. 2. Object level video compression.

Secondly, block-level motion compensation can be applied to the original image sequence so that the object recognition and tracking can use the generated motion information. This information can be used to extract the object streams in a video segment, composed of the images of the tracked object. This process is illustrated in Fig. 3.

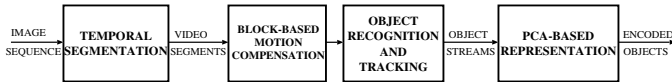


Fig. 3. Object level video compression on motion compensated video sequence.

One additional observation is that whenever object tracking is used for separate encoding, the resulting sequence of object images through the frames of the video segment may have variable size due to variation in the object size. If this variation is small and the tracking frame is fixed in size, the object bounded by its tracking frame can be directly processed by the PCA. In the case where there exists variation in the size of the tracking frame, and thus in that of the object images, additional processing is required to prepare the resulting images for the input to the PCA-representation module. If the variation is relatively small the largest size of the tracked object can be used to determine the dimensionality of the data vectors to be processed by the PCA.

For all cases presented, the PCA-based compression of video sequences offers the advantage of a high compression ratio due to the optimality of energy compaction for a given number of transform coefficients. The effectiveness of this approach is dependent on the structure (locality) of video data obtained as a result of temporal segmentation and object tracking, as well as on the image resolution at which the algorithm operates.

5 Simulation Results

The original video sequences utilized in the simulations were encoded using the MPEG-2 video compression standard. The size of the video frame was 240 x 352 pixels. The encoding pattern was 'IPBBP', with a GOP of length 12.

The objective of simulations of the PCA-based video representation was to evaluate the allocation of transform coefficients for each frame of the video segments determined as discussed in the previous Sections, and the objective quality of the reconstructed frames of the video sequences. The use of PCA representation in a motion compensation context is part of on-going work. Following the presentation in Sect. 4, a low resolution version of video segments was used for simulations. For simplicity, only the luminance part of the image data was retained. These low resolution video sequences were created by extracting the DC coefficients of blocks from the compressed video frames. Thus, the dimensionality of the resulting DC vectors is 1320×1 (corresponding to DC images of size 30×44). As described in Sect. 3 and 4, the original set of DC vectors comprising the low-resolution video shot is processed by the PCA-based algorithm. The DC images can also be seen as a raw image sequence that are subject to a transform. The video shots utilized fall into three categories ranging from low (shot 1), moderate (shot 2), and high activity (shot 3).

Each DC image in the video shot was also represented using both frame level and block-level DCT transforms. A subset of the DCT coefficients of each DC image was retained either at the frame or block level. For example, for frame level DCT, there are 1320 DCT coefficients corresponding to a 30×44 DC image. A subset of these coefficients can be utilized for a lossy representation of the image, e.g., 900 coefficients. Similarly, at block DCT level, if blocks are taken to be of size 4×4 , there are 16 DCT coefficients per block, of each one could retain only the first 9 in plane order. Blocks of size 4×4 yield the best representation performance for DC images (the spatial correlation present in the full resolution images is diminished in their DC representation). The lossy representations of each image in the video shot are then used for its reconstruction, through the inverse transformation.

In the first part of the simulations we use the PCA-based representation of the DC images in a video segment by utilizing a training set of samples. The information needed for reconstruction of the images comprises the M eigenvectors of the representation space, and the M -dimensional representation (coordinates) of each DC image in the determined space. As presented in Sect. 3, 4, a training sample of $M + 1$ DC vectors in each video segment must be selected to enable the TS-PCA representation. If additional information is available about the visual changes taking place in a video shot such as given by a statistical change estimator, the sample can be selected accordingly (see Sect. 4). Once the PCA-representation of the video shot is obtained, the images in the video shot are reconstructed as presented in Sect. 3.

Figs. 4 - 5 show the SNRs of reconstructed DC images in the video shots considered. These images were encoded using a TS-PCA-based representation, and lossy frame and block DCT-based representations of each DC image (identified by the abbreviations DCT and BDCT). The number of coefficients retained for each representation, as well as the total number of coefficients per frame are also indicated in the figures.

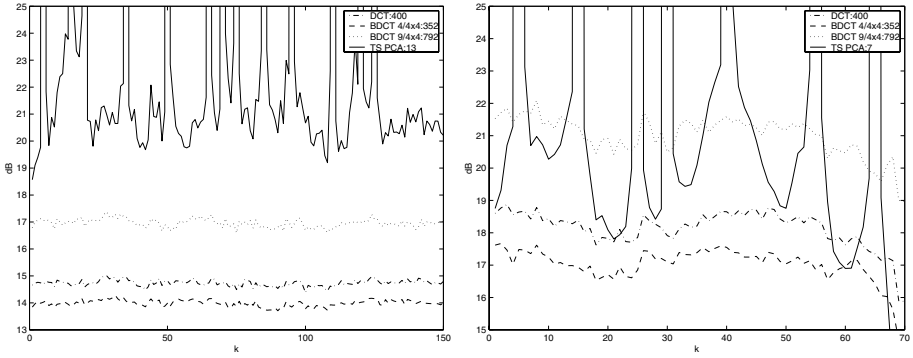


Fig. 4. Reconstructed sequence SNR: Training set PCA - Video shot a) 1, b) 2.

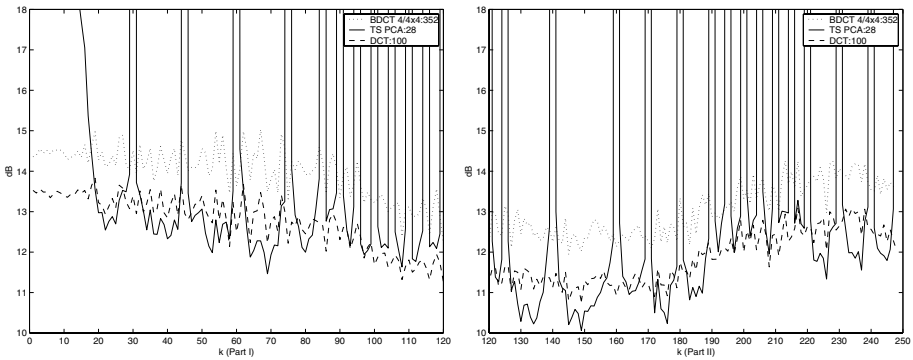


Fig. 5. Reconstructed sequence SNR: Training set PCA - Video shot 3.

A characteristic of the training sample-based PCA (TS-PCA) is that the images that were part of the training sample used for the representation are perfectly reconstructed, and thus are present in the figures as spikes in the SNR. For the video shots with low to moderate visual activity, the performance of the PCA representation is comparable to that of the DCT-based representations using a much larger number of coefficients, as seen in Fig. 4. The video shot in Fig. 5 is more difficult to represent for a TS-PCA-based representation (especially if processed as one shot), due to significant and continuous changes in the visual information. Two approaches can be used to improve performance. The most effective remedy is provided by the appropriate selection of the video segments to be encoded, i.e., the original video shot should be split into two sub-shots which are much more consistent visually. The second method to improve representation performance is to increase the dimensionality of the subspace, with the cost of increasing the number of eigenvectors that have to be transmitted.

An alternative to the training sample methodology is made computationally efficient by the iterative algorithm discussed in Sect. 3. An *initial* estimate of the representation space having the desired dimensionality M is obtained by

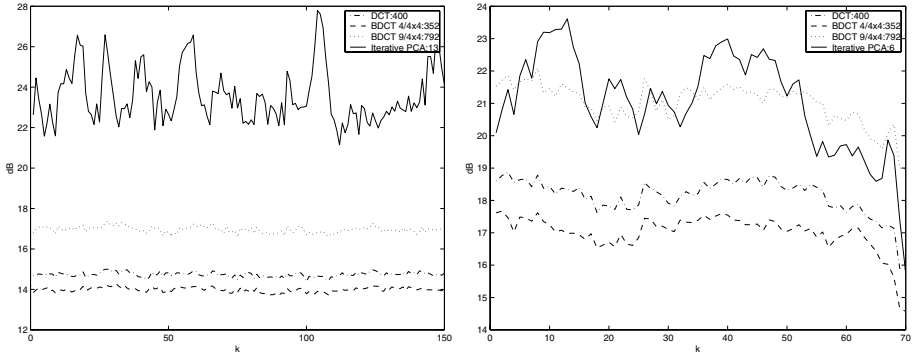


Fig. 6. Reconstructed sequence SNR: I-PCA - Video shot a) 1, b) 2.

computing the most significant M eigenvectors of $M + 1$ images from the video shot by the direct method presented in Sect. 3. Once the initial representation is determined, the algorithm refines the representation space iteratively, with each new image from the video sequence, until all images have been processed. At the end of its operation, the iterative PCA algorithm (I-PCA) produces approximations of the most significant M eigenvectors corresponding to the *entire set* of images.

The SNRs of reconstructed images corresponding to the PCA-based representation using the I-PCA algorithm and the DCT-based representation are shown in Figs. 6-7. The images used to compute an initial estimate of the space are selected to be the first ones in the video shot. Other choices would be a random selection, or a selection based on an activity estimator, similarly to the training sample methodology. As expected, there is a marked improvement in the reconstructed video quality in the case of I-PCA, reflecting a more precise description of the image sequence space. The performance of the representation is also improved for the third video shot in Fig. 7(a), although the previous considerations regarding additional gain that can be obtained through separate processing of its two component sub-shots remain valid. The performance of the I-PCA computed using a spread sample initialization instead of selecting the first frames in the shot for the initial estimation of the eigenspace, is shown in Fig. 7(b) for the last video shot considered. As expected in this case, the SNR is slightly improved compared to Fig. 7(a), using a selection of the sample that spans the entire video shot. For the case of the PCA-based encoding, similarly to intra-coded images in a motion compensation context, the M eigenvectors must be intra-coded for transmission. For all other images in the video sequence, only the $M \times 1$ -dimensional vector representing coordinates of the corresponding image in the subspace must be transmitted. For the I-PCA algorithm, M was chosen so as not to exceed the number I of intra-coded images needed for a motion compensated sequence.

As discussed in Sect. 4, by taking into account information regarding the scene contents, the proposed algorithm can be extended to operate at object

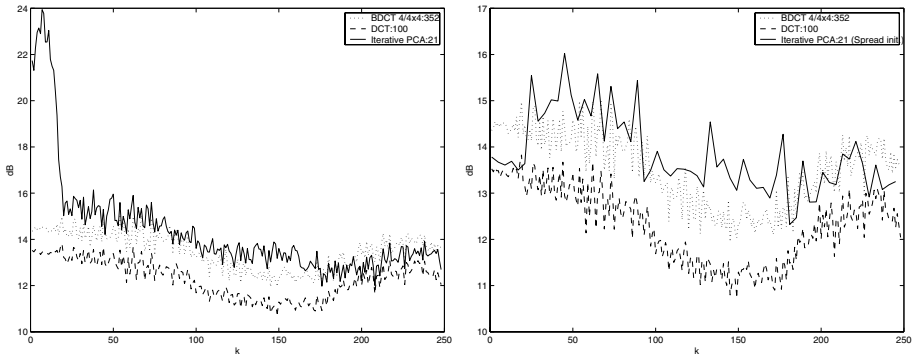


Fig. 7. Rec. SNR (Shot 3): a) I-PCA, b) spread sample I-PCA (at higher SNR resolution.)

level. Through extraction and tracking of moving objects, their PCA-based representation has better performance due to the virtual registration of the objects inside a tracking bounding box. In Fig. 8(a), sample frames show two objects marked by their corresponding tracking frames, which were tracked through the frames of a video shot by the algorithm presented in [3]. Thus, these objects can be extracted and encoded separately using the PCA-based algorithm. Sample original object images and their reconstructed counterparts are also shown in Fig. 8(b). Even though while being tracked the objects suffer a moderate degree of change inside the tracking box, their PCA-based representation and reconstruction is very good.

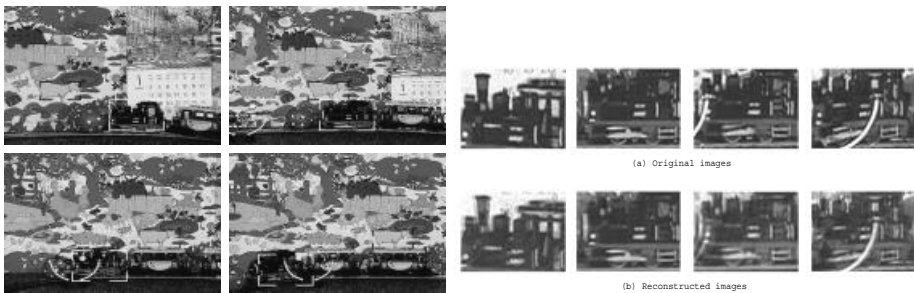


Fig. 8. Snapshots of video tracking for retrieval of multiple templates.

6 Conclusion

In this paper, a new algorithm for the efficient representation of video segments based on Principal Component Analysis was presented. This approach can be used in the context of content-based multimedia processing and retrieval, as well

as a basis for developing video data compression algorithms. For both application domains, the PCA algorithms can be hierarchically applied at both resolution and scene contents level. Simulations of the PCA algorithms show the potential for a very economical representation of video segments, using a small average number of transform coefficients for the images of the sequence. The performance of the approach can be maintained by a temporal segmentation of the video sequence. With increasing resolution of the image sequence, the computational cost of the algorithm increases as well. Depending on the nature of the video segment, a variable number of eigenvectors may be used for representation. It is of interest to determine the optimal number of eigenvectors to be used for each video segment subject to a constrained number of eigenvectors used for the entire video sequence. Also, a hybrid encoder can utilize different encoding methods (PCA-based, motion compensation) depending on the resolution of the image sequence, and degree of visual changes in a video segment.

References

1. Wactlar H., "Informedia-Search and Summarization in the Video Medium," *Proceedings of Imagina 2000 Conference*, Monaco, 2000.
2. Smith J.R., Chang S.F., "VisualSEEK: A Fully Automated Content-Based Image Query System," *Proceedings of ACM Multimedia '96*, ACM Press, Boston, November 1996.
3. Schonfeld D., and Lelescu D., "VORTEX: Video Retrieval and Tracking from Compressed Multimedia Databases-Multiple Object Tracking from MPEG-2 Bitstream," *Journal of Visual Communications and Image Representation (JVCIR'2000)*, 2000 Vol.11, No.2
4. Lelescu D., and Schonfeld D., "Real-Time Scene Change Detection on Compressed Multimedia Bitstream Using Statistical Sequential Analysis," *IEEE International Conference on Multimedia and Exposition (ICME2000)*, 2000 New York.
5. Vetterli M., "Multi-Dimensional Subband Coding: Some Theory and Algorithms," *Signal Processing*, 1984, Vol.6, No.2, pp.97-112.
6. Woods J.W., "Subband Image Coding," *Kluwer Academic Publishers*, Boston, 1991.
7. Salembier P., Marques F., and Gasull A., "Coding of Partition Sequences," *Video Coding: The Second Generation Approach (Torres L. and Kunt M., eds.) Kluwer Academic Publishers*, Boston, 1996.
8. Katsaggelos A. K., Kondi L.P., Meier F.W., Ostermann J., Schuster G.M., "MPEG-4 and Rate Distortion Based Shape-Coding Techniques," *IEEE Proceedings Special Issue on Multimedia Signal Processing*, 1998, Vol.86, No.6, pp.1126-1154.
9. Archer C., Leen T.K., "Optimal Dimension Reduction and Transform Coding with Mixture Principal Components," *Proceedings of the IEEE International Joint Conference on Neural Networks*, Washington, 1999.
10. Meinicke P., Ritter H., "Local PCA Learning with Resolution-Dependent Mixture of Gaussians," *Proceedings of 9th International Conference on Artificial Neural Networks (ICANN'99)*, 1999, Edinburgh, UK, pp 497-502.
11. Murakami H. and Kumar V., "Efficient Calculation of Primary Images from a Set of Images," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1982, Vol. PAMI-4, No. 5, pp. 511-515.

Distributed Resource Management to Support Distributed Application-Specific Quality of Service

Gary Molenkamp, Michael Katchabaw, Hanan Lutfiyya, and Michael Bauer

Department of Computer Science
The University of Western Ontario
London, Ontario, Canada N6A 5B7

Abstract. This paper presents a policy-based software architecture and implementation for application Quality of Service (QoS) management. The approach is dynamic in that the application is started with an initial resource allocation, and if it does not meet its QoS requirement, a resource manager attempts to allocate more resources to it until its QoS requirement is met. One key problem is that of determining the corrective action(s) to be taken when an application's QoS requirements are not being satisfied. For example, one possible action is to adjust the application's CPU priority when its QoS requirement is not being met. However, if there are several applications on the same host with QoS requirements, this may not be feasible. In other words, there may be constraints on the corrective actions depending on other factors. This paper explores the use of administrative policies to guide decisions on actions to take to meet QoS requirements and presents a prototype and initial experimentation.

Keywords: Dynamic Quality of Service, Policies, Control Actions, Multimedia

1 Introduction

There has been an increase in distributed applications that involve the exchange of data which is time-sensitive. These types of applications arise in a number of areas including video-on-demand, distance education, tele-medicine, tele-conferencing, electronic commerce, and several others [7]. 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.

QoS management techniques, such as resource reservation and admission control techniques, can be used to guarantee the QoS requirements, but these techniques usually base the resource reservation on worst-case needs which leads to inefficient resource utilization. In addition, these approaches often require that

the developer or user have intricate knowledge of resource needs (e.g., the number of CPU cycles needed). 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. Generally, the adjustments to resources refers to making adjustments to the behaviour of the service providing the resources (e.g., frame dropping will cause a server to back off). Thus, monitored resource usage is used to determine adaptations.

As will be described in detail in the next section, most of the related research is not suitable for environments and applications with the characteristics just described. (i) Often, the QoS requirements are considered to be *soft*, i.e., applications with these requirements are still considered functionally correct if the QoS requirement is not satisfied. This is opposed to *hard* QoS requirements where applications (e.g., flight control systems, chemical process control systems, patient-monitoring systems) must meet the QoS requirement to avoid disastrous consequences. (ii) Many distributed applications are dynamic in nature. For example, the QoS requirements may vary for different users (and different sessions) of the same application, for the same user of the application at different times, or even during single session of the application. Also, during a session the QoS requirements may change. For example, in a multicasting application it is not unusual to have new clients join a multicasting session. (iii) The user or developer will not have intricate knowledge of the hardware architecture and system software in the target environment. This is especially difficult to do for widely heterogeneous environments. (iv) It is also likely that the users of these applications will have multiple applications on their systems, likely running concurrently, and that these applications will also have to co-exist with more traditional applications for transaction processing, data processing, and software development.

This paper describes an approach to QoS management for an environment and applications with the above characteristics.

Section 2 describes related work. Section 3 describes the strategy taken by the our approach to QoS management. Section 4 provides a brief discussion of QoS requirements and policies. A proposed architecture for a system dynamically allocate resources to meet QoS requirements is presented in Section 5 and details of a prototype are presented in Section 6. Some experimental results are presented in Section 7. A conclusion is presented in Section 8.

2 Related Work

There is a large body of work related to QoS management that is briefly described in this section.

2.1 Single Resource Algorithms

Over the years, a number of algorithms have been developed for CPU, disk and network resource allocation (e.g. [20,3,22,16,15]). A single algorithm is not

enough to support QoS management. Users and developers should not interface with or be exposed to these algorithms and approaches directly. Our work addresses this issue by allowing the user and application developers to focus on defining the QoS requirements e.g., “The number of video frames per second displayed to the user must be at least 25”, but not having to define low-level resource needs such as the number of CPU cycles. Our QoS management approach hides these details.

2.2 Static QoS Infrastructures for Network Resources

There are a number of approaches for providing network quality of service for multimedia applications. An excellent review can be found in [1]. Most of these approaches are relatively static, although some support some form of renegotiation of the resources allocated to them for satisfaction of their QoS requirements. These approaches have two disadvantages: (i) They require major modifications to the operating systems or (ii) The technique is based on worse-case needs (as the result of using resource reservation) which often leads to inefficient resource utilization. Our work makes use of the existing features provided by the operating system and does not need to use resource reservation which is generally more applicable for systems with hard real-time requirements.

2.3 Dynamic QoS Management Approaches for Application Adaptation

One class of application adaptation mechanisms adapts the resource consumption needs. An example of this can be found in [9]. Another type of application adaptation mechanism alters the topology of the end-to-end processing. Yet another application mechanism allows the user to prioritise services. If there are not enough computing resources then the behaviour of the application changes by changing or eliminating a service. For example, an application may be told to reduce the resolution that it is displaying an image at or eliminate audio and leave video.

Many of the techniques are either used or modified for use in middleware services. For example, the work in [12] introduces a Task Control Model for QoS application adaptations in the Agilos middleware control architecture. The adaptations usually take the form of changing request or delivery rates in applications based on control theory models. The *QuO* project [17] proposes extensions to standard CORBA components and services for supporting application adaptation. Other work [6] includes that which combines resource reservation with application adaptation.

Our approach can support application adaptation (although we do not extensively focus on this in this paper). Our approach does not rely on existing middlewares like CORBA before it can be applied. In fact, our approach can be used to manage multiple and concurrent applications that use CORBA, Java or sockets. As a result, our work can be more easily extended to QoS management

across multiple administrative domains. This is not the case with using a system like QuO which assumes that the CORBA services are in all administrative domains.

2.4 Dynamic QoS Management Approaches for Dynamically Changing Resource Allocation

There is relatively little work on dynamic QoS management approaches for changing resource allocations of an application. The work in [2] proposes a quality of service management scheme that dynamically adjusts resource allocations (e.g. the operating system adjusts its computing resource allocations). The applications make most of their own resource management decisions, instead of the operating system or other services. While this may get better results for individual applications, it places additional burden on application developers to develop this logic, and can be dangerous since it relies upon all applications and users to behave in a fair and equitable manner when resources are being allocated. Unfortunately, the implementation and validation of this approach has been self-described as ad hoc and incomplete; further refinements of this work have not been done. Our work is different in that it will allow for the potential to optimise performance across all applications.

The work in [4,11] describe dynamic approaches to QoS management where the environment adapts as much as possible to application needs. The user is central to this process, as it is up to the user to decide when quality is unacceptable, and what to do to correct the problem, with the assistance of feedback from the system. This degree of user involvement places a great deal of burden on the user.

3 Strategy

Our approach to QoS management does not require user or developer knowledge of their application's computing resource needs. Computing resources are allocated as needed (if available), if the application does not satisfy its QoS requirements. For example, the CPU priority of an application may be adjusted if that application is not satisfying its QoS requirements. This is different than having the application re-adjust its behaviour. Rather, the computing resource allocation is dynamically changed.

This approach entails several problems: (1) The application needs to be monitored in order to determine if the QoS requirements are being met. If they are not, then this must be detected (*violation detection*). (2) Once a QoS requirement has been detected by the monitoring system, the cause of the degradation must be determined (*location*). For example, a violation of the QoS requirement would occur if a video application was expecting to receive at least 23 frames per second, but was not. This could be caused by several situations, e.g., the video application might not be getting enough local processor cycles, the server process might not be getting enough cycles, a process failed, or there is an unexpected

load on a network link. Locating the cause is an important step in determining the appropriate resource allocation. For example, if the problem is that there is an unexpected load on the network, then there is no need to adjust resources on the video application process's host. (3) The resource allocation should be adjusted. Examples include providing more processor cycles or restarting a failed process. The actions to be taken depend not only on the cause of the violation, but also depend on the constraints imposed on how to achieve the QoS requirement. For example, one possible corrective action is to adjust the CPU priority of the video application receiving the video stream. However, if there are several multimedia applications on the same host, then perhaps attaining the desired level of service for all is not possible and a different action is necessary, e.g. adjust the priority based on who the user of the video application is.

These constraints constitute a second category of requirements. These are, however, not QoS requirements on the expected behaviour of an application. Rather, they are administrative or organizational requirements. These requirements will vary among different administrative domains and will vary over time.

4 Types of Requirements and Policies

We have identified two different types of requirements: requirements on the run-time behaviour of an application (hence referred to as *application QoS requirements*) and requirements that express constraints on the set of possible corrective run-time actions (hence referred to as *administrative requirements*). Clearly, for both types of requirements to be useful in the management of distributed applications, and given the run-time nature of both sets of requirements, it must be possible to represent both within the management system and it must be possible to determine the specific components that they apply to.

Application QoS requirements for a particular application will change. For example, the requirements of an application may depend on the user who has invoked the application. Thus, different sessions of the same application may have different QoS requirements. Administrative requirements will also change, since the constraints on the possible adaptations will also change during the lifetime of the system.

As described earlier, an application QoS requirement may be violated. In these cases, it must be possible to specify an action. In many cases we will want this action to include sending a notification of the violation to another entity that is doing the diagnosis and determining the adaptation to be taken.

The implication of these two observations is that it must be possible to store both application QoS and administrative requirements. The application QoS requirements should be accessible by an application when that application is started up. It must also be possible to specify the action(s) to be taken if the application QoS requirement is not satisfied.

This suggests that we allow the requirements to be expressed as *policies*. A *policy* can be defined [19] as a rule that describes the action(s) to occur when specific conditions occur. There has always been some use of some form of

policies to manage networking resources in most enterprises. For example, system administrators often define policies regarding what types of traffic a firewall can reject. The Internet Engineering Task Force (IETF) is currently developing a set of standards (current drafts found in [14,18,21]) associated with policies.

A policy for an application QoS requirement has as its condition the negation of the QoS requirement which means that the specified action(s) is to take place when it has been detected that the QoS requirement has been violated. We call these *expectation* policies. The action(s) will include where to send the notification of a violation of the QoS requirement.

Policies can also be used for administrative requirements. These are called *enforcement* policies. A more detailed description of how these policies can be formalised and supplied to the application upon start-up is described in [13].

5 Architecture

This section describes the key architectural components (depicted in Figure 1). For illustrary purposes, we shall use the following QoS requirement in examples in the remainder of the paper: “The application deliver video at a frame rate of between 23 and 27 frames per second.” (lower and upper bounds are used to indicate when the frame rate is too slow or too fast).

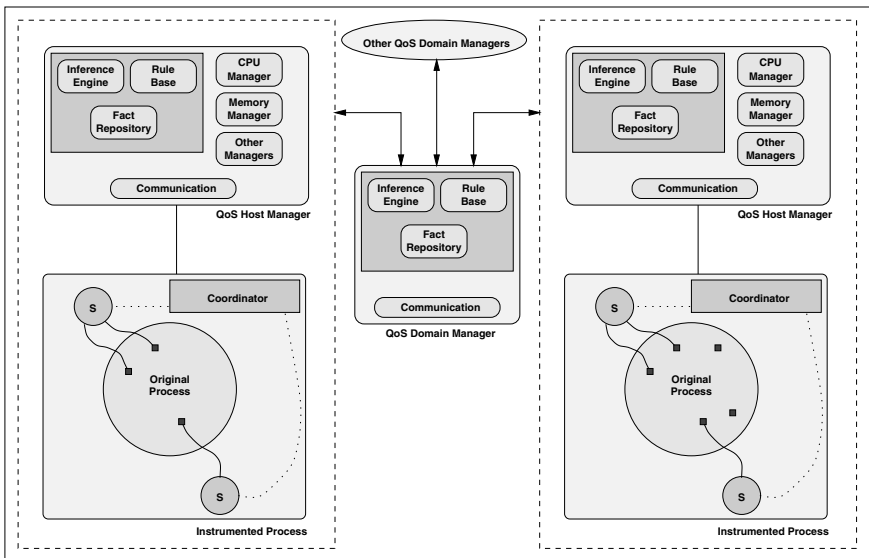


Fig. 1. Management Architecture

5.1 Instrumented Process

Expectation policies express application QoS requirements which are characterised by conditions that must be satisfied about attributes of the application. For example, the frame rate of a video application is an attribute and the statement “a frame rate of between 23 and 27 frames per second” is a condition about that attribute that should be satisfied by the application at run-time.

Instrumentation code is code inserted into an application at strategic locations to collect the values of attributes. While some measurements of attributes can be taken by observing external application behaviour and rudimentary control can be achieved through operating system interactions, work in this area has found that these approaches are limiting in both accuracy and the kinds of metrics and control available.

An instrumented process is an application process with embedded instrumentation code. It is this instrumentation code that enables the management of the application process. This instrumentation is provided by several components.

Sensors. Sensors (represented as S in Figure 1) are used to collect, maintain, and (perhaps) process a wide variety of attribute information within the instrumented processes. Each attribute is part of a policy (or policies) being applied to the application; policies specify constraints on process attributes. For simplicity, we associate a sensor with one attribute. Sensors get their input data from probes inserted at strategic points in the application code or by reading other sensors. During run-time, sensors can be enabled or disabled, reporting intervals can be adjusted and thresholds can be modified.

Probes. Probes are embedded in process code to facilitate interactions with sensors. Probes allow sensors to collect metric information. Each probe is specific to a particular sensor. Probes are the only instrumentation component that must be injected into the original process source code—all other components need only reside in the same address space as the process, and do not require code modifications.

Coordinators and sensors are defined as classes. Probes are methods of the sensors.

Example 1.

As an example, consider the sample pseudo-code for a video playback application in Figure 2 that has the QoS requirement stated at the beginning of this section. This QoS requirement is translated into initial thresholds for sensor s_1 , capable of measuring the frame rate. The target threshold is 25 frames per second, the upper threshold is 27 frames per second and the lower threshold is 23 frames per second.

Sensor s_1 includes at least the following two probes which are also methods of s_1 : (1) An initialisation probe that takes as a parameter the default threshold target value. This probe gives a value to the target, as well as the upper and lower thresholds. (2) A probe that (i) determines the elapsed time between frames. (ii)

checks to see if this time falls within a particular range defined by the lower and upper acceptable thresholds, and (iii) informs the coordinator if the constraints on the values of the frame rate are not satisfied c . \square

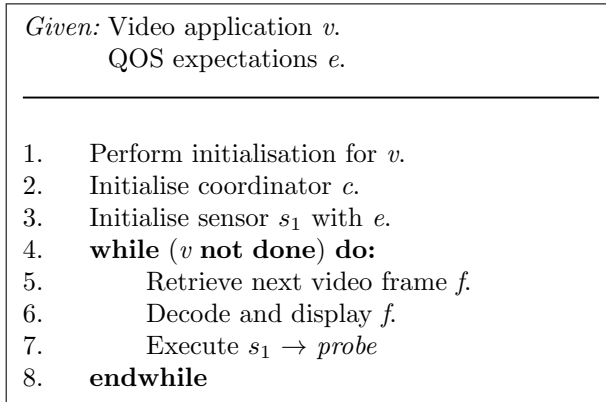


Fig. 2. Instrumentation Example

How does a sensor relate to policies that express application QoS requirements? A sensor collects values for an attribute of the process. This attribute is part of a policy (or policies) being applied to the application. Application policies specify constraints on process attributes. We assume that a sensor is responsible for monitoring a specific attribute. If the sensor finds that a constraint involving the frame rate attribute is violated, it reports this to the coordinator as an *alarm report*. For our example QoS requirement, it is assumed that one sensor is needed for the frame rate. We note that a sensor may provide values to be used in more than one policy. This means that there is a many to many relationship between policies and sensors. An application may also have more than one policy and it is possible that these policies share attributes. We note that not all sensors measure attributes that are directly used in the specification of a QoS requirement of a policy (this will be illustrated later).

Coordinator. A coordinator maintains information about each policy including the following: (i) The policy identifier. (ii) A condition list where a condition is represented by the attribute, comparison operator and value that the attribute is to be compared to using the comparison operator. (iii) An action list. Since a policy may be specified using more than one condition on attributes, when the coordinator receives an alarm report from a sensor it needs to check to see if a policy has been violated. If so, it takes the actions specified in the action list which most likely include a message notifying the QoS Host Manager. It also

disables the policy – this means that violations of the policy will not send out any more messages. This prevents a flood of messages at the QoS Host Manager.

Example 2.

There is a policy that has the negation of QoS requirement stated at the beginning of this section. Thus, when this condition is true (meaning that the QoS requirement is not being satisfied), the associated action list (maintained by the coordinator) specifies that the following actions take place: (i) The size of the communication buffer is requested from the buffer sensor (ii) The communication buffer size, the current frame rate, the identifier of the policy are among information sent to the QoS Host Manager for further analysis.

5.2 Quality of Service Host Manager and Domain Manager

The QoS Host Manager receives notifications from a process (through the process's coordinator) when a policy has been violated. The QoS Host Manager has a set of rules that are used to determine the corrective action(s). This involves determining the cause of the policy violation and then determining a corrective action(s). The process of determining the rules to be applied is called inferencing. Inferencing is used to formulate other facts or a hypothesis. Inferencing is performed by the Inference Engine component of the QoS Host Manager. The inferencer chooses which rules can be applied based on the fact repository. The inferencing that can take place can either be as complex as backward chaining (working backwards from a goal to start), forward chaining (vice-versa) or as relatively simple as a lookup. In this work we used forward chaining.

Consider the QoS requirement stated at the beginning of Section 3. A possible corrective action is to increase the CPU priority of the process receiving the frames so that it can process more frames in shorter time. However, what if the problem is not local e.g., the server process feeding the frames is on a host machine that is overloaded or the network connections between the server and the client are congested. If the problem is not local, then changing the CPU priority of the client process will not improve the frame rate. Worse yet, it will have a negative impact on the performance of other processes.

One rule for the QoS Host Manager is informally stated as follows: *If the communication buffer size is above some threshold (implying that the process is not able to process frames fast enough), then the CPU Manager is invoked to adjust the CPU priority of the violated process.* Additional rules are used to determine how much to increase CPU priority based on how close the priority is to being satisfied.

Another rule for the QoS Host Manager can be informally stated as follows: if the communication buffer size is below some threshold then send a notification to the QoS Host Manager, which can then locate the source of the problem, perhaps by interacting with the QoS Host Managers. The implication of a small buffer size is that the video client is able to process the received frames fast enough and that the problem is either a network problem or a server problem.

The QoS Domain Manager also has a rule set that is used to drive the location process and guide the formulation of corrective actions. One such rule for the QoS Domain Manager is informally stated as follows: Upon receiving an alarm report from the client-side QoS Host Manager, ask the corresponding server-side QoS Host Manager for CPU load and memory usage. Another QoS Domain Manager rule states that if the CPU load exceeds some predefined threshold or the memory usage exceeds some threshold then an alarm report is sent to the server-side QoS Host Manager.

This is a relatively simple set of rules. A more complex set of rules would include rules that reflect administrative requirements. We will describe some of these in section on experimental results.

6 Details of the Prototype Implementation

To evaluate the approach, we have developed a prototype system based on the architecture presented in the previous section. This prototype has been implemented for Solaris 2.6.

6.1 Instrumentation

We have built a C++ instrumentation library that implements a hierarchy of sensor classes. The base of the hierarchy provides a registration method that provides the ability to have all sensors register with the coordinator in a uniform manner. Other base methods are for enabling/disabling and read/report.

A coordinator retrieves policies from a file. The coordinator creates a random port number with which to listen for messages from the QoS Host Manager. The port number is registered with the QoS Host Manager. Policy violation reports are issued by the coordinator using an IPC message queue that the QoS Host Manager creates.

6.2 QoS Host Manager

The QoS Host Manager was implemented to permit multiple violation reports to be examined concurrently by maintaining a violation queue with a policy identifier, server host, client host, application type and application mode (e.g., is it in client mode or server mode).

The inference engine, rule set and fact repository are implemented using CLIPS [8]. CLIPs rules are used to do the following: (i) Distinguish between local and non-local causes of the policy violation and (ii) Implement the enforcement policies. These rules represent the enforcement policies.

One rule has as its condition that the complaining process is registered (if it hadn't then this is not a QoS-enabled process), the frame rate is low and the size of the buffer is high or the trend is that the buffer size is increasing relatively quickly. This indicates that the frame rate is low and that the problem is local since a full buffer or a buffer whose size is increasing indicates that the process is

not processing the frames fast enough. If this condition is true, then the action to be taken is briefly described as follows: Determine if there are enough CPU resources to be allocated to the complaining process. If so, then allocate and if not then assert a `serv_diff` fact. This causes the condition of another rule to become true which in turn has the associated action that the complaining process scales back its threshold for a specified period of time.

6.3 CPU Manager

The CPU Manager is organised as follows. There is a communication component that deals with communication with the inference engine. Requests from the inference engine like are received using message queues. The results are available through shared memory. Requests include the registration and deregistration of processes, allocation adjustment requests, enabling/disabling requests, and service differentiation requests. These requests are dispatched to the `scheduler manager` component whose functionality is explained shortly.

The `scheduler manager` component is an application running at the superuser level and has the highest priority that maintains information about the number of processes running at each of the real-time priority levels as well as the maximum allowed (which can be changed). By default, all user initiated processes start out as time sharing. Resource allocation is done by having the `scheduler manager component` change the priority of a process using the `prcntl` system call. This has the effect of moving a process to the end of a queue at a different level of the array. Different classes at the user level are mapped to a set of priority levels associated the real time class. If the maximum is reached then they remain as a time-sharing process.

6.4 QoS Domain Manager

There is one QoS Domain Manager in the local network that receives reports from QoS Host Managers. It has rules that can distinguish between a remote machine problem and a network problem (this usually requires a query of other QoS Host Managers) and can tell a QoS Host Manager on a remote machine to adjust CPU and memory allocations where appropriate. Communication between QoS Host Managers and the QoS Domain Manager is through TCP.

7 Experiments

We have implemented a prototype system based on the architecture described in Section 3. The prototype is implemented in a Sun environment running Solaris 2.6.

To examine the effectiveness of this approach, we evaluated the prototype in a number of experiments involving three different applications - a real time video conferencing application (`rtvc`), a video player application (`mpeg`), and a multi-person game with video (`doom`). The applications were chosen because

a) they involve multiple systems, b) they have significant demands for system resources and c) they have soft QoS requirements for the display of video and images. Each application was instrumented with a sensor (fps) that measures frames per second.

The experiments presented in this section show results when the problem causing the QoS violation is the client CPU load. The applications were run under different scenarios and the delivered frames per second on a client system were collected during execution. We graphically present several experiments and briefly discuss each.

Experiment 1: No Management

As a baseline, we ran all the application concurrently, as a single user might do. Figure 3 depicts the frames per second as measured over several minutes. The quality of each, as measured in terms of frames per second varies widely for each application. Given that all are competing for system resources, this is not surprising.

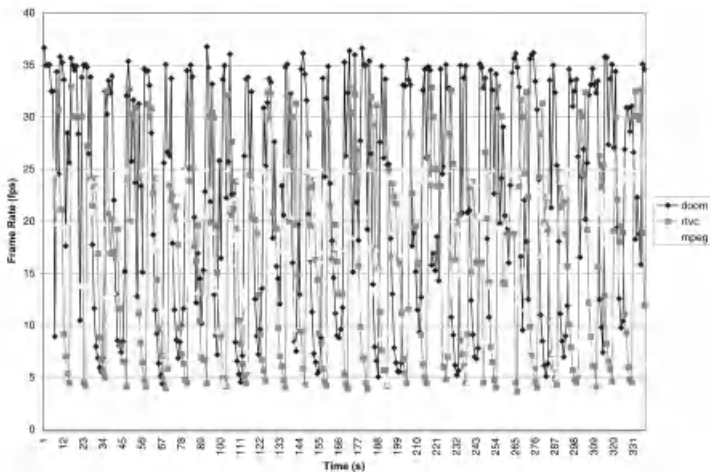


Fig. 3. All Applications with No Management

Experiment 2: Managed Environment

In a second experiment, we introduced our management prototype and defined policies. The QoS requirements for the real time video conferencing application and the video player application was a defined frame rate of 25 frames per second, plus or minus 2 frames per second, i.e., it was acceptable if the frame rate was between 23 and 27 frames per second. The game (doom) was assigned no specific policy – in effect, it was designated as “best effort”.

We introduced an administrative policy which favoured the video conferencing application over the video player application.

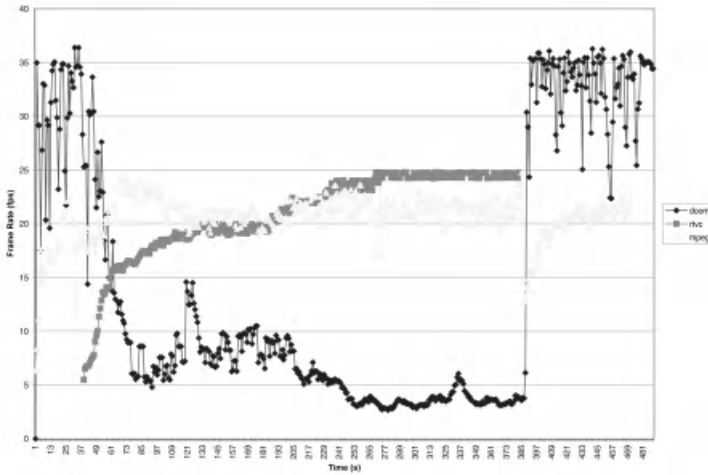


Fig. 4. All Applications in Managed Environment

The results are presented in Figure 4. *Doom* is started first, followed by *mpeg* and then *rtvc*. Within the first minute of the start of *rtvc*, the system quickly begins to differentiate among the applications, allocating more resources to the video conferencing and video player applications. It is able to manage system resources to enable *rtvc* to meet its QoS target of 25 frames per second and does a fair job of ensuring that *mpeg* meets its QoS requirements.

Once *rtvc* terminates, resources are freed. The system continues to manage the video player application, but now the game can grab available resources. The variability in the delivered frames per second for *doom* is understandable - it is a best effort approach for getting resources. The reason for the brief decrease in the frames per second for the *mpeg* application after *rtvc* terminates and for the variability of the frames per second subsequently is not clear. It could be a side effect of our algorithm, its implementation or unexpected behaviour arising because of the interplay between the real-time scheduler and the time-sharing scheduler in Solaris. We are currently exploring this in greater detail.

Experiment 3: Loaded Processor

In the third experiment, we introduced a load on the local processor. A locally developed load generator was started before the applications which created and maintained a run queue of 6. The “load” measured used the systems load monitor and represents an average of the run-time queue length.

Even with the load, the prototype manages to allocate resources to `rtvc` and `mpeg`. The results are shown in Figure 5.

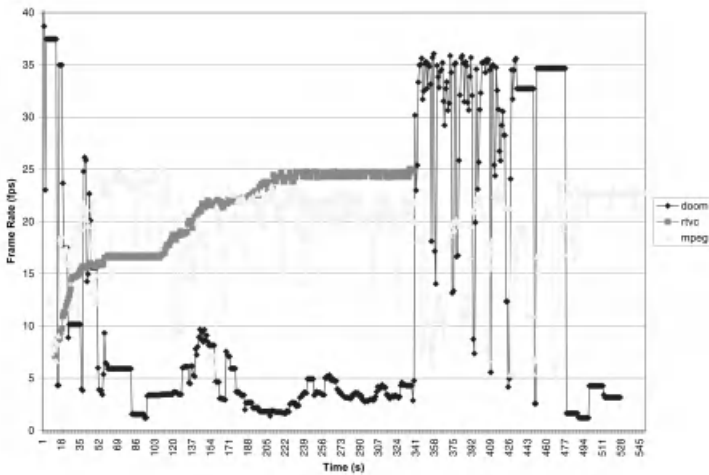


Fig. 5. Managed Environment with Load

Once `rtvc` terminates, `doom` is able to grab more available resources. Interestingly, there is less variability with load on the system than without (Experiment 2). We speculate that this is because the prototype is forced to continue to control resources with a load in place than without, where it “lets go”.

Experiment 4: Multiple QoS Demands

As a final experiment, we considered the situation in which more than two applications were executing and which demanded QoS. In this case we created somewhat artificial situation in which two “users” would try to run the video player application and the game on the same system. The QoS associated with each `mpeg` application was the same, namely, a frame rate of between 23 and 27 frames per second. Each game was treated as “best effort”. One user (user a) also ran the video conferencing application. As with the previous experiments, the prototype had an administrative policy which favoured the video conferencing application. The results are depicted in Figure 6.

The prototype was able to allocate resources to meet the QoS requirements of `rtvc`. This meant that both `mpeg` applications and both `doom` applications were given less resources. As the graph illustrates, the frames per second for both `mpeg` applications was around 10-15 frames per second. Since `doom` is “best effort”, the prototype also allocated more of the resources to both `mpeg` applications, leaving `doom` to cope with less computing resources.

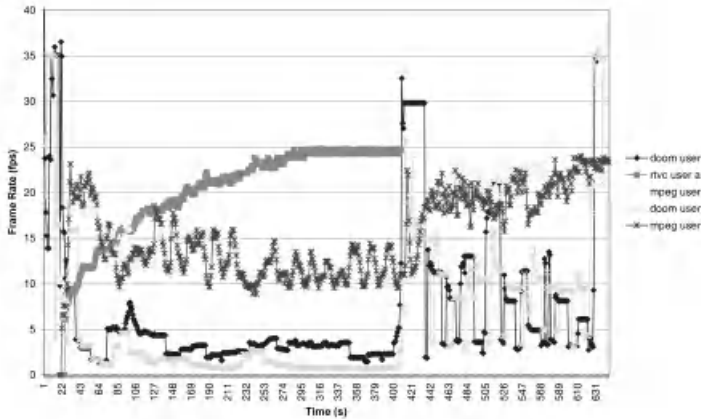


Fig. 6. Multiple Applications with Same QoS Requirements

Once the video conferencing application terminated, the system allocated more resources to both video player applications bringing to about 23-24 frames per second.

8 Conclusion and Future Work

The results of the experiments with the prototype are promising. First, the prototype was able to manage resources to meeting the QoS requirements of multiple applications on a single host, even under loaded conditions. Enforcement policies were successfully used for differentiating between applications. This is significant in that it shows that it is possible to manage resources on behalf of all applications running on a host, server or across a network. This allows administrators to determine their goals in optimising resource allocation to applications. These goals can be mapped to administration policies. As seen in the CLIPS rules, if the service differentiation rule is applied, the thresholds of the policy attribute are scaled back. This requires that the QoS Manager send a message to the coordinator which causes the coordinator to invoke a method in the sensor that causes a scale back of the threshold. This has the effect of reducing the resource needs of the application which is a form of application adaptation. Other application adaptations are possible and more discussion of how our work supports other adaptations can be found in [10]. Although not shown in this paper, we are able to diagnose between client and server machine overloads and affect the CPU allocation on the server machine.

Second, our work supports satisfying QoS requirements in a dynamic environment as follows: (i) QoS requirements for an application can be retrieved by the application when the application starts up. When the application starts executing it registers with a *policy agent* process. The application process passes information about that process that is relevant in determining the expectation

policies that are applicable to that process. This includes a process identifier, an application identifier, an executable identifier and a role identifier. The policy agent uses this information and maps it to the appropriate policies (which is retrieved from an LDAP directory). This is sent to the coordinator component that creates a list of policy objects that it maintains. More information can be found in [13].

Third, in our work neither the user nor developer needs to have intricate knowledge of the hardware architecture or system software. We have instrumented a number of third party applications besides those described in the experimental part of the paper. This includes the Apache Web Server. The only knowledge needed was the name of the probes and knowing which libraries to link in. It was not necessary for the instrumentor to have any knowledge of what sort of QoS management was taking place.

Fourth, our work can be extended to support other languages. We are currently developing an instrumentation library for Java software. The instrumented Java programs will be able to communicate with the QoS Host Manager.

Future work includes the following:

- The prototype focussed on one resource. We will consider other resources, such as memory, network bandwidth and input-output throughput. We currently have implemented a resource manager for memory and for networking. These will be integrated into the current prototype. This raises many questions in terms of how different resource managers would interact.
- We will further study the prototype across multiple systems. There are two types of studies: When the multiple systems are in the same administrative domain and when they are in different administrative domains.
- The CLIPS rules are relatively difficult to specify. We have used a formalism based on [5] for specifying expectation policies (for more detail see [13]) and we are currently writing specifications for enforcement policies and a translator that takes the enforcement policies and translates them to CLIPS rules. This should make it easier for administrators to specify enforcement policies.
- In our work, we found that enforcement policies across different hosts to be difficult to define. We will study the informal policies used at our university to give help better formulate reasonable enforcement policies.

References

1. C. Aurrecochea, A. Campbell, and L. Hauw. A Survey of QoS Architectures. *Multimedia Systems Journal, Special Issue on QoS Architecture*, May 1998.
2. G. Beaton. A Feedback-Based Quality of Service Management Scheme. *Proceedings of the Third International Workshop on High Performance Protocol Architectures*, Uppsala, Sweden, June 1997.
3. J. Bruno, J. Brustoloni, E. Gabber, M. McShea, M. Ozden, and A. Silberschatz. Disk Scheduling with Quality of Service Guarantees. *Proceedings of the 1999 IEEE International Conference Multimedia Computing and Systems*, Florence, Italy, June 1999.

4. H. Cho and A. Seneviratne. Dynamic QoS Control without the Knowledge of Resource Requirements. *Submitted to IEEE Trans. Computing*.
5. N. Damianou, N. Dalay, E. Lupu, and M. Sloman. Ponder: A language for specifying security and management policies for distributed systems: The language specification (version 2.1). Technical Report Imperial College Research Report DOC 2000/01, Imperial College of Science, Technology and Medicine, London, England, April 2000.
6. I. Foster, A. Roy, and V. Sander. A Quality of Service Architecture that Combines Resource Reservation and Application Adaptation. *Proceedings of the Eight International Workshop on Quality of Service (IWQOS 2000)*, pages 181–188, June 2000.
7. N. D. Georganas. Multimedia Applications Development: Experiences. *Journal of Multimedia Tools and Applications, Volume 4, Number 3*, May 1997.
8. J. Giarratano and G. Riley. *Expert Systems: Principles and Programming*. PWS Publishing Company, 1998.
9. A. Goel, D. Steere, C. Pu, and J. Walpole. Adaptive Resource Management via Modular Feedback Control. *Tech. Report. 99-03, Oregon Graduate Institute, Computer Science and Engineering*, Jan 1999.
10. M. Katchabaw, H. Lutfiyya, and M. Bauer. Using User Hints to Guide Resource Management for Quality of Service. *The 1999 International Conference on Parallel and Distributed Processing Techniques and Applications*, July 1999.
11. B. Landfeldt, A. Seneviratne, and C. Diot. User Services Assistant: An End-to-End Reactive QoS Architecture. *Proceedings of the Sixth International Workshop on Quality of Service*, Napa, California, May 1998.
12. B. Li and K. Nahrstedt. QualProbes: Middleware QoS Profiling Services for Configuring Adaptive Applications. *Proceedings of IFIP International Conference on Distributed Systems Platforms and Open Distributed Processing (Middleware 2000)*, 2000.
13. H. Lutfiyya, G. Molenkamp, M. Katchabaw, and M. Bauer. Issues in Managing Soft QoS Requirements in Distributed Systems Using a Policy-Based Framework. *2nd Workshop on Policies in Distributed Systems and Networks*, pages 461–468, January 2001.
14. B. Moore, J. Strassmer, and E. Elleson. Policy core information model – version 1 specification. Technical report, IETF, May 2000.
15. S. Rampal, D. Reeves, and I. Viniotis. Dynamic Resource Management Based on Measured QoS. *Technical Report TR 96-2, North Carolina State University*, 1996.
16. H. Sariowan, R. Cruz, and G. Polyzos. Scheduling for Quality of Service Guarantees via Service Curves. *Proceedings of the 1995 International Conference on Computer Communications and Networks*, Las Vegas, Nevada, September 1995.
17. R. Schantz, J. Zinsky, J. Loyall, R. Shapiro, and J. Megquier. Adaptable Binding for Quality of Service in Highly Networked Applications. *SSGRR-2000, International Conference on Advances in Infrastructure for Electronic Business, Science and Education on the Internet*, August 2000.
18. Y. Snir, Y. Ramberg, J. Strassner, and R. Cohen. Policy framework qos information model. Technical report, IETF, April 2000.
19. Stardust.com. Introduction to qos policies. Technical report, Stardust.com, Inc., July 1999.
20. I. Stoica, H. Abdel-Wahab, K. Jeffay, S. Baruah, J. Gehrke, and C. Plaxton. A Proportional Share Resource Allocation Algorithm for Real-Time, Time-Shared Systems. *Proceedings of the Real-Time Systems Symposium*, Washington, DC, December 1996.

21. J. Strassner, E. Ellesson, B. Moore, and Ryan Moats. Policy framework ldap core schema. Technical report, IETF, November 1999.
22. H. Zhang and S. Keshav. Comparison of Rate-Based Service Disciplines. *Proceedings of SIGCOMM'91*, Zurich, Switzerland, September 1991.

Implementation of a Bandwidth Broker for Dynamic End-to-End Capacity Reservation over Multiple Diffserv Domains

Ibrahim Khalil and Torsten Braun

Computer Networks and Distributed Systems (RVS)
Institute of Computer Science and Applied Mathematics (IAM)
University of Berne
Neubrückstrasse 10, CH-3012 Bern, Switzerland
`{ibrahim,braun}@iam.unibe.ch`

Abstract. As Virtual Leased Line (VLL) type point-to-point connections over Internet are now possible with the Expedited Forwarding (EF) Per Hop Behavior (PHB), Internet Service Providers are looking for ways to sell bandwidth services to prospective corporate customers with the so called Bandwidth Brokers (BB). Based on this approach most of the recent implementations consider providing such services only in a single provider domain. However, many of the ultimate customers might actually want to extend VLLs up to the periphery of other providers. In this paper, we describe the implementation of a BB that uses simple signaling mechanism to communicate with other cooperative Brokers to enable customers to dynamically create VLLs over multiple Diffserv domains.

1 Introduction

Virtual Leased Line (VLL) type point-to-point connections can now be built over various IP segments with the recently proposed Expedited Forwarding (EF) [1] Per Hop Behavior (PHB) in the differentiated services [2] architecture. To take advantage of this new technology Bandwidth Brokers that can dynamically create VLL on demand have been proposed in [3], [4] and refined in [5], [6]. New Bandwidth Broker models based on the existing architectures have also been proposed in [7], [8], [9] and several implementations have been reported in [7], [5], [6], [10]. Most of these implementations of Bandwidth Brokers have the following characteristics in common:

- They are mostly responsible for a single Diffserv Domain. The Bandwidth Brokers are capable of advance or immediate reservation only in the domains they maintain.
- All the concepts propose policing users traffic at the ingress edge only. Except [11] most of the BBs don't consider interior provisioning.
- Almost all except [7] and [11] rely on RSVP for signaling.

While the existing Bandwidth Broker implementations don't yet have mechanisms to communicate with other neighboring domains, they mostly propose modified RSVP or similar mechanisms as the method for both inter-domain and intra-domain signaling. Although both of these have the potential to be integrated in the future advanced Bandwidth Brokers, at the moment when there are core Service Level Agreements (SLAs) and resource provisioning issues yet to be solved [12], they (i.e. the use of RSVP like signaling) might further complicate implementation issues and delay easy and rapid deployment. For example, as mentioned in [4], for resource reservation over a Diffserv network using such Bandwidth Brokers, both sending and the receiving hosts need to be present during reservation and also during the period the reserved interval starts. In reality, the sender or receiver might not even exist during the reservation process.

In this paper, keeping this in mind, we present a simple approach to make advance reservations in the absence of senders or receivers in a multi-domain scenario. Rather than using RSVP in inter-domain signaling to reserve capacity across domains, we use a novel method to identify domains, and hence the Bandwidth Brokers that are responsible for maintaining them. Section 2 presents basic components and ingredients for making reservations over several Diffserv domains with Bandwidth Brokers. Section 3 describes implementation architecture and the components in that architecture. In section 4, we describe operational details and system flows of the BB, and in section 5 we clarify the operational details by presenting some real examples. Finally, in section 6, we conclude our paper with a summary and future research directions.

2 End-to-End Capacity Reservation

2.1 An Example Scenario

Consider the scenario as shown in Figure 1. The domains are Diffserv [2] enabled and under different administrative control. This means that if stub networks C or D in domain 1 want to establish VLL with Stub network A in the same domain or with stub network B in domain 2, traffic entering domain 1 is classified and possibly conditioned at the boundaries (edge router 2) of the network, and assigned to different behavior aggregates. Each behavior aggregate is identified by a single DS codepoint (DSCP for EF). In the interior (and also egress) of the network, with the help of DSCP- PHB mapping certain amount of node resources can be allocated for this quantitative traffic.

2.2 An Automated System: Bandwidth Broker

In the example above if the administrative control of each ISP is given to an automated system like a Bandwidth Broker, its responsibilities will be :

- Check request validity. In the example, for the VLL over domains, BB 1 needs to check the validity of stub network A's request and BB 2 needs to check the request of ISP domain 1.

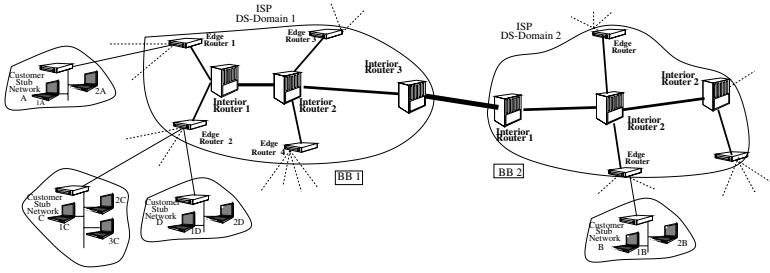


Fig. 1. Diffserv Deployment Scenario across Multiple Diffserv Domains

- Perform admission control in its domain. In a simple case, this can be only checking resource availability at the border routers as these are the obvious points that will be traversed by a VLL connection. In a more advanced case this can be checking resource availability at all the routers along the path from ingress towards egress.
- Coordinate with other separately administered Bandwidth Brokers. In the example, Bandwidth Broker 1 will need to signal to Bandwidth Broker 2 for resource reservation on behalf of stub network A. If there are several ISP domains each managed by such Brokers, the job of this coordination also means identifying the right domains and corresponding brokers for a certain resource allocation request (RAR).
- Configure ingress router of its domain if a request is accepted. Configuring ingress router means dynamically marking the traffic as EF and policing to limit the volume of traffic.

2.3 Service Level Agreements

Service Level Agreements are generally contracts between network service providers and their customers to guarantee particular quality of service levels for network performance. SLAs exist between a customer and its provider (called intra-domain or customer-ISP SLA) and also between two providers (called inter-domain or inter-ISP SLA).

A customer normally has a contract with the local ISP regarding the maximum amount of traffic he can send or receive for a VLL service. Such customer-ISP SLA, however, doesn't automatically guarantee that a customer will always receive the service upon request - it only indicates the upper limit of the request and successful reservation of the requested VLL depends on admission tests at different points along the VLL. Referring to Figure 1, if a customer wants to establish a VLL between stub network A and C, an intra-domain SLA would suffice. However, for a VLL to be established between stub network A and B and inter-domain SLA between domain 1 and 2 must be in place. Based on inter-domain SLA the sending domain can send a maximum, say X Mbps aggregated traffic, to a receiving domain, and ensures that it doesn't send more than X Mbps by shaping at the outgoing interface connected to receiving domain. Receiving

domain polices the incoming traffic at the ingress router's inbound interface to make sure that the sending domain doesn't send more than X Mbps.

2.4 Service Provisioning and Call Admission Control

Determination of resources required at each node for quantitative traffic needs the estimation of the traffic volume that will traverse each network node. While an ISP naturally knows from the SLA the amount of quantitative VLL traffic that will enter the transit network through a specific edge node, this volume cannot be estimated with exact accuracy at various interior nodes that will be traversed by VLL connections. However, if the routing topology is known, this figure can be almost accurately estimated. For simplified provisioning and admission control we assumed the following:

- Pre-configure interior and other border routers with scheduling mechanism like Priority Queuing (or CBQ, WFQ) so that traffic marked as EF are served by high priority queue.
- Traffic follows a known path.

If the domain is QoS rich, for a simple model it might suffice only to perform CAC at the edge points. For a more sophisticated model, considering the necessity of interior provisioning the BB may also check the availability of resources at the interior points that would be traversed by a VLL. In such a case, virtual core provisioning [11] might be suitable that only requires a capacity inventory of interior devices to be updated based on VLL connection acceptance, termination or modification.

2.5 End-to-End Signaling

A user sends a request to the Bandwidth Broker that maintains the user's ISP domain. The request contains source and destination addresses and also the requested bandwidth. While the source naturally resides in the stub networks attached to the ISP's network, the destination might well be in the stub network that is attached to another ISP's domain. That domain might not be the final domain and there might be one or more domains in between. If both the source and destination addresses are in the stub networks of the same ISP domain, the Broker that maintains the domain can find the ingress and egress routers by some simple lookup in the various Broker related databases (explained in the next section). The Broker performs admission control at the points (ingress and egress) before deciding whether the request should be granted or not. If the destination is in another domain other than the source domain, then the Broker must identify the followings:

- the domain that has the destination stub connected to it.
- intermediate domains (if any) that will be traversed by VLL connection if the request is accepted.

Before we investigate the above two issues it would be useful if we give a brief overview of Resource Allocations Requests.

BB Resource Allocation Request Format. A Resource Allocation Request (RAR) may come from an individual user to a BB or from one BB to another neighbor BB. We call the first one intra-domain RAR while the latter one is referred as inter-domain RAR. Their formats are:

- **Intra-domain RAR:** To setup a new VLL this request contains user id and password, source and remote addresses of the VLL to be established and the bandwidth required for it:
Newflow -u userid -p password -s source -d remote -b bandwidth
- **Inter-domain RAR:** Inter-domain RAR is automatically generated by a broker when it detects that the remote address indicated in the request is not attached to its domain. The request is then sent to another neighbor domain. Since the actual requester is a domain broker, the recipient broker needs to check its validity as an inter-domain request.
Newflow -bb brokerid -p password -s source -d remote -b bandwidth -tbb final_domain

Domain Identification. A scalable and simple way for each Broker would be to send boundaries of the domain that it maintains to other cooperative domains. By boundaries we mean the IP addresses of the edge devices. Lets consider domain 1 and 2 in Figure 2 where each of the domain is actually constituted from several edge devices. All these edge devices have unique IP addresses. If we can identify an edge router by the destination IP address in the RAR, then we can readily identify the domain, and hence the Bandwidth Broker that represents the domain. For example, when a user wants to establish a VLL to send traffic from any of the stub networks 7.7.x.x to one of the sub networks 5.5.x.x, Broker BB1 can easily identify that it has to finally communicate with BB7 by reading a domain identification database.

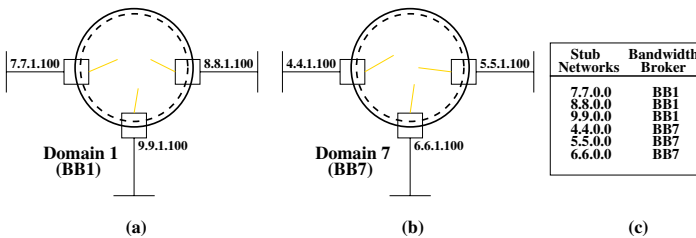


Fig. 2. (a) Domain 1 (b) Domain 2 (c) Domain Identification Database

Bandwidth Broker Message Forwarding. When the Bandwidth Broker identifies the Final Broker there might be one or more intermediate Brokers that need to be contacted as well for end-to-end capacity reservation. How does

Broker (first Broker or any intermediate Brokers) determine the next appropriate Broker when there are several neighbor Brokers and a VLL needs to be established over several domains? In the previous example the VLL needs to be established over domains that are managed by BB1, BB2 and BB7. If each Broker knows the neighbor brokers and by exchanging that information every Broker can build a message forwarding table as shown in Figure 3(c) and 3(d). From the table is obvious that BB2 is the intermediate broker that needs to be contacted first by sending an Inter-domain RAR from BB1 before BB2 finally sends another inter-domain RAR to BB7 on behalf of BB1.

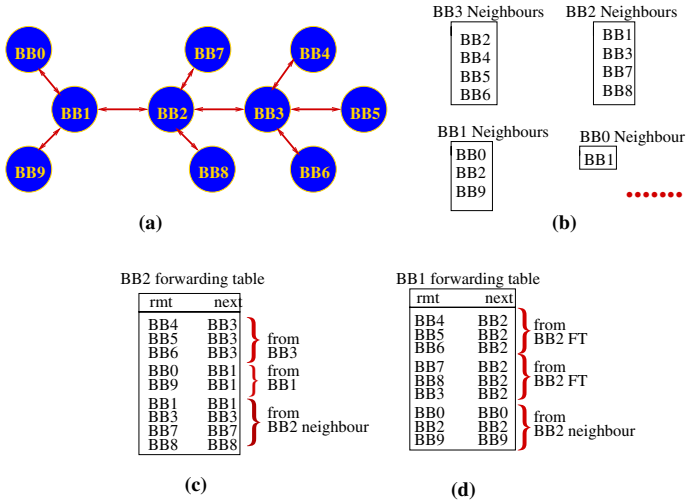


Fig. 3. (a) Several Diffserv Domains represented by Bandwidth Brokers (b) Neighbors tables of some Brokers (c) BB Message forwarding table in BB2 (d) BB Message forwarding table in BB1.

3 Implementation of the System Components of the Broker

3.1 Architecture

Based on the requirements for end-to-end capacity reservation the Bandwidth Broker has been developed to dynamically establish VLL on customer's request. Our earlier analysis and functional requirements of BB resulted in a four layer implementation architecture of Figure 4. The top layer is responsible for validating both intra- and inter-domain requests. The two middle layers are composed of several databases that are invoked for admission and signaling purposes of valid requests. The bottom layer decides and configures edge routers based on

processing of requests in the three above mentioned layers. In the next few sections we will describe the components of these layers.

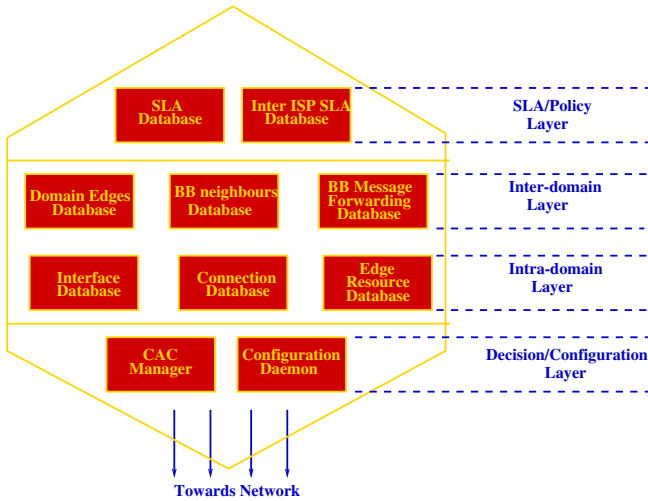


Fig. 4. Layered Implementation Architecture: Components of a BB for Resource Provisioning, Admission Decision and end-to-end Signaling

3.2 The Databases of the Bandwidth Broker

The **customer-ISP SLA database** contains not only the user's identification, but also specifies the maximum amount of traffic one can send and/or receive over a VLL. As VLL might be established between two points (i.e. source and destination) out of several possibilities, a SLA also contains the boundary of a valid VLL area and are put in this database as source and remote stub addresses. User authentication process prohibits malicious users to setup unauthorized VLL and access network resources illegally. It contains the following tuple:

<User ID, Password, Maximum BW in Mbps, Source Stub Address, Remote Stub Address>

The **inter-ISP SLA database** is invoked by a domain when it receives inter-domain RAR. By doing so the receiving domain can check the validity of the request sent by the sending domain. Here this validity means identification of the sending domain and the maximum amount of bandwidth it can reserve on a certain edge interface in the receiving domain that is directly connected to it.

<Domain ID, Domain Password, Maximum BW>

The **interface database** contains necessary records of edge routers that are used as VLL end-points for the outsourced VLL model. In such a model since some customer stub networks are connected to the ISP edge router we need to

specify which stub networks are connected to a particular edge router. Also, an edge router might have one or more inbound and out-bound interfaces which also need to be specified for each stub network that is actually connected to a particular in-bound interface of a router. This is important because normally at the in-bound interface VLLs are policed on individual basis and at the out-bound they are shaped on an aggregated basis. The tuples are :

< stub network, edge router, generic router name, in-bound interface, out-bound interface >

The **connection database** contains a list of currently active VLLs. When a request for a new VLL connection or termination of an existing connection arrives, the BB can check if that connection already exists or not and then make its decision. The storage of detail connections indicates the amount of resources consumed by VLL users at various edge and interior nodes.

< user id, source address, VLL ID, rmt address, bandwidth, activation time >

The **edge resource database** contains information regarding resource provisioned (C_{TOTAL}) for different router interfaces and used (allocated) capacity ($C_{allocated}$) to existing VLL connections. The difference between the two is the spared capacity that can be allocated to incoming connections. The tuples are, therefore:

< edge router, C_{TOTAL} , $C_{allocated}$ >

The **VLL ID database** maintains a list unique VLL IDs and their status for each edge router. An ID that is available is marked as 1, and the one that is used is marked as 2. The tuples are: *< edge router, Tunnel ID, Status >*

The **BB Neighbor Database** hold records of neighbor Bandwidth Brokers IP addresses as well as IP address of the router interfaces (both in-bound and out-bound) that interconnect the peer domains.

< Neighbor BB, InsideInterface, OutsideInterface >

The **Domain Edges (or Identification) Database** hold records of the networks that reside at the periphery of a domain. Its purpose is described in details in the previous section . An example entry of this database is also shown in Figure 2.

< Stub Network, BB >

The **BB Message Forwarding Database** contains next hop BB's IP address to send resource allocation request to final Remote Bandwidth Broker.

< Remote BB, NextBB >

3.3 CAC Manager and Configurations Daemons

CAC manager is a functional engine of BB that basically invokes the databases described above and decide the fate of an incoming request by performing admission control at various network nodes. *Configuration Daemons* are intelligent provisioning agents that are able to translate user request and BB generated pseudo rules into device specific rules to configure the routers/switches since we might have several different devices from various vendors.

3.4 Inter-domain Signaling

From earlier sections we have seen that a Bandwidth Broker not only receives RARs from a customer of its own domain or other BBs, but also sends RAR to neighbor BBs. Therefore, we have designed a Bandwidth Broker that consists of server and a client Socket program. When a Broker's Server receives a request from a client and finds itself to be the final destination BB it can convey the CAC decision back to the client, otherwise the Server tells the client of that Broker to talk to the appropriate neighbor Broker's server. So, there is a chain of communications which are handled by client-server concatenations. This is shown in Figure 5.

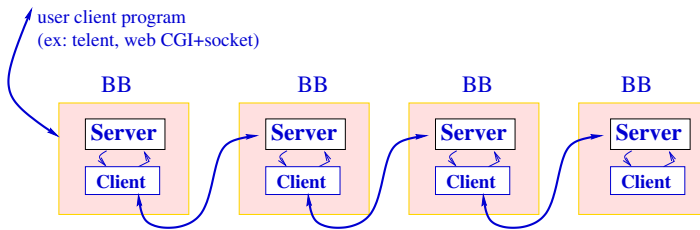


Fig. 5. Client-Server Concatenation for Inter-domain Signaling

4 Operational Details and System Flows

In this section we will describe how a connection is established or terminated, how various components interact with the BB, and under which circumstances a new connection request or termination request gets refused.

4.1 VLL Establishment

Figures 6 and 7 show all the communications involved in setting up a VLL connection between two stub networks or simply between an originating host and a remote host. Both intra and inter-domain cases are explained. Although an intra-domain scenario is not the focus in this paper, yet we describe it because its similarities in system dynamics with an inter-domain case, and many of the communications involved in an intra-domain scenario are actually repeated in the latter one. We will start describing the operational details by referring to the communications marked on Figure 6. Considering each communication in turn :

- 1) A user sends a VLL connection request message to the BB via http or other interfaces able to communicate to the BB server.

- 2,3) The BB contacts the customer SLA database that is responsible for validating the user and his request. If the user is identified correctly, his source and remote address conforms the contract, and also the bandwidth requested is less than or equal to the agreed traffic contract, it proceeds further.
- 4,5) The BB contacts the configuration daemon to check its status. The status can be busy, available, or down. Only in the case of availability the user request can be processed further.
- 6,7) The BB contacts the connection database to check the existence of an exactly similar VLL. This is because for a source and destination pair only one VLL can remain active.
- 8,9) The BB reads domain edges database to find out whether the VLL is needed to be created only in the domain under its supervision or might well span over other autonomous domains.

Intra-domain Case. If (Figure 6(a)) the BB finds that both source and destination are in the same domain,i.e. the VLL is needed to be created over a single domain, it proceeds as follows :

- 10,11) BB reads the interface database to find out ingress and egress edge routers. One or both are configured depending on a traffic contract.
- 12,13) Once the edge routers are detected from the interface database the BB communicates with the resource database and performs admission control on certain router interfaces to allocate a VLL of the requested amount. It might perform admission control on only the appropriate edge router interfaces or even on the interior routers interfaces that can be detected from the topology database. The resource database responds to the BB and either allocates the resource or denies based on resource availability.
- 14) The BB tells the configuration daemon to create appropriate configuration scripts. This is to be noted that configuration script is created only for the ingress edge router because this the only router that is configured to mark and police the incoming traffic. In the case of double edged SLA the egress router is configured as well for allocating QoS in the other direction. In the meantime, the resource and the connection database update their records. Another point to note is that BB also fetches a VLL ID from the VLL ID database that is unique for each VLL and needed while configuring the router. The new connection request data is appended to the connection database and the VLL ID that has just been allocated from the VLL ID database is marked as used.
- 15,16) The CD puts a busy signal on itself and creates the routing scripts. It then sends configuration scripts to the routers. The routers send signals to the CD.
- 17,18) The CD removes the busy signal from itself and sends acknowledgment to BB which sends a notification to the user.

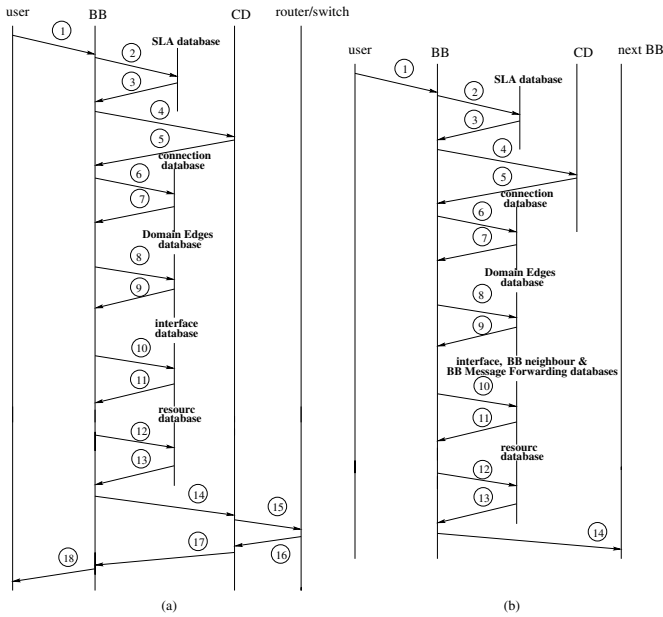


Fig. 6. VLL Establishment System Flow: (a) Intra-domain Case, (b) Inter-domain Case

Inter-domain Case. In the case a VLL (Figure 6(b)) is supposed to be established over several DiffServ domains the BB follows the steps described below:

- 10,11) Once the final destination domain has been determined the Bandwidth Broker finds out the next hop BB by reading the BB Message Forwarding Database. Now a search in the BB Neighbor Database gives current domain's outgoing interface towards the next hop BB. The BB also fetches the appropriate ingress router interface from the interface database.
- 12,13) These steps are similar to the steps 12 & 13 in the previous case. As the BB now knows the ingress and egress router interfaces, it performs admission control on those interfaces.
- 14) A positive CAC response leads to sending an inter-domain RAR to the next hop BB.

Next Hop BB as Final BB

If the next hop BB finds that the destination stub is in the domain maintained by it, the following steps are followed (Figure 7(a)):

- 15,16) Upon receiving an inter-domain RAR the next hop BB contacts inter-ISP SLA database to check the validity of the request.
- 17,18) It reads the BB Neighbor and interface databases to identify ingress and egress interfaces.
- 19,20) BB contacts the resource database and performs admission control on the previously identified ingress and egress interfaces.

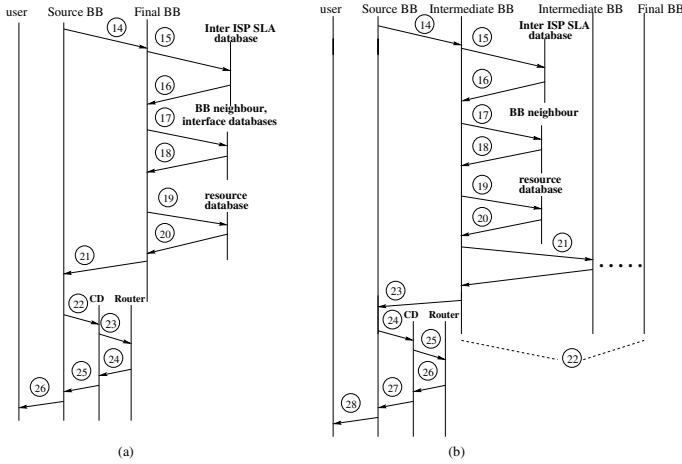


Fig. 7. VLL Establishment System Flow:Inter-domain Case cont'd. (a) Next Hop BB as Final BB (b) Next Hop BB as Intermediate BB

- 21) The BB sends CAC decision to the sender BB.
- 22-26) If the response received by the sender BB is positive then it contacts the appropriate configuration daemon to configure the ingress edge router. These steps are essentially the same like what we have seen in steps 14-18 in the intra-domain case.

Next Hop BB as Intermediate BB

The behavior of an intermediate BB is similar to that of a final BB with the exception that this one generates an inter-domain RAR based on positive CAC response from the resource database . The RAR is sent to the next BB which might be another intermediate BB or a final one. Figure 7(b) illustrates this case.

4.2 VLL Termination and VLL Request Rejection

VLL termination process involves the followings:

- The VLL connection entry is deleted from the connection database of the origin domain. Only the ingress edge router is configured to reflect the connection release.
- The resource databases are updated in all the domains that are traversed by the VLL, i.e. as resources are released $C_{allocated}$ is update as $C_{allocated} + C_{vll}$ where C_{vll} is the capacity of terminated connection.

A VLL request is rejected if

- user's SLA profile doesn't match in the origin domain, or in the case when inter-domain RAR is sent from one domain to the next neighbor domain,

interISP SLA profile of ISP that sends RAR doesn't match in the received domain.

- VLL connection already exists in the connection database of the origin domain.
- Admission control fails in any of the domains that are traversed by the VLL.

5 Examples of Dynamic Admission Control and Configuration with a BB

To test our implementation of the Broker System and its capabilities to setup VLL we ran some experiments over the public SWITCH [13] network between Bern and Geneva. The topology we used is shown in Figure 8. We have two domains with several end-systems that have private addresses and all these machines are connected to routers having public IP addresses. The domain in Geneva is represented by Broker 130.92.65.29 and the domain in Bern is managed by Broker 130.92.65.40. We also statically created VPN tunnels between these private stub networks so as to allow transparent connections between them. A Bandwidth Broker is only expected to dynamically configure ingress edge router assuming that the routers along the way from source to destination have been pre-configured with CBQ or WFQ.

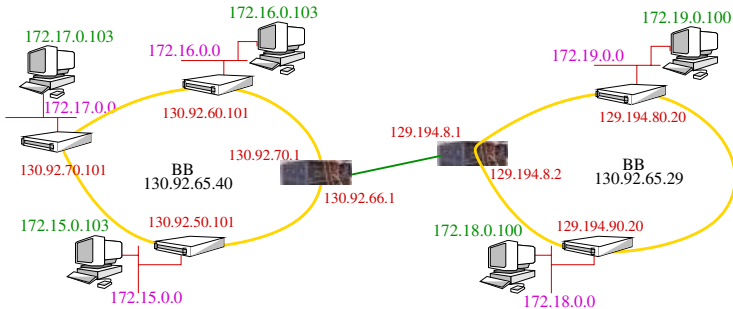


Fig. 8. Experimental Setup for Demonstration of Dynamic VLL Creation over Multiple Domains

Based on the setup as shown in Figure 8 we will now explain when a VLL is established over several Diffserv domains (Figure 9, 10). Assume that user *ibrahim* plans send traffic from 172.17.0.103 to 172.18.0.100. The broker 130.92.65.40 receives the request as: `newflow -u ibrahim -p ***** -s 172.17.0.103 -d 172.18.0.100 -b 3`. As the Broker realizes that 172.18.0.0 is in domain Geneva it performs admission control in its domain (i.e. at 130.92.66.1) and then send an inter-domain RAR to 130.92.65.29 in form of `newflow -bb 130.92.66.29 -p ***** -s 172.17.0.103 -d 172.18.0.100 -b 3 -tbb 130.92.65.29`. When the broker 130.92.65.29 receives this request it knows that the request has come from another neighbor Broker (because of the tagging -bb) and therefore checks the interISP SLA

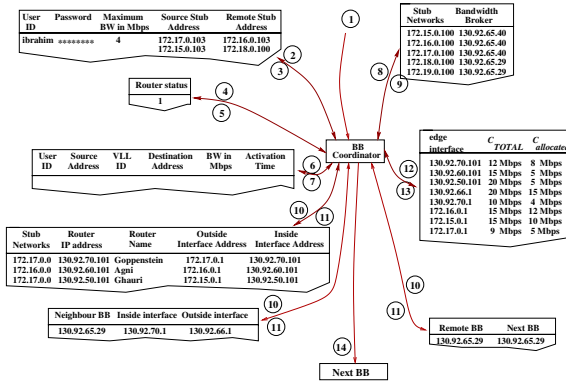


Fig. 9. An example of VLL Setup in Multi-Domain Scenario.

database to check the validity of the request. The Broker also identifies that 172.18.0.0 is located in the stub network attached to its domains since this is the final domain (from -tbb 130.92.65.29). While following the steps as described in the previous section it identifies the ingress and egress router interfaces to be 129.194.8.2 and 172.18.0.1, performs admission control on those and finally conveys the decision to the sender Broker 130.92.65.40. Upon receiving the decision Broker 130.92.65.40 talks to VLL ID database to pick up an ID, configures the edge router 130.92.70.101, and then conveys acknowledgment to the user.

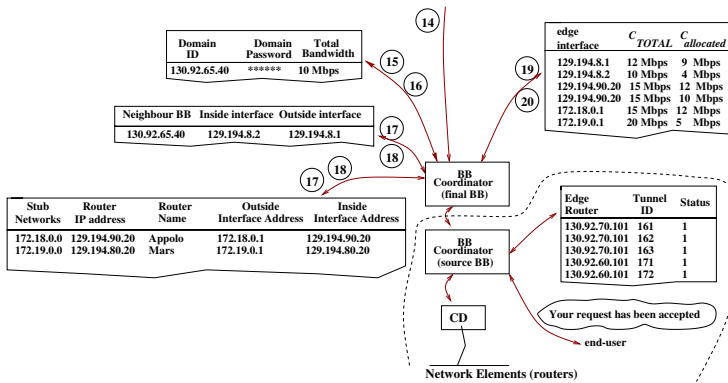


Fig. 10. Multi-domain VLL Setup Example Cont'd from Figure of 9

6 Conclusion

In this paper we have described the implementation of a Bandwidth Broker that uses a simple signaling mechanism to communicate with other cooperative

Brokers to enable customers to dynamically create VLLs over multiple Diffserv domains. We have presented a simple approach to make advance reservations in the absence of senders or receivers in a multi-domain scenario. Rather than using RSVP or COPS in inter-domain signaling to reserve capacity across domains, we used a novel method to identify domains, and hence Bandwidth Brokers that are responsible for maintaining them. A detailed implementation of the system and its operational details and some practical examples show how a simple resource reservation can be made dynamically over several cooperative Diffserv capable domains. Further simulation work might be useful to examine the scalability and effectiveness of our approach and is a topic of future research.

References

1. V. Jacobson, K. Nichols, and K. Poduri. An Expedited Forwarding phb, June 1999. RFC 2598.
2. S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, and W. Weis. An Architecture for Differentiated Services, December 1998. RFC 2475.
3. K. Nichols, Van Jacobson, and L. Zhang. A Two-bit Differentiated Services Architecture for the Internet, July 1999. RFC 2638.
4. Olov Schelen. *Quality of Service Agnets in the Internet*. PhD thesis, Lulea University of Technology, August 1998.
5. Andreas Terzis, Lan Wang, Jun Ogawa, and Lixia Zhang. A Two-Tier Resource Management Model for the Internet. In *IEEE Global Internet'99*, December 1999.
6. Benjamin Teitelbaum and et al. Internet2 QBone: Building a Testbed for Differentiated Services. *IEEE Network*, September/October 1999.
7. Ibrahim Khalil, Torsten Braun, and M. Günter. Implementation of a Service Broker for Management of QoS enabled VPNs. In *IEEE Workshop on IP-oriented Operations & Management (IPOM'2000)*, September 2000.
8. Hemann De Meer, Aurelio la Corte, Antonio Puliafito, and Orazio Tomarchio. Programmable Agents for Flexible QoS Management in IP Networks. *IEEE Journal on Selected Areas in Communications*, 18(2), February 2000.
9. Zhi-Li Zhang, Zhenhai Duan, Lixin Gao, and Yiwei Thomas Hou. Decoupling qos control from core routers: A novel bandwidth broker architecture for scalable support of guaranteed services. *ACM SIGCOMM 2000*, August 2000.
10. Ibrahim Khalil and Torsten Braun. Edge Provisioning and Fairness in DiffServ-VPNs. *IEEE International Conference on Computer Communication and Network (I3CN)*, Oct 16-18 2000.
11. I. Khalil and T. Braun. Implementation of a Bandwidth Broker for Dynamic End-to-End Resource Reservation in Outsourced Virtual Private Networks. *The 25th Annual IEEE Conference on Local Computer Networks (LCN)*, November 9-10 2000.
12. Y. Bernet, J. Binder, M. Carlson, B. E. Carpenter, S. Keshav, E. Davies, B. Ohlman, D. Verma, Z. Wang, and W. Weiss. A Framework for Differentiated Services. Internet Draft `draft-ietf-diffserv-framework-02.txt`, February 1999. work in progress.
13. SWITCH. The switchlan backbone. <http://www.switch.ch/lan/national.html>.

The Impact of Confidentiality on Quality of Service in Heterogeneous Voice over IP Networks

Johnathan M. Reason and David G. Messerschmitt

University of California
Berkeley, Electrical Engineering and Computer Sciences Department
Berkeley, CA 94720
{reason, messer}@eecs.berkeley.edu

Abstract. With the advent of ubiquitous access to multimedia content via wireless networks, users are more likely to have their data traverse a heterogeneous internetwork. Given the open nature of the Internet and wireless links, more users will demand end-to-end confidentiality. However, because of inherent error-expansion properties of secret-key encryption, it is a challenge to provide good subjective quality and end-to-end confidentiality for multimedia data, particularly in network environments subject to both loss and corruption impairments. This paper analyzes the affect end-to-end confidentiality has on the quality of service (QoS) in Voice over IP (VoIP) networks. To satisfy a given QoS objective, we show that mitigating the error-expansion caused by confidentiality comes at a cost. We measure this cost by increased delay, reduced bandwidth, and reduced traffic capacity. Thus, for this class of applications, we motivate the need for error-robust encryption and introduce one such technique in this paper.

1 Introduction

The Internet is emerging as the network of choice to advance the convergence of computer and telephony communications, and voice over IP (VoIP) is the predominate service leading the way. However, because the Internet was not designed with real-time traffic as a target, there are still a number of challenges to providing good voice quality. In particular, the lack of quality of service (QoS) guarantees for delay, jitter, packet loss, and bandwidth affect the voice quality attainable over the Internet. The Real-Time Transport Protocol (RTP) provides some functionality suited for carrying real-time content (e.g., a sequence number and timestamp) [6], but more work is needed to provide voice quality comparable to that of the Public Telephone Switching Network (PTSN).

In recent years, many authors have contributed to the literature regarding QoS for VoIP networks [8]. Most of these articles focus on mitigation techniques for the aforementioned impairments. With recent advances in robust, header

compression to promote wireless IP [7], VoIP over wireless is also viable. Thus, future VoIP networks will likely be heterogeneous networks.

Heterogeneity introduces another impairment, corruption, which can further degrade voice quality. Natural phenomenon present in wireless channels (e.g., multipath fading), but not in packet-switched networks, manifest this impairment. A heterogeneous VoIP network must contend with the combination of all these impairments.

This observation raises a special QoS challenge in providing both good subjective quality and end-to-end confidentiality for VoIP. In particular, secret-key encryption techniques have inherent error-expansion properties that can degrade subjective quality. Historically, the purpose of confidentiality has been viewed solely from the perspective of its strength. Improper deployment of confidentiality can lead to weakened security, and encryption that is not secure is useless. In this paper, we show that improper deployment of confidentiality can also lead to a trade-off in the QoS targets for delay, bandwidth and traffic capacity. While the primary concern will always be the strength of confidentiality, its impact on QoS is also of importance. We motivate the need for error-robust encryption to provide confidentiality in VoIP networks and introduced one suitable technique in this paper.

2 Error-Expansion Properties of Encryption

Secret-key algorithms perform bulk encryption of sensitive data in real-time applications such as VoIP. In contrast, because of their slower performance, public-key algorithms are usually reserved for non-real-time applications, such as providing confidentiality of sensitive credentials exchanged during authentication protocols.

We classify secret-key algorithms in two categories: block and stream. Block algorithms encrypt/decrypt blocks of bits at a time and stream algorithms encrypt/decrypt a single bit at a time. In the cryptography literature, the term *cipher* describes a family or grouping of algorithms. For example, the term *block cipher* refers to the grouping of a block encryption algorithm with a block decryption algorithm for enciphering and deciphering data. On the other hand, its plural, *block ciphers*, refers to the entire family of possible groupings. We will adhere to this convention in this paper. Furthermore, we call the device that performs encryption the *encryptor* and the device that performs decryption the *decryptor*. The original messages are called *plaintext* and the encrypted messages are called *ciphertext*.

2.1 Expansion of Bit Errors

Cryptographers design block algorithms to satisfy the *avalanche effect* [1]. The avalanche effect states that an average of one-half of the output bits should

change whenever a single input bit changes. Although it has not been proven that this property is a necessary condition for security, this property is exhibited by all block algorithms that have found their way into practice [2,3]. It is a desirable property because it says that each output bit must depend on all the input bits. In other words, an algorithm that has this property does not exhibit any statistical correlation between input and output that an adversary might use in an attack. The consequence of this property is that block ciphers multiply bit errors. That is, a single bit error in the ciphertext received at the input to the decryptor will result in multiple bit errors in the recovered plaintext at the output of the decryptor.

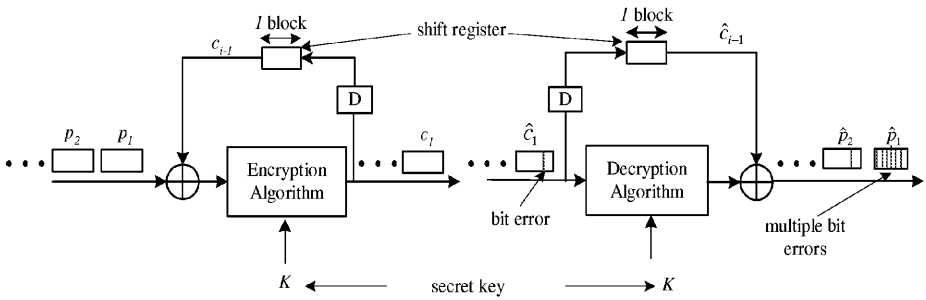


Fig. 1. Example of bit-error expansion for a block cipher in cipher block chaining (CBC) mode. Because of the avalanche effect, one or more bit errors in ciphertext block, \hat{c}_1 , leads to multiple bit errors in the corresponding recovered plaintext block, \hat{p}_1 .

Fig. 1 illustrates this property for a generic block cipher in cipher-block-chaining (CBC) mode. The encryptor produces N -bit ciphertext blocks then transmits them to the decryptor. During transmission, impairment along the path between encryptor and decryptor corrupts one bit of ciphertext block c_1 . Because of the avalanche effect, the decryptor multiplies this error by corrupting half of the recovered bits in plaintext block p_1 . Additionally, the next recovered block (p_2) will have a single corrupted bit in the same bit position as the bit error in c_1 . For a VoIP service using block encryption for confidentiality, in the worst case, if there is at least one bit error in every block of ciphertext, then the recovered voice would make a persistent static sound (i.e., pseudo random noise). It does not matter how badly a ciphertext block is corrupted; because of the avalanche effect, the decryptor will always garble the recovered plaintext block.

Because of this property, block ciphers are seldom used for real-time services in environments that have appreciable corruption (e.g., the wireless, land-mobile channel of cellular systems). However, block ciphers do perform well in envi-

ronments that have appreciable loss, but negligible corruption (e.g., landline, packet-switched networks). In the case of VoIP, the latter statement is true as long as the VoIP service maintains the block framing within a voice packet.

Although there are many modes of operation for block ciphers, there are four modes that have gained wide acceptance [4]. Each of these modes has a particular error structure [5]. We highlight CBC mode because it is most often used as the default operating mode for packet communications. In particular, the Data Encryption Standard (DES) in CBC mode is specified as the default encryption method for RTP communications [6].

2.2 Propagation of Synchronization Errors

In contrast, stream ciphers do not multiply bit errors, but do propagate synchronization errors caused by insertion or deletion of bits. That is, if the decryptor loses synchronization with the encryptor, the decryptor will garble all the recovered plaintext bits after the synchronization error until the system restores synchronization.

Fig. 2 illustrates this property for a *synchronous* stream cipher: a stream cipher that requires some external mechanism to maintain synchronization between encryptor and decryptor. The encryptor produces a stream of ciphertext bits (formatted as packets) then transmits them to the decryptor. Ciphertext packet c_1 is lost in transmission, thereby causing a synchronization error. Until the encryption system restores synchronization between the encryptor and decryptor, the decryptor will produce a pseudo-random stream of bits at its output. For a VoIP service using stream encryption for confidentiality, in the worst case, if the system does not restore synchronization, then the recovered voice from the voice codec would sound like static.

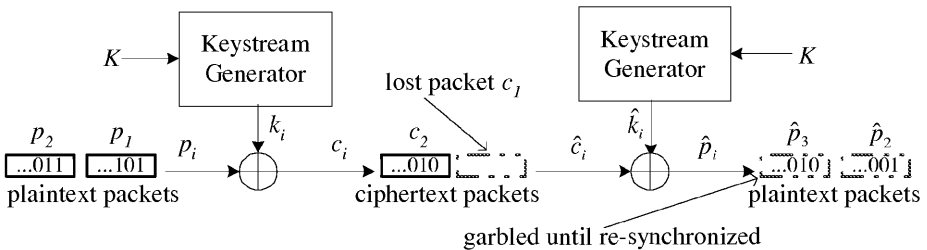


Fig. 2. Propagation of a synchronization error for a synchronous stream cipher. A lost ciphertext packet will cause loss of synchronization between encryptor and decryptor. The decryptor will garble all recovered plaintext packets ($\hat{p}_2, \hat{p}_3, \dots$) until re-synchronization.

Because of this property, environments where packet re-ordering is common and packet loss is appreciable confound the task of re-synchronization. Packet-switched networks exhibit these characteristics, particularly when data is transported using a real-time, unreliable, connection-less transport protocol (e.g., RTP/UDP). As a result, VoIP applications that provide confidentiality opt to avoid the synchronization issues of stream ciphers by exclusively using block ciphers. Stream ciphers do perform reasonably well over connection-oriented transport, where there might be appreciable loss and/or corruption, but no re-ordering of packets. Wireless links exhibit this combination of impairments. However, stream ciphers used in these environments often derive their synchronization means from physical-layer parameters that are not available to higher layers. For example, Bluetooth derives its cipher's synchronization from the master clock [9]. Moreover, the absence of packet re-ordering over the wireless link allows the use of heuristics to make reasonable guesses at the cipher synchronization bits, in the event they are corrupted by the channel.

2.3 Heterogeneity

In a heterogeneous network that includes wireless links, it is not clear which type of cipher to use for end-to-end confidentiality. In such an environment, the end-to-end path is subject to loss, corruption, and re-ordering of packets. Because of corruption, block ciphers are not well suited for this scenario. Similarly, the combination of these impairments exacerbate the synchronization problems of stream ciphers. In particular, many systems implement synchronization by providing cipher synchronization bits in each packet. As we show in Section 4, this method works fine in the absence of corruption, but when corruption is present this technique leads to degraded QoS. Thus, for VoIP over wireless, end-to-end confidentiality is a challenge.

3 Why End-to-End?

Confidentiality can be deployed on a link-by-link basis or on an end-to-end basis. For link-level confidentiality, each end-system and intermediate node (e.g., bridges, routers, and gateways) of each link performs encryption and decryption of each packet passing through that node. Whereas for end-to-end confidentiality, only the end-systems perform the encryption and decryption operations.

Link encryption is effective for point-to-point communications, where the two end-systems communicate directly with each other over a single link. However, this level of confidentiality does not scale well over many links and many networks. For example, over an end-to-end path that consists of N secure links, each packet would be encrypted and decrypted N times. Thus, link encryption adds processing delay that grows linearly with the number of secure links over the

end-to-end path. This increased delay can adversely affect the QoS in VoIP networks. In contrast, for end-to-end confidentiality, each packet is only encrypted and decrypted once, regardless of how many links comprise the end-to-end connection. Link encryption might also be effective when there are only a small number of untrusted links in the end-to-end path. Cellular telephone communications follow this scenario, where the wireless link between the base station and the cellular telephone is the only encrypted link. Voice data is sent in the clear over the backbone network (i.e., PTSN), which is assumed to be a trusted network. However, one can debate the validity of this assumption. Anyone with the right equipment can eavesdrop on a telephone conversation by tapping into the termination box, which is usually not secured and is located on the exterior of the telephone customer's premises.

End-to-end confidentiality is more appropriate for heterogeneous, VoIP networks because this approach assumes that the entire end-to-end path is untrustworthy. Additionally, unlike link encryption, it is flexible and scalable. End systems (which are usually maintained by end users) have more flexibility in upgrading to new technology than intermediate network nodes (which are usually maintained by system administrators).

4 Quantifying the Error-Expansion Properties

In this section, we summarize results that quantify the error expansion caused by block ciphers in CBC mode and synchronous stream ciphers. We refer the reader to [5] for the details of our analysis.

4.1 Preliminaries

For the discussion that follows, we assume uncorrelated bit errors as seen at the input to the decryptor. For a typical VoIP system, this is usually a valid assumption because bit errors at this point in the system are residual errors. Corruption enters the system as a result of channel phenomenon (e.g., multipath fading) experienced from any wireless links over the end-to-end path. However, the raw channel errors undergo mitigation techniques such as interleaving and forward error-correction (FEC). If deployed properly, these techniques tend to mask correlation in any residual errors. Even in a slow fading channel, where the fade length might exceed the interleaving depth, other diversity techniques, such as spread spectrum or multiple antennas, can be used to provide the receiver with uncorrelated samples of the signal [11].

For stream encryption, we assume the T^{th} packet contains N bits for cipher synchronization (where $T \in \{1, 2, \dots\}$) and packet loss is primarily caused by corrupt packet headers. We consider packet losses from congestion or re-ordering of packets negligible. Although these other loss mechanisms do exist in practice,

inclusion of them here confound the analysis. It suffices to say that any increase in packet losses beyond this limitation would further exacerbate error propagation.

4.2 Summary of Results

To quantify the error-expansion properties, we used a three prong approach: 1) we developed stochastic models that describe the error structure of the decryptors for each cipher, 2) using these models, we derived the statistics to describe the length and frequency of error events (defined below), and 3) from these statistics, we derived an expression for the average error expansion.

We define an error event as the state in which the decryptor garbles an integral number of consecutive blocks (or packets) at its output in response to one or more bit (or synchronization) errors at its input. Thus, an error event starts when the decryptor begins garbling recovered plaintext and ends when the decryptor stops garbling recovered plaintext. For block encryption in CBC mode, a corrupt ciphertext block starts an error event, which ends when the decryptor receives the next error-free ciphertext block. For synchronous stream ciphers, a lost ciphertext packet starts an error event, which ends upon re-synchronization. We measure the length of an error event in blocks for block ciphers and packets for stream ciphers.

If we consider the case of T equals one, then both these error structures can be modeled by a Markov Chain with two states: an error state and an error-free state. This stochastic process is stationary and ergodic, such that the mean time spent in the error state (\bar{H}) and mean time spent in the error-free state (\bar{R}) can be derived as

$$\bar{H} = \frac{1}{1 - \rho} \quad \text{and} \quad \bar{R} = \frac{1}{\rho}, \quad (1)$$

where ρ is the transition probability to the error state and $1 - \rho$ is the transition probability to the error-free state. The variance of these quantities is given by

$$\sigma_H^2 = \frac{\rho}{(1 - \rho)^2} \quad \text{and} \quad \sigma_R^2 = \frac{1 - \rho}{\rho^2}, \quad (2)$$

respectively. Alternatively, we can interpret \bar{H} as the mean length of an error event and \bar{R} as the mean length between successive error events.

For block ciphers in CBC mode, ρ is the probability the k^{th} ciphertext block c_k is in error. For synchronous stream ciphers, ρ is the probability that in the k^{th} ciphertext packet c_k at least one of the synchronization bits are in error. In both cases, we can express this quantity as

$$\rho = 1 - [1 - \overline{BER}_{in}]^N, \quad (3)$$

where \overline{BER}_{in} is the average, pre-decryption BER and N is the block size in bits (for block ciphers) or N is the number of synchronization bits in each packet (for stream ciphers).

Using these statistics, we can show that the average post-decryption BER (\overline{BER}_{out}) is given by

$$\overline{BER}_{out} = \frac{1 - [1 - \overline{BER}_{in}]^N}{2}. \tag{4}$$

For \overline{BER}_{in} small ($< 10^{-2}$), we can approximate this expression as

$$\overline{BER}_{out} \approx \frac{N}{2} \overline{BER}_{in}. \tag{5}$$

Re-arranging (5), we define the error expansion as

$$\delta \equiv \frac{\overline{BER}_{out}}{\overline{BER}_{in}} = \frac{1 - [1 - \overline{BER}_{in}]^N}{2 \overline{BER}_{in}} \approx \frac{N}{2}. \tag{6}$$

Fig. 3 plots (4) over a range of N and \overline{BER}_{in} . We observe that \overline{BER}_{out} , is directly proportional to N . Additionally, for given N , the error expansion is approximately constant for \overline{BER}_{in} less than 10^{-2} . The two most typical block sizes and number of synchronization bits are N equal to 64 and 128 bits.

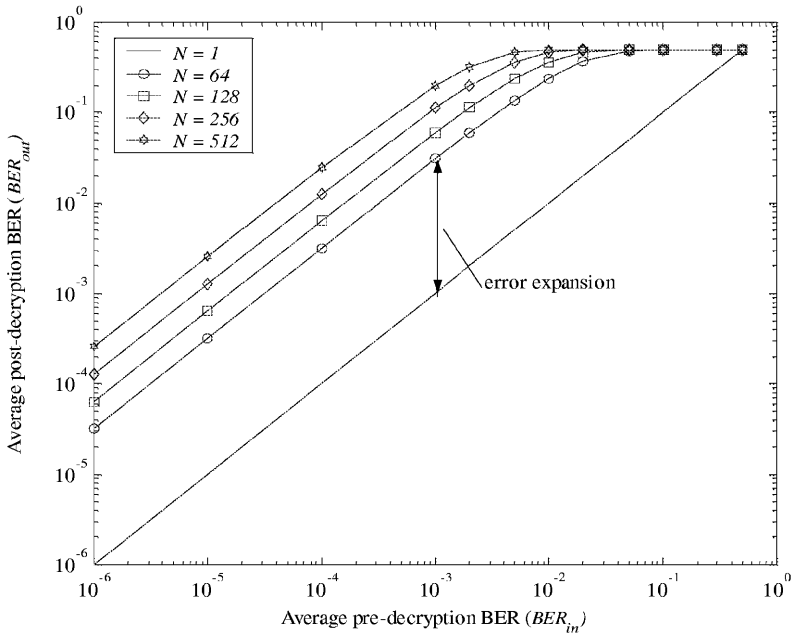


Fig. 3. Comparison of bit error expansion in CBC modes for varying block sizes. The amount of error expansion increases with block size.

Fig. 4 compares the theoretical \overline{BER}_{out} from (4) to an empirical \overline{BER}_{out} measured using DES in CBC mode. The figure shows error expansion of more than an order of magnitude for both cases. We obtained similar results for DES in 64-bit output feedback (OFB) mode, which can be shown to have identical error properties to any synchronous stream cipher that requires 64 synchronization bits [5].

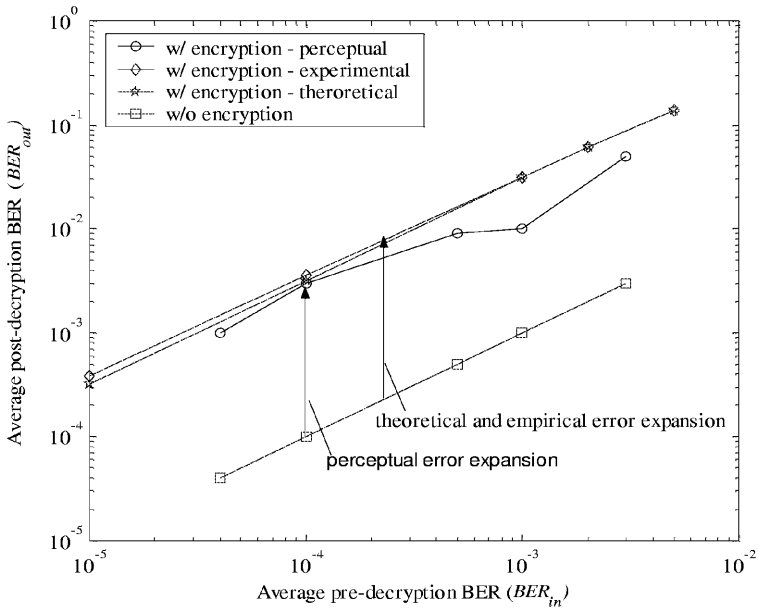


Fig. 4. Comparison of theoretical, experimental and perceptual bit-error expansion for block ciphers in CBC mode.

Fig. 4 also compares the results of a subjective assessment to the theoretical and empirical results. This assessment was conducted using the voice codec of the Global System for Mobile communications (GSM) [10]. The line with circular markers represents a set of twenty users mean opinion regarding their perception of degradation in the average, post-decryption BER. More precisely, the degradation they perceived to maintain the same subjective quality between GSM voice with and without encryption. These results also show error expansion of at least an order of magnitude. Similar results were obtained using 64 kb/s pulse code modulation (PCM).

5 Cost of Confidentiality

To provide good subjective quality for VoIP, it is necessary to define QoS targets for delay, jitter, bit-error rate, and packet-loss rate. We define a QoS target for a given parameter as the minimum (or maximum) value that is acceptable to the VoIP service. Satisfying these targets comes at a cost, which we measure in terms of resources (e.g., memory, processing cycles, power, frequency spectrum, traffic capacity, and channel capacity). For a given QoS target, we show that mitigating error expansion caused by confidentiality leads to increased consumption of resources.

5.1 Reduced Channel Capacity

Forward-error-correction (FEC) coding can mitigate the affect error expansion has on subjective quality. However, for a given modulation scheme, voice codec, and fixed frequency spectrum, applying FEC to compensate for error expansion reduces channel capacity. In particular, enough FEC coding must be applied to compensate for error expansion such that the post-decryption BER satisfies the BER target ($< 10^{-4}$ for VoIP). For a typical scenario (e.g., a 64-bit block cipher), the error expansion caused by confidentiality can be more than an order of magnitude. To improve the BER performance by an order of magnitude, communication systems typically require rate 1/2 (or smaller) FEC codes, especially for high BERs ($> 10^{-3}$). Thus, mitigating error expansion using FEC can reduce the channel capacity by a factor of two or more. In an environment where additional frequency spectrum is scarce, reduction in channel capacity is undesirable, especially for high-bitrate, multimedia applications.

5.2 Increased Delay

Adding any processing in the end-to-end path increases the overall delay. Thus, using FEC coding to mitigate error expansion will increase processing delay. Another way to avoid error expansion is to exclusively use stream encryption over wireless links and block encryption elsewhere. However, this is not an end-to-end approach. This approach requires transcoding at each wired/wireless node to convert from stream encryption to block encryption (and vice versa). Unfortunately, transcoding also increases the end-to-end delay. In delay-sensitive multimedia applications like VoIP, meeting the delay target (which is about 150 milliseconds) is paramount for good QoS. Managing delay for multimedia applications in IP networks is a challenge. Thus, it is desirable to find other means to compensate for error expansion that do not trade-off delay.

Additionally, transcoding can lead to weakened security. At each intermediate node that performs transcoding, the data is temporarily in the clear. Thus, an adversary who is trying to eavesdrop on a multimedia conversation has another

potential point of attack. These intermediate nodes might have vulnerabilities, such as swapping secret keys out of main memory to an insecure swap file.

5.3 Reduced Traffic Capacity

In wireless systems, one precious resource is traffic capacity, which is defined as the number of users the system can support simultaneously. In this section, we summarize results that show error expansion can reduce the traffic capacity of a direct-sequence, code-division, multiple-access (DS-CDMA) system by more than an order of magnitude.

We consider a DS-CDMA system that supports variable QoS for VoIP users via the optimal power control algorithm in [12]. The algorithm presented in this paper is optimum in the sense that it minimizes the interference each user experiences from other users (within a given cell), while satisfying each user’s reliability requirement. Following this paper, we consider multidimensional 8-PSK, which is one example of combined modulation and coding (with low complexity) that is compatible with the power control algorithm. We consider only two types of users: those with confidentiality and those without. Thus, the power control algorithm provides two distinct levels of reliability: \overline{BER}_1 for users without confidentiality and \overline{BER}_2 for users with confidentiality. There are a total of M users, M_1 users without confidentiality and with reliability requirement, \overline{BER}_1 and M_2 users with confidentiality and with reliability requirement, \overline{BER}_2 . We wish to derive an expression that relates the total system traffic capacity, M , to the fraction of users requiring confidentiality, M_2/M .

We want the power control algorithm to provide all users with the same subjective quality. Therefore, to compensate for error expansion, we use (6) to obtain

$$\overline{BER}_2 \approx \frac{\overline{BER}_1}{\delta}. \tag{7}$$

A feasible solution for this power control algorithm exists and this solution is unique and optimum if and only if

$$\sum_{m=1}^M \beta_m \alpha_m < 1, \tag{8}$$

where

$$M = \text{the total number of users}, \tag{9}$$

$$\beta_m = \begin{cases} 1, & \text{if user } m \text{ is transmitting} \\ 0, & \text{otherwise} \end{cases}, \tag{10}$$

$$\alpha_m = \frac{(E_b/N_0)_m}{G + (E_b/N_0)_m}, \tag{11}$$

$$(E_b/N_0)_m = \text{the energy per bit to interference for user } m, \quad (12)$$

$$G = \text{spread spectrum processing gain}. \quad (13)$$

This feasibility condition is identical for both uplink and downlink cases. Applying (8) to our setup and writing the equation in terms of $M1$ and $M2$ yields

$$M < 1 + \frac{G}{(E_b/N_0)_1} - M2 \left[\frac{\frac{(E_b/N_0)_2}{G + (E_b/N_0)_2}}{\frac{(E_b/N_0)_1}{G + (E_b/N_0)_1}} - 1 \right]. \quad (14)$$

Equation (14) assumes that G and $(E_b/N_0)_1$ are expressed in absolute units, not in decibels. Since it is often desirable to express parameters in a communication system in decibels, we can show that (14) is equivalent to

$$M < 1 - 10^{[G - (E_b/N_0)_1]/10} - M2 \left[\frac{1 + 10^{[G - (E_b/N_0)_1]/10}}{1 + 10^{\{G - [(E_b/N_0)_1 + \Delta]\}/10}} - 1 \right], \quad (15)$$

where G and $(E_b/N_0)_1$ are expressed in decibel units. The parameter Δ represents the perceptual, error expansion specified in (E_b/N_0) . That is, Δ is the additional (E_b/N_0) needed by users with confidentiality such that they can have the same subjective quality as users without confidentiality.

Equation (15) expresses the reliability requirement in terms of the energy-to-interference ratio. For a DS-CDMA system, we can approximate the sum of interference from other users and along multiple propagation paths as being additive white, Gaussian noise (AWGN) [12,13,14]. A fading channel and interference from adjacent cells complicates the analysis, but we can still achieve an error performance that approaches an AWGN approximation with the right techniques to mitigate distortion and provide diversity [11]. We use the AWGN formulation from [15] for 8-PSK modulation to map BER into (E_b/N_0) .

Capacity-Percent Confidentiality Curve. Fig. 5 illustrates a capacity-percent confidentiality cost curve for the two cases: high reliability with \overline{BER}_1 equal to 6×10^{-5} and low reliability with \overline{BER}_1 equal to 3×10^{-3} . We can interpret this figure as follows. For a fixed reliability requirement (and therefore fixed subjective quality), the capacity decreases as the fraction of users with confidentiality ($M2/M$) increases. When all users require confidentiality, the traffic capacity is at its minimum point. In contrast, the traffic capacity is at its maximum point when all users do not require confidentiality. Furthermore, the greater the perceptual error expansion, the faster the drop off in capacity. This follows our mathematical analysis well, since a larger Δ yields a larger multiplicative constant in the second term in (15). With a larger multiplicative constant, the traffic capacity drops off rapidly with more users requiring confidentiality.

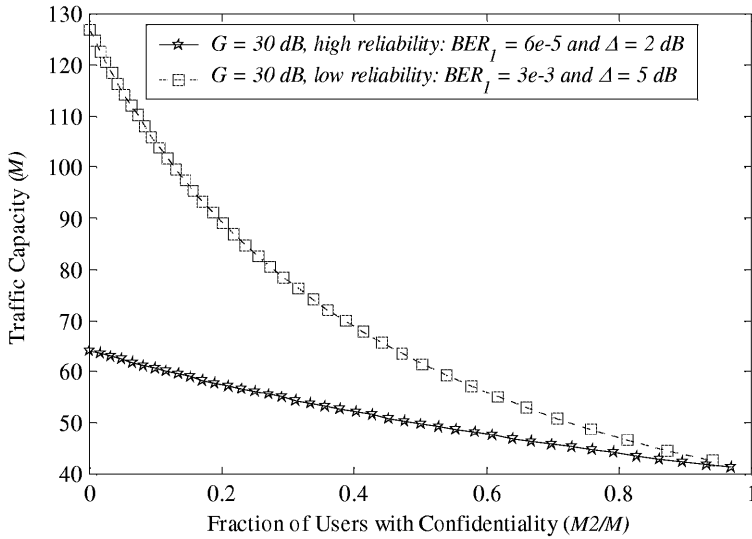


Fig. 5. Traffic capacity-percent confidentiality curve for two cases: 1) high reliability requirement (star markers) and 2) low reliability requirement (square markers).

In addition, the figure shows that the low-reliability curve has a steeper drop-off than the high-reliability curve. This results from the fact that, when all users do not require confidentiality, the system can support almost twice as many users at the lower reliability. Thus, there is more flexibility in resource utilization and admissions control when users require low reliability.

6 Error-Robust Encryption for Voice over IP

To eliminate this trade-off between confidentiality and QoS, we propose error-robust encryption. In this section, we introduce one such method called the automatic synchronization protocol (ASP), which is a technique that makes synchronous stream ciphers robust to synchronization errors.

6.1 Design Goal and Philosophy

Our goal is to design a synchronization technique for synchronous stream ciphers suitable for VoIP over wireless. Our approach to designing ASP is to use existing cryptographic mechanisms as much as possible, while achieving our design goal and satisfying the following properties:

1. Packets should not contain any additional bits specifically for cipher synchronization.

2. The technique should be memory less. That is, correct synchronization of the i^{th} packet should depend only on the integrity of the i^{th} packet's header and not on previous (or future) packets.
3. The security strength of the cipher should be no less than the strength of the underlying cryptographic mechanisms.
4. The design should be flexible enough to be used with any synchronous stream cipher or block cipher in OFB mode.
5. The design should be efficient.

6.2 Description

For each received ciphertext packet, ASP re-initializes the cipher to a different starting point by seeding it with a new key. Fig. 6 depicts how ASP maintains synchronization for synchronous stream ciphers. For each received ciphertext packet, the ASP module produces a d -bit secret key (K_i). Each K_i then re-initializes the keystream generator to a new starting point. ASP takes as input an s -bit sequence number (S_i) that it extracts from each packet header, an n -bit random initialization vector (IV), and a k -bit secret key ($K1$). Both IV and $K1$ are exchanged during session establishment.

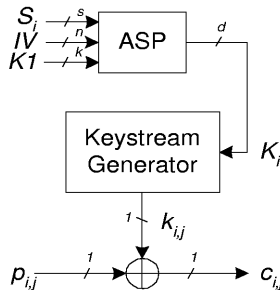


Fig. 6. ASP operation with a generic synchronous stream cipher. S_i is a s -bit sequence number for the i^{th} packet. IV is a n -bit random initialization vector. $K1$ is a k -bit secret key for session. K_i is a d -bit secret key for the i^{th} packet. $k_{i,j}$ is the j^{th} keystream bit for the i^{th} packet. $p_{i,j}$ is the j^{th} plaintext bit for the i^{th} packet. $c_{i,j}$ is the j^{th} ciphertext bit for the i^{th} packet.

ASP is a pseudo random number generator (PRNG) that produces a new secret key for each packet. It is based on existing cryptographic mechanisms, namely block ciphers in counter mode and hash functions. Using PRNGs for generating secret keys and IVs is common practice. The literature contains numerous PRNG methods that use existing cryptographic mechanisms [17,18,19, 20,21]. However, all the methods we reviewed were incompatible with our design

goal because they were designed for reliable transport. That is, they were designed with the intent that the sender would generate the pseudo random output, then reliably communicate that output to the receiver. This approach works fine during session establishment, but is inadequate for generating a pseudo random output to be used with each packet in a continuous stream of packets. Yarrow [17], discloses a well designed PRNG from a security perspective, which could be adapted to be compatible with our target applications. However, such a modified Yarrow PRNG would require additional state to maintain its own internal synchronization. Since our goal is to provide cipher synchronization for our target applications, we thought it best to devise a new method that is optimized in this sense.

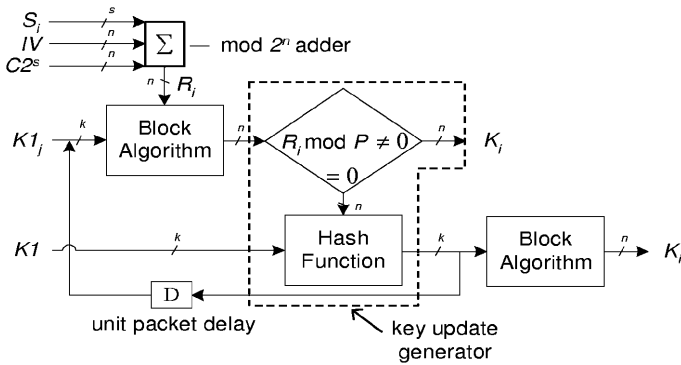


Fig. 7. Internal structure of ASP. S_i is a s -bit sequence number for the i^{th} packet. IV is a n -bit random initialization vector for the session. $K1$ is a n -bit secret key for the session. $K1_j$ is a n -bit secret key for packets between j^{th} and $(j + 1)^{th}$ update. $K1_0 = K1$. C is a sequence number overflow counter.

Fig. 7 illustrates the internal structure of ASP. There are three major components: 1) an n -bit block encryption algorithm with a k -bit key, 2) a modulo 2^n adder, and 3) a hash function. The block algorithm is configured in counter mode, where its input is the output of the adder block. The adder block is essential to the technique. Its inputs are chosen in such a way as to transform a rather small sequence number into a much larger n -bit index (R_i). This is done so that the pseudo random output of the block algorithm will have a long period (i.e., 2^n instead of 2^s). The hash function is used every P packets to update the j^{th} session key, $K1_j$. This key-update generator protects ASP output against key compromises. If an adversary compromises the j^{th} session key (for $j > 1$), this adversary will only recover data between the j^{th} and the $(j + 1)^{th}$ session keys.

6.3 Voice over IP Example

We implemented ASP in a VoIP application called Speak Freely [22]. Using simulated errors, PCM voice coding, and DES encryption in 64-OFB mode, we performed computer-to-computer VoIP communications over the Internet for three test cases: 1) no encryption, 2) 64-OFB encryption with 128-ASP and 3) 64-OFB encryption with 64 synchronization bits per packet. In Fig. 8, we plot the results of this experiment for the average, post-decryption BER (\overline{BER}_{out}) versus the average pre-decryption BER (\overline{BER}_{in}). The results for the test case with 128-ASP was indistinguishable from the test case without encryption. That is, there was zero empirical or perceptual error expansion with ASP. In contrast, the third test case exhibits more than an order of magnitude error expansion.

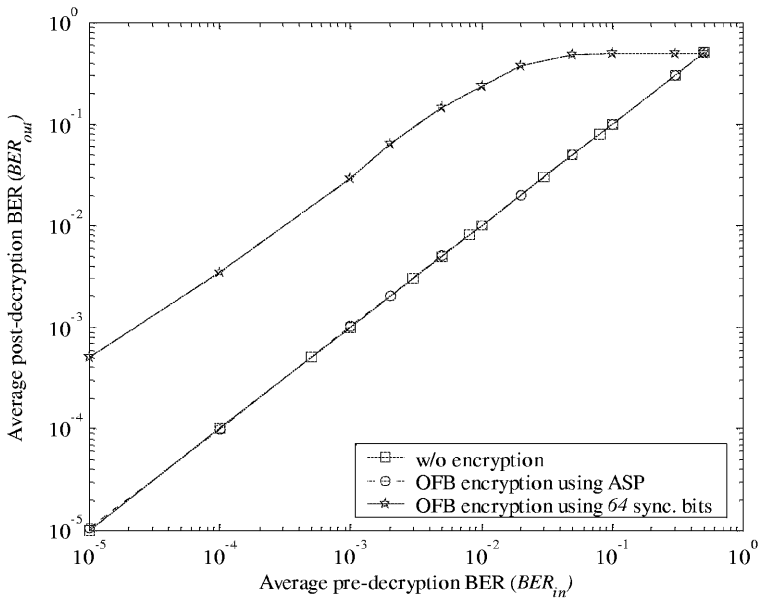


Fig. 8. The average post-decryption BER for three test cases: 1) no encryption (square markers), 2) 64-OFB DES encryption with 128-ASP (circle markers) and 3) 64-OFB DES encryption with 64 synchronization bits per packet (star markers).

7 Conclusions

In this article, we showed how the error properties of secret-key ciphers leads to error expansion for real-time, multimedia applications, particularly in heterogeneous networks. For typical secret-key ciphers, this error expansion can be more

than an order of magnitude. For this class of applications, we showed that traditional techniques to combat corruption are inadequate because they consume precious resources. Alternatively, we proposed error-robust encryption and introduced one technique called the automatic synchronization protocol (ASP). This technique is a flexible synchronization means that makes synchronous stream ciphers robust to errors. ASP was designed to work with any synchronous stream cipher or block ciphers configured in OFB mode. We demonstrated its utility for computer-to-computer VoIP communications, but it should also be applicable to digital IP telephones. We hope this article will encourage the wireless networking and multimedia applications communities to view security at the system level, from the perspective of good QoS as well as strong security.

References

1. A. Webster and S. E. Tavares, On the design of S-boxes, in H. C. Williams (Ed.): *Advances in Cryptology - Proc. of CRYPTO '85*, Springer-Verlag, NY, pp. 523-534, 1986.
2. National Institute for Standards and Technology, Data Encryption Standard (DES), FIPS PUB 46-2, US Department of Commerce, Dec. 30, 1993.
3. X. Lai, On the design and security of block ciphers, *ETH Series in Info. Proc.*, vol. 1, Konstanz: Hartung-Gorre Verlag, 1992.
4. National Institute for Standards and Technology, DES Modes of Operation, FIPS PUB 46-2, US Department of Commerce, Dec. 2, 1980.
5. J. Reason, *End-to-end Confidentiality for Continuous-media Applications in Wireless Systems*, Dissertation, UC Berkeley, December 2000.
6. S. Casner, R. Frederick, and V. Jacobson, RTP: A Transport Protocol for Real-Time Applications, IETF, RFC 1889, January, 1996.
7. C. Bormann et al, RObust Header Compression (ROHC): Framework and four profiles: RTP, UDP, ESP, and uncompressed, IETF, draft-ietf-rohc-rtp-09, Feb. 2001.
8. Bo Li; Hamdi, M.; Dongyi Iang; Xi-Ren Cao; Hou, Y.T., QoS enabled voice support in the next generation Internet: issues, existing approaches and challenges, *IEEE Communications Magazine*, Volume: 38 Issue: 4, April 2000
9. The Bluetooth Special Interest Group, Bluetooth Specification, <http://www.bluetooth.com>, Dec. 1999.
10. Recommendation GSM 06.10, GSM Full Rate Speech Transcoding, ETSI, Feb. 1992.
11. B. Skylar, Rayleigh fading channels in mobile digital communication systems part ii: mitigation, *IEEE Communications Magazine*, pp. 102-109, July 1997.
12. L. Yun and D.G. Messerschmitt, Power control and coding for variable QOS on a CDMA channel, *Proc. IEEE MILCOM Conf.*, Fort Monmouth, NJ, Oct 2-4, 1994.
13. R. L. Pickholtz, et. al., Spread spectrum for mobile communications, *IEEE Transactions on Vehicular Technology*, Vol. 40, NO. 2, pp. 313-321, May, 1991.
14. M. B. Pursley, Performance evaluation for phase-coded spread-spectrum multiple-access communication-part I: system analysis, *IEEE Trans. on Comm.*, Vol.COM-25, NO. 8, pp. 795-799, August 1977.

15. P. J. Lee, Computation of the bit error rate of coherent M-ary PSK with Gray Code bit mapping, IEEE Trans. on Comm., Vol. COM-34, NO. 5, pp. 488-491, May, 1986.
16. B. Schneier, J. Kelsey, D. Whiting, D. Wagner, C. Hall, and N. Ferguson, The Twofish Encryption Algorithm, John Wiley & Sons, 1999.
17. J. Kelsey, B. Schneier, and N. Ferguson, Yarrow-160: Notes on the Design and Analysis of the Yarrow Cryptographic Pseudorandom Number Generator, Sixth Annual Workshop on Selected Areas in Cryptography, Springer Verlag, August 1999.
18. ANSI X9.17 (Revised), American national standard for financial institution Key management (wholesale), American Bankers Association, 1985.
19. P. Gutmann, Software generation of random numbers for cryptographic purposes, Proceedings of the 1998 Usenix Security Symposium, 1998.
20. National Institute for Standards and Technology, Key Management Using X9.17, FIPS PUB 171, US Department of Commerce, 1992.
21. P. Zimmermann, The Official PGP User's Guide, MIT Press, 1995.
22. J. Walker and B. C. Wiles, Speak Freely, <http://www.speakfreely.org>, 1999.

Queue Length Based Fair Queueing in Core-Stateless Networks¹

Mingyu Zhai, Guanqun Gu, and Yuan Yuan

Department of Computer Science and Engineering, Southeast University, Nanjing 210096, China

The Key Laboratory for Computer Networks and Information Integration, Ministry of Education, China

{myzhai, ggq, yyuan }@seu.edu.cn

Abstract. In this paper, we present Queue Length based Fair Queueing (QLFQ), a scheme to approximate fair bandwidth allocation without per-flow state. We have evaluated QLFQ together with CSFQ and RFQ with several different configurations and traffic sources. The simulation results show that QLFQ is able to achieve approximately fair bandwidth sharing in all of these scenarios. The performance of QLFQ is comparable to that of CSFQ, and it performs much better than RFQ.

1 Introduction

Many router mechanisms [1, 2] have been developed to achieve fair bandwidth allocations. Generally, most of these mechanisms need to maintain per-flow state and therefore are more complex and less scalable than simple FIFO queueing when they are used in the interior of a high-speed network. Consequently, design of algorithms that can provide per-flow fair rate without requiring per-flow functionality in the network core routers has become an active area of research.

Recently several frameworks and mechanisms have been proposed to overcome the implementation complexity problem and address the scalability and robustness requirements. Stoica *et al.* propose a new scheme called Core-Stateless Fair Queueing (CSFQ)[3], which approximates fair queueing while limiting flow state to the edge of network and removing it from core routers. Cao *et al.* proposed Rainbow Fair Queueing (RFQ) [4], a similar scheme that avoids fair share rate calculation in core routers and that is better adapted to layered encoding applications. Both schemes remove flow state but still require *computation* to determine dropping thresholds for example.

RFQ is a combination of a layering scheme and a buffer management mechanism. This is the direction we pursue in this paper, where we take it to its simplest configuration. In our approach, in order to determine the dropping threshold, we *only*

¹ This work is supported by Chinese NSF (item number: 69896240).

need to know queue occupancy information (i.e. Queue Length) and do few *comparison* operations in core routers (so we call our scheme Queue Length based Fair Queueing (QLFQ)). Compared to RFQ and the other approaches of achieving reasonable fairness, we believe that our scheme has one of the lowest implementation cost in core routers. The simulation results demonstrate that the performance of QLFQ outperforms RFQ and is comparable to CSFQ.

2 Queue Length Based Fair Queueing (QLFQ)

2.1 Network Architecture

The network architecture is the same as that used in CSFQ, RFQ and in Differentiated Services: namely, a network comprised of edge routers and core routers. The edge routers perform packet classification and encode certain state in packet headers, and the core routers use the state for packet discarding.

2.2 Computation of Flow Arrival Rate

At the edge routers, the flow arrival rate must be estimated, in order to assign a label to a packet. To estimate the flow arrival rate, we use the same exponential averaging formula as in [3,4]. This scheme requires an edge router to keep state for each active flow. Specifically, let t_i^k and l_i^k be the arrival time and length of the k^{th} packet of flow i . The estimated rate of flow i , γ_i , is updated every time a new packet is received:

$$\gamma_i^{\text{new}} = (1 - e^{-T_i^k / K})(l_i^k / T_i^k) + e^{(-T_i^k / K)} \gamma_i^{\text{old}} \quad (1)$$

where $T_i^k = t_i^k - t_i^{k-1}$ and K is a constant. Using an exponential weight $e^{-T_i^k / K}$ gives more reliable estimation for bursty traffic, even when the packet inter arrival time has significant variance.

2.3 Flow Layering

Flow layering is an important part in both QLFQ and RFQ, by which edge routers partition a flow and assign a number to each layer of the flow. The numbered layers have two purposes. First, they reflect the rate of the flow: the larger the number of layers is, the higher the rate of the flow is; flows with the same rate have the same number of layers. Second, the numbered layers provide a structure for controlled discarding in the network when congestion occurs. That is, when congestion is detected, the core routers will first discard the largest layer and then the second largest layers and so on.

The preliminary requirement of flow layering is to select layer rates for a flow. If a flow has an arrival rate r , and c_i is the rate of layer i , then the flow will be partitioned

into $(j + 1)$ layers, where j is the smallest value satisfying $\sum_{i=0}^j c_i \geq r$. Contrary to RFQ, which uses a *non-linear* layering scheme, QLFQ uses a *linear* layering scheme. More precisely, QLFQ make all layers have *equal* rate. Specifically, if P is the maximum flow rate in the network and N is the total number of layers, the layer rate c is P/N . For example, if a flow has a maximum rate of 16Kbps, and RFQ partitions it into eight layers, then the rate for each layer is 1, 1, 1, 1, 2, 2, 4, 4 (unit: Kbps). Under the same conditions, for QLFQ the layer rate will be 2, 2, 2, 2, 2, 2, 2, 2 (unit: Kbps). Apparently the QLFQ scheme can simplify operations of edge routers and is amenable to hardware implementation.

2.4 Packet Labeling

After estimating the flow arrival rate, each packet is assigned a label, with the constraint that the average rate of packets in each layer is at most c . In QLFQ, to label a packet we use a scheme similar to that of RFQ. But our scheme requires lower computational overhead than RFQ because of the simple flow layering scheme used in QLFQ. Specifically, if the current estimate of the flow arrival rate is r , and j is the smallest value satisfying $j * c \geq r$. Then the current packet is assigned a label $i : 0 \leq i \leq (j-1)$ with probability $1/j$. Whether all packets have fixed size or not, it is easy to see that the probabilistic label assignment will cause the rates for each layer to approach the required rates c . Moreover, the probabilistic label assignment can smooth a burst of packets and avoid temporary buffer overflow.

2.5 Core Router Algorithm

Using QLFQ core routers can achieve approximate fair queueing while maintaining high utilization. In order to do so, the core routers need to monitor queue occupancy and accordingly set a dropping-threshold. When network is congested, the core routers begin to discard layers whose value exceeds the dropping threshold.

The main task of the core router algorithm is to update the dropping threshold according to the current congestion status. The principle behind this core router algorithm is that the queue occupancy reflects the congestion status of the core router, so we can use the queue length information to set the dropping threshold. Namely, the dropping threshold $Drop_thresh^{new}$ is updated as following:

$$Drop_thresh^{new} = f(Drop_thresh^{old}, qlen^{old}) \quad (2)$$

where $Drop_thresh^{old}$ and $qlen^{old}$ are the dropping threshold and queue length before a new packet arrives respectively.

According to the above principle, in the current implementation of QLFQ we logically divide the queue of an output link into four segments using three queue length thresholds (see Figure 1):

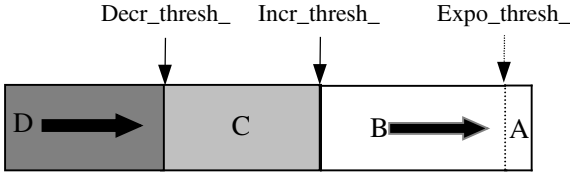


Fig. 1. Queue Partition

Segment A where queue length $qlen$ is less than $Expo_thresh_$ - Because the queue is nearly empty in the segment, we drastically increase the dropping threshold and make the aggregate arriving rate match the link capacity as soon as possible.

Segment B where $qlen$ lies between $Expo_thresh_$ and $Incr_thresh_$ - In the segment, the link is uncongested and the buffer capacity has not been sufficiently utilized, so we smoothly increase the dropping threshold.

Segment C where $qlen$ stays between $Incr_thresh_$ and $Decr_thresh_$ - With both the link and buffer capacity at full use, we keep the dropping threshold unchanged and make the system work steadily.

Segment D where $qlen$ is larger than $Decr_thresh_$ - When the queue length exceeds $Decr_thresh_$, the link becomes congested, and we reduce the dropping threshold to get more packets dropped.

Initially, the dropping threshold $drop_thresh$ is set to the maximum layer value Max_layer . When the current queue is in Segment A and its length $qlen$ has been reduced since the last $drop_thresh$ update, the new $drop_thresh$ is computed as:

$$drop_thresh^{new} = drop_thresh^{old} + 2^i \tag{3}$$

where i is the number of *continually* arrived packets under the above conditions. And also we track the variation of current queue length $qlen$ so as to prevent $drop_thresh$ from changing very frequently and ensure stability. If the current queue stays between $Expo_thresh_$ and $Incr_thresh_$ and its length $qlen$ has been reduced since the last $drop_thresh$ update, we only increase $drop_thresh$ linearly using a simple way as following:

$$drop_thresh^{new} = drop_thresh^{old} + 1 \tag{4}$$

We decrease $drop_thresh$ when the current queue length $qlen$ is larger than $Decr_thresh_$. But here we discuss it with two aspects: (i) the queue is not full and the current queue length has increased since the last $drop_thresh$ update - Under the condition, if $drop_thresh$ equals to Max_layer , we set $drop_thresh$ to $current_max_layer$, which is updated to record the largest layer number *recently* seen when the queue is not in the Segment D. Subsequently, we linearly decrease $drop_thresh$ as following:

$$drop_thresh^{new} = drop_thresh^{old} - 1 \tag{5}$$

(ii) the queue is full – Under this condition, we need reduce $drop_thresh$ quickly to alleviate congestion. So we first use a non-linear formula to compute $drop_thresh$:

$$drop_thresh^{new} = drop_thresh^{old} - \frac{MAX(1, 0.1 * drop_thresh^{old} * drop_thresh^{old})}{Max_layer} \quad (6)$$

Secondly, we set $drop_thresh$ to $current_max_layer$ if $drop_thresh$ is still larger than $current_max_layer$. To show the efficiency of expression (6), we give an example. Assuming that the maximum layer number is 10000, and the current dropping threshold is 8000, then the new dropping threshold is 7360. If the current dropping threshold is 1000, then the dropping threshold is reduced to 990. That is, the larger the dropping threshold is, the more it is reduced.

At core routers, both the time and space complexity of QLFQ are constant with respect to the number of competing flows, and thus QLFQ could be implemented in very high speed core routers. In fact, In core routers QLFQ only needs queue occupancy information and conduct few comparison operations to determine the dropping threshold in the steady status. It should be pointed out that the *multiplication* operation in expression (6) is conditioned, and there is little probability to execute it unless the traffic is very bursty (the *division* operation in (6) can be avoided because Max_layer is a constant). Our simulations also prove this point.

3 Conclusion

In this paper, we present Queue Length based Fair Queueing (QLFQ), a scheme to approximate fair bandwidth allocation without per-flow state. We believe that, by its simplicity and efficiency, QLFQ is an interesting approach for future high-speed network.

References

1. Parekh, A. A generalized processor sharing approach to flow control - the single node case. In : Proceedings of the IEEE INFOCOM'92, 1992.
2. Lin, D. Dynamics of random early detection. In : Proceedings of ACM SIGCOMM '97,Cannes, France, October 1997, 127-137
3. Stoica, I. Core-stateless fair queueing: Achieving approximately fair bandwidth allocations in high-speed networks. In : Proceeding of ACM SIGCOMM'98,1998,118-130
4. Cao, Z. Rainbow fair queueing: Fair bandwidth sharing without per-flow state. In : Proceedings of IEEE Infocom '2000,March 2000

Study of TCP and UDP Flows in a Differentiated Services Network Using Two Markers System¹

Sung-Hyuck Lee, Seung-Joon Seok, Seung-Jin Lee, and Chul-Hee Kang*

Department of Electronics Engineering, Korea University
5-Ka, Anam-dong, Sungbuk-ku, 136-701, Seoul, Korea
{starsu, ssj, linuz}@widecomm.korea.ac.kr, *chkang@korea.ac.kr

Abstract. There are two cruxes of issues identified in differentiated services (diffserv) networks: One is TCP dynamics over Assured Services, and the other is the interaction of TCP and UDP flows for Assured Forwarding Per Hop Behavior (PHB). Therefore, we argue unfair distribution of excess bandwidth in an over-provisioned networks as well as unfair degradation in an under-provisioned network for TCP and UDP flows traffic. First, we consider *Two Markers System (TMS)* that we have proposed, which is able to properly mark packets and fairly share the bandwidth to each flow for their targeted sending rates. Next, we present simulation results to illustrate the effectiveness of TMS scheme over TCP and UDP interaction. That is, flows in Two Markers System somewhat fairly share the excess bandwidth and experience degradation in proportion to their target rates.

1 Introduction

TCP's complex response primarily to packet losses in a differentiated Services Network affects the Assured Services. TCP reacts to congestion by halving the congestion window (cwnd) and increases the window additively when packets are delivered successfully[1]. However, in the diffserv network these additive-increase and multiplicative-decrease make it hard to protect the reservation rate for Assured Services. When TCP reacts to an OUT packet drop by halving its congestion window and increases additively, it may not reach its reservation rate. In [2], in order to alleviate the issue, it focused on several strategies used to mark packets in order to consider TCP dynamics and adapt fairness for sharing a bottleneck link of a network, and proposed a modified marking scheme, so called, Two Markers System (TMS); the first marker (TMS_I) is located at sources of a network to adapt TCP congestion control algorithm, and the second marker (TMS_II) at edge to fairly mark the aggregated flow as shown Figure 1. In addition, one of the cruxes identified in the diffserv network is the effect of congestion insensitive flows such as UDP when they share the same AF class with TCP

¹ This research was supported in part by Information and Telecommunication Research Center(ITRC) contracts. We are grateful to Asian Pacific Advanced Network-Korea Consortium(APAN-KR) members for technical discussion on Diffserv related issues.

flows. TCP and UDP interaction for the AF PHB have become the important issue in the fairness of diffserv context.

In this paper, we take the problem into consideration between the transport control protocol (TCP and UDP) and the differentiated drop policies of the network in realizing the reserved throughputs, using modified scheme called, Two Markers System for improving the realization of target rates in a differentiated services network.

The rest of this paper is organized as follows: Section 2 reviews the state of the art in the Two Markers System. Section 3 explores for responsive traffic flows such as TCP and non-responsive traffic flows such as UDP interaction and presents the results using TMS algorithms in simulated environments, and performs analysis for simulated results. Section 4 concludes our work.

2 Two Markers System

This system has two marking modules that are located in the source and at the edge of differentiated services network, respectively, and each marking module plays different roles to achieve the reservation rate and the target rate of Assured Services. First, a virtual- source marker (TMS_I) carries out two main roles: One is to control a flow and congestion, so called *suitable-marking strategy* and the other is to give the marking probabilities to the edge-embedded marker (TMS_II). In [3], it showed that a source-integrated packet marking engine (PME) properly kept up marking rate rather than a source-transparent PME, because the measurement of throughputs against the reservation rate at the source is accomplished more exactly than at the edge of a network. Therefore, TMS_I decreases TCP impacts in the underlying AF services, and helps the TMS_II to properly mark packets. So to speak, TMS_I can be not a marker used to mark packets in order to classify the service in the core of a network, but an indicator that notifies TMS_II in the edge of a network of the marking rate. Second, the edge-embedded marker (TMS_II) elaborates a fairness strategy for sharing excess bandwidth of a bottleneck link called *marking rate-based fairness*, so it monitors incoming traffic flows from users at the edge of the network against the profile that the users have agreed with the service provider. It measures the number of the marked packets (m_i) from sources and partly re-marks aggregated flows for the profile that the users have agreed with the service provider. IN marking (In-profile), for example, may change into OUT (Out-of-profile) or reverse. Therefore, a datum point of fairness strategy is the marking information (X_{mi}) of traffic flows from users, as follows:

$$X_{mi} = m_i / (2E[m_i]) \quad (1)$$

where $E[m_i]$ represent the average marking rate of all the flows at the edge of network. Therefore, X_{mi} is used in the computation of a flow's target rate(T_i) in the edge of a network, as follows:

$$T_i = R_i + X_{mi} (C - \sum R_i) \quad (2)$$

where, C and R_i represent the capacity of a bottleneck link in the network and reservation rate of each flow, respectively.

3 Interaction of TCP and UDP Flows

In this section, we present the simulation results for TCP and UDP traffics in Two Markers System we have described in the previous section. The simulation was done with the Network Simulator-v2(ns-v2.1b8a). For the sake of simulation, we used a network with the configuration shown in Figure 1. In the simulation, we have 6 sources (1 through 6 counting downwards) that communicate with one of six different destinations. Long-lived packet streams generated by an infinite file transfers are originated at source 1 through 4, and destined to source 7 through 10. Constant rate UDP packet streams are originated at source 5-6, and destined to source 11-12. We carried out two scenarios: over-provisioned and under-provisioned network. In the first scenario, the aggregate reservation rate is 6Mbps, and the bottleneck capacity is set to 8Mbps so that the bottleneck is not oversubscribed. In the second scenario, the aggregate reservation rate is 6Mbps, and the bottleneck capacity is also set to 3Mbps so that the bottleneck link experiences congestion. The UDP flows source traffic at the rate of 1Mbps. We assume that the RTT without queuing delay of each flow is randomly pocked from 80 to 220 ms. The sources 1 through 4 are all TCP-Reno sources (unless specified otherwise). For the RIO implementation, the routers use RED with the values of 200 packets, 400packets, and 0.02 for min_in , max_in , and P_{max_in} and 50 packets, 100 packets and 0.5 for min_out , max_out , and P_{max_out} .

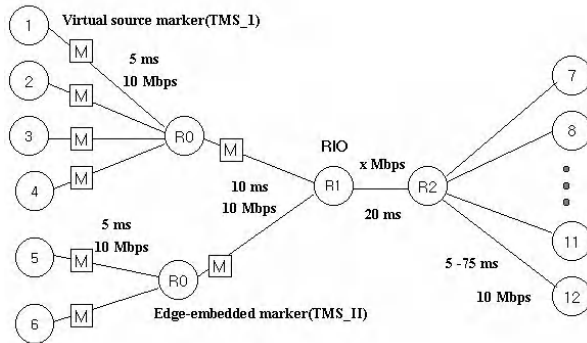


Fig. 1. Simulation topology using the Two Markers System

First of all, we investigate the simulation results of the issues for sharing excess bandwidth according to the link utilization. We define the link utilization(ρ), as follows:

$$\rho = \sum T_i / C \tag{3}$$

where T_i and C represent target rate of a flow and the capacity of a bottleneck link in the network, respectively. As the network approaches an under-provisioned state, the TCP flows suffer acute degradation compared to UDP flows and fall under their target rates at ($> 80\%$) as shown in Figure 2(a). However, UDP flows meet their target rates and the only degradation they experience is reduction of excess bandwidth they receive, and in the figure 2 the excess bandwidth a flow receives is expressed as percentage of its target rate. We assume as follows: (i) UDP flows in TMS are only dealt with OUT marking (out-of-profile), (ii) the magnitude of marking rate represents increase or decrease in demand for bandwidth. If the number of marked packets, for example, exceeds the threshold value, that is the average marking rate, $E[m_i]$, the edge-embedded marker considers that the flow wants more bandwidth than others in order to achieve its reservation rate. Therefore, the flow is marked more and has a higher target rate than others. The distribution of excess bandwidth in the over-provisioned network is more fair than in the Figure 2(a). That is, the idea behind TMS is to preferentially drop UDP and TCP packets marked OUT which are outside of their contract when congestion occurs. The excess bandwidth in the Figure 2(b) is percentage-wise almost equally divided between TCP and UDP flows. Both traffic types achieve their target rates in the over-provisioned network, and both suffer the same level of service degradation in the under-provisioned network.

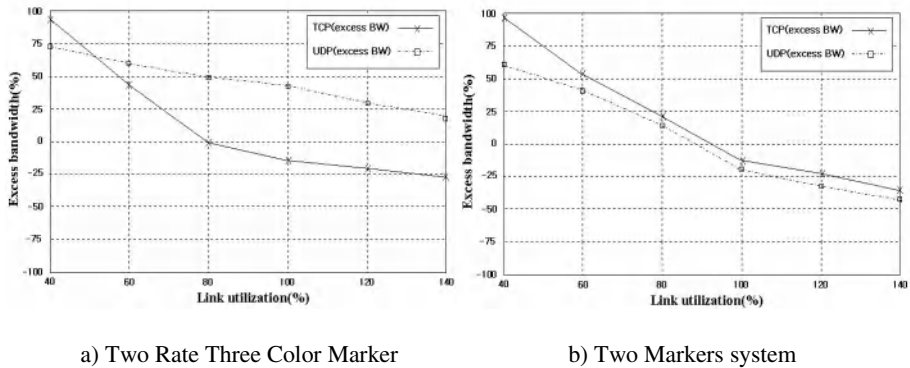


Fig. 2. Sharing excess bandwidth according to the link utilization

Next, we present the simulation results considering of marking rate-based fairness strategy described in the previous section. We set that reservation rate of each flow is 1Mbps, and compare two marking schemes: One is the TRTCM, the other is marking strategy of TMS. As stated above, the target rate of a flow i in TMS varies in proportion to the probability of marking from the sources. Figure 3 represents the results that the throughputs of all individual flows of aggregated traffic realize their target rates in over-provisioned networks. In the figure 3(a), UDP flows captured all excess bandwidth, but TCP flows in the figure 3(b) fairly share excess bandwidth elaborating *the marking rate-based fairness strategy* against UDP flows. Each flow also satisfies its reservation rate and shares the excess bandwidth of the bottleneck link according to

the probability of his marking in over-provisioned network. Each TCP flow in Figure 3 (a) often fails to achieve their target rates in under-provisioned networks, because UDP flows are transmitted constantly irrelative to congestion. That is, UDP gains unfairly at the advantage of TCP flows. However, Figure 3 (b) shows that each flow in under-provisioned network is distributed fairer than in the Figure 3 (a), by dint of dealing with UDP as out-of-profile.

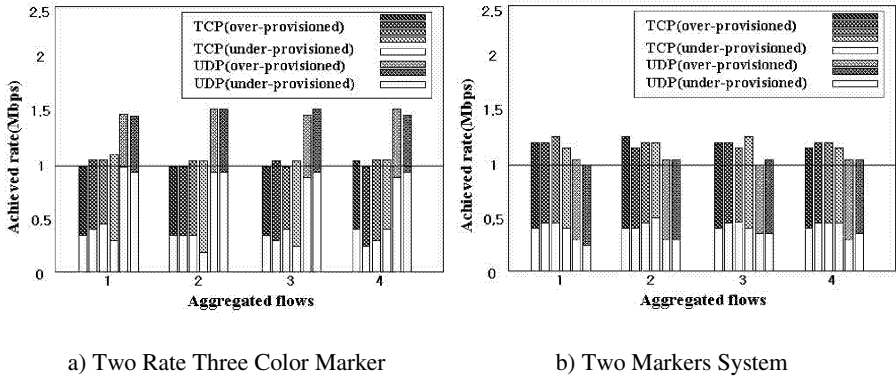


Fig. 3. Throughput according to network conditions

4 Conclusion

We have described the analysis for the interaction of TCP and UDP flows in a diffserv Network using the Two Markers System. We have also simulated a TMS model to study the effects of several factors on the throughput rates of TCP and UDP flows in a RIO-based Differentiated Services network. First, in over-provisioned network, as TMS elaborates *the marking rate-based fairness*, TCP could fairly share excess bandwidth that UDP almost dominated and achieve their target rates. Next, although all the flows couldn't achieve their target rates, they fairly experienced degradation in proportion to their target bandwidth.

In the near future, we will study the effect of the interaction of TCP and UDP flows for values of scaling factor, α , β , and γ issued in the Two Markers System.

References

1. V. Jacobson and M. Karels: Congestion Avoidance and Control, in Proc. SIGCOMM'88, Stanford, CA, August (1998) 314-329
2. S. H. Lee, S. J. Seok, S. J. Lee, and C. H. Kang: Two-differentiated Marking Strategies for TCP flows in a Differentiated Services Network, unpublished manuscript, April. (2001)
3. W. Feng, D. Kandlur, D. Saha, and K. Shin: Adaptive Packet Marking for Providing Differentiated Services in the Internet, In Proc. ICNP'98, Austin, TX, Oct. (1998) 108-117

4. N. Seddigh, B. Nandy, P. Peda: Bandwidth Assurance Issues for TCP Flows in a Differentiated Services Network, in Proc. GLOBECOM'99, Rio De Janeiro, Dec. (1999)
5. N. Seddigh, B. Nandy, P. Peda: Study of TCP and UDP Interaction for AF PHB, IETF draft, draft-nsbnpp-diffserv-udptcpaf-01.pdf, August (1999)

Equation-Based Dynamic Shaping Algorithm

Halima Elbiaze, Tijani Chahed, Gérard Hébuterne, and Tulin Atmaca

Institut National des Télécommunications

9 rue Charles Fourier 91011 Evry CEDEX - France

{halima.elbiaze, tijani.chahed, gerard.hebuterne, tulin.atmaca}@int-evry.fr

Phone : +33 1 60 76 47 95 , Fax : +33 1 60 76 47 80

Abstract. The focus of this paper is the traffic shaping at the access of an network node. We propose a novel algorithm that dynamically shapes the incoming traffic, based on service curves equations, in order to meet the QoS constraints in terms of buffer size or delay. We first estimate arrival parameters within various time intervals in order to make the incoming traffic fit into a token bucket traffic specification (Tspec) format. We then derive the shaping parameters based on deterministic service curves. Those shaping parameters vary dynamically according to the Tspec of every time window.

1 Introduction

The traffic shaping takes place in the network edges to maintain the logical performance to its highest level and thus respect the traffic contract. The problem studied in this paper¹, is motivated by the desire to obtain applicable performance bounds for a very high-speed network. To achieve this aim, a tool for studying end-to-end, bounded performance is needed. Classical queuing analysis studies average performance for aggregate traffic. It focuses on single server environments for traffic with somehow periodic inter-arrival times. However, in the packet-switching, integrated services models, bounds on the end-to-end performance need to be studied in a networking environment with traffic dynamics, interactions and burstiness far more complex than in the previous case. In this work, we use the deterministic version of the service curves method [2] and particularly Network Calculus (NC) [1], its Min-Plus algebra formulation.

2 End-to-End System

For source modeling, we consider the Deterministic Bounding Interval-Dependent (D-BIND) model, found in [4]. It uses a family of rate-interval pairs where the rate is a bounding rate over the corresponding interval length. The

¹ supported by the “Réseau National de Recherche en Télécommunications” under the decision number 99S 0201-0204 and the European IST Project DAVID (Data and Voice Integration over DWDM)

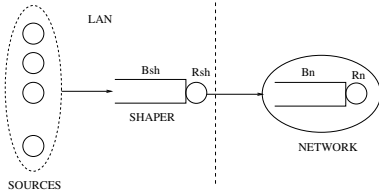


Fig. 1. End-to-end System

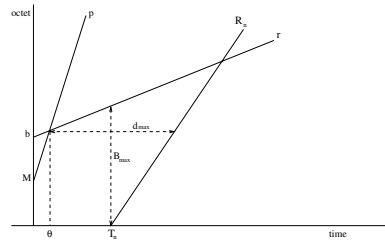


Fig. 2. Arrival Tspec and service curve

bounding rates used in this study are base on Traffic Specification (Tspec) formulation. Introduced by the IETF for IP, Tspec takes the form of a token bucket, with rate r and depth b , plus a peak rate p and maximum packet size M . We now show how sources traffic can be made to fit into a token bucket Tspec format. We partition the whole process into N equal, non-overlapping blocks, corresponding to time intervals $(I_i)_{i=1,\dots,N}$, of length l_N , and approximate the traffic volume within each interval I_i by a corresponding set of Tspec parameters. In doing so, we obtain a piece-wise Tspec formulation of the global process which approximates the actual process.

Using the service curve approach, the arrival curve corresponding to the Tspec given in terms of p, M, r, b , equals $\min(pt + M, rt + b)$. The network is modeled by n elements in tandem, each one with service rate R_i and starting service at time $T_i, i = 1, \dots, n$, one possible concatenation scenario is a network element with a service curve $c(t) = R_n(t - T_n)^+$ where $R_n = \min(R_1, R_2, \dots, R_n)$ and $T_n = \sum_i^n T_i$. Figure 2 represents an arrival Tspec and service curve, showing the corresponding fundamental bounds of Network Calculus [1], namely the backlog B_{max} and the delay d_{max} .

3 Shaping

Adding a shaper to the arriving traffic prior to its entrance to the network is done as follows. A new service curve, corresponding to the actions of the shaper, with parameters (R_{sh}, T_{sh}) is set between the arrival curve and the network service curve. In what follows, we assume, without loss of generality, $T_n = 0$.

3.1 Shaping to Meet Buffer Requirement and/or Delay Constraint

Let us suppose that the maximum buffer size, B_c , at the network level is smaller than the maximum backlog bound, B_{max} , caused by the non-shaped arriving traffic. The point of introducing a shaper in this case is to assure that the incoming traffic does not exceed B_c for a loss-free network performance.

Schematically, and considering the setting of Figure 3, the idea is to vary the shaping curve through the segment indicating B_c . In this case, the shaded region, given in Figure 3, shows the region of shaping. Through Network Calculus

[1] bounds formulations, we easily derive maximal R_{sh} (buffer constraint) corresponding to optimal shaping case.

In delay constraint case, the point of shaping is to reduce the maximum delay to be experienced at the network region from the original d_{max} to a new delay constraint d_c . Let us note that introducing a shaper does not add to the end-to-end delay. This is again achieved by setting appropriate values to R_{sh} (delay constraint). Figure 4 shows the shaping region varying the line indicating d_c . $R_{sh}(\text{buffer constraint}) = \frac{B_c r - R_n b}{B_c - b}$ (*)

; $R_{sh}(\text{delay constraint}) = \frac{R_n(b - r d_c)}{b - R_n d_c}$ (**)

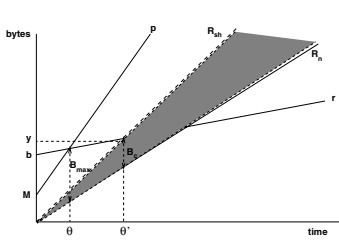


Fig. 3. Region of shaping to meet buffer requirement B_c

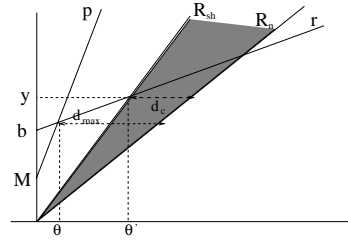


Fig. 4. Region of shaping to meet delay requirement d_c

3.2 Equation-Based Dynamic Shaping Algorithm

So far, we considered shaping within every interval I_i of length l_N . Our ultimate aim is however to shape the global incoming traffic. Parallel to the idea of partitioning the arrival process so as to locally bound each interval, the shaping scheme introduced in the previous section shall apply to each interval.

It is clear that the shaping rate R_{sh} depends on the arrival curve parameters throughout the whole process. The task in this case is to find optimal, i.e. maximal, shaping rate R_{sh} for each interval I_i such that the buffer constraint and/or delay constraint are satisfied. This is achieved by dynamically changing the shaping rate from one interval to the next. The dynamic shaping algorithm is then as follows.

1. Set observation window size equal to l_N
2. Determine corresponding T_{spec} in interval $(I_i)_{i=1, \dots, N}$
3. Apply Equations (*) and (**). to set shaping parameters such that
 - i. shaping is optimal : minimal buffer size B_{sh} and maximal shaping rate R_{sh}
 - ii. requirements are met, i.e., buffer or delay constraint at network level
 - iii. no loss at shaper, i.e. B_{sh} not exceeded.

4 Numerical Results

The end-to-end system studied is shown in Figure 1. We consider self similar traffic resulting from the LAN sources with mean = 100 Mbit/s, variance = 10^9 , Hurst parameter $H = 0.7$, and M equal to 1540 bytes. At the network level, let $R_n = 227$ Mbit/s be the rate of the server, with buffer capacity B_n equal to 100 packets.

4.1 Estimation of Arrival Parameters

We first estimate the parameters of the incoming traffic into a Tspec format, for different observation windows of size l_N .

Figures 5 and 6 shows the average rate r_i for intervals I_i of lengths equal to 300ms and 1s respectively. We notice that for a given window size l_N , r_i^N varies from one interval to the next keeping the same behavior as the original traffic, i.e., incorporating correlation. On the other hand, the family of $(r_i^N)_{i=1,\dots,N}$ behaves

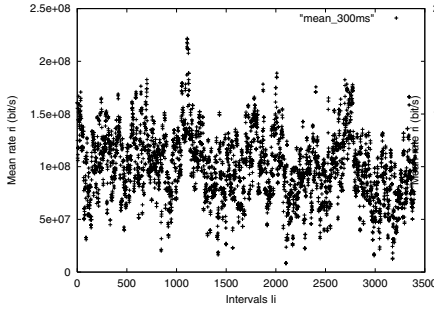


Fig. 5. Average rate r_i for $l_N = 300ms$

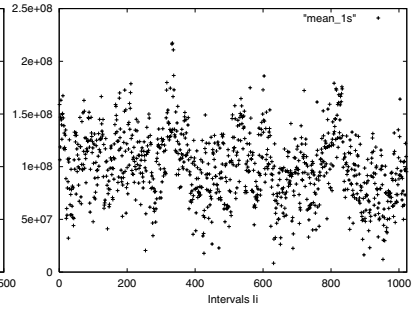


Fig. 6. Average rate r_i for $l_N = 1s$

in the same way in different lengths l_N of intervals $(I_i)_{i=1,\dots,N}$, i.e., in many time scales. That means the presence of self-similarity property in the sequence $(r_i^N)_{i=1,\dots,N}$. Figures 5 and 6 confirms that for two times scales $(r_i^N)_{i=1,\dots,N}$ behaviors : 300ms and 1s. The same remarks remain valid for p.

The burst size b estimation consists on the sum of consecutive interarrivals smaller or equal to the interarrival time within the corresponding peak rate p_i corresponds to the burst size. The obtained values for b vary from 85 packets for small window size l_N to 60 for larger ones.

4.2 Performance

If no shaping is used at the access of the network, a 55 packet buffer size is needed at the network server to achieve no-loss performance. The maximum delay in the network level in this case is equal to 0.00268 sec.

Let us assume that in fact, the buffer size $B_c=10$ at the network level cannot be as large as to hold 55 packets, i.e., if no shaping is used, there will be some loss. We derive the shaping parameters for every interval $(I_i)_{i=1,\dots,N}$ of length l_N . For intervals of length $l_N = 100$ ms, Figure 7 shows the mean arrival rate r_i versus shaping rate R_{sh} throughout the duration of the connection (100 seconds). We

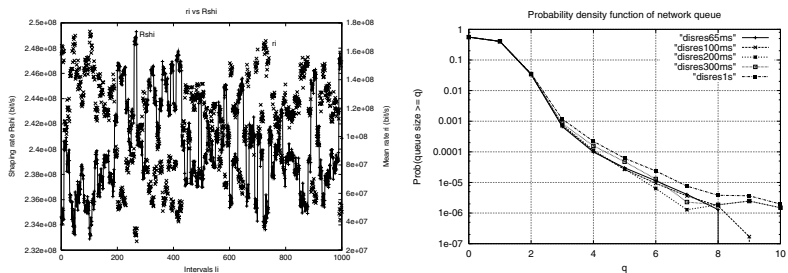


Fig. 7. Average rate r_i vs Shaping rate R_{sh} during the Interval I_i **Fig. 8.** Network queue PDFs for Interval I_i lengths: 65ms, 100ms, 200ms,300ms, 1s

notice that r_i and R_{sh} are inversely proportional; for every large values of r_i , i.e., high input rate, the value of R_{sh} is small, i.e., a severe shaping is needed to accommodate the buffer constraint and the loss-free performance. This done, we plug the equation-based shaping parameters back into the simulation model. No loss is observed at the shaper for different observation window lengths l_N because we consider shaper size is largest value over all intervals derived from the equations. This independence between the shaping queue PDF's and the interval sizes l_{Ns} , can be explained by the fact that the shaping rate R_{sh} is adaptive with respect to the incoming traffic in order to meet the non-loss performance. Figure 8 shows the probability density function of the buffer occupancy at the network level for different observation window lengths l_N . We notice that the smallest interval lengths $l_N = 65$ and 100 ms yield a non-loss performance. This is explained by the fact that at those interval lengths, we obtain higher precision for estimation of arrival parameters and hence shaping parameters. For larger interval lengths, $l_N = 200$ and 300 and 1000 ms, some loss, on the order of $2.4 \cdot 10^{-7}$, is observed. This is explained by the fact that for small precision, the arrival parameters are under-estimated. Put in the equations, they yield high shaping rates, or equivalently, soft shaping. This in turn results in loss at the network level. We follow the same steps as buffer-constrained performance to respect the tolerated maximum delay $d_c = 0.0005$ at the network level.

Table 1 illustrates the maximum delay values obtained by simulation for different window lengths l_N : 65ms, 100ms, 200ms, 300ms and 1s. Again, we notice that the smallest interval lengths $l_N = 65$ and 100 ms yield the target maximum delay.

For larger interval lengths, $l_N = 200$ and 300 and 1000 ms, the maximum values

of the observed delay exceed the constraint. This is explained by the fact that for small precision, the arrival parameters are under-estimated.

Table 1. Maximum delay at the network level for different window lengths l_N : 65ms, 100ms, 200ms,300ms, 1s

window lengths l_N	65ms	100ms	200ms	300ms	1s
maximum delay a the network	0.0004102 s	0.0004219 s	0.0005413 s	0.0005414 s	0.0005414 s

5 Conclusion

In this paper, we focused on self-similar traffic [5] at the input of an network node. If this traffic is left as is, it cannot satisfy the buffer and/or delay constraint at the network level, which may be very stringent in the optical case. In order to meet those requirements, shaping is essential. In this work, we proposed an equation-based dynamic shaping with three key steps: 1) estimation and fitting of interval-wise incoming traffic into arrival curves, 2) solving into the service curve approach for the shaping parameters in an adaptive manner and 3) fitting the later back into the original model.

As of the first step of our algorithm, we notice that the input estimate reproduce the same self-similar, correlated nature of the original traffic. The shaping parameters derived in step 2 are typically conservative owing to the deterministic nature of the service curve. However, when put back into the original model, i.e., step 3, they are shown to be numerically not very conservative. This may be explained by the correlated nature of the original self-similar traffic.

References

1. J.-Y. Le Boudec *Application of Network Calculus To Guaranteed Service Networks*. IEEE Trans on Information theory, (44) 3, May 1998.
2. R. L. Cruz *Quality of Service Guarantees in Virtual Circuit Switched Networks*. IEEE JSAC, 1995.
3. C. Kamanek, H. Kanakia and S. Keshav. *Rate controlled servers for very high-speed networks* In Proceedings of IEEE Global Telecommunications Conference, pp. 300.3.1-300.3.9, San Diego, California, December 1990.
4. E. Knightly and H. Zhang. *Traffic characterization and switch utilization using deterministic bounding interval dependent traffic models* In Proceedings of IEEE INFOCOM'95, Boston, MA, April 1995.
5. V. Paxson and S. Floyd. *Wide-Area Traffic: The failure of Poisson Modeling* IEEE/ACM Transactions on Networking, 3(3), pp; 226-244, June 1995.

A Novel Content-Based Video Streaming Algorithm for Fine Granular Scalable Coding¹

Li Zhao, Qi Wang, Yuwen He, Shiqiang Yang, and Yuzhuo Zhong

Department of Computer Science and Technology, Tsinghua University,
Beijing 10084, China

Li.Zhao@media.cs.tsinghua.edu.cn

Abstract. Most of current video streaming technologies, which protect the video stream by rate allocation or CRC means according to the bandwidth of network and the video quality user expected, is unrelated with the content of video. This paper proposes a new content-based video streaming method, which is based on video content analysis. The method discusses the rate distortion function of Fine Granular Scalable (FGS) enhancement layer bitstream based on the video content, and allocates the suitable bits for video units by the importance of video content. The experimental results indicate that our content-based rate allocation method can protect the transmission of video efficiently, improve the subjective quality of video and use the network bandwidth in maximum utility.

1 Introduction

Video streaming application over the Internet booms rapidly in recent years. Because of the lack of Quality-of-Service (QoS) guarantees over the current Internet, the media streaming data undergoes the dynamic variations of bandwidth, delay and packet loss. Due to the bandwidth of the channel between the encoder and decoder is not a fixed number but changes constantly along with time.

To resolve the issues of video streaming over Internet mentioned above, there are two general types of approaches have been proposed. The one is Network-Centric, which requests the router/switcher of the Internet to provide the guarantees of the bandwidth, delay, jitter, packet loss rate for video streaming applications. The other is End System-based, which does not rely on the implement of network at present or in the future and be achieved easily. In this paper, we focus on the second (End System-based) approach.

From the End System-based perspective, there are some conventional methods proposed to solve the issue mentioned above: (1) Transcode approach is to encode the video with the highest quality and store the bitstream at the server. Then the encoded bitstream is transcoded into different bitrate for different users [3]; (2) Simulcast approach is to generate multiple bitstreams of multiple bitrates at the encoder time. (3) Scalable encoding or adaptive encoding methods [5] [6]. (4) Rate-shaping approach is a technique to adapt the rate of compressed video bit streams to the target rate constraint[1]. Almost all of the current video streaming technologies, which

¹ This work is supported in part by China National 973 Projects G199032704.

protect the video stream by rate allocation or CRC means[2] according to the bandwidth of network and the video quality user expected, is unrelated with the content of video.

In this paper, we propose a content-based video streaming algorithm for FGS coding, which combines the video streaming with video content analysis technique. This approach allocates the bits by video content based on video content analysis result and can protect the important video content by selectively and actively discarding some bits of unimportant video data while the bandwidth descends sharply.

The rest of this paper is organized as follows. In section 2 we focus on our content-based video streaming algorithm. Section 3 proposes our experimental results. Concluding remarks are in section 4.

2 Algorithms Description

The rate control is a fundamental technique in the encoding process, which is based on the rate-distortion theory[7][8]. The objective of rate control is to make encoded bitstream meet the bandwidth of channel. The rate allocation for the FGS enhancement is different from the traditional rate control in the encoding process for that it is independent of the FGS encoding progress and is executed by the feedback information from the channel before transmission.

In this paper, we propose a content-based video rate allocation algorithm for video streaming over Internet. In our method the video content is analyzed at first, including video structure analysis, motion information extraction, video objective retrieval/identification etc. Then allocate the bits for FGS enhancement based on the result of video content analysis. In FGS coding, the base layer restructures the low quality VOP and the enhancement layer, improves the quality of the VOP. In our rate allocation algorithm, we assume that the decoder can receive all base layer bits. We focus on how to allocate the enhancement bitstream by video content.

Segmented video clips are used to represent the video sequence: {Seg(1), Seg(2), ... Seg(N - 1), Seg(N)}, here N denotes the total number of segmented video clips.

We propose a video content-weighted rate-distortion (R-D) function:

$$D(R) = w * \sigma^2 * e^{-aR} \tag{1}$$

Here, $D(R)$ denotes the mean square value of the error between decoded video clip and original video clip; w denotes the weight coefficient of the video clip, which is determined by the result of video content analysis; σ^2 is a constant number; σ^2 is the variance of the encoding signal; R is the bits number used to encode the video clip.

If R_i denotes the allocated bits number for encode video clip $Seg(i)$ and D_i denotes the mean square error of $Seg(i)$, the essence of the problem of rate allocation is a global optimization issue and can be described by the following formula:

$$f(R_1, R_2 ..R_3) = D_1(R_1) + D_2(R_2) + \dots D_N(R_N) = \sum_{i=1}^N w_i * \sigma_i^2 * e^{-aR_i} \tag{2}$$

Subject to $R_1 + R_2 + \dots + R_N = S$ (const), S is the given total number of bits used for the sequence decoding.

Where $f(R_1, R_2..R_3)$ denotes the total degree of distortion of all video clips.

In order to minimize total distortion $f(R_1, R_2..R_3)$, we can apply Lagrange multiplier method to solve it. Firstly, we define the following function:

$$g(R_1, R_2..R_N) = f(R_1, R_2..R_N) - \lambda(S - (R_1 + R_2 + \dots + R_N)) \tag{3}$$

Then, we obtain some equations:

$$\frac{dg}{dR_1} = \frac{dg}{dR_2} = \dots = \frac{dg}{dR_N} = \frac{dg}{d\lambda} = 0 \tag{4}$$

We solve above equations:

$$R_i = \frac{C}{N} + \frac{2}{aN} \sum_{i \neq j} \ln \frac{\sigma_i}{\sigma_j} + \frac{1}{aN} \sum_{i \neq j} \ln \frac{w_i}{w_j} \tag{5}$$

Here, $i = 1, 2, \dots, N$.

However calculating $\frac{1}{a} \ln \frac{\sigma_i}{\sigma_j}$ is a not easy thing during the rate allocation progress. So a simple method for estimating $\frac{1}{a} \ln \frac{\sigma_i}{\sigma_j}$ must be found. Similarly, we think the distortions approximately equal when the all enhancement layers are fully decoded, i.e.

$$\sigma_i^2 e^{-aR_i} = \sigma_j^2 e^{-aR_j} \tag{6}$$

Now we can get:

$$\frac{1}{a} \ln \frac{\sigma_i}{\sigma_j} = \frac{\widehat{R}_i - \widehat{R}_j}{2} \tag{7}$$

Here, \widehat{R}_i and \widehat{R}_j denote the used bits number to fully encode the enhancement layer of video clip $Seg(i)$ and $Seg(j)$. In addition, this information can be generated during encoding.

Bring equation (7) into equation (5), the bit allocation can be simply represented as:

$$R_i = \frac{C}{N} + \frac{1}{N} \sum_{i \neq j} (R_i - R_j) + \frac{1}{aN} \sum_{i \neq j} \ln \frac{w_i}{w_j} \tag{8}$$

Here $i = 1, 2, \dots, N$.

3 Experimental Result

To evaluate the performance of the content-based video streaming algorithm we proposed in this paper, we use the *Stefan* sequence that is the standard test sequence of MPEG-4 as our experimental sequence. The parameters are set as follows: The sequence format is SIF (352*240). The encoding frame rate is 30f/s. The quantization scheme of the base layer is the same as in H.263. The bit rate of base layer is 320Kbits/s.

In detail, in our experiment, we adopt motion activity to describe the importance of video content: we assume that the greater the motion activity is, the higher the importance of video content is. Here global motion estimation (GME) method is adopted to measure the motion activity. Detailed algorithm of robust GME can be referred in[4]. Then, according to the motion activity, the sequences are segmented into several segments with different importance degree. The segment result is given in Fig.1.

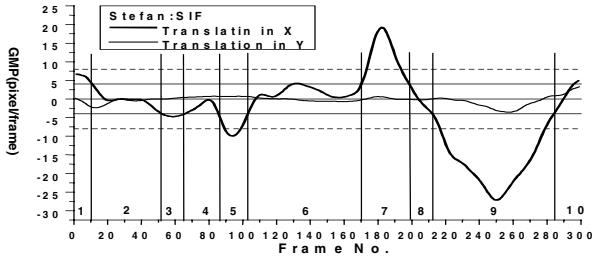


Fig. 1. The motion activity segmentation result according to global motion estimation method.

In our experiment, the enhancement layer bits are allocated with two methods: One is to allocate the bits to each frame averagely. The other is to allocate the bits with our algorithm. Fig.2 shows the experimental result under the bitrates of enhancement layer at 320Kbits/s. Here the horizontal axis denotes the frame number and vertical axis denotes the SNR.

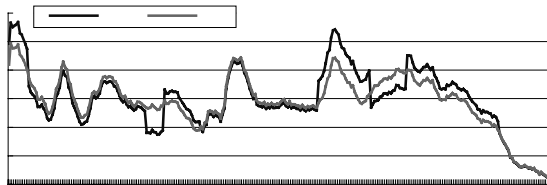


Fig. 2. The curves of PSNR versus frame number of “Stefan” (320Kbits/s)

The experimental results prove that the proposed bit allocation scheme can improve the performance of FGS to 0.5~1.0 dB at the important parts. For example, the part of No. 175~200 of sequence “Stefan” has very high motion activity (See fig. 1). We can think this part is more important. From fig.2, we can see that the PSNR of our method at this part is 1.0 dB higher than contrasting method’s.

We can draw the conclusion that the content-based rate allocation scheme proposed for video streaming can protect important video content while transmitting. In other words, this method can improve the video quality at the receiver and increase the utilization ratio of the channel.

4 Conclusions

In this paper, we propose a new content-based video rate allocation algorithm for video streaming over Internet, which is based on video content analysis and fine granular scalable video coding techniques. This scheme can stream video by content, so that it can protect the important video content while transmitting under the conditions of limited bandwidth of channel and improve the utilization of the channel. The experimental results show that the method proposed can protect video content efficiently and improve the subjective quality of important video segments.

In the future, we will discuss the content-based video streaming in big picture, including rate control, error resilience and error concealment etc research areas, and propose a whole architecture of content-based video streaming/delivery.

References

- [1] Jacobs, S., Eleftheriadis, A.: Real-Time Video on the Web Using Dynamic Rate Shaping, Image Processing, 1997. Proceedings., International Conference on Volume: 3 , 1997 , Page(s): 14 -17 vol.3.
- [2] Tan, W.T., Zakhor, A.: Real-Time Internet Video Using Error Resilient Scalable Compression and TCP-Friendly Transport Protocol, Multimedia, IEEE Transactions on Volume: 12 , June 1999 , Page(s): 172 -186.
- [3] Wang, L.M., Luthra, A., Eifrig, B.: Adaptive Rate Control for Mpeg Transcoder, Image Processing, 1999. ICIP 99. Proceedings. 1999 International Conference on Volume: 4 , 1999 , Page(s): 266 -270 vol.4.
- [4] He, Y.W., Qi, W., Yang, S.Q., Zhong, Y.Z.: Feature-based Fast and Robust Global Motion Estimation Technique for Sprite Coding, *ISO/IEC JTC1/SC29/WG11, MPEG00/M6226*, July 2000.
- [5] Yu, Y., Chen, C.W.: SNR Scalable Transcoding for Video over Wireless Channels, Wireless Communications and Networking Conference, 2000. WCNC. 2000 IEEE Volume: 3 , 2000 , Page(s): 1398 -1402
- [6] Lee, B.R., Park, K.K.: H.263-Based SNR Scalable Video Codec, Consumer Electronics, 1997. Digest of Technical Papers. ICCE., International Conference on , Page(s): 254 -255
- [7] Chiang, T., Zhang, Y.Q.: A new rate control scheme using quadratic rate distortion model, IEEE Transactions on CSVT, 7(1): 246-250, Feb, 1997.
- [8] Mallat, S., Falzon, F.: Analysis of low bit rate image transform coding, IEEE Transactions on SP, 46(4): 1027-1042, Apr 1998.

Topological Synthesis of Mobile Backbone Networks for Managing Ad Hoc Wireless Networks¹

Izhak Rubin and Patrick Vincent

Electrical Engineering Department, University of California (UCLA)
Los Angeles, CA 90095-1594
rubin@ee.ucla.edu

Abstract. In this paper, we overview our approaches and methods to the topological synthesis and management of Mobile Backbone Network (MBN) based architectures. Such backbone networks are intended to aid in the support of multimedia applications across mobile ad hoc wireless networks. An MBN based ad hoc network employs backbone nodes, operating in a manner similar to that used by base-stations in a cellular network. In contrast with the latter, in an MBN system these nodes are mobile and are selected dynamically to best track the mobility trends of the involved units and best adapt to the environmental topography of the area of operations.

Such networks integrate the MBN with ad hoc networking and routing approaches to support non-realtime and realtime multimedia applications. In this manner, network services offer applications a choice of best-effort transport as well as guaranteed end-to-end quality of service (QoS) performance.

To design the topology of the mobile backbone network, we incorporate mobile-user location distribution information. We consequently introduce an approach based on the use of pattern recognition techniques in synthesizing cluster regions that serve as key components of the mobile backbone. Intra-cluster and inter-cluster topology design methods are then used to complete the construction of a reliable backbone network. Simplified hierarchical routing mechanisms are then invoked for distributing flows across the backbone.

1 Introduction

Providing reliable multimedia communications in mobile multihop wireless networks is a formidable challenge, which has received much research attention. Consider communication networks in which all users are potentially mobile and untethered to any fixed wireline infrastructure. Such networks can arise in a variety of contexts. The mobile users of the system desire reliable communications, including real-time voice and video, notwithstanding the lack of a fixed infrastructure.

Recent investigations into the development of next generation wireless networks for commercial and military applications have led to the consideration of multihop ad hoc network architectures. Yet, as an ad hoc multihop network grows large, the performance tends to deteriorate. If the network management and traffic handling responsibilities are

¹ This work was supported by Army Research Office (ARO) Contract DAAG55-98-1-0338, and by Office of Naval Research (ONR) Contract N00014-01-C-0016.

fully distributed, then each individual mobile user must maintain network connectivity information, perform routing calculations and updates, and monitor network performance. This operation, in the presence of mobility, requires sophisticated terminal equipment and excessive control traffic overhead. The already difficult task of attempting to guarantee any type of performance on a path with many wireless hops—a particular concern for the support of real time applications—is made more difficult because the dynamic nature of the network topology causes inconsistencies in the routing and connectivity information possessed by each host. Some approaches have been developed to reduce the traffic control overhead by constructing routes between specified source-destination pairs only when needed. These schemes, while reducing control traffic overhead, have other problems, including the latency introduced by the computation of a new route, and poor efficiency when many sources are simultaneously transmitting to many distinct destinations.

In the fully distributed wireless network described above, there is no fixed (wired) backbone that can be exploited to centralize some of the network management and routing functions. In order to provide the desired functionality, the mobile terminals must be organized into a network that has some hierarchical organization or reliable structure that is maintained under varying network connectivities. This structure should, if possible, be dynamically constructed in a distributed fashion, without reliance on a centralized controller. The network organization scheme must be designed to cope with the multiple-access and collision problems inherent in the broadcast nature of the radio channel, as well as the scarcity of resources present in the wireless scheme.

One method of organizing a large group of mobile users into a network consists of hierarchically partitioning the overall network into subnetworks called clusters. Some network nodes are selected to be local controllers, called clusterheads, while other nodes join a clusterhead to form a cluster [1]. Clustering facilitates the use of techniques to resolve scheduling conflicts and facilitates the reuse of CDMA codes and TDMA time slots. Once nodes are partitioned into clusters, clustering maintenance techniques are employed to maintain the cluster organization in the presence of mobility.

Another approach to solving the ad hoc network problem is to use an embedded hierarchical structure, with physically different networks at the various levels. Recently, we have proposed [2]-[4] the employment of a mobile backbone to support guaranteed QoS (as well as best effort) applications for mobile networks. Certain mobile terminals are assigned to effectively serve as mobile base stations, and these selected users ("backbone nodes") together with their interconnecting communication links constitute the mobile backbone. The backbone network consists of a mesh topology with point-to-point or multipoint (broadcast) links used to interconnect neighboring backbone nodes. The mobile backbone, which has a functionality analogous to a fixed backbone in a cellular network, is then exploited to make the routing, access control, scheduling and congestion control problems tractable. We are carrying out research investigations aimed at developing backbone architectures, routing methodologies, medium access control techniques, queueing algorithms and scheduling policies for mobile backbone networks (MBNs).

It must be noted that the two network backbones mentioned above, the clusterhead backbone (CB) and the mobile backbone (MB), have distinct designs and different purposes. The CB construction algorithms do not generally attempt to select a backbone

with targeted capacity and connectivity features and do not explore all possibilities for using the backbone as a primary means for routing traffic that requires a specified QoS. The established mobile backbone is employed to provide for the support of multimedia applications, including real-time streams and high intensity imaging and data files. The mobile backbone (MB) must be synthesized to offer the mobile stations the required transport services at acceptable QoS levels, while incorporating the processing/routing rate and power supply limitations of the nodes and the capacity constraints of the links..

Our investigations allow network nodes to be categorized into a number of classes. Single module nodes (identified also as regular nodes, RNs, or terminal nodes TNs) employ a single module radio that enables them to operate at relatively lower power and lower data rate levels. Such nodes are more constrained in their power resources. In turn, multi module nodes employ a multi-module radio system. They are outfitted with a lower power radio module that allows them to communicate with regular nodes, as well as with higher power radio module that enables them to communicate at higher data rate and over a longer range with other such nodes. The latter nodes are identified as high power (HP) nodes. It is desirable to construct the MBN by using HP nodes. We identify the latter as backbone capable nodes (BCNs).

The MB guarantees QoS connections, a stable operation, a mechanism for network management and control and a significant simplification in the complexity of the routing problem. In constructing a mobile backbone there exists a key design tradeoff between backbone accessibility (i.e., a certain percentage of nodes should be within a fixed small number of hops from the backbone) and backbone minimality (also related to overall power utilization levels).

The backbone must satisfy user-specified accessibility and connectivity requirements. Once a relatively stable backbone is synthesized, it can then be used to manage the networks; for example, allocate network resources during route set-up; and assign MAC layer resources. Since the backbone terminals form a small subset of the network terminals, a simplified routing algorithm can be employed. Those streams that require a specified quality of service will use the stable backbone, not just any route. Best-effort message traffic may use the backbone or any other access and ad hoc routing technique. The backbone can be used for dynamic reallocation of resources to satisfy varying demands of users and varying link capacities. Routing algorithms will, of course, need to incorporate connectivity identification operations, terminal location tracking and management, and handoff control. Medium access-control algorithms are used to regulate the access of terminals to their associated backbone nodes.

2 Mobile Backbone Requirements

We observe the integrated network to consist of the following three hierarchies (Fig. 1):

Level 1: The **backbone network** is established to provide for communications between the backbone nodes. This network consists of a mesh topology with links used to interconnect neighboring backbone nodes. The backbone nodes implement packet switching functions that provide for the transport of packets on a best effort basis as well as for the provision of QoS guarantees to network flows.

Level 2: Access nets are used for communications from each backbone node to its mobile clients and from the terminal nodes to their associated backbone node(s). A multiplexing scheme is used to establish forward control and traffic channels that carry backbone node to terminal node flows. Multiple access protocols are used to implement the reverse control and traffic channels that carry message flows from the terminal nodes to their associated backbone node.

Level 3: Mobile to mobile communications can take place along a direct multi-hop ad hoc network path, in which case backbone nodes and links are not used for message transport. The BNs can however be used to set up and control the underlying routes, as well as to coordinate the MAC resources allocated in each cluster. Ad hoc networks are typically used to provide best-effort services. Such a shared medium network is identified at times as a **terminal net**. Note the latter to me often embedded into access net(s).

In the following, we define the qualities of an effective mobile backbone.

- a) **Covering requirement:** Since the primary mode of routing information in the MBN is through the backbone, terminal nodes must communicate with backbone node(s) either directly, or through other terminal nodes. We permit a “probabilistic” approach to synthesizing backbones. In selecting the backbone topology, we specify that a fraction p ($p < 1$) of network terminals should be within a distance of h hops from a backbone node. If we specify a backbone for which h equals 1.0 while $p=0.9$, then 90% of all users will be within one hop of one or more backbone nodes. If, on the other hand, we specify that $h=1$ and $p=1$, then all users can communicate directly with at least one backbone node, and, in graph-theoretic terms, the nodes on the backbone are said to form a dominating set for the underlying graph.
- b) **Connectivity requirement:** Since the backbone will be used for routing throughout the network, it is necessary for the MBN to have a connected topology. We may further specify that the backbone should be k -connected. For example, if we specify that $k=2$ for node connectivity, then there should exist at least 2 node disjoint paths between each pair of backbone nodes, in which case the failure of any single node will not disconnect the backbone. If we specify that $k=1$, $h=1$, and $p=1$ (all nodes must be within one hop of the backbone, and the backbone must be connected), the set of backbone nodes is said to be, in graph-theoretic terminology, a connected dominating set.
- c) **Minimality requirement:** We desire that the number of nodes selected for the backbone be minimal. If we specify that $k=1$, $h=1$, $p=1$ and, additionally, we require the number of nodes in the backbone be minimal, the set of backbone nodes is said to be, in graph-theoretic terminology, a minimum connected dominating set (MCDS). Determining the MCDS for a general graph is known to be an NP-hard problem.
- d) **Robustness requirement:** The backbone topology should be selected such that it will not change too often in the face of node movements and node connectivity variations.

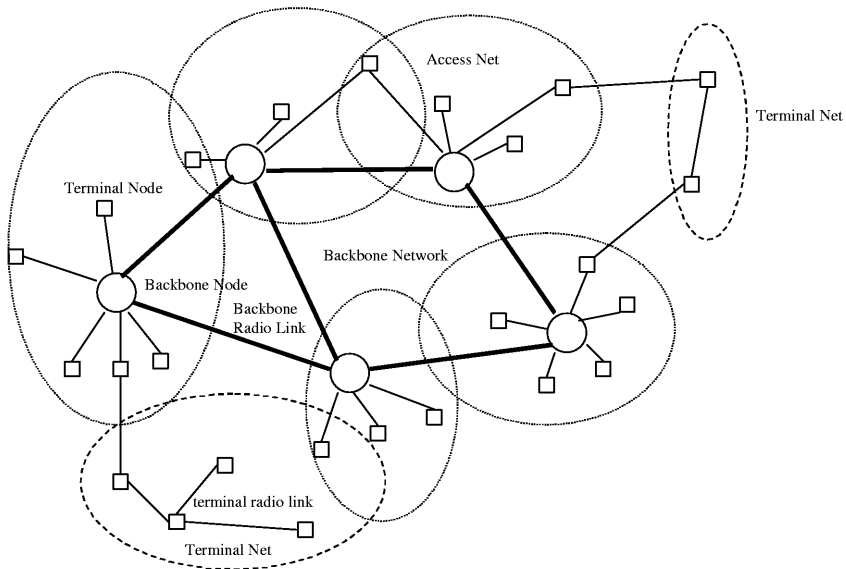


Fig. 1. Architecture of a Mobile Backbone Network (MBN)

- e) **Degree requirement:** The degree of each backbone node may be specified to be no higher than a value of DEG . This places a limit on the number of radio links that may be incident at each backbone node.
- f) **Diameter requirement:** We may specify that the length of the shortest route between all pairs of backbone nodes be limited to d radio links. In graph-theoretic terms, if we compute the length of the shortest path joining each pair of nodes in a graph, the maximum value of the shortest path over all node pairs is termed the graph diameter.
- g) **Loading requirement:** Traffic loading requirements and nodal processing/routing capacities must be taken into consideration when constructing the backbone. Specifically, the backbone must be able to provide sufficient bandwidth to support the required best effort and guaranteed traffic loading. In an area that contains a larger number of high intensity terminals, a larger number of backbone nodes may be needed in comparison with an area that contains a larger number of low intensity terminals. Note that QoS guarantee requirements will affect the backbone construction.

It is noted that several of the above requirements conflict with each other. For instance, it is intuitively clear that a backbone chosen for its robust, unchanging topology (requirement d) is not likely to constitute a minimal set of nodes (requirement c). Similarly, the diameter, loading and degree requirements adversely affect backbone minimality. Therefore, when synthesizing a backbone algorithm, we consider which "requirements" are most important in a particular application, and what compromises and trade-offs have to be made to loosen other constraints.

3 Network Partitioning Using Pattern Recognition

Most techniques and algorithms presented in the literature that seek to impose a backbone on a mobile wireless network are graph-theoretic in nature. That is, known backbone construction algorithms model the communications network as an unweighted graph wherein the users comprise the nodes of the graph, and an edge exists between two nodes (users) if they can communicate with each other. These algorithms then use graph parameters (such as node degree) or graph-theoretic concepts (such as dominating sets) to determine which nodes should be placed on the backbone.

An actual real-world mobile wireless network consists of a certain number of mobile terminals, each with a certain transmitter power level, and each located (at a given instance of time) in a certain geographical position. Users operate in an existing geographical terrain under certain existing atmospheric conditions. When this real-world system is modeled as an unweighted graph, information is lost. With the advent of worldwide commercially available global positioning systems (GPS), it is a simple matter for hosts to determine their geographic position. The geographic distance between two hosts, which may affect, for instance, the link strength and the optimal route selection, is not considered in the graph model where an edge either exists or doesn't exist.

We use **pattern recognition** techniques in designing mobile backbones. We avail ourselves of “positional” information (with a generalized interpretation of the positional information to incorporate other metrics, such as communications quality descriptors). Pattern recognition algorithms are then employed to discern compact sections of the network, and to choose appropriate representative gateway nodes for these compact sections.

Our MBN topological synthesis process thus consists of three steps: network partitioning into regional clusters (compact regions) using location aware pattern recognition techniques; synthesis of intra-cluster networks; layout design of inter-cluster networks using reliability and connectivity objectives. Note that when higher-power backbone nodes are used to provide access to lower-power terminals, the intra-cluster network is dominated by the shorter ranges induced by the use of lower power radio modules. In turn, the inter-cluster network contains communications links that are established through the use of mainly higher power radio modules and thus involves longer range links that use higher data rate communications channels. In addition, we make use of relay nodes (including unmanned vehicles, UVs, such as UAVs) to aid in the construction of the backbone network.

We further use the backbone nodes to control and manage the allocation of MAC resources (such as time/frequency/code slots) inside each cluster and across clusters. The **XBONE** algorithm being developed by Prof. Izhak Rubin ensures that such MAC-layer slot allocation controls the occurrence of interferences that may be caused by simultaneous transmissions among terminals located inside a cluster, or in neighboring clusters. This algorithm integrates the topological synthesis process with the allocation of MAC resources, the selection of the flow's route, and the support of quality-of-service guarantees to realtime flows. It combines the use of direct source-destination multi-hop paths (coordinated by the backbone nodes) that do not employ

the MBN for message transport and the use of the MBN for message transport among nodes that are not too close to each other, as well as for terminal flows that require strict QoS performance guarantees, as is the case for realtime applications.

References

1. C.E. Perkins, Editor, Ad Hoc Networking, Addison-Wesley, 2001.
2. I. Rubin and P. Vincent, "Effective Backbone Architectures for Mobile Wireless Networks," UCLA Technical Report, Annual report submitted to ARO, Contract No. DAAG55-98-1-0338, December 22, 1998.
3. I. Rubin and P. Vincent, "Design and Analysis of Mobile Backbone Networks," UCLA Technical Report, Annual report submitted to ARO, Contract No. DAAG55-98-1-0338, ARO Report No. 37637-CI-SAH, March 31, 2001.
4. I. Rubin and P. Vincent, "Topological Synthesis of Multimedia Wireless Mobile Backbone Networks," presented at IEEE MILCOM, October 2000.

QoS Monitoring System on IP Networks

Marcelo Borges Ribeiro, Lisandro Zambenedetti Granville,
Maria Janilce Bosquirolí Almeida, and Liane Margarida Rockenbach Tarouco

Federal University of Rio Grande do Sul – Institute of Informatics
Av. Bento Gonçalves, 9500 – Bloco IV – Porto Alegre, RS – Brazil
{mribeiro, granville, janilce, liane}@inf.ufrgs.br

Abstract. At the current development stage of computer networks, the emergence of new applications that use the high performance available is unavoidable. In this context, any service that requires high performance also requires network QoS (Quality of Service). To help the maintenance of QoS services and QoS provisioning mechanisms, this work proposes a QoS monitor. This monitor operates on IP-based networks and its goal is to measure current QoS parameters observed on the network and compare them with the negotiated QoS parameters. Thus, network managers can be informed about degradations, and proceed with proper actions in order to provide adequate conditions to applications that require strict time warranties in order to operate properly.

1 Introduction

Nowadays, there is a clear need to have IP networks offer more appropriate services to transport differentiated information. Multimedia applications, for instance, have more demanding time restrictions than the usual networked applications (e-mail, web browsing, etc.). In order to make differentiation possible and to allow time-restricted applications to run properly, IP networks must offer services with some level of warranty, i.e., IP networks must feature QoS services.

However, having the user or the application request services, or having the service provider and other involved parties acknowledge user demands is not enough for the service to work effectively according to needs. It is also necessary to have processes to monitor QoS and the network manager must bear in mind that the desired QoS might not be the obtained QoS. This difference causes performance degradation of the applications, which no longer perform properly. Thus, the presence of monitoring mechanisms that inform the manager of QoS service performance is a real necessary.

This work presents a QoS monitoring architecture that uses monitors developed to operate on IP networks. With the aid of the QoS monitor, the network manager can determine monitoring policies for some network links, so as to be warned when any of them presents a difference greater than a preset limit. Because QoS monitoring is distributed for increased efficiency it is necessary to have a central element that gathers such information and compiles it so that the manager can understand it. The development of a central element that collects and processes data from network monitors and detects degradation is also presented.

This paper is structured as follows: section 2 discusses QoS monitoring on IP networks. Section 3 is concerned with the monitor-related architecture and section 4 explains how the QoS monitor operates and section 5 finishes the paper with conclusions and future work.

2 Related Work

Due to the increase of QoS needs, some monitoring solutions present ways to analyze traffic on the managed network. Waldbusser [1] has proposed the extension of the RMON MIB functionalities. With RMON2, the agent (probe) has its reach (monitoring) extended beyond the link level. An RMON2 probe can analyze packets by accessing information on the network and transport levels (TCP and UDP). This is very useful to have a monitoring activity by analyzing the media occupation of each application through the observation of the ports used. However, RMON2 probes cannot perceive degradation because they cannot check delay, jitter, or loss. Since these parameters are critical in QoS, RMON2-based QoS monitoring is a poor option.

Brownlee et al. [2] have proposed an architecture called RTFM (Real Time Flow Measurement) to measure and monitor flow on networks. A flow is the path of data that starts at a point A and ends at a point B. Because it depends on source and target, this flow measure is also appropriate to monitor the application level. The risk of running path-based monitoring between end points is that we are working on a packet-switch-oriented environment. This feature harms the idea of flow if only end points are considered because packets of the same flow may even switch places halfway, causing delays. At best, in case a performance loss is detected, it is not possible to determine the node that has caused it.

Other tool for monitoring bandwidth utilization proposed by Deri et al [3] is called Ntop. It was proposed to act like the „top“ command used in UNIX systems that shows the amount of memory used by processes. The main difference between all the monitoring application and Ntop is the concern about how its data are presented to the user. It features a web interface showing statistics about link occupancy, protocols utilization, machines involved, etc. Because it is not a distributed architecture, the concept of flow used in Ntop is different of the used (and needed) by other applications. The manager itself must provide all calculations with the information from every machines running Ntop in consequence of that it cannot monitor flows outside the local network, thus any flow formed by an external host cannot be used.

3 Architecture of the Monitoring System

As seen before, QoS monitoring is a need, and distributed monitoring is specially important to determine degradation points. Given this context, this section presents a monitoring system that is part of the QAME QoS management environment [4]. The QoS monitor is subdivided into a central element and monitor agents (fig. 1). This division is necessary because monitoring is distributed. The central element will receive the QoS parameters as policies from the user environment, find out which agents operate on the „area“ to be monitored and divide the monitoring flow into tasks for each relevant agent, according to its location.

3.1 Central Element

The central or centralizing element supplies a communication interface with the user environment. The central element, from a technical point of view, is a service that is run on a machine the manager interacts with. The central element has to receive the monitoring policies and determine the best strategy to collect the relevant data for that policy from the agents, compile this information and then send it to the network

manager. For this constant interaction mechanism to work, the central element not only has to wait constantly for network manager requests, but also to program the agent(s) at run-time. The central element is also responsible for assessing degradation of QoS parameters such as loss and delay, since it only accesses the aggregate data of the monitoring agents.

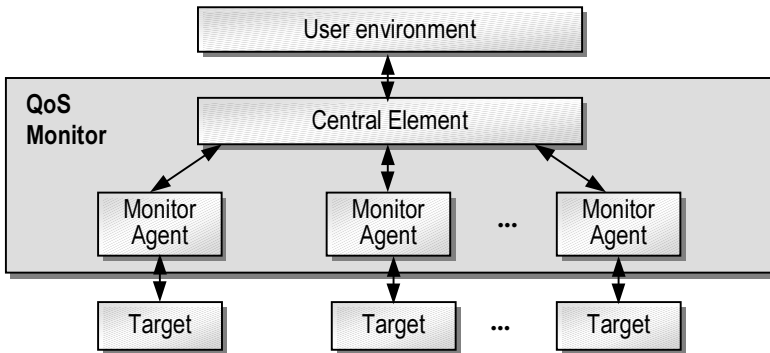


Fig. 1. QoS monitoring system

3.2 Monitoring Agents

The main tasks of a monitoring agent are to monitor flows, collect data and send information about the monitoring requests made by the central element. It is important to notice that monitors do not exchange messages with one another but with the central element only and they are able to inform about the link and current flow only.

If a critical flow has a smaller band than necessary, the monitoring agents associated to the problem node identify this condition and inform the central element. Another QoS parameter, which can be checked directly on agents, is jitter. When packets of the same flow or aggregate arrive at the interfaces of the monitored devices at irregular intervals, the variation is detected and the monitoring agent once again informs the central element of this event. Besides that, they are unable to process the collected information. They do not know what policy is under use at that time and, because they are limited to a sub-network, they would need data from other agents to generate any useful information, which is a task of the central element.

The degradation of other QoS parameters can only be detected at the central element. For instance, a packet delay is determined by an analysis of the departure time of this packet from the interface of a device and its arrival time at another interface of another device. This information is collected by accessing the monitoring agents located on each of the involved devices.

4 System Operation

Figure 2 shows an example of the operation of the QoS monitor in two sub-networks. The used network is made up of two collision domains which form segments A and B. Each segment has several workstations, represented by the squares. Two monitoring agents were placed on each segment, represented by the circles. Each

monitoring agent is, in fact, a workstation with a network interface operating on promiscuous mode.

The segments were connected to a main backbone through the use of two routers with at least two interfaces. Besides the routers, the backbone has a station which holds the central element of the QoS monitor, represented by the central rectangle. The dotted lines represent the indirect communications between monitoring agents and the central element.

Several IP flows/aggregates were monitored during the tests. To do so, each monitoring activity was defined by: sender and receiver IP addresses, sender and receiver ports, transport protocol and priority (represented by the DS field).

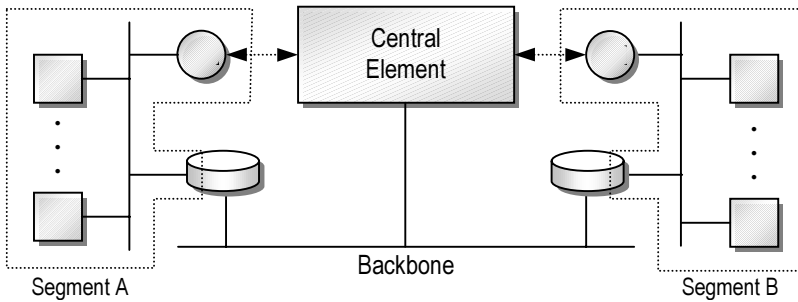


Fig. 2. Operation of the QoS monitor

The direct analysis on the monitoring agents allowed the assessment of flow bandwidth and jitter. The association of the information collected by the agents allowed the central element to verify delay and lost rate.

The analyses that were carried out were relevant, but limited by the type of implementation of the monitoring elements, which were, in this case, workstations with network interfaces in promiscuous mode. That limits the precision of the results obtained, since the verification time of the monitored packets on each agent is different from the real moment when the packet left the source, or arrived at the destination station. A more effective monitoring activity could be achieved if the monitoring agents were implemented directly on the routers.

The monitoring process starts when the network manager chooses a flow to be monitored (table 1). The management application must then send this information to the central element to have QoS monitoring started. As table 1 shows, the central element uses the monitoring policies to obtain enough information in order to distribute tasks to each agent that should be monitoring. The rows on table 1 are examples of the information the central element sends to an agent.

Table 1. Flows to be monitored

Sender IP	Receiver IP	Sender port	Receiver port	Protocol	DS field
192.168.0.101	192.168.0.4	1111	4444	TCP	7
192.168.0.101	192.168.0.23	2345	80	TCP	3

The network manager interacts with the central element only. When a monitoring policy is sent from the network manager to the central element it is broken down into simpler instructions, which are distributed to the monitoring agents that perform the

necessary measures within their monitoring scope. The collected data are then sent back to the central element, which processes the information and sends it to the network manager.

The only concern of the network manager is the communication with the central element (the whole QoS monitoring system, from the manager's viewpoint, is the central element only).

In case a data transmission service needs continuous flow, which may and most likely will go through more than one computer network, it is not enough to collect data on the client's end and to compare it with the sent data.

As previously discussed, performance loss may be observed throughout the entire „extension“ of the transmission, i.e., monitors should be distributed along the transmission. This analysis causes a few problems. One must know which monitors are part of the monitored connection because it makes no sense to receive data from monitors that are not part of the connection under assessment. Besides that, the receipt of monitor messages has to be synchronized and the frequency of value sampling has to be chosen. A higher frequency results in a more precise analysis, but an excessive number of messages interferes with performance.

5 Conclusions and Future Work

QoS services are extremely important on a network, but very complex to design and install. The best effort paradigm of IP networks is not enough to ensure quality to performance-dependent services such as multimedia.

As the action to be taken most of the times a transmission degrades is already known (decrease the number of frames on a video, degrade sound quality on a song, etc.), what matters is to be aware of the link conditions of connections. The QoS monitor presented in this paper was designed for an environment that needs quality of service. We presented techniques to develop and install monitoring agents that would work as network „thermometers“ by measuring QoS and submitting it to be compared with the hired value. This work also proposes a central element that „bridges the gap“ between the network manager, via management application, and the agents, besides combining information gathered from several sources and giving the data meaningful interpretation.

References

1. Waldbusser, S.: Remote Network Monitoring Management Information Base Version2 using SMIv2. IETF RFC2021, (1997)
2. Brownlee, N.: Traffic Flow Measurement: Experiences with NeTraMet. IETF RFC 2123. (1997)
3. Deri, L.; Carbone, R; Suin, S.: Monitoring Networks Using Ntop Netikos S.p.A. (2001).
4. Granville, L. Z.; Tarouco, L.: QAME - An Environment to Support QoS Management Related Tasks on IP Networks. In: Proc. IEEE International Conference on Telecommunications - ICT (2001). Bucharest-Romania. (2001)

A Framework for Supporting Intelligent Fault and Performance Management for Communication Networks

Hongjun Li and John S. Baras

Center for Satellite and Hybrid Communication Networks
Department of Electrical and Computer Engineering
University of Maryland, College Park, MD 20742
{hjli, baras}@isr.umd.edu

Abstract. In this paper, we present a framework for supporting intelligent fault and performance management for communication networks. Belief networks are taken as the basis for knowledge representation and inference under evidence. When using belief networks for diagnosis, we identify two questions: When can I say that I get the right diagnosis and stop? If right diagnosis has not been obtained yet, which test should I choose next? For the first question, we define the notion of right diagnosis via the introduction of intervention networks. For the second question, we formulate the decision making procedure using the framework of partially observable Markov decision processes. A heuristic dynamic strategy is proposed to solve this problem and the effectiveness is shown via simulation.

1 Introduction

In a communication network environment, we categorize the term fault as either *hard* or *soft*. Hard faults consist of hardware or software faults [20]. Hardware faults include incorrect or incomplete logic design, damage, wear or expiry, etc. Software faults usually come from incorrect or incomplete design and implementation. However, there are still some other important kinds of faults that need to be considered. For example, the performance of a switch is degrading or there exists congestion on one of the links. Another example is to model faults as deviations from normal behavior [21]. Since there might not be a failure in any of the components, we call such faults *soft* faults. Hard faults can be solved by replacing hardware elements or software debugging. Such diagnosis is called re-active diagnosis. Soft faults are in many cases indications of some serious problems and for this reason, the diagnosis of such faults is called pro-active diagnosis. Handling soft faults is typically part of the functionality of performance management [8][17] and in the sequel, we use the term fault to represent both hard and soft faults for convenience.

The task of fault management is to detect, diagnose and correct the possible faults during network operations. Fault detection can be thought of as an online

process that gives indication of malfunctioning. Such indications of malfunctioning are manifested in the form of events, which must be correlated to diagnose the most likely fault(s) [5][12][16]. Finally, corrective actions are taken to restore the normal operations. In this paper, we focus on fault diagnosis issues.

Efficient fault management requires an appropriate level of automation. Knowledge based expert systems, as examples of automated systems, have been very appealing for communication networks fault diagnosis [14]. Usually, such systems are based on deterministic network models. A serious problem of using deterministic models is their inability to isolate primary sources of faults from uncoordinated network events. Observing that the cause-and-effect relationship between symptoms and possible causes is inherently nondeterministic, *probabilistic* models can be considered to gain a more accurate representation.

Hood and Ji [9] proposed a pro-active network fault detection scheme based on AR models and belief networks. However, their belief network model is over simplistic in that there is only one root node, which will explain whatever anomalies as detected by the AR modeling. It estimates the network status in a snapshot; there is no further test suggested. In [10], Huard and Lazar used a more general belief network model with multiple root nodes as the candidate faults. They also presented a dynamic programming (DP) formulation for the network troubleshooting problem. However, single fault assumption was made, which limits the applicability. In this paper, we develop a framework that supports fault diagnosis for communication networks. General belief network models with multiple root nodes are chosen as the knowledge representation scheme. We handle multiple faults and formulate the fault diagnosis procedure as a Partially Observable Markov Decision Processes (POMDP) problem with optimal stopping. To help solve the problem, we introduce the notion of right diagnosis for optimal stopping and provide a dynamic, heuristic strategy for test sequence generation.

The rest of the paper is organized as follows. In section 2, we introduce belief networks and identify two problems when using belief networks for diagnosis. We introduce the concept of intervention networks and right diagnosis for the first problem in section 3, and the decision theoretic fault diagnosis strategies are studied in section 4. We run simulation in section 5, and conclude the paper in section 6.

2 Fault Diagnosis Problems Using Belief Networks

A belief network, also called a Bayesian network or a causal network, is a graphical representation of cause-and-effect relationships within a problem domain. More formally, a belief network $\mathcal{B}=(V, L, P)$ is a Directed Acyclic Graph (DAG) in which: The nodes V represent variables of interest (propositions); The set of directed links L represent the causal influence among the variables; The strength of an influence is represented by conditional probability tables (CPT). For any node in the DAG, given its parents, that node is conditionally independent of any other node that is not its descendent. This conditional independence makes a belief network model a compact representation of the joint probability dis-

tribution P over the interested variables. Belief networks can also serve as the inference engine, and can compute efficiently any queries over the variables modeled therein[11][22]. Let us look at one example.

Suppose we are handling the problem call failure and identify the possible causes as follows: Server, link and switch may fail, and there might be heavy traffic that causes the network congestion. Luckily, we have access to the alarms associated with link failure and switch failure. This scenario is modeled as a belief network, as shown in figure 1. Each node takes binary value and the table associated with it represents the conditional probability distribution, given its parent nodes' instantiations.

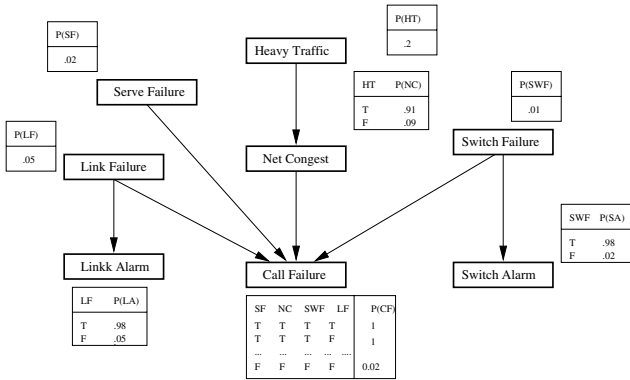


Fig. 1. An example Belief Network

In communication networks, probes are attached to some hardware/software components to get operation status. Typically the raw data returned from the probes will be grouped into vector form $\mathbf{d} \in \mathbf{R}^n$ and then processed to get some aggregated values (e.g. average, peak value, etc.). A statistics is a function from \mathbf{R}^n to \mathbf{R} that maps the raw data vector \mathbf{d} to a real number. Such statistics will usually be quantified and represented using discrete values. We use 0 to represent normal status, and other positive integers to represent abnormal status with different level of severity. A node v in a belief network model $\mathcal{B}=(V, L, P)$ is called observable if and only if it represents the health status of a statistics, or corresponds to a user report. The set of observable nodes is denoted by O . The non-observable set is simply $\tilde{O} = V \setminus O$. We restrict these observable nodes to be leaf nodes only, and vice versa. The regular evidence set R contains those nodes that we observe during regular network monitoring operations. Each $r \in R$ is called a symptom node. The test set ST contains all other observable nodes that are not currently in R , namely $ST = O \setminus R$. The fault set F is the set of root nodes, and they are not observable, $F \subseteq \tilde{O}$. We restrict that all root nodes are binary valued. The hidden node set H contains all nodes in \tilde{O} but not in fault

set F , $H = \tilde{O} \setminus F$. Hidden nodes are intermediate nodes between faults and symptoms and we don't usually put queries on them during diagnosis.

The problem domain is said to be working in normal status with respect to regular evidence set R if and only if every node in R takes value 0, or vector $\mathbf{r} = \mathbf{0}$, where $\mathbf{r} = (r_1, r_2, \dots, r_{|R|})$. The problem domain is said to be working in abnormal status with respect to regular evidence set R if and only if there is at least one $r \in R$ whose value is other than 0. There might be cases when multiple symptom nodes in R take nonzero values. The syndrome with respect to regular evidence set R is simply the nonzero vector \mathbf{r} . Any syndrome can trigger the diagnosis process.

After fault diagnosis is triggered, the initial evidence is propagated and the posterior probability of any $f \in F$ being faulty can be calculated. It would be ideal if we can locate the fault with efforts up to this. But most of the time, similar to what happens in medical diagnosis, we need more information to help pinpoint the fault. So naturally, we identify two important problems associated with belief network based fault diagnosis: When can I say that I get the right diagnosis and stop? If right diagnosis has not been obtained yet, which test should I choose next? We address these two problems in the next sections. In our work, we postulate that all the observations and tests are constrained within the belief network model.

3 Right Diagnosis via Intervention

Consider what a human usually think during diagnosis. After obtaining one possible reason, one may naturally ask, for example, "Will the problematic circuit work normally if I replace this suspicious component with a good one?" He/she then goes ahead and sees what will happen after the replacement. If the syndrome disappears, one can claim that he/she actually found and trouble-shooted the fault. If the problem domain is tiny, not very complex, and the replacement burden is light, this paradigm will work well. But for communication networks, the story is totally different. We would like to do intelligent diagnosis via *computation*, rather than brutal replacement before we are very confident what the fault is.

To do this, we need to distinguish between two kinds of semantics for the instantiation of a node in a belief network: passive observation and active setting. All the instantiations of nodes we have talked about so far are passive observations, and we would like to know the consequences of, and the possible causes for such observations. The alternative semantics is that we can also *set* the value of a node via active experiment. One example is the above question, where external reasons (the human diagnoser) explain why the suspicious component becomes good and thus all the parent nodes for this node should not count as causes during belief updating. Other belief updating like evaluating consequences, however, are not influenced by this active setting. This external force is called *intervention* in [23].

With this *set* semantics, we could do virtual replacement in our belief network model. For simplicity, we assume here that the single symptom node is $S1$. For each node in F , we could get its posterior probability of being faulty given $S1 = 1$. Let $f = \operatorname{argmax}_{g \in F} P(g = 1 | S1 = 1)$, and we would evaluate $P(S1 = 0 | \operatorname{setting}(f = 0))$. Other nodes in F are treated as background variables and they keep at the same status as what has just been updated. In our work, we introduce the so-called intervention belief network to help this virtual replacement.

Definition 1. An *intervention belief network* $\tilde{\mathcal{B}} = (V, L, P, S, Fs)$ is obtained from the original belief network $\mathcal{B} = (V, L, P)$ with the same V, L, P . S is the symptom set and $Fs \in F$ is the set of suspicious nodes. We compute for each $s \in S$ the probability $P(s = 0 | \operatorname{setting}(Fs = \mathbf{0}))$ using $\tilde{\mathcal{B}}$.

For our particular example above, the virtual replacement procedure is as follows. First, in $\mathcal{B} = (V, L, P)$, update for each node $f_i \in F$ the probability $p_i \triangleq P(f_i = 1 | S1 = 1)$. Suppose $f_1 = \operatorname{argmax}_{g \in F} P(g = 1 | S1 = 1)$. Then in intervention belief network $\tilde{\mathcal{B}} = (V, L, P, S1, f_1)$, set node $f_1 = 0$, and with $P(f_i = 1) = p_i, i = 2, \dots, |F|$, compute $P(S1 = 0 | \operatorname{setting}(f_1 = 0))$. To determine whether or not this virtual replacement has led $S1$ to an acceptable status, we need a reference value for the computed $P(S1 = 0 | \operatorname{setting}(f_1 = 0))$ to compare with. Without any evidence input, the belief network model \mathcal{B} itself gives the marginal probability of each leaf node to be normal. We use these values as the reference in our work.

Definition 2. Given a small number ϵ , we say that node $S1$ becomes ϵ -normal via intervention on f_1 if and only if $P(S1 = 0) - P(S1 = 0 | \operatorname{setting}(f_1 = 0)) < \epsilon$.

Note that during diagnosis process, some of the testing nodes chosen may already manifested themselves as values other than “normal”. These nodes should also be included in intervention network $\tilde{\mathcal{B}}$.

Definition 3. A nonempty set of suspicious nodes Fs is called the explanation or right diagnosis if and only if every node in set S , including both initial and newly-found symptoms, becomes ϵ -normal if we set every node in Fs to normal in the intervention belief network $\tilde{\mathcal{B}} = (V, L, P, S, Fs)$. It is when Fs explains the set S that we terminate the diagnosis process.

4 Decision Theoretic Fault Diagnosis Strategies

We formulate the test selection procedure as a partially observable Markov decision processes (POMDP) problem with optimal stopping. At each decision epoch, we could either choose a node to test or stop there. Test is rarely free, and termination incurs some costs. The goal is to find a good test sequence and the right time to stop. We will show that by choosing termination cost appropriately, the optimal stopping rule matches our notion of right diagnosis.

4.1 POMDP Formulation

State Space \mathcal{S}

The state is the status of the root nodes $F = \{F_1, \dots, F_{|F|}\}$, and for a particular $s \in \mathcal{S} = 2^{|F|}$, $s = \{f_1, \dots, f_{|F|}\}$. We use S_k to denote the state at time k . In our diagnosis case, the current state, which is unobservable, does not change regardless what tests will be chosen. The goal of diagnosis is to *identify* this state by using initial symptoms and subsequent test results. So here we have

$$P(S_{k+1}|S_k) = \begin{cases} 1 & \text{if } S_{k+1} = S_k \\ 0 & \text{otherwise} \end{cases} \quad (1)$$

History Process

If we choose one test per decision epoch, the time step set is defined as $N = \{1, 2, \dots, |ST|\}$. The active evidence set AE contains the nodes that are instantiated during the process of diagnosis. Initially $AE = R$ and it expands as more test nodes in ST are added into it. Nodes in AE are not to be considered for future use. The candidate test set C_{st} contains the nodes in ST that are available to be chosen and tested. Initially $C_{st} = ST$ and it shrinks as instantiated nodes are removed from it. The action set $A = C_{st} \cup \{STOP\}$. Let Z_{a_t} denote the value obtained by observing a_t , and we define the history process up to time k as $I_k = (Z_0, (a_1, Z_{a_1}), \dots, (a_k, Z_{a_k}))$, where $Z_0 = ((r_1, Z_{r_1}), \dots, (r_{|R|}, Z_{r_{|R|}}))$ represents the regular evidence set and corresponding instantiations. I_k grows with diagnosis and obviously, $I_k = (I_{k-1}, (a_k, Z_{a_k}))$, the Markov property. We can simply take I_k as the *state* at time k and obtain a completely observable Markov decision problem. But the growing state process makes this approach impractical.

Belief / Information State

Given I_k , we define $b_k = P(\mathbf{F}|I_k)$ as the probability distribution of states in \mathcal{S} . It is proven that b_k is a sufficient statistics that contains all information embedded in the history process for control, and we call it belief or information state [2]. Using Bayes rule, we can easily verify that the process $\{b_k\}$ is also Markov. If we choose b_k as the state at time k , we avoid the growth of the state space; but now, the state space is *continuous*, and we call it \mathcal{B}_c . In our case, if we are given I_k, a_k , and Z_{a_k} , the next belief state b_{k+1} is uniquely determined via belief network propagation, and we define $\Psi(b_k, a_k, Z_{a_k}) \triangleq Pr(b_{k+1}|b_k, a_k, Z_{a_k})$. If we let $\mathcal{X} = \mathcal{B}_c \cup \{T\}$ and x_k be the state at time k , then the augmented states evolve according to

$$x_{k+1} = \begin{cases} \Psi(x_k, a_k, Z_{a_k}) & \text{if } x_k \neq T \text{ and } a_k \neq STOP \\ T & \text{if } x_k = T \text{ or } (x_k \neq T \text{ and } a_k = STOP) \end{cases} \quad (2)$$

The observation model for $a_k \neq STOP$ is $P(Z_{a_k}|I_k, a_k) = Pr(a_k = Z_{a_k}|I_k)$.

Choosing Suspicious Nodes

After we obtain x_k , it will not suffice to give out this probability distribution directly as the result. What is needed is the explanation. To see if we could obtain the explanation as defined above, we need to extract from x_k the suspicious

nodes. However, in our belief network model and the parallel intervention model, we should be discreet in choosing multiple nodes. If we simply choose all nodes in F and do the intervention, the symptom nodes will all become ϵ -normal for sure. But clearly, calling every node in F as faulty is not acceptable; One of the most important aspects of fault diagnosis in general is to bias among the many possible faults and locate the real one(s)! In our work, we used the following scheme.

We first compute the belief state and get a table that contains the joint distribution of the root nodes given I_k . Then we choose the largest entry from the table and mark the index of the entry. The suspicious nodes are obtained from the index. For example, if we only have four root nodes and the binary string corresponding to the index of the largest entry is 0101, then the second and fourth nodes are chosen. In this scheme, there is no need to find a good η , and it adapts to multiple causes easily.

Cost Structure

There is an immediate cost associated with each $s_i \in ST$. The cost function $C(s_i, t)$ entails careful deliberation about many factors like the difficulty and time to be consumed for the test, etc. Here we assume that the cost function is of form $C(s_i)$. This is usually the case in that the cost is normally associated with the test itself only, and the test itself does not usually change with time. Also, we wish to diagnose promptly and we penalize on diagnosis steps. If $a_k = STOP$ at time k , no penalty. Otherwise, we penalize this extra step using function $g(k)$. Here, we simply take $g(k) = 1$ for all k . At time k with state $x_k \neq T$, if we choose $a_k = STOP$, we incur $t(x_k)$ as the termination cost. Note that $t(T) = 0$. Given $x_k \neq T$ and suspicious node set F_s , we compute $t(x_k)$ as follows. First, in original belief network, let $K = F \setminus F_s$ and compute for each node in K the probability of being faulty as $q_i \triangleq Pr(K_i = 1|I_k)$. Second, in intervention network, set the root nodes that correspond to those in K with the same probabilities as those in $\{q_i\}$, and set the root nodes that correspond to those in F_s to state "normal". Finally, in intervention network for each node S_i in the active symptom set S , and for some given small ϵ , define $\Delta = P(S_i = 0) - P(S_i = 0|Setting\ root\ nodes\ as\ above)$. If $\Delta < \epsilon$, $t_{S_i}(x_k) = 0$, else $t_{S_i}(x_k) = CONST[\Delta - \epsilon]$, where $CONST$ is a constant to make $t_{S_i}(x_k)$ large. The total cost is $t(x_k) = \sum_{S_i \in S} t_{S_i}(x_k)$. So, the immediate cost of choosing action a_k at time k with state $x_k \neq T$ is

$$g_k(x_k, a_k) = \begin{cases} c(a_k) + g(k) & \text{if } a_k \neq STOP \\ t(x_k) & \text{otherwise} \end{cases} \tag{3}$$

At the last step N , the terminal cost $g_N(x_N)$ is defined as

$$g_N(x_N) = \begin{cases} t(x_N) & \text{if } x_N \neq T \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

Note that both $g_k(x_k, a_k)$ and $g_N(x_N)$ are deterministic functions. Now we have the finite horizon problem

$$\min_{a_k, k=0, \dots, N-1} \left\{ g_N(x_N) + \sum_{k=0}^{N-1} g_k(x_k, a_k) \right\}. \tag{5}$$

4.2 Solution for the Problem

Define $J_k(x_k)$ as the cost-to-go at state x_k and time k [2]. At termination state T , $J_k(T) = 0, \forall k = 0, \dots, N - 1$. For $x_k \neq T$, we have the dynamic programming algorithm:

$$J_N(x_N) = g_N(x_N) \tag{6}$$

$$J_k(x_k) = \min \left[t(x_k), \min_{a_k \in A_k} [c(a_k) + g(k) + \sum_j P(a_k = j | I_k) J_{k+1}(x_{k+1})] \right], \tag{7}$$

where $x_{k+1} = \Psi(x_k, a_k, Z_{a_k})$. So the optimal stopping policy is: Choose STOP if

$$t(x_k) \leq \min_{a_k \in A_k} [c(a_k) + g(k) + \sum_j P(a_k = j | I_k) J_{k+1}(\Psi(x_k, a_k, Z_{a_k} = j))], \tag{8}$$

at current state x_k and time k . If we choose $t(x_k)$, as shown above, such that $t(x_k) = 0$ in the case of right diagnosis and let $t(x_k)$ be very large otherwise, then the optimal stopping policy is: STOP if and only if we obtain the right diagnosis. Now let us look at the test selection strategies.

As discussed above, we need to extract from state $x_k \neq T$ the suspicious node set F_s . We ignore those root nodes that are not very fault-prone and this is our first approximation. Now, given that F_s does not explain the current active symptoms, we need some heuristics to help choose the next test. Let us begin with a simpler problem for intuition.

Suppose the concern here is to locate the single faulty component. There are symptoms indicating the malfunction (e.g. car doesn't start) and for each possible faulty component there is a *direct* test associated with it. The cost for testing component i is c_i . Based on the symptoms, we obtain P_i , the probability that component i is in failure, for every component. We are supposed to test those components one at a time. As soon as one component fails its associated test, we claim that we find the single fault and stop. By interchange argument [2], it is easy to see that in an optimal strategy, all elements must be in non-decreasing sequence of c/P values, see also [13].

Our problem is different from this scenario in the following aspects. It tackles failures while our problem integrates both hard and soft faults. It assumes the existence of direct test while we don't have that luxury. For a communication network environment which is distributed, complex and heterogeneous, it is impossible to predefine and store a direct test for each possible cause. Actually one of the goals here is to *generate* dynamically the test sequence on the fly. In our setup, right diagnosis is determined through computation, rather than brutal replacement. Finally, our algorithm should be able to tackle multiple faults.

But the c/P algorithm does provide insight in that it reflects the following observation: in order to minimize the total cost, people are more likely to try those more fault-prone, cheaper components before the less-probable, expensive ones. In our diagnosis algorithm, we wish to find an appropriate test node st if F_s could not explain the active symptom set S . In particular, we would like

to choose the test node from candidate test set C_{st} that is cheapest and most relevant to F_s . To achieve this, we need a measure for relevance between a test node in C_{st} and a fault node in F_s .

Definition 4. Given I_k , the relevance of random variable Y relative to random variable X is defined as

$$R(X; Y|I_k) = \frac{I(X; Y|I_k)}{H(X|I_k)},$$

where $H(X|I_k) = -\sum_{x \in \mathcal{X}} p(x|I_k) \log p(x|I_k)$ is the conditional entropy of a random variable X , $I(X; Y|I_k) = \sum_{x \in \mathcal{X}} \sum_{y \in \mathcal{Y}} p(x, y|I_k) \log \frac{p(x, y|I_k)}{p(x|I_k)p(y|I_k)}$ is the conditional mutual information between random variable X and Y [4]. $R(X; Y|I_k) \in [0, 1]$ indicates to what extent Y can provide information about X . $R(X; Y|I_k) = 1$ means that Y can uniquely determine X , while $R(X; Y|I_k) = 0$ indicates that Y and X are independent, given current I_k . Note that $R(X; Y|I_k) \neq R(Y; X|I_k)$. More generally,

Definition 5. Given I_k , the relevance of random variable Y relative to a set of random variables \mathbf{X} is

$$R(\mathbf{X}; Y|I_k) = \frac{I(\mathbf{X}; Y|I_k)}{H(\mathbf{X}|I_k)},$$

where $H(\mathbf{X}|I_k)$ and $I(\mathbf{X}; Y|I_k)$ are defined similarly as above.

With the relevance measure, our next test node given I_k at time k is simply

$$st = \operatorname{argmax}_{g \in C_{st}} R(\mathbf{F}_s; g)/c(g), \tag{9}$$

and our fault diagnosis process is summarized as follows, also shown in figure 2.

- Step 1. Initialization
 - Set time step $tp = 0$, $AE = R$, $C_{st} = ST$.
 - Input evidence by setting the nodes in set AE according to current active values ae .
 - Step 2. Belief Propagation in belief network \mathcal{B} and get the set of suspicious nodes F_s according to scheme one or two.
 - Step 3. Set the root nodes in $\tilde{\mathcal{B}}=(V, L, P, S, F_s)$ accordingly, and execute the intervention. If F_s explains S , update total cost and TERMINATE.
 - Step 4. Get next testing node
 - If $C_{st} = \emptyset$, update total cost and give out the set F_s and say "Didn't find the right diagnosis, but here is the list of possible faults in decreasing order".
 - Else: Get node st according to (9).
 - Step 5. Observing test node st and get observation Z_{st}
 - Input this evidence $st = Z_{st}$ to original belief network \mathcal{B} . Update tp , C_{st} , and AE .
 - Goto Step 2.
-

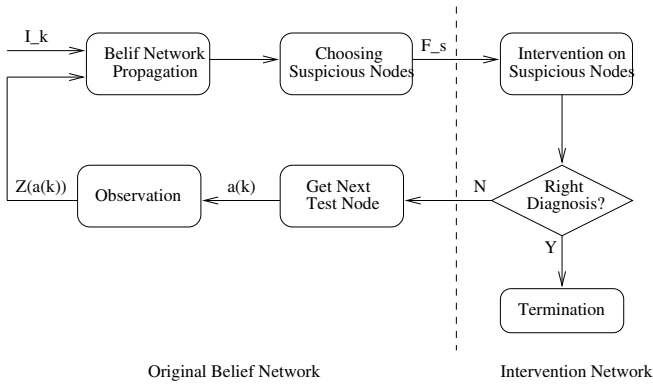


Fig. 2. Illustration of the diagnosis process using intervention belief network

5 Simulation

To illustrate the effectiveness of our fault diagnosis algorithm, consider the example network in figure 3. Two switches *SW1* and *SW2* are connected via link *L1*. We have a probe *a* hooked at the end of *SW2* to measure the traffic throughput going out of *SW2*. Suppose the information we could obtain during network operation include whether or not: *SW1* alarm is normal, *A* could connect *SW2*, *B* could connect *SW2*, *A* could connect *C*, *C* could connect *SW1*, throughput at probe *a* is normal, and *D* could connect *SW1*. The possible faults are identified as: *SW1* works normal or not, *L1* normal or congested, *SW2* normal or not, and source pumped from *C* to *L2* is normal or not. We set up a belief network model for such situations, and figure 4 shows the structure and initial probability distributions.

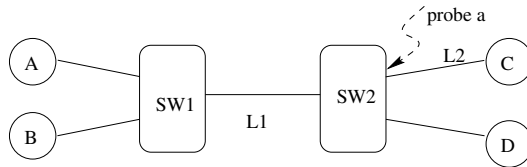


Fig. 3. Example Network

Let us look at one diagnosis scenario. Suppose we observe that *A_Conn_SW2* goes wrong, and we get the updated distribution as shown in figure 5. We see that *SW1* is the suspicious node and the intervention result is $P(A_Conn_SW2 = yes|Intervention) = 0.78$. Initially, $P(A_Conn_SW2 = yes) = 0.83$, and we have not yet got the right diagnosis for $\epsilon = 0.4$. Based on our test selection scheme, node *SW1_Indicator* is chosen and the observation of it is “normal”.

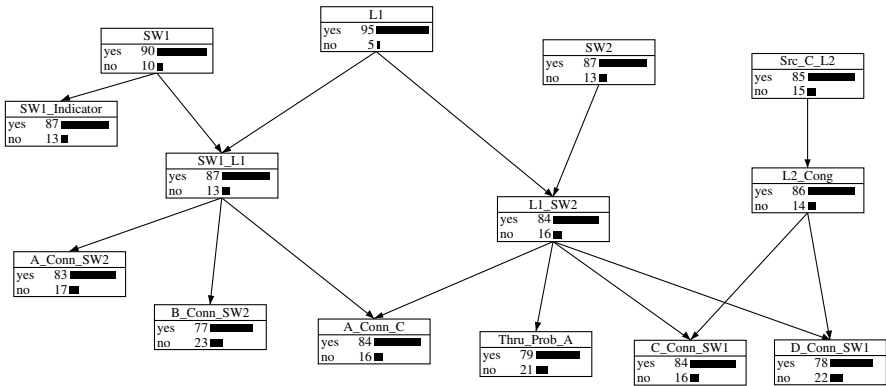


Fig. 4. Belief Network for Example Network

The updated distribution is shown in figure 6. Again, *L1* is intervened and no right diagnosis is obtained. The next node selected this time is *A_Conn_C* and the observation is “abnormal”. We got the updated distribution again in figure 7. If we intervene node *L1*, we have $P(A_Conn_SW2 = yes|Intervention) = 0.87 > 0.83$, and we obtain the right diagnosis!

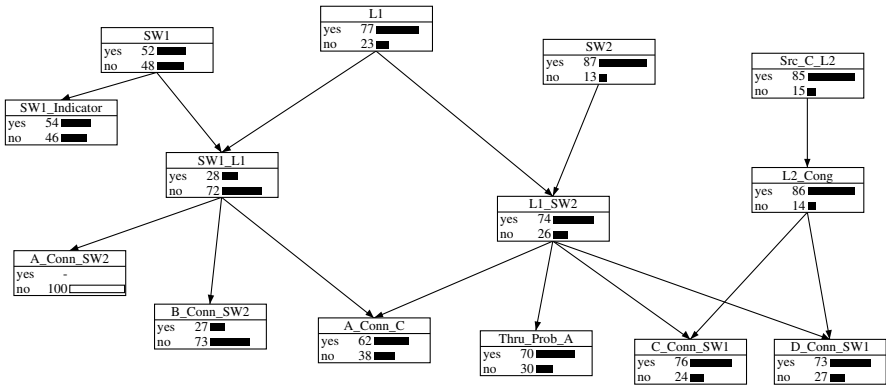


Fig. 5. After *A_Conn_SW2* Goes Wrong

As a comparison to our node selection scheme, we use the random scheme meaning that each time we need a test node, we simply choose one uniformly from all current available nodes in C_{st} . In our simulation, the outcome of chosen test node st is uniformly generated as either 0 or 1. The costs for testing each leaf node is shown in Table 1, with 40 as the penalty for not being able to find the right diagnosis. Table 2 shows for three scenarios the comparisons of the two test generation schemes with 2000 runs, which take only about 40 milliseconds

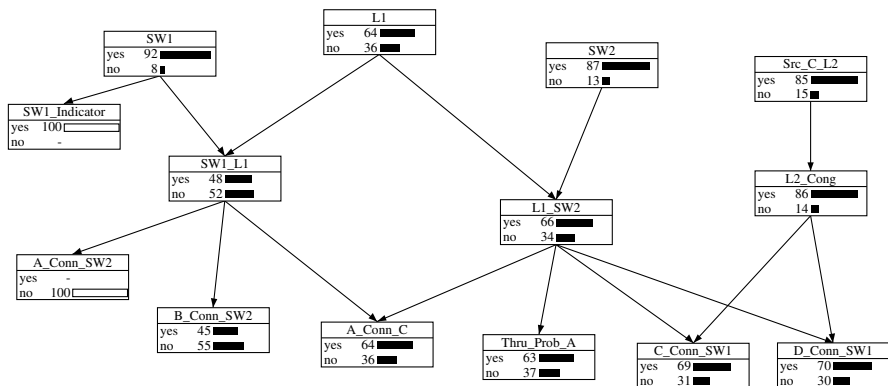


Fig. 6. After SW1_Indicator Observed as Normal

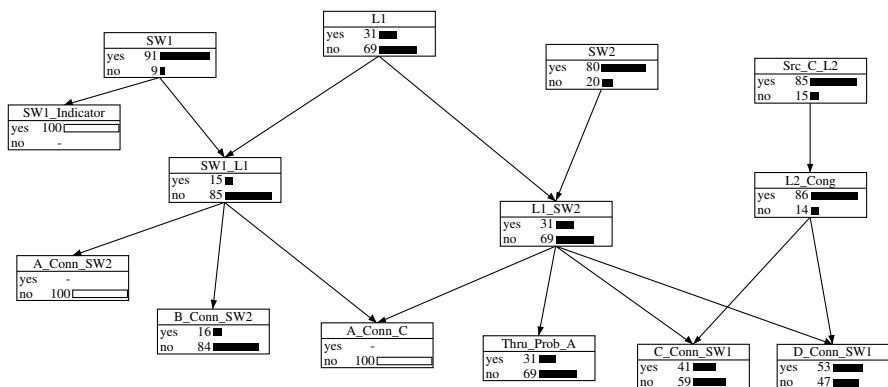


Fig. 7. After A_Conn_C Observed as Abnormal

per run for each scenario on a SUN Ultra2 running Solaris 8. We see that node selection via relevance is much better than that via random selection.

Table 1. Cost for All Leaf Nodes

SW1_Indicator	A_Conn_SW2	B_Conn_SW2	A_Conn_C	Thru_Prob_A	C_Conn_SW1	D_Conn_SW1
2	1	7	1	3	1	3

6 Conclusions

In this paper, we presented a framework that supports intelligent fault and performance management for communication networks. We used belief networks as the knowledge representation scheme and inference engine for the problem

Table 2. Comparison of Node Selection Schemes

Symptom Nodes	Random Selection		Relevance Selection	
	Avg. Cost	Success Rate	Avg. Cost	Success Rate
A_Conn_SW2	15.38	84.5%	9.13	94%
A_Conn_C	26.21	70.1%	14.22	88%
A_Conn_SW2 and A_Conn_C	24.68	67.8%	3	100%

domain. The optimal stopping problem is tackled by using the notion of right diagnosis via intervention, and test selection is based on a heuristic dynamic strategy. Simulation shows that this scheme is much superior than a random selection scheme.

This framework is quite general. The belief network model and the associated decision making algorithm could exist at any management station in a network management system. After a test node is chosen, the observation for this test may take advantage of the traditional SNMP paradigm by polling appropriate MIB variables; or, delegated (mobile) agents could be sent to the network elements to collect the data by using the management by delegation paradigm [6]. As one example of such an agent-based environment, the authors presented in [19] a couple of system designs for adaptive, distributed network monitoring and control. Further, the managed system could be divided into domains [24], and for each domain we could assign such an “intelligent” module that take charge of the fault and performance management for it [1][18].

The dynamic heuristic strategy could be improved via reinforcement learning [3][25], and in particular, Q -learning techniques [26]. The idea is that, by interacting with the environment, the decision making module could accumulate experience and improves its performance. We could use the above dynamic strategy as the starting point. We will discuss the details in a forthcoming paper. Also, simulation on a more complex network environment is on the way.

References

1. J. S. Baras, H. Li and G. Mykoniatis, “Integrated, Distributed Fault Management for Communication Networks”, Technical Report, CSHCN TR 98-10, University of Maryland, 1998
2. D. P. Bertsekas, *Dynamic Programming and Optimal Control, Vol. I and II*, Athena Scientific, Belmont, MA, 1995
3. D. P. Bertsekas, and J. N. Tsitsiklis, *Neuro-Dynamic Programming*, Athena Scientific, 1996
4. T. M. Cover, and J. A. Thomas, *Elements of Information Theory*, Wiley Interscience, 1991
5. Gabrijela Dreo, “A Framework for Supporting Fault Diagnosis in Integrated Network and Systems Management: Methodologies for the Correlation of Trouble Tickets and Access to ProblemSolving Expertise”, PhD Dissertation, Department of Computer Science, University of Munich, 1995
6. G. Goldszmidt, Y. Yemini, “Distributed Management by Delegation”, in *Proceedings of 15th International Conference on Distributed Computing Systems*, 1995

7. D. Heckerman, J. S. Breese, and K. Rommelse, "Decision-Theoretic Troubleshooting", *Communications of the ACM*, vol. 38, pp. 49-57, 1995
8. H.G. Hegering, S. Abeck, and B. Neumair. *Integrated Management of Networked Systems: Concepts, Architectures, and Their Operational Application*. Morgan Kaufmann, San Francisco, CA, USA, 1999.
9. C. S. Hood, and C. Ji, "Probabilistic Network Fault Detection", *GlobalCom*, pp. 1872-1876, 1996
10. J. Huard, and A. A. Lazar, "Fault Isolation based on Decision-Theoretic Troubleshooting", Tech. Rep. TR 442-96-08, Center for Telecommunications Research, Columbia University, 1996
11. <http://www.hugin.dk>
12. G. Jacobson, M. Weissman, "Alarm Correlation", *IEEE Network*, Vol. 7, No. 6, 1993
13. J. Kalagnanam and M. Henrion, "A Comparison of Decision Analysis and Expert Rules for Sequential Diagnosis", in *Uncertainty in Artificial Intelligence 4*, pp. 271-281, Elsevier Science Publishers B. V., 1990
14. L. Kerschberg, R. Baum, A. Waisanen, I. Huang and J. Yoon, "Managing Faults in Telecommunications Networks: A Taxonomy to Knowledge-Based Approaches", *IEEE*, pp. 779-784, 1991
15. L. P. Kaelbling, M. L. Littman, and A. R. Cassandra, "Planning and acting in partially observable stochastic domains", *Artificial Intelligence*, Vol 101, pp. 99-134, 1998
16. S. Kliger, S. Yemini, Y. Yemini, D. Ohsie, and S. Stolfo. "A Coding Approach to Event Correlation." In Sethi, Raynaud, and Faure-Vincent, editors, *Integrated Network Management*, no. 4, pp. 266-277. May 1995.
17. A. Leinwand and K. F. Conroy, *Network Management, A practical perspective*, second edition, Addison-Wesley, 1996
18. H. Li, J. S. Baras and G. Mykoniatis, "An Automated, Distributed, Intelligent Fault Management System for Communication Networks", *ATIRP'99*, 1999
19. H. Li, S. Yang, H. Xi, and J. S. Baras, "Systems Designs for Adaptive, Distributed Network Monitoring and Control", *IFIP/IEEE International Symposium on Integrated Network Management*, Seattle, Washington, May 2001, to appear.
20. G. Mahamat, A. Das, G.V. Bochmann, "An overview of fault management in telecommunication networks", *Advanced Information Processing Techniques for LAN and MAN Management*, 1994 IFIP
21. R. Maxion, "A case study of ethernet anomalies in a distributed computing environment", *IEEE Trans. on Reliability*, Vol. 39, No. 4, pp. 433-443, Oct 1990
22. J. Pearl, *Probabilistic Reasoning In Intelligent Systems: Networks of Plausible Inference*, Morgan Kaufmann, 1988
23. J. Pearl, *Causality*, Cambridge Press, 2000
24. M. Sloman and K. Twidle. "Chapter 16. Domains: A Framework for Structuring Management Policy". In M. Sloman (Ed.). *Network and Distributed Systems Management*, pp. 433-453. Addison-Wesley, Wokingham, UK, 1994.
25. R. S. Sutton, A. G. Barto, *Reinforcement Learning: An Introduction*, MIT Press, 1998
26. C.J.C,H. Watkins, P. Dayan, "Q-learning", *Machine Learning*, 8, pp. 279-292, 1992

Architecture of Generalized Network Service Anomaly and Fault Thresholds

Zheng Zhang¹, Constantine Manikopoulos¹, and Jay Jorgenson²

¹ECE Department, New Jersey Institute of Technology, University Heights,
Newark, NJ 07102, USA

zzz9622@njit.edu, manikopoulos@adm.njit.edu

²Department of Mathematics, CCNY, Convent Ave. at 138 ST.,

New York, NY 100031, USA

jjorgenson@mindspring.com

Abstract. In this paper we introduce GAFT (Generalized Anomaly and Fault Threshold), featuring a novel system architecture that is capable of setting, monitoring and detecting generalized thresholds and soft faults proactively and adaptively. GAFT monitors many network parameters simultaneously, analyzes statistically their performance, combines intelligently the individual decisions and derives an integrated result of compliance for each service class. We have carried out simulation experiments of network resource and service deterioration, when increasingly congested in the presence of class-alien traffic, where GAFT combines intelligently, using a neural network classifier, 12 monitored network performance parameter decisions into a unified result. To this end, we tested five different types of neural network classifiers: Perceptron, BP, PBH, Fuzzy ARTMAP, and RBF. Our results indicate that BP and PBH provide more effective classification than the other neural networks. We also stress tested the entire system, which showed that GAFT can reliably detect class-alien traffic with intensity as low as five to ten percent of typical service class traffic.

1 Introduction

Network faults can be classified into two types: *hard failures*, in which the network, or some of its elements, are not able to deliver any traffic at all; *soft failures*, network/service anomaly or performance degradation in various performance parameters, i.e., decrease in bandwidth, increase in delay, etc. A hard fault can be easily noticed by all, the system administrators as well as the users. However, defining and otherwise characterizing and detecting soft faults is difficult. Wireless networks are particularly vulnerable to soft faults in that they have much lower bandwidth than their wired counterparts, while they are more prone to overloads, noise, congestion, etc.

In the past few years, some research progress has been made in soft fault detection. A proactive fault management system is presented in [3]. In [2] a study was carried out of path failure detection. In [4], neural networks are applied in billing fraud detection in telecommunication voice networks. Performance anomaly detection in Ethernet is discussed in [5]. In [1], Cabreta et. al. described their practice of using Kolmogorov-Smirnov (K-S) statistics to detect Denial-of-Service and Probing attacks.

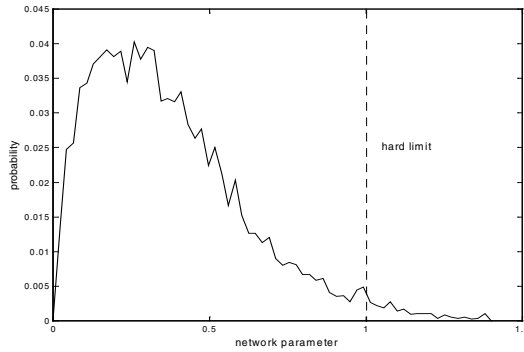


Fig. 1. PDF of a Network Parameter

In this paper, we propose a hierarchical, multi-tier, multi-window, soft fault detection system for both wireless and wired networks, which operates automatically, adaptively and proactively. The system uses statistical models and neural network classifiers to detect anomalous network conditions. The statistical analysis bases its calculations on PDF algebra, in departure from the commonly used isolated sample values or perhaps their averages. The use of PDFs is much more informative, allowing greater flexibility than just using averages. The sample PDF of some network performance parameter, shown in Fig. 1, helps illustrate that point. The figure depicts a hard service limit at some parameter value, drawn as a vertical line. Such a limit might apply for a real-time service, for example, the total packet delay parameter for packetized voice. From the point of view of the hard limit, the network condition described by the PDF in this figure would represent a service failure, in that some packets exceed the limit.

However, the PDF shows that the totality of such failing packets, those in the tail of the PDF to the right of the limit, may be small. Thus, if in fact, this total is smaller than the maximum packet loss rate specification, the service may remain in compliance to the delay limit simply by discarding the laggard packets. Our system generalizes further by combining information of the PDFs of the monitored performance parameters, either all of them or subgroups of them, in one integrated and unified decision result. This combining is powerful in that it achieves much higher discrimination capability that enables the monitoring of individual service classes in the midst of general traffic consisting of all other classes. It is also capable of discriminating against intrusion attacks, known or novel. In fact, network intrusion

detection is one of the promising applications of this technology. Others, include transaction oriented e-commerce networks, that typically support large numbers of service classes concurrently, and, as mentioned, various wireless networks, where monitoring the service classes may be challenging due to adverse network conditions.

GAFT gathers data from network traffic, the system log and hardware reports; it statistically processes and analyzes the information, detects network anomaly and failures, for each parameter individually, or in combined groups using neural network classification; it generates the system alarms and event reports; finally, GAFT updates the system profiles based on the newly observed network patterns and the system outputs (see Fig. 2).

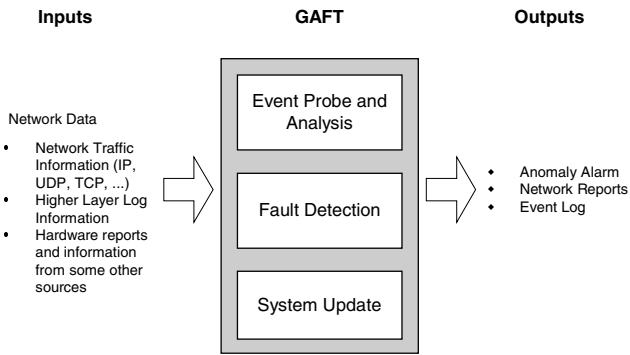


Fig. 2. The Inputs and Outputs of GAFT

The rest of the paper is organized as follows. The system architecture is outlined in Section 2. Section 3 introduces the statistical algorithms that we are using. Section 4 describes the five neural networks we tested. The simulation environment is given in Section 5. The experimental results using neural network classifiers are presented in section 6. Section 7 reports the results of stress testing on GAFT. Section 8 draws some conclusions and outlines future work.

2 System Architecture

Our system is a distributed application, deployed hierarchically, which consists of several tiers, with each tier containing several Fault Management Agents (FMAs). FMAs are management components that monitor the activities of a host or the network it is attached to. Different tiers correspond to different network scopes that are monitored by the agents affiliated to them.

Generally speaking, each FMA collects network status reports from FMAs one tier below and combines them with data that it collects from its own system logs, as well

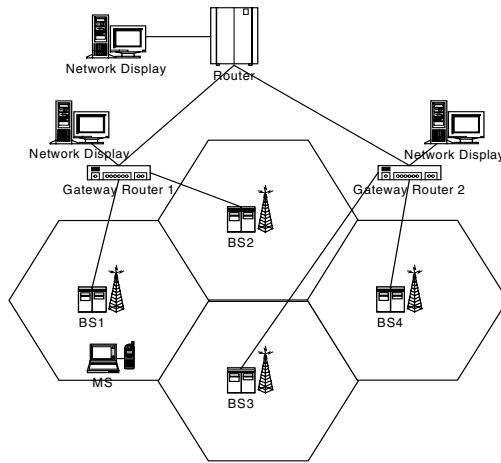


Fig. 3. Sample Wireless Network

as its monitoring of the network; it then generates a network status report, that it sends to the FMA of the next higher tier.

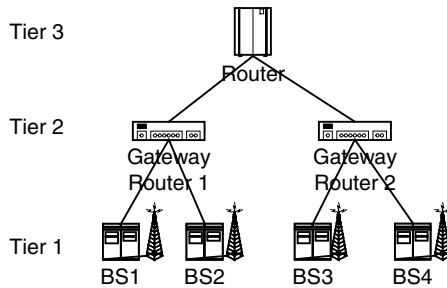


Fig. 4. System Hierarchy

For the sample IP wireless network shown in Fig. 3, the fault detection system can be divided into 3 tiers. Tier 1 agents monitor system activities of the base stations, as well as the nodes in the subnet that the base station supports; they also periodically generate reports for the next higher tier, i.e. Tier 2, agents. Tier 2 agents combine network data that they gather within their cell, based on the network traffic that they observe, as well as the reports they receive from the Tier 1 agents, to formulate their network status decision; they report this to Tier 3 agents. Tier 3 agents collect reports from Tier 2 agents and combine them with their own network observation data, to generate status reports to send to the next higher tier, and so on, tier by tier. The system hierarchy is shown in Fig. 4.

The FMAs of all tiers have the same structure, thus simplifying the system. A diagram of an FMA is illustrated in Fig. 5, which consists of the following

components: the probe, the event preprocessor, the statistical processor, the neural network classifier and the post processor. The functionalities of these components are described below:

- *Probe*: Collects data of host activity or associated network behavior, abstracts the data into a set of statistical variables to reflect the network status, and periodically generates reports to the *event preprocessor*.
- *Event Preprocessor*: Receives reports from both the *probe* and *FMA*s of the next lower tiers, and converts the information into the format required by the *statistical model*.
- *Statistical Processor*: Maintains a reference model of the typical network and host activities, compares the reports from the *event preprocessor* to the reference models, and (1) generates a decision for one or each of the monitored parameters individually, or (2) forms a stimulus vector that combines all or groups of parameters to feed into the neural network classifiers.
- *Neural Network Fault Classifier*: Processes the stimulus vector from the *statistical model* to decide whether the network status and traffic adhere to the service requirements for each class. Section 4 introduces the neural network classifiers used in the system in more detail.
- *Post Processor*: Generates reports for the agents at higher tiers. At the same time, it may display the results through a user interface.

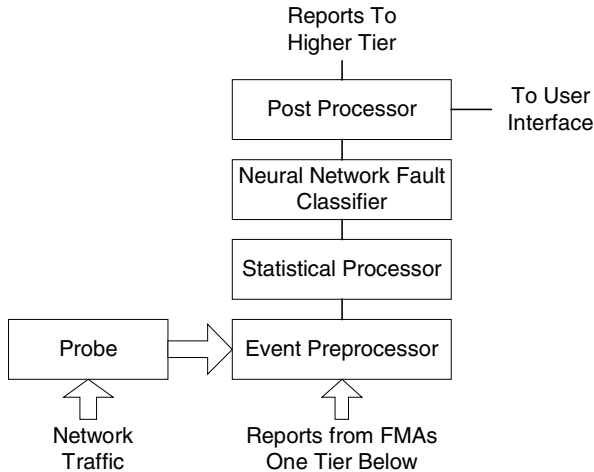


Fig. 5. Fault Management Agent

Because network traffic is not stationary and network soft faults may have different time durations, varying from a few seconds to several hours or longer, we need a statistical model, which is capable of efficiently monitoring network traffic with different simulation observation time windows. Based on the above, we designed and deployed a layer-window statistical model, shown in Fig. 6, with each layer-window corresponding to a monitoring time slice of increasing size.

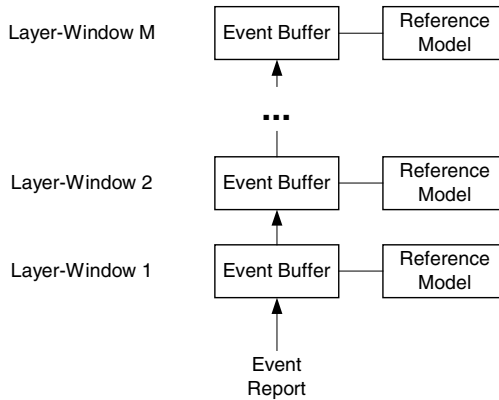


Fig. 6. Statistical Model

The newly arrived events will first be stored in the event buffer of layer 1. The stored events are compared with the reference model of that layer and the results are then fed into the neural network classifier to decide the network status during that time window. The event buffer will be emptied once it becomes full, and the stored events will be averaged and forwarded to the event buffer of layer 2. This process will be repeated recursively until the top level is reached where the events will simply be dropped after processing.

3 Statistical Algorithm

Statistical methods have been used in fault management systems to detect anomaly network activities; however, most of these systems simply measure the means and the variances of some variables and detect whether certain thresholds are exceeded. SRI's NIDES [7][8] developed a more sophisticated statistical algorithm by using a χ^2 -like test to measure the similarity between short-term and long-term profiles. Kolmogrov-Smirnov (K-S) statistics is used in [1] to model and detect anomaly traffic patterns.

Our current statistical model uses a similar algorithm as the above two algorithms but with major modifications. Therefore, we will first briefly introduce some basic information about the χ^2 -like test and the KS test.

3.1 χ^2 -Like Test

In NIDES, user profiles are represented by a number of probability density functions (PDFs). Let S be the sample space of a random variable and events E_1, E_2, \dots, E_k a mutually exclusive partition of S . Assume P_i is the expected probability of the occurrence of the event E_i , and let P_i^{\wedge} be the frequency of the occurrence of E_i during a

given time interval. Let N denote the total number of occurrences. The NIDES statistical algorithm used a χ^2 -like test to determine the similarity between the expected and actual distributions through the statistic:

$$Q = N \times \sum_{i=1}^k \frac{(P_i' - P_i)^2}{P_i} \tag{1}$$

When N is large and the events E_1, E_2, \dots, E_k are independent, Q approximately follows a χ^2 distribution with $(k - 1)$ degrees of freedom. However in a real-time application the above two assumptions generally cannot be guaranteed, thus, empirically, Q may not follow a χ^2 distribution. NIDES solved this problem by building an empirical probability distribution for Q , which is updated daily in a real-time operation.

3.2 Kolmogrov-Smirnov Statistic

In using the Kolmogrov-Smirnov test, the reference models and the observed system activities are represented by a number of cumulative density functions (CDF). The equation to compare a theoretical, completely specified target cumulative distribution function (CDF), $F_T(x)$, and a sample CDF, $F_S(x)$, is

$$D = \max_{-\infty < x < \infty} |F_T(x) - F_S(x)| \tag{2}$$

What makes the K-S statistic powerful is the fact that, in the case of the null hypothesis (data sets drawn from the same distribution), the distribution of D is independent of $F(x)$ and can be calculated with useful accuracy. This is very important property for network monitoring, when little may be known about the underlying distribution for either the typical or the anomalous traffic.

There are several variants on the K-S tests. Among them, the Anderson-Darling statistic calculates a weighted K-S measure:

$$D^* = \max_{-\infty < x < \infty} \frac{|F_T(x) - F_S(x)|}{\sqrt{F_T(x) \cdot [1 - F_T(x)]}} \tag{3}$$

As another example, Kuiper's statistic is the sum of the maximum distance of $F_T(x)$ above and below $F_S(x)$.

$$V = D_+ + D_- \tag{4}$$

$$= \max_{-\infty < x < \infty} [F_T(x) - F_S(x)] + \max_{-\infty < x < \infty} [F_S(x) - F_T(x)]$$

3.3 The Algorithms Used in GAFT

Similarly to the approach taken by NIDES, in our system, the network activities are sampled and abstracted into PDFs. However, since we are using a neural network fault classifier, that is capable of learning from examples by training, to discern network degradations, we are not so concerned with the actual distribution of Q . Nevertheless, because network traffic is not stationary and network faults may have different time durations, the algorithm must be reliable in detecting the differences between the observed and the reference PDFs, so as to adapt effectively to the observed traffic patterns.

The similarity-measuring algorithm that we are using is shown below:

$$Q = f(N) \cdot [\sum_{i=1}^k |p_i' - p_i| + \max_{i=1}^k (|p_i' - p_i|)] \quad (5)$$

where $f(N)$ is a function that takes into account the total number of occurrences during a time window. This similarity algorithm may be interpreted as incorporating the KS statistic in equation (2) along with the area of the difference between the target and sample PDF curves.

Besides similarity measurements, we also designed an algorithm for the real-time updating of the reference model. Let \bar{p}_{old} be the reference model before updating, \bar{p}_{new} be the reference model after updating, and \bar{p}_{obs} be the observed user activity within a time window. The formula to update the reference model is

$$\bar{p}_{new} = s \times \alpha \times \bar{p}_{obs} + (1 - s \times \alpha) \times \bar{p}_{old} \quad (6)$$

where

- α is the predefined learning rate
- s is the dynamic adaptation step based on the output of the neural network classifier.

Assume that the output of the neural network classifier is a continuous variable u between -1 and 1 , where -1 means fault with absolute certainty and 1 means normalcy again with complete confidence. In between, the values of u indicate proportionate levels of certainty. The function for calculating s is

$$s = \begin{cases} u, & \text{if } u \geq 0 \\ 0, & \text{otherwise} \end{cases}$$

Through the above equations, we ensured that the reference model would be updated actively for typical traffic while kept unchanged when failures occurred. The fault events will be diverted and stored for future neural network learning.

4 Neural Network Fault Classifiers

Neural networks are widely considered as an effective approach to classify patterns. In [10], BP neural networks were used to detect anomalous user activities. Jiang et al [6]

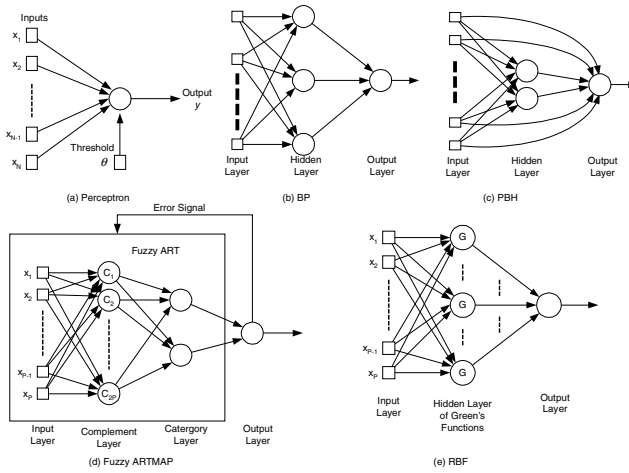


Fig. 7. Neural Network Architectures

built a network fault management system using artificial intelligence technologies. In order to comprehensively investigate the performances of neural network classifiers, we examined five different types of neural networks: Perceptron, BP, PBH, Fuzzy ART MAP and RBF.

The perceptron [11], Fig. 7(a), is the simplest form of a neural network used for the classification of *linearly separable* patterns. Although our data sets will not, in general, be linearly separable, we are using the perceptron as a baseline to measure the performances of other neural networks.

The Backpropagation network [11], or BP, Fig. 7(b), is a multi-layer feedforward network. BPs have strong generalization capabilities and have been applied successfully to solve a variety of difficult and diverse problems. Here we tested BP networks with the number of hidden neurons ranging from 2 to 8.

The Perceptron-backpropagation hybrid network [9], or PBH, Fig. 7(c), is a superposition of a perceptron and a small backpropagation network. PBH networks are capable of exploring both linear and nonlinear correlations between the input stimulus vectors and the output values. We tested PBH networks where the number of hidden neurons ranged from 1 to 8.

The Fuzzy ARTMAP [12] in its most general form is a system of two Fuzzy ART networks ART_a and ART_b , whose F2 layers are connected by a subsystem referred to as a “match tracking system”. We are using a simplified version of Fuzzy ARTMAP [13], Fig. 7(d), which is implemented for classification problems. We tested ARTMAP networks with the number of category neurons ranging from 2 to 8.

The Radial-basis function network [11], or RBF, Fig. 7(e), involves three entirely different layers. We tested RBF networks with hidden neurons ranging from 2 to 8.

5 The Simulation Environment

To validate our statistical models and to test the system architecture, we built a virtual network using simulation tools. The experimental testbed that we built using OPNET, a powerful network simulation facility, is shown in Fig. 8. The testbed is a 10-BaseX LAN that consists of 11 workstations and 1 server. This might correspond to the lower tier (Tier 1) only of the system hierarchy diagram shown in Fig. 4. It adequately illustrates the performance of a typical GAFT agent at Tier 1, as well as provides insight for the GAFT agent to be basically the same at all tiers.

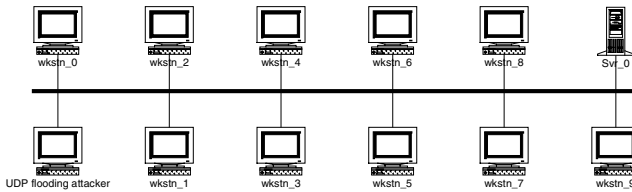


Fig. 8. Simulation Testbed

In the real network environment, a network/service degradation or failure may be caused by equipment malfunctions, transmission media breakdowns, malicious external or internal attacks, etc. In this work, we simulated network condition degradation by injecting high volumes of class-alien UDP traffic into the simulation testbed. This causes increased delays and may lead to decreased throughput and increased packet loss rates. The simulation generates class-normal traffic using standard driving distributions, such as Poisson, uniform, constant, etc. The class-alien traffic is also simulated using the standard distributions but different parameters. Other methods of simulating network degradation can be employed as well. Because in Ethernet all classes of traffic are transmitted over the same media, an anomaly behavior of one class will interfere with other network services. In other words, the behavior of different classes of traffic becomes correlated, as is often the case in many real-life networks.

To extensively test the performances of the system, we ran several independent scenarios with different typical and anomaly traffic loads. For each simulation scenario, we collected 10,000 records of network traffic. We divided these data into two separate sets, one set of 6000 records for training and the other of 4000 records for testing. The training records included typical as well as traffic tainted by the injected alien packets and thus labeled. In each scenario, the system was trained for 100 epochs.

6 Results on Neural Network Classifiers

We evaluated the performance of each of the neural networks based on the *misclassification rates* of the outputs. The misclassification rate is defined as the percentage of the network traffic that is misclassified by neural network fault classifiers during one epoch, which includes both *false positive* and *false negative* misclassifications.

TABLE 1 lists the traffic loads of the four simulation scenarios that we ran for neural network testing.

Table 1. Traffic Loads of the Four Simulation Scenarios

	Typical Class Traffic	Class-Alien Traffic
Scenario 1	600kbps	50kbps
Scenario 2	600kbps	100kbps
Scenario 3	2Mbps	50kbps
Scenario 4	2Mbps	100kbps

In the rest of this section, we will present and analyze the simulation results of the neural network classifiers one by one.

6.1 Perceptron

The mean squared root errors and the misclassification rates of the perceptrons within the four simulation scenarios are tabulated in TABLE 2.

Table 2. The simulation results for perceptrons

	Scenario 1	Scenario 2	Scenario 3	Scenario 4
Misclassification rate	0.167	0.202	0.234	0.119

We can see that the perceptron performed poorly in all the four scenarios: Mean squared root errors are between 0.6 and 0.7; and misclassification rates are between 0.1 and 0.2. Both the MSR errors and the misclassification rates are unacceptably high for a fault detection system.

6.2 Fuzzy ARTMAP and RBF

The results of Fuzzy ARTMAP and RBF nets are shown in Fig. 9. The x-axis values of the figures represent the number of category neurons in Fuzzy ARTMAP and the hidden neurons in RBF. The y-axis values represent the lowest Misclassification Rates that these neural classifiers achieved within the 100 epochs.

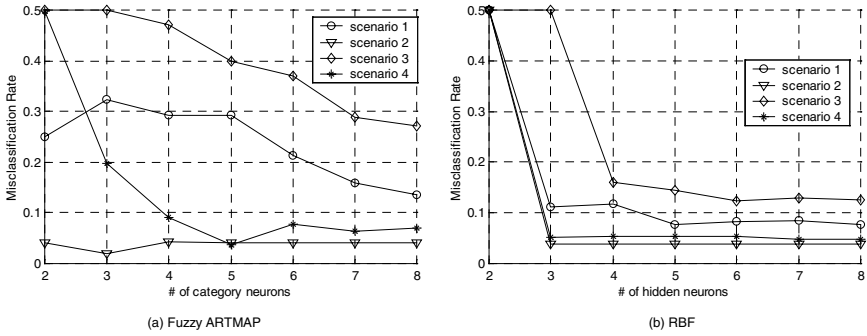


Fig. 9. Results of Fuzzy ARTMAP and RBF

From the above figures, we can see that, as the number of hidden neurons increases, the performance of both, ARTMAP and RBF networks, improves. In most of the cases, both of them outperformed the perceptron.

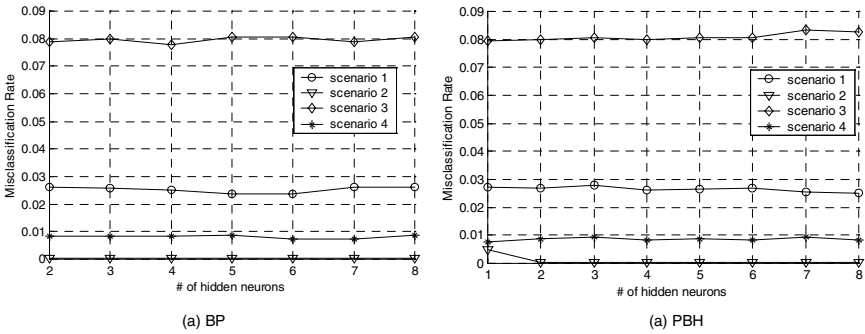


Fig. 10. Results of BP and PBH

6.3 BP and PBH

The results of BP nets are illustrated in Fig. 10. The figures indicate that BP and PBH networks have similar performance, and that both neural networks consistently perform better than the other three types of neural networks. The curves in these figures are flat: the MSR errors and misclassification rates do not decrease as the number of hidden neurons increases. We believe the reason is that, the stimulus vectors provided to the neural network classifier are very powerful and telling, especially in the sense that multiple performance parameter K-S similarity measures are combined together in one integrated pattern. Thus, there is ample room for the classifier to undertake more challenging network conditions.

7 Stress Testing the GAFT System

In this section, we will stress test the sensitivity, and thus effectiveness, of our system. We simulated various network fault scenarios using the test bed we introduced in section 5. The traffic loads of the simulations we ran for stress testing are specified in TABLE 3. From section 6, we can see that BP and PBH performed the best among the five neural networks tested, and that, for these two neural networks, increment in the number of hidden neurons did little to help with their performances. Therefore, the neural network that we chose for stress testing was a BP network with 2 hidden neurons.

Table 3. The traffic loads for stress testing

Typical-Class Traffic	Class-Alien Traffic
600 Kbps	10Kbps, 20Kbps, 30Kbps, 40Kbps, 50Kbps, 70Kbps, 100Kbps, 200Kbps
2Mbps	10Kbps, 50Kbps, 100Kbps, 150Kbps, 200Kbps

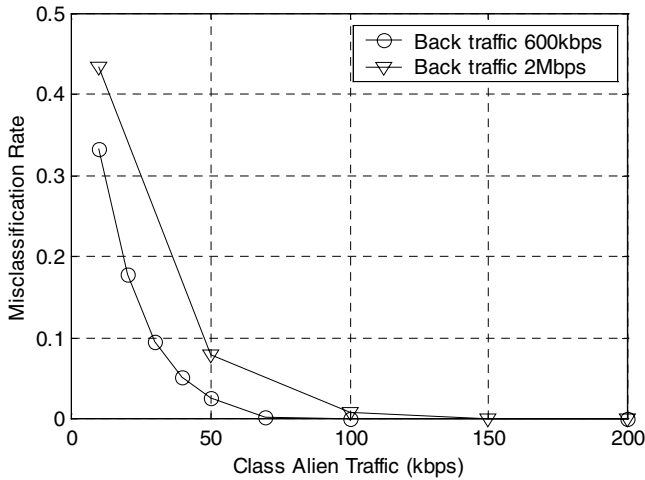


Fig. 11. The results of Stress Testing

Through the simulations, we expect to see the changes in the *Misclassification Rates* of the system as functions of the typical and anomaly traffic volumes.

The results of stress testing are shown in Fig. 11. From the figure, we can see that both the MSR errors and the misclassification rates decrease as the class-alien traffic level increases. This is because traffic patterns of higher-volume foreign traffic yield greater differences from the reference model than those created by lower-volume. We can also notice that, for a certain class-alien traffic level, the performance for the 600Kbps typical traffic background is consistently better than that of 2Mbps, as one may reasonably expect.

Fig. 12 shows the ROC (Receiver Operating Characteristic) curves of some selected simulation scenarios. The x-axis of the figure is the false alarm rate, which is the rate

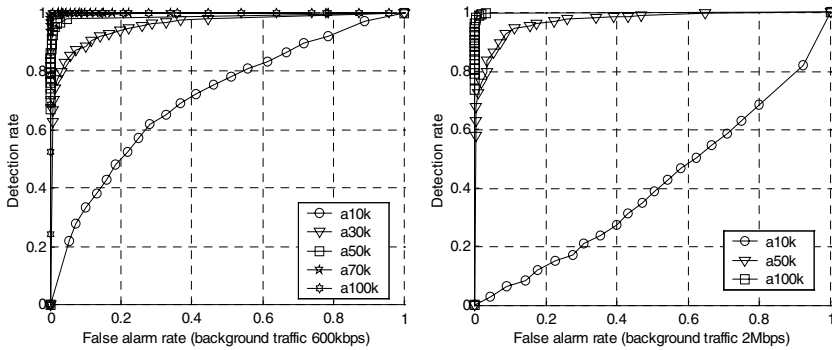


Fig. 12. ROC Curves

of the typical traffic events being classified as faults or anomalies; the y-axis of the figure is the detection rate, which is calculated as the ratio between the number of correctly detected faults/anomalies to their total number. For each curve, the point at the upper left corner represents the optimal detection with high detection rate and low false alarm rate. From the figure, we can observe the same tendency as in Fig. 11: The detection performance improves as the alien traffic intensity increases. In fact, we can see that, when the alien traffic level is 70Kbps for 600Kbps typical class traffic and 100Kbps for 2Mbps typical traffic, the system performance approaches the optimum.

8 Conclusions

In this paper, we introduced the framework of a hierarchical, multi-tier, multi-window fault detection system, using statistical preprocessing and neural network classification. We described our experiments using five different neural network classifiers. The results showed that BP and PBH nets outperform Perceptron, Fuzzy ARTMAP and RBF networks. Thus, the classification capabilities of BP and PBH classifiers are more desirable for statistical anomaly fault detection systems. We also presented some of the results of stress testing. The results indicate that the system is very efficient. It can reliably detect soft faults with traffic anomaly intensity as low as five to ten percent of the typical service class traffic intensity.

Acknowledgements. Our research was partially supported by a Phase I SBIR contract with US Army.

We would also like to thank OPNET Technologies, IncTM, for providing support for the OPNET simulation software.

References

- [1] Joao B.D. Cabrera, B. Bavichandran, R.K. Mehra, "Statistical Traffic Modeling for Network Intrusion Detection", *Proceedings of 8th International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication systems*, Aug. 2000, pp. 466-473.
- [2] N. Dawes, J. Altoft, and B. Pagurek, "Network Diagnosis by Reasoning in Uncertain Nested Evidence Spaces", *IEEE Trans. Commun.*, vol. 43, 1995, pp. 466.
- [3] L.L. Ho, D.J. Cavuto, S. Papavassiliou, A.G. Zawadzki, "Adaptive and Automated Detection of Service Anomalies in Transaction-oriented WANs: Network Analysis, Algorithms, Implementation, and Deployment", *IEEE Journal on Selected Areas in Commun.*, vol. 18, May 2000, pp. 744-757.
- [4] J.R. Dorrnsoro, F. Ginel, and C. Sanchez, "Neural fraud Detection in Credit Card Operations", *IEEE Trans. Neural Networks*, vol. 8, 1997, pp. 827.
- [5] R. Maxion and F.E. Feather, "A Case Study of Ethernet in a Distributed Computing Environment", *IEEE Trans. Reliability*, vol. 39, Oct. 1990, pp. 433-443.
- [6] S. Jiang, D. Siboni, A.A. Rhissa, G. Beuchot, "An Intelligent and Integrated System of Network Fault Management: Artificial Intelligence Technologies and Hybrid Architectures", *Proceedings of IEEE Singapore International Conference on Networks*, July 1995, pp. 265-268.
- [7] A. Valdes, D. Anderson, "Statistical Methods for Computer Usage Anomaly Detection Using NIDES", *Technical report*, SRI International, January 1995.
- [8] H. S. Javitz, A. Valdes, "The NIDES Statistical Component: Description and Justification", *Technical report*, SRI International, March 1993.
- [9] R. M. Dillon, C. N. Manikopoulos, "Neural Net Nonlinear Prediction for Speech Data", *IEEE Electronics Letters*, Vol. 27, Issue 10, May 1991, pp. 824-826.
- [10] A.K. Ghosh, J. Wanken, F. Charron, "Detecting Anomalous and Unknown Intrusions Against Programs", *Proceedings of IEEE 14th Annual Computer Security Applications Conference*, 1998, pp. 259-267.
- [11] Simon Haykin, *Neural Network A Comprehensive Foundation*, Macmillan College Publishing Company, 1994.
- [12] G.A. Carpenter, et al, "Fuzzy ARTMAP: An adaptive resonance architecture for incremental learning of analog maps", *International Joint Conference on Neural Networks*, June 1992.
- [13] NeuraWare Inc., *Neural Computing A Technology Handbook for NeuralWorks Professional II/PLUS and Neural Works Explorer*, NeuralWare Inc., 1998.

Providing Scalable Many-to-One Feedback in Multicast Reachability Monitoring Systems

Kamil Saraç and Kevin C. Almeroth

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

Abstract. With the deployment of native multicast in commercial networks, multicast is getting closer to becoming a ubiquitous service in the Internet. The success of this deployment largely depends on the availability of good management tools and systems. In this paper, we focus on reachability monitoring as an important multicast management task. First, we provide a general architecture for multicast reachability monitoring systems and focus on three critical functions: agent configuration, monitoring and feedback collection. For each component, we provide a number of alternative approaches to implement the required functionality and discuss their advantages and disadvantages. Then, we focus on the feedback collection component. To a large extent, it determines the complexity and the overhead of a monitoring system. We compare a number of alternative approaches for feedback collection using simulations and make suggestions on when to use each. We expect our work to provide insight into the issues and considerations in designing and developing multicast reachability monitoring systems.

1 Introduction

With the deployment of native multicast in commercial networks, multicast is getting closer to becoming a ubiquitous service in the Internet. However, before multicast can be used as a revenue-generating service, its robust and flawless operation needs to be established across both the intra- and inter-domains. This requires availability of good management tools to help network administrators configure and maintain multicast functionality within and between multicast enabled domains.

While unicast management is well established and provides good support for robust network operation, there exists a need for good multicast management tools and systems. The difference between *unicast* management and *multicast* management derives mainly from the simple fact that multicast traffic can be destined to multiple receivers. Multicast data sent to multiple receivers travels along a logical tree that is dynamically built on top of the underlying network infrastructure. The construction and maintenance of this tree topology is automatically done by network devices (e.g. routers) depending on the joins and

leaves of multicast session receivers. It is this dynamic behavior that makes multicast more difficult to manage and monitor compared to unicast services in the network.

The management function that we focus on in this paper is reachability monitoring. Reachability monitoring is one of the most important yet one of the most difficult multicast management tasks. Reachability monitoring enables a network operator to test availability and quality of multicast service in a domain. In this paper, we provide a general architecture for multicast reachability monitoring systems. First, we identify the functional components of this architecture and then the steps to configuring a reachability test. Next, we provide a number of alternative approaches to implementing these steps. The variations in the implementation of these steps are mainly due to the type and scope of different monitoring tasks. In this respect, we provide some guidelines about when to use each alternative implementation. We believe that this analysis is useful as guidance for reachability monitoring system designers and developers.

The rest of this paper is organized as follows. Section 2 motivates the multicast reachability monitoring problem. Section 3 provides a general architecture for reachability monitoring systems. Section 4 gives a detailed discussion on alternative approaches for implementing the functional steps of monitoring systems. In Section 5, we provide simulation-based comparisons for different feedback collection approaches and present guidelines for their applications in Section 6. The paper concludes in Section 7.

2 Motivation

From network management point-of-view, successfully deploying multicast requires the ability to instill confidence that the network is working in a correct and predictable manner. This requires mechanisms to monitor and verify successful multicast data transmission within and between multicast-enabled domains. For a globally-scoped multicast application, a number of potential receivers may be located in other domains and the delivery of data to these receivers may be affected by reachability. Network operators must have the ability to ensure multicast reachability to all potential receivers. There are two properties that make this difficult: (1) the dynamic nature of multicast trees, and (2) anonymity of group receivers.

The goal of reachability monitoring is to verify and help maintain the robust operation of multicast. Reachability monitoring within a domain is most effective when both routers and end hosts are used as monitoring agents. Monitoring agents can be configured to source test data to a multicast group and/or join and collect data reception statistics about the group. This information can then be used by the Network Operations Center (NOC) personnel for network management purposes. In general, management personnel not only have full access to all devices in the network but they are also expected to solve problems by changing parameters or configurations. Even with this capability, NOC personnel cannot control or easily monitor networks outside their own domain.

Therefore, intra-domain reachability monitoring may not be enough to maintain a robust multicast service. Availability of data from sources external to the local network depends on proper multicast operation in and between other domains. This requires the ability to perform inter-domain reachability monitoring tests. These tests require access to agents located in remote domains. In general, due to security and privacy issues, network operators are not willing to share their network resources with others. Therefore, inter-domain monitoring test scenarios are generally limited to using end-host-based monitoring agents. In any case, the information that is made available can be very useful for confirming that (1) problems do exist, and (2) problems are located in a remote domain.

3 An Architecture for Multicast Reachability Monitoring

In this section we present a general architecture for multicast reachability monitoring systems. In this architecture, we focus on the basic system components and their functionality. In general, a multicast reachability monitoring system has a *manager* component and one or more monitoring *agent* components. The manager is a software component that can either be a stand alone program or can be part of a complex management system that interacts with the other modules in performing network management. Monitoring agent functionality can be implemented both in the routers and in end-hosts. Figure 1 shows an example of this architecture. In this figure, the manager uses both router and end-host agents in the intra-domain and uses end-host agents in inter-domain. In either case, the overall monitoring task can be divided into three steps: (1) the manager configures agents for a monitoring task, (2) agents perform monitoring and collect statistics, and (3) the manager collects statistics from the agents. In the next section, we discuss these steps in more detail.

4 Reachability Monitoring Steps

In this section, we discuss the steps involved in performing reachability monitoring. For each step, we briefly discuss the main functionality. Then, we provide alternative approaches to implement this required functionality. When appropriate, we compare these alternatives based on various criteria such as scalability, functional overhead and security.

4.1 Step I: Agent Configuration

In this step, the manager configures the agents for a particular monitoring task by sending necessary configuration parameters to the agents. These include the session address, duration of monitoring, packet encoding format, types of feedback reports to collect, etc. The type of communication used between the manager and the agents is an important consideration. Depending on the scale and complexity of the system, this communication may be as simple as a UDP-based

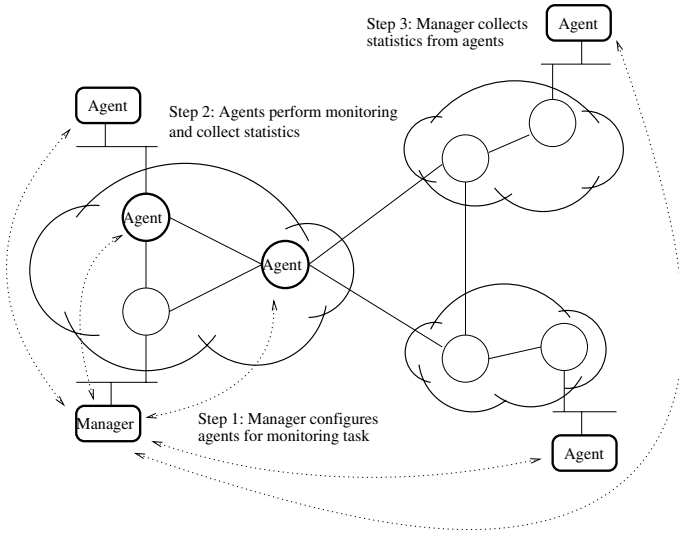


Fig. 1. A general case for intra- and inter-domain reachability monitoring.

message exchange (with or without reliability) or it may be as complex as some type of authenticated and encrypted message exchange. Important issues in this step are (1) to protect monitoring agents from un-authorized users and (2) to protect monitoring agents from being overloaded by monitoring requests.

Conducting reachability monitoring introduces both network overhead and processing overhead. Preventing the overload of router-based monitoring agents is important as it may interfere with the overall performance of the router in forwarding traffic. Therefore, the agent configuration step should use an appropriate mechanism to control access to monitoring agents. In this paper, we discuss two alternative approaches for agent configuration communication: the Simple Network Management Protocol (SNMP) and an IP Security (IPsec) based authentication mechanism:

- *SNMP-based approach.* SNMP provides a standard approach to communicate configuration or control information between a management station and monitoring agents. Traditionally, SNMP is used to access network devices to exchange management information[1]. It is a common piece in many network management systems. Most NOC personnel are comfortable with SNMP and most are familiar with SNMP-based systems. However, SNMP is not without its deficiencies. Until recently, SNMP has been primarily used for monitoring and performance management within domains. The current protocol version, SNMPv2, provides poor support for inter-domain network management tasks and suffers from lack of scalability and security. To a large extent, these problems are being addressed by two threads of effort. First, there is effort in the Internet Engineering Task Force (IETF) to develop a SNMPv3 protocol with better support for security and device configuration[2]. Second, there

are research efforts looking to improve SNMP's distributed management capabilities—primarily by adding scalability[3]. We believe that with these modifications, SNMP can provide solid vehicle for control/configuration communication for reachability monitoring.

- *IPsec-based approach.* This approach refers to cryptographically strong but non-SNMP-based techniques. As an example, the Multicast Reachability Monitor (MRM) protocol uses *access control lists* and the IPsec Authentication Header (AH) mechanism to control accesses to monitoring agents[4]. Agent platforms keep a list of authorized sites for agent use and agent configuration messages of these sites use AH mechanism to protect against spoofing attacks. Another example that uses a solution other than SNMP is the National Internet Measurement Infrastructure (NIMI) project. In NIMI, monitoring platforms are protected by a password mechanism and (AH-based) authenticated message exchanges[5].

The second issue for agents is to protect them from being overloaded by excessive simultaneous monitoring requests. This is necessary because each monitoring task introduces network overhead (in terms of sourcing or receiving multicast data streams) and processing overhead (in terms of producing statistical information). The management system should be aware of the agents' load and avoid them. However, because end-host-based agents can be used by multiple managers simultaneously, the agents also need to protect themselves. Monitoring agents can regulate their overhead by limiting their network bandwidth usage in monitoring tasks and by limiting the maximum number of test scenarios that they participate in simultaneously. When an agent receives a new test request that it can not satisfy, it should deny the request and inform the requester about its decision and its reason. This protects agents from being overloaded by requests of authorized users.

4.2 Step II: Reachability Monitoring

In this step, agents perform the actual monitoring and collect data about the results. The specific functions performed by monitoring agents depend on various factors such as the *type* of the monitoring task (active vs. passive), agent platform capability (router vs. end-hosts) and granularity of collected data. In general, monitoring agent functionality can be divided into two parts: *test sender* and *test receiver*. A *test sender* is configured to source multicast data to a test multicast group address. A *test receiver* is configured to join and monitor data reception quality (either for a test session or for a real session).

Monitoring tasks can be divided into two types: *active* monitoring and *passive* monitoring. In active monitoring, network operators create a test multicast session among a number of agents. Some agents are configured to function as test senders and some agents are configured to function as test receivers. An advantage of active monitoring is that test receivers know beforehand all the important parameters about characteristics of the data stream (e.g. starting time, duration, delay interval between data packets). This information usually

simplifies the computation of statistics about the session. In addition, network operators can create and run various type of monitoring tests, e.g. packet bursts, constant bit rate, or single packet transmissions. These tests are useful to verify reachability after an initial deployment or periodically to confirm proper operation. The disadvantage of these tests is that they create additional traffic in the network.

In passive monitoring, network operators configure monitoring agents to join and monitor an actual, ongoing multicast session. One advantage of passive monitoring is that it does not introduce additional traffic in the network. However, the disadvantage is that passive monitoring requires the agents to be able to interpret application layer data encoding schemes. This is necessary because the agent needs to track losses, duplicates and out-of-order packets. In addition, the monitoring depends on an active source which may not always be available or transmitting a stream with the desired characteristics.

Agent functionality can have different complexity levels and capabilities depending on the platform. Router-based agents are likely to only be able to provide a minimum support for reachability monitoring tasks. This is because routers are mainly used as production network components and any type of secondary tasks should add only minimum overhead. This requirement puts a limitation on the types of statistics that can be collected by router-based agents. As an example, MRM uses the Realtime Transport Protocol (RTP) to provide a minimum set of feedback information: packet loss, jitter and delay information[4,6].

Compared to network devices, end-host-based agents can support the collection of more detailed statistical information. In general, end-host-based reachability monitoring systems are more extensible and can provide more flexibility for various types of monitoring tasks. For example, there are research studies that use end-host-based multicast reachability monitoring tests to collect per packet loss information and use this information to estimate loss characteristics of individual network links[7]. In a related joint study, we proposed a number of extensions to the RTP reporting mechanism[8] that could provide a standard model to collect more detailed information about multicast data reception quality. Even though end-host-based agents are better than router-based agents in almost every way, routers provide a view from inside the network that is hard to obtain otherwise.

4.3 Step III: Feedback Report Collection

In this step, agents send their feedback reports back to the manager site. The reporting mechanism can be divided into two types: off-line reporting and real-time reporting. In off-line reporting, agents store the statistics that they generate during the monitoring step and then send them to the manager at a later time. Off-line reporting requires local storage and therefore it is more suitable for end-host-based agents. Real-time reporting can be divided into two parts: polling-based reporting and alarm reporting. In polling-based reporting, agents create and send a feedback report based on an explicit request by the manager.

In alarm reporting, agents create and send feedback reports based on detecting a threshold violation event. The alarm reporting mechanism has potential scalability problems. In a monitoring task where a large number of agents are configured to send alarm reports back to the manager site, a threshold violation close to the session source can cause a large number of agents to create and send feedback reports to the manager. If not controlled, these feedback reports can create scalability problems both in the network and at the manager site.

Depending on the type of test scenario, a number of different approaches can be used for alarm report collection. In this paper, we discuss four different approaches: (1) plain reporting, (2) report suppression, (3) delayed reporting and (4) probabilistic reporting. Each of these alternatives have different characteristics in terms of processing overhead, scalability and information granularity. We describe each of these alternatives below:

1. *Plain reporting.* This is the most straightforward reporting technique. We use this technique as the baseline in our analysis. In plain reporting, agents send their feedback reports directly to the manager site on detecting a threshold violation. This approach has the most potential for causing scalability problems. Depending on the size of the monitoring scenario, feedback reports may cause report implosion at the manager site.
2. *Report suppression.* In this approach monitoring agents cooperate with each other to minimize the number of feedback reports that reach the manager site. Similar approaches have been used in various reliable multicast routing protocols. Walz and Levine use a version of this approach in the Hierarchical Passive Multicast Monitor[9] (HPMM). In report suppression, agents do not send their feedback reports directly to the manager site. Instead they use a reporting hierarchy. In this hierarchy, there exists a number of nodes performing feedback suppression. Monitoring agents are configured to send their reports to appropriate suppression nodes. A suppression node can be either an agent participating in the monitoring task or an end-host specially elected for this purpose. Based on the type of suppression node, the suppression functionality may have different implementations. Suppression functionality implemented in routers should have minimum processing overhead and therefore should use a simple method for suppression. On the other hand, end-host-based suppressors can perform more complex functions. In general, the report suppression approach aims to inform the manager site about the existence and (approximate) location of a threshold violation event. However, it may not inform the manager site about which individual agents are affected by the event. This information can easily be obtained by the manager provided that it knows the monitoring session tree topology.
3. *Delayed reporting.* In this approach, on detecting a threshold violation, agents do not send their feedback reports immediately to the manager but delay them for a period of time. Delayed feedback techniques have been used in several other multicast protocols. For example, RTP uses a dynamic algorithm to compute an interval during which session receivers collect statistics about data transmission quality of the session. At the end of the interval,

group members send this information back to the group. In reachability monitoring, agents select a random delay period from a given maximum wait interval. The maximum wait interval is assigned by the manager as part of the agent configuration. This approach relaxes the real-time requirement of feedback reporting and provides more scalability in the report collection mechanism. That is, scalability is gained by extending the time interval for agent feedback reports so that they do not cause report implosion problems at the manager site. This technique is best used when there are infrequent threshold violations. On the other hand, in the case of frequent threshold violations, the scalability gain of this approach may deteriorate if several reports are generated with overlapping report delays.

4. *Probabilistic reporting.* In this approach, monitoring agents send their reports based on some probability. The reporting probability is assigned by the manager during agent configuration. In the probabilistic reporting approach, scalability is gained at the expense of potentially losing some feedback information (when the reporting probability is less than one). Therefore, this approach is only useful for collecting an approximation of the reachability status of a multicast network. A similar approach was used by Ammar to address scalability issues in distributed computing applications that require many-to-one response collection[10].

In the above list, we briefly discussed alternative approaches for alarm report collection. We pointed out the differences between these approaches in terms of their scalability characteristics and feedback information granularity. In addition to these characteristics, the processing overhead that these different approaches put on monitoring agents is also an important consideration. In terms of processing overhead, test receivers have similar behavior in all of these approaches. Mainly, they join and monitor a session; on detecting a threshold violation, they create feedback reports and send them to the appropriate destination(s). In the delayed approach, there is additional functionality that requires test receivers to keep timers to delay transmission of their reports. Also, in the suppression-based approach test receivers may have to perform suppression in addition to monitoring. In the next section, we compare the performance of these approaches using simulations.

5 Simulations

The feedback reporting mechanism used in a reachability monitoring system has an important effect on the overall performance of the system. In this section, we quantitatively compare the overhead of the feedback reporting approaches that we discussed in the previous section. We compare these approaches based on their messaging overhead at the manager site. In our comparisons, we use two metrics: (1) the number of feedback reports reaching the manager site during a 100 second simulation and (2) the peak number of feedback reports reaching the manager site in a small time interval. For the second metric, we collect data

for 0.1, 0.5 and 1.0 second intervals. In this paper, we present results for the 1.0 second interval and the results from the other intervals exhibit similar behavior.

In the simulations, we use topologies with different sizes and different threshold violation values. We generate random flat graphs for 100, 200, 300, 400 and 500 node topologies using the Georgia Tech Internet Topology Modeler (GT-ITM)[11]. In these topologies, we identify one of the nodes as the *test sender* and all the others as *test receivers*. Then, we assign loss probabilities for each link in the network. For loss assignment, we use the guidelines presented in Yajnik et. al.[12]. According to this approach, most of the internal links are assigned small loss probabilities and a few edge (or close to the edge) links are assigned larger loss probabilities. The last parameter in our simulations is the threshold value necessary to generate an alarm report. We use loss-based threshold values with $>0\%$ and 10% losses. Briefly considering the effects of our choices for the parameter ranges, system behavior outside of these ranges are relatively intuitive. In general, larger graph sizes mean more agents and therefore more feedback reports for a threshold violation. More losses in the backbone means more synchronized feedback reports for the threshold violation. On the other hand, higher threshold values mean less violations and therefore less feedback reports in the system.

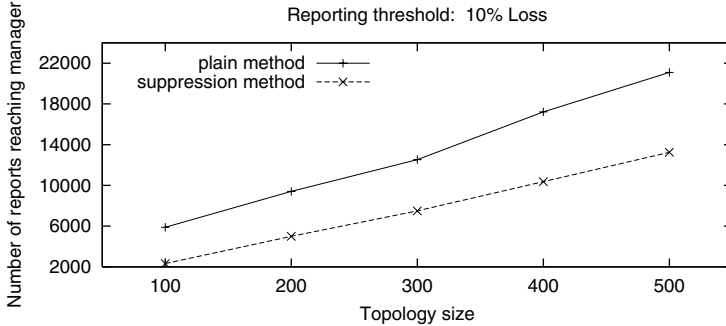


Fig. 2. Plain vs. Suppression: Reports reaching manager during 100 sec simulation.

The first set of simulations are run to compare the plain report collection approach and the suppression based report collection approach. These two approaches have a common property in that the manager site receives feedback information for *all* threshold violation events. In other words, these approaches do not lose any information during feedback reporting. As we mentioned above, the suppression functionality can have different implementations. We implement a simple suppression function at agent nodes. According to this implementation, when an agent detects a threshold violation event, it will create and send a feedback report to its upstream neighbor. If and when this report reaches the agent at the root of the tree, this agent sends it to the manager. In our simulations, the root agent is always the manager node and at the same time it is the *test sender* of the session. After sending a report, an agent *A* will receive feedback reports

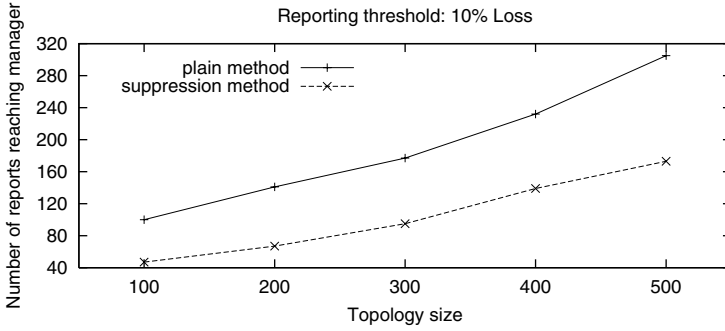


Fig. 3. Plain vs. Suppression: Maximum reports reaching manager in 1 sec interval.

from its downstream neighbors. This is because, any threshold violation that *A* detects should also be detected by the agents below it. On receiving a report from a downstream neighbor *B*, *A* will forward this report if the report is for another threshold violation event that has occurred between *A* and *B*. Otherwise, *A* will suppress this report. Figure 2 compares the plain and the suppression based approaches in terms of the overall number of reports that reach the manager site during a 100 second monitoring interval. The differences in the number of reports between the two lines show the saving of the suppression-based feedback collection technique. According to the figure, in this particular monitoring scenario, suppression-based feedback collection provides around 50% savings in the number of reports reaching the manager site. This saving is gained at the expense of having suppression functionality at every monitoring agent. Figure 3 compares these two approaches based on the maximum number of feedback reports reaching the manager site within a one second interval. These results present the peak number of reports that the manager must process in one second. Figure 3 shows that the maximum messaging overhead of plain feedback reporting is two times more than that of suppression-based approach. These results indicate that suppression-based feedback collection approach provides more scalability for feedback report collection without introducing any loss of feedback information.

In the next set of simulations, we compare the plain report collection approach with the delayed feedback collection approach. In this set of simulations, we use $>0\%$ loss threshold value and configure the test sender to introduce packet losses by not sending randomly selected data packets to the group address. This causes all the agents to detect threshold violation events and therefore to send feedback reports to the manager. In the simulations, we use 2, 4, 6, 8 and 10 second delay intervals. The actual delay values assigned to agents are randomly selected within these intervals. Figure 4 shows the results of these simulations. In this figure, we see that delayed feedback collection approach loses some feedback reports. According to this figure, as the delay period increases, it introduces more feedback losses into the system. Information loss can be avoided by keeping track of pending feedback reports and sending them on their delay expiration.

But this introduces extra overhead into the system as agents need to keep track of all pending reports. Figure 5 shows the maximum number of feedback reports reaching the manager site within one second interval. According to this figure, the messaging overhead at the manager site goes down as we increase the delay period. In the figure, the 2 sec delayed case performs close to the plain method. We believe that this behavior is due to the randomization in assigning delay values to monitoring agents. These results indicate that in case of infrequent losses, the delayed feedback reporting approach provides more feedback scalability in the sense that the manager does not have a report implosion problem. On the other hand, it may cause feedback information losses depending on the frequency of threshold violations.

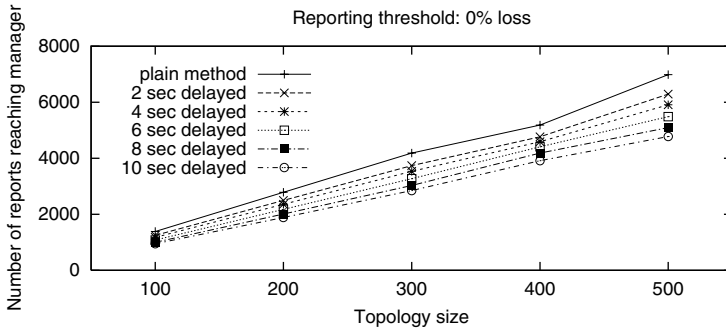


Fig. 4. Plain vs. Delayed: Reports reaching manager during 100 sec simulation.

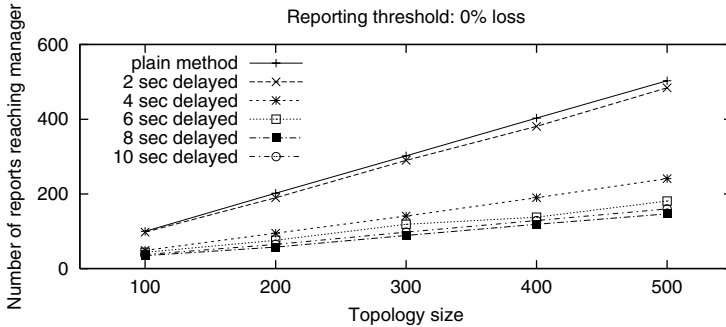


Fig. 5. Plain vs. Delayed: Maximum reports reaching manager in 1 sec interval.

In the last set of simulations, we compare the plain report collection approach with the probabilistic report collection approach. As we mentioned above, in the probabilistic approach, the manager configures agents to send feedback reports based on some probability. In our simulations, we use reporting probabilities as 80%, 60%, 40% and 20%. Figure 6 compares the two approaches based on the number of feedback reports arriving at the manager site during a 100 second sim-

ulation. As the reporting probability decreases, the number of messages arriving the manager decrease and therefore the amount of feedback information loss increases. Similarly, Figure 7 shows that as the reporting probability decreases, the maximum number of reports reaching the manager site within a one second interval decreases. These results show that the probabilistic approach loses a significant number of feedback reports. Therefore, this method should only be used to get some approximate information about the reachability status of the network.

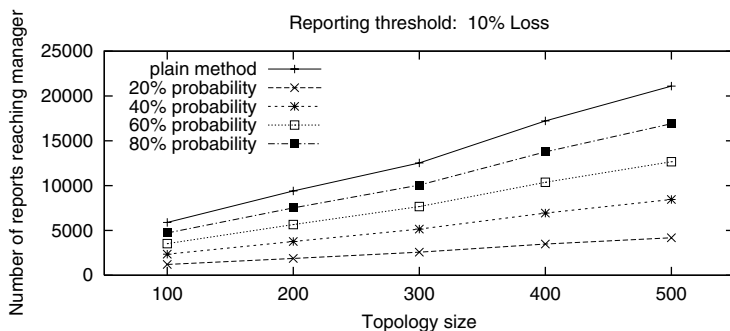


Fig. 6. Plain vs. Probabilistic: Reports reaching manager during 100 sec simulation.

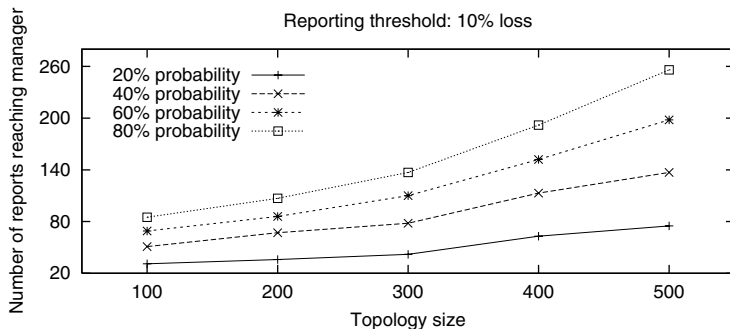


Fig. 7. Plain vs. Probabilistics: Maximum reports reaching manager in 1 sec interval.

6 Discussion

The feedback report collection mechanism is the most important step in a reachability monitoring system. The complexity and the functional overhead of a

monitoring system mostly depends on this component. Therefore, it is very important to choose the most appropriate feedback report collection technique. This decision usually depends on the needs of individual monitoring tasks. It may often be necessary for a reachability monitoring system to implement more than one feedback collection approach. It is then up to the user of this system to choose the correct alternative for a particular monitoring task. In the rest of this section, we provide some general suggestions about usage scenarios for each of the techniques that we discussed in this paper:

- *Plain reporting.* Plain feedback reporting should be used for test scenarios that have a small number of test receivers and the monitoring task requires collecting feedback information from all the agents.
- *Report suppression.* For monitoring tasks with a larger number of test receivers, report suppression approach provides more scalability. This approach should be used for monitoring scenarios where it is necessary to collect all the feedback information with better scalability than the plain reporting approach. As we mentioned previously, the report suppression method enables the manager to learn the existence and the approximate location of a threshold violation event but it does not provide information about which agents are affected by this event.
- *Delayed reporting.* Delayed reporting is most suitable for monitoring tasks where threshold violation events occur infrequently. In the delayed reporting approach, the manager receives feedback reports from all the test receivers. Furthermore, feedback reporting scalability can be adjusted by using different delay interval values. As the delay interval increases, the chance of report implosion at the manager site decreases. In this approach, test receivers may lose some feedback reports due to frequent threshold violations. For threshold violation events that occur in the backbone of the multicast tree (rather than the edge of the tree), feedback loss at a test receiver may not be very important. This is because another agent that has been affected by this threshold violation event can inform the manager about it. On the other hand, feedback losses occurring at the edge of the tree may not be recovered. Therefore, for monitoring tasks that are interested in learning threshold violation events in the core of the tree, this approach should work quite well.
- *Probabilistic reporting.* The probabilistic approach can be used for monitoring tasks that involve a large number of test receivers and the manager needs only approximate information about reachability. Furthermore, feedback implosion at the manager site can be controlled by using different reporting probabilities for the agents. In addition, agents can be assigned different reporting probabilities based on their strategic locations in the network.

As we have seen above, monitoring operations have potential scalability problems. These tasks may require configuring and managing a large number of monitoring agents. In addition, monitoring agents may be configured to send feedback reports based on some threshold violation and this often worsens the

scalability problem inherent in reachability monitoring. In a related recent study, Al-Shaer and Tang proposed adding multicast as a communication mechanism in more traditional protocols like SNMP[3]. This improvement provides scalability in the one-to-many communication between a management station and a set of monitoring agents during the agent configuration step. The issue then becomes improving scalability of the many-to-one feedback collection mechanism. In our discussion, we use reachability monitoring tasks as example and evaluate a number of alternative approaches for many to one report collection. We believe that our results applies to other monitoring tasks having many to one feedback collection components.

7 Conclusions

In this paper, we have presented a general architecture for multicast reachability monitoring systems and provided a detailed discussion about the functional components and alternative implementations for these components. We have described different approaches to implementing the functionality in each step. The differences between these implementations are mainly based on the type and scope of different monitoring tasks. For the agent configuration and actual monitoring steps, the two important considerations are the active vs. passive nature of the monitoring task and the computational resources available at the agent platforms. On the other hand, the feedback report collection step is the most important step of a monitoring system as it affects the overall performance of the monitoring system. In the paper, we presented four different approaches with different scalability and information completeness characteristics for the feedback report collection step. In addition, we performed simulations to compare the performance of these different feedback report collection techniques in terms of their scalability characteristics. Based on the simulation results, we provided suggestions on when to use which feedback reporting technique.

References

1. J. Case, K. McCloghrie, M. Rose, and S. Waldbusser, "Protocol operations for version 2 of the simple network management protocol (SNMPv2)." Internet Engineering Task Force (IETF), RFC 1905, January 1996.
2. J. Case, R. Mundy, D. Partain, and B. Stewart, "Introduction to version 3 of the internet-standard network management framework." Internet Engineering Task Force (IETF), RFC 2570, April 1999.
3. E. Al-Shaer and Y. Tang, "Toward integrating IP multicasting in internet network management protocols," *Computer Communications-Integrating Multicast into the Internet*, 2000.
4. K. Almeroth, L. Wei, and D. Farinacci, "Multicast reachability monitor (MRM)." Internet Engineering Task Force (IETF), draft-ietf-mboned-mrm-*.txt, October 1999.
5. V. Paxson, J. Mahdavi, A. Adams, and M. Mathis, "An architecture for large-scale internet measurement," *IEEE Communications*, August 1998.

6. H. Schulzrinne, S. Casner, R. Frederick, and J. V., "RTP: A transport protocol for real-time applications." Internet Engineering Task Force (IETF), RFC 1889, January 1996.
7. A. Adams, R. Bu, R. Caceres, N. Duffield, T. Friedman, J. Horowitz, F. Lo Presti, S. Moon, V. Paxson, and D. Towsley, "The use of end-to-end multicast measurements for characterizing internal network behavior," *IEEE Communications*, May 2000.
8. T. Friedman, R. Caceres, K. Almeroth, and K. Saraç, "RTCP reporting extensions." Internet Engineering Task Force (IETF), draft-ietf-avt-rctp-report-extns*.txt, March 2000.
9. J. Walz and B. Levine, "A hierarchical multicast monitoring scheme," in *International Workshop on Networked Group Communication (NGC)*, (Palo Alto, California, USA), November 2000.
10. M. H. Ammar, "Probabilistic multicast: Generalizing the multicast paradigm to improve scalability," in *IEEE Infocom*, (Toronto, CANADA), June 1994.
11. E. Zegura, C. K., and D. M., "Modeling internet topology," tech. rep., College of Computing, Georgia Institute of Technology, 1999.
12. M. Yajnik, J. Kurose, and D. Towsley, "Packet loss correlation in the Mbone multicast network," in *IEEE Global Internet Conference*, (London, ENGLAND), November 1996.

A Framework for Event Correlation in Communication Systems

Mouayad Albaghdadi¹, Bruce Briley², Martha Evens³, Rafid Sukkar⁴,
Mohammed Petiwala¹, and Mark Hamlen¹

¹ Motorola Inc., 1301 E. Algonquin Road, Room # 3346, Schaumburg, IL 60196, USA
{albaghdadi, Mohammed.H.Petiwala, CSPP03}@motorola.com

² Motorola Inc., 50 Northwest Point Blvd., Elk Grove, IL 60007, USA
bbriley@ieee.org

³ Illinois Institute of Technology, CS Department, 10 West 31st Street, Chicago, IL 60616,
USA
evens@iit.edu

⁴ Lucent Technologies, 2000 N. Naperville Road, Naperville, IL 60566, USA
sukkar@lucent.com

Abstract. This paper introduces a framework for event correlation in communication systems. We will show how the concept of a class in object-oriented methodology can be used to provide scalability to the framework. Events and system topology information are combined to generate the causal information needed for correlation. Geometric representation of codewords is used to overcome the noise factor. Temporal reasoning is explored to reduce noise and increase the number of event patterns that can be detected. The framework has been applied to a wireless communication system.

1 Introduction

Event correlation in communication systems is the process of analyzing the information (events) received to determine a higher-level picture of the system behavior. The stream of events arriving at the centralized location is the system operator's main view of the managed system. The operator uses this information in a variety of ways. He can check the status of periodic test results performed on certain nodes, look for small sets of events indicating the failure of a node, reconfigure a remote site (in a wireless system) and wait for a reaction (in terms of events) as to how successful the reconfiguration process was, etc. A more typical task of event correlation in large systems is to correlate events from multiple nodes throughout the system to determine a higher-level picture of the system. Obviously, such nodes are related in some fashion and information from all of them can be combined to give a complete picture.

To meet users' demands for access to information, today's systems are in constant change. In addition to the large size and increased complexity, a system's topology changes frequently. As the system grows in size and complexity, more events will be generated that need to be analyzed and acted upon by the system's operator.

Moreover, the change in topology changes the relationships between nodes in the system. Thus, correlating events across nodes becomes a time dependent task that the operator has to keep track of.

A final challenge to event correlation in communication systems is the factor of noise. Noise problems can be attributed to one or more of the following: (1) events lost due to links down, which prevent events from arriving at their centralized location; (2) limited resources at different levels leading to dropped events; (3) spurious events; (4) related events that are delayed and/or arriving out of sequence; and (5) unrelated events arriving at the same time a pattern is being formed.

The system operator's main task is to operate and maintain the system. Such a task becomes very difficult to perform effectively given the above conditions. The result is that important information about the health of the system is misinterpreted or totally ignored.

A primary use of event correlation in communication systems is the detection and identification of faults. These two functions are part of fault management, which consists of a set of functions that enables the detection, isolation, and correction of abnormal operation of the system [1].

Due to the importance of event correlation, researchers have applied a variety of techniques that span many fields in their investigation of this problem. Artificial Intelligence (AI) techniques have been used by [2, 3, 4, 5, 6, 7]. In [8, 9, 10], the authors utilize the causal relationships between events to generate a causality graph. A coding technique is then used to identify event patterns. In [11], the authors used the causality graph outlined in [9] to model the problem, and applied algebraic operations of sets to identify event patterns. In [12], graph theory and propositional relations between events were used. Finite-State Machines (FSM) and probabilistic FSMs have been used by [13, 14, 15, 16]. Probabilistic models have been used by [17, 18]. Petri Nets have been used by [19, 20].

In this paper, we will investigate the problem of event correlation. We will exploit the object-oriented concept of a class to provide scalability to our approach. Techniques using graph and coding theories are presented to correlate events in distributed, dynamic, and noisy communication systems. Concepts of temporal reasoning and their application to event correlation are discussed. These techniques have extended the work of [9] in many ways. This investigation is being applied to a wireless communication system.

This paper is organized as follows: Section 2 describes the system alphabet. Section 3 presents the correlation language of the system. Section 4 gives the details of the event correlation model. Section 5 deals with related research. Section 6 and 7 contain our conclusion and references respectively.

2 The System Alphabet

The structure of a typical event is shown in Fig. 1 below. Here, the event type and error code together indicate the reason for the event. The network element and object names identify the source of the event. The reason field provides additional information about the reason for the event in some cases. The time field indicates when the event was generated.

Event Type	Error Code	Element Name	Object Name	Reason	Time
------------	------------	--------------	-------------	--------	------

Fig. 1. Event structure

We can think of all the possible events generated by a system as its alphabet. Let's define the set of all events (the alphabet) generated by node i as Σ_i . The total set of all the Σ_i in a system with n nodes is $\Sigma_T = \Sigma_1 \cup \Sigma_2 \cup \dots \cup \Sigma_n$. It is clear that the system's alphabet will increase linearly as the number of nodes in the system increases.

Communication systems can be characterized as a collection of objects organized in a way to provide specific services to the end user. An object can be defined as a distinct entity in the system. It can be a hardware entity, a software process, a communication link, etc. A node consists of one or more objects. A system consists of a finite set of node classes (e.g., router, server, etc.). For each of these classes, there exist one or more instances of such a class.

Given the structure of the events and the environment in which they exist, it is obvious that nodes belong to certain classes and each class has the same alphabet. This can be seen in Fig. 2 where we divided an event into two parts: topology information and event information. The event information is the class's alphabet.

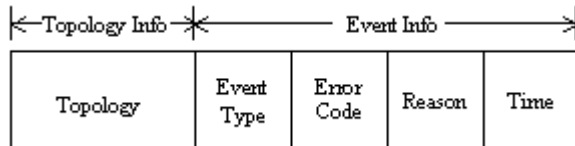


Fig. 2. Class event structure

The restructuring of events in this fashion shows clearly that the system's alphabet can be reduced to that of classes of nodes and objects within nodes. Therefore, a system's alphabet is dependent only upon the types of nodes and objects within nodes and not upon the instances of such entities.

3 The Language

We define a pattern (word) as a concatenation of events, e.g., $p_1 = e_1e_2e_3$ without regard to the arrival sequence. Therefore, $e_1e_2e_3, e_2e_1e_3, e_3e_2e_1$, etc., are the same pattern. Furthermore, the multiplicity of an event within a pattern is equivalent to a single occurrence. Let Σ^k denote the set of words with length k . Thus, $\Sigma^k = \{w \mid w \text{ is a word over } \Sigma \text{ and } (|w| = k)\}$. For example, if $\Sigma = \{e_1, e_2, e_3\}$,

$\Sigma^2 = \{e_1e_2, e_1e_3, e_2e_3\}$. Let $\Sigma^+ = \Sigma^1 \cup \Sigma^2 \cup \Sigma^3 \cup \dots = \bigcup_{k=1}^{\infty} \Sigma^k$. Note that

Σ^+ is the set of words that might be constructed from one or more events of Σ , and is the largest set of possible words we might observe from an entity being modeled.

Let $L(C)$ be the language of a class, e.g., an object. A language is a set of words that can be constructed from the alphabet. Therefore, $L(C)$ is a subset of Σ^+ .

4 The Model

An event pattern may consist of tens or hundreds of events. These events are the contribution of many objects throughout the system. The number of events generated by each object and the number of objects involved in a pattern are dependent on the type of the pattern and the size and complexity of the system.

Given this framework, the correlation process is divided into two phases: local correlation, which is an object level correlation, and global correlation, which combines the outcomes of the local correlation process in a way to determine a higher-level picture of the system behavior. This model is appropriate for the problem at hand. What we are saying is that objects generate events in two situations: (1) changes in the state of the object itself; and (2) the object’s reaction to external changes. In both cases, an object’s behavior is based on its own limited view of the system. The global correlation process collects and assembles these objects’ views to determine the system view.

The correlation language can be modeled as a bipartite graph, where one set of nodes is the alphabet and the other is the words (patterns). Fig. 3 below depicts an example of such a graph. The figure shows three patterns and eight events, and the causal relationships between them. The causality is depicted by the arrow “ \rightarrow ” pointing from the pattern to the event.

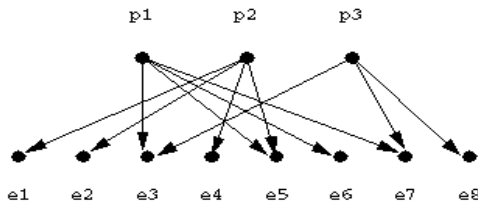


Fig. 3. Relationships between patterns and events

The figure above shows that patterns share common events. This is typical of real systems where subsets of the alphabet overlap but identify different patterns. Sometimes we are not that lucky and the same subset may indicate more than one pattern. In that case, the correlation result will have multiple outcomes.

There is more than one way to process the graph above. As an example, our initial but brief investigation was directed toward the use of neural networks using the

software package described in [21]. Our approach follows that of [9]. The language can be modeled as a set of patterns (a template) for an object class. This template can be seen as a matrix where the columns are the different patterns and the rows are the types of events making up the patterns. The above graph is shown below as a matrix, where there is a 1 in row i and column j iff there is an arrow from p_j to e_i .

	p1	p2	p3
e1	0	1	0
e2	0	1	0
e3	1	0	1
e4	0	1	0
e5	1	1	0
e6	1	0	0
e7	1	0	1
e8	0	0	1

Fig. 4. Correlation matrix

4.1 Pattern Identification in a Noisy Environment

Given the noisy environment in communication systems, there is a need for an algorithm to compensate for this inevitable problem. Coding and information theory is in large part concerned with noise when transmitting and storing information. Techniques for error-detecting and error-correcting codes are well established in this area and are suited to address this problem.

In [9], Hamming codes were to represent and identify (decode) event patterns. However, their work does not specify the type of the decoder being used. Our investigation utilizes the same technique and at the same time provides a suitable decoder for the problem at hand. First we start with some definitions of the principles of error-control coding.

A codeword is a fixed length vector of 0s and 1s. Fig. 4 depicts three codewords p_1 , p_2 , and p_3 . The Hamming distance weight $w(p_i)$ of codeword p_i is the number of 1s in p_i . The Hamming distance $d_{ij} = d(p_i, p_j)$ between codewords p_i and p_j is defined as the number of positions in which p_i and p_j differ. The greater this number, the greater the geometric distance between them. The distance between two codewords can be obtained by applying the modulo-2 addition \oplus (the logical operator Exclusive-OR in Boolean Algebra) to the two codewords p_i and p_j as follows: $d_{ij} = w(p_i \oplus p_j)$. For example, the Hamming distance between the two codewords p_1 and p_2 in Fig. 4 is 6.

The power of a code has been stated in [22] as follows:

$$d_m = \begin{cases} e + 1 & \text{error detection} \\ 2e + 1 & \text{error correction} \end{cases} \quad (1)$$

Here a code that can detect up to e errors per word and correct up to e errors per word must have a minimum Hamming distance d_m . It is obvious that a code having all distinct codewords must have a minimum Hamming distance of at least 1.

Given this equation for a minimum Hamming distance, we can construct a code with different levels of resilience to noise. This concept can be used to tailor a given code to different operating environments, i.e., different systems facing different noise levels. It also helps optimize the size of the correlation matrix, which [9] refers to as the “codebook”.

The following equation is our Hamming decoder:

$$\cos \theta = \frac{u \cdot v}{\|u\| \|v\|} \quad (2)$$

This decoder corresponds to a geometric representation of two vectors u and v in n -space. Assume that the two vectors have been positioned so their initial points coincide. The angle between them satisfies $0 \leq \theta \leq \pi/2$. The term $u \cdot v = u_1v_1 + u_2v_2 + \dots + u_nv_n$ is the dot product of the two vectors, while $\|x\| = \sqrt{x_1^2 + x_2^2 + \dots + x_n^2}$ is the norm of a vector x . When $\cos \theta$ is equal 1, i.e., θ is zero, we have a perfect match while for a value of 0 ($\theta = \pi/2$) the two vectors are complete opposites. To compensate for noise, a pattern match is declared once $\cos \theta$ exceeds some threshold, e.g., $\cos \theta \geq 0.9$.

4.2 Local Correlation

Our approach is to identify the correlation matrices for all object classes in the system under study. As events arrive at the Operations and Maintenance Center (OMC), or are read from the event log offline, we use the topology information part of the event to identify the source of the event. The topology information provides us with the node instance and the object name of the source of the event as well as the node’s hierarchy in the system. The event will be stored in the object’s queue in a FIFO fashion. If this is the first event generated by the object, a new queue is created and the event is placed first in the queue. The event is assigned a number based on the object’s alphabet and is labeled as an uncorrelated event. This means that the event has not been identified as part of a pattern yet. The queue length is a system dependent parameter. We have used a length of twenty based on our system’s empirical evidence. Using this technique, old events will be pushed out of the queue as more events arrive at the queue.

With the arrival of every new event in an object's queue, uncorrelated events are used to form a vector called a "running vector". A similarity measure is applied to the running vector and every vector (pattern) in the object's class template. The vector that is most similar to the running vector is declared as its match. All events that contributed to the pattern are labeled as "correlated" and can be removed from the queue. A local correlation result, i.e., a composite event, will be generated as a result of the match. If however, no match is found, no action is taken.

Defining bipartite graphs in local correlations start with the definition of event-pattern pairs. This knowledge is stored in text files, which are then read into the system during initialization.

In local correlation where an object class within a node class is the focus, each event-pattern pair consists of the three fields:

- LCEC: Local Composite Event Code which is the result of a local correlation;
- PEC: Primitive Event(s) Code is a set of primitive events represented numerically. This is a subset of the object's alphabet;
- LCED: Local Composite Event Description. It is a text description of the local correlation result. It is used by the human operator to describe the correlation result if needed.

To illustrate the above, we use Fig. 4 as an example. First, assign a unique LCEC value for each pattern as follows: $p1 = 1$, $p2 = 2$, and $p3 = 3$. Second, assign a unique PEC value for each event as follows: $e1 = 1$, $e2 = 2$, $e3 = 3$, $e4 = 4$, $e5 = 5$, $e6 = 6$, $e7 = 7$, and $e8 = 8$. Third, assign a unique LCED field to each pattern. This can be a simple description of the pattern, which can be used by the operator. Finally, each field is separated by a delimiter (e.g., vertical bar |).

1 | 3 5 6 7 | BSC link object is down

2 | 1 2 4 5 | BSC link object is up

3 | 3 7 8 | BSC link object is active.

4.3 Global Correlation

We define a correlation zone (CZ), as a logical entity that represents a collection of objects. A CZ is modeled as a bipartite graph where one set of nodes represents the correlation patterns and the other represents the composite events. A CZ subscribes to the composite events, which are received from objects within the system. The bipartite graph must consist of one or more pattern nodes and the composite events must be from two or more objects within the system. The objects can be located within the same node or different nodes or a combination of the two. A CZ is represented by a queue, which receives composite events from the objects it subscribes to. Like the local correlation decoder, the CZ uses the same approach to identify arriving patterns. The CZ is identical to the concept of an object and its approach to event correlation. The only difference is that a CZ does not generate events and therefore it does not have its own alphabet. Instead, the user defines its alphabet and its language according to known causal relationships. The global correlation process is recursive. This means that a CZ class can be made of smaller set

of CZs. This concept fits well given the hierarchical nature of communication systems.

While the events in local correlation are the object's alphabet, the global correlation's alphabet is made up of the composite events subscribed to by the CZ based on casual relationships. Therefore, a CZ can be represented by any arbitrary bipartite graph that consists of patterns and composite events. Our objective in this step of the correlation process, however, is to address the scalability issue while finding a solution that works. This can be achieved by using the class concept we explained in the local correlation step.

In wireless communication systems the concept of hierarchy is evident. A system consists of thousands of base stations and hundreds of site controllers, both of which are located remotely. Other types of nodes including the OMC are located in the central office. These nodes belong to a few node classes, which make them ideal candidates for object-oriented methods.

Defining bipartite graphs in global correlations is similar to that of local correlation. It starts with the definition of event-pattern pairs. This knowledge is stored in text files, which are then read into the system during initialization. However, there are two problems that need to be addressed.

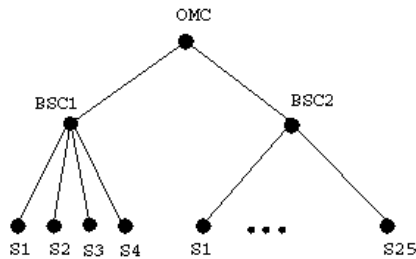


Fig. 5. A small segment of a system

First, patterns of a CZ are dependent on the system's size and complexity. Therefore, the bipartite graph of a CZ must be built dynamically with input provided by the topology information collected from the OMC. To illustrate this concept, consider, for example, Fig. 5, where a small segment of a wireless system is depicted. Here, BSC1 controls four sites while BSC2 controls twenty-five sites. There is an X.25 connection between each site and the OMC that passes through a BSC. Moreover, an X.25 connection exists between every BSC and the OMC. If we consider a BSC outage pattern to be all the X.25 link failure events between the OMC and the sites controlled by that BSC, then BSC1 has a different outage pattern than BSC2. This information can only be captured by accessing the topology information stored at the OMC.

Second, we need a way of collecting all the composite events subscribed to by a CZ in a single location for correlation. This can be achieved through the following steps:

- During system initialization, for each global pattern, identify the node(s), the objects, and the local result LCEC(s) that make up the global pattern;

- Send a message to each object identified above indicating the need to subscribe to the specified LCEC(s);
- Objects receiving such messages will store the destination address of the CZ along with the corresponding LCEC in a table;
- During normal operation, once an object detects a pattern, an LCEC is assigned to it. The object will search through its subscription table to see whether it needs to be routed to a destination for global correlation or not.

Each composite event is defined as follows:

1. CZ Class: The name of the class that the CZ belongs to;
2. Object Class: The object class within the CZ class;
3. LCEC: Local Correlation Event Code of the object's result.

Using topology information, classes are converted into instances with which a CZ has a causal relationship. Once this conversion is completed, a global bipartite graph is generated, which is similar to that of a local correlation. Like the local correlation step where a code is assigned for each pattern (LCEC), we will assign a code for a global pattern and call it a Global Correlation Event Code (GCEC).

The relationship between an object and the global correlation patterns may be one-to-one or a one-to-many relationship. This means that a single LCEC can be sent to one or more CZs. The type of relationship is governed by the causal relationships between them.

An example illustrating these concepts appears in Fig. 6, which depicts a segment of a wireless system. There is an X.25 link between every S node and the OMC which passes through a BSC. Let us define a CZ class to be an S node. We define an S node failure as one of the patterns within this CZ.



Fig. 6. Simple segment example

The composite events making up this pattern are as follows: a link object failure in the controlling BSC (between BSC and an S node), a BSC controller object failure which controls a set of S nodes, a link object failure between a D1 and an S node, and the X.25 link object failure in the OMC that controls a specified S node. The pattern can be specified as follows:

- SI BSC-S-Link| 1
- SI BSC-Cntrl|1

S| D-S-Link|1

S| OMC-S-X25|1

The next step is to convert this template pattern into an instance for a specified CZ. Let us take the S1 node as an example. The template pattern becomes:

S1|BSC1-S1-Link|1

S1|BSC1-Cntrl|1

S1|D1-S1-Link|1

S1|OMC-S1-X25|1

The CZ for S1 then sends a message to subscribe to all the objects in the pattern. It is important to note that the BSC1-Cntrl has a one-to-many relationship with the S node class. Therefore, when the controller object generates an LCEC of 1, it will send it to all the S nodes it controls, i.e., CZ of S1, S2, and S3, even though CZ S2 and S3 do not need it. This is not a problem to be concerned about since S2 and S3 will flush out these unwanted events either due to the limited size of the queue holding the events or through temporal reasoning.

4.4 Temporal Reasoning

The temporal relationships between events are as important as causal relationships. We plan to utilize the temporal relationships in two different areas, (1) reducing noise and therefore improve the accuracy of the correlation results; and (2) solving the problem of variable length codes.

Local and global entities are represented by queues where events arrive for correlation. Spurious and delayed events among others are the cause of noise. The events stay in the queue until they are flushed out by the FIFO effect. This FIFO effect is helpful but not enough to solve the noise problem completely. Next, we introduce ways to incorporate the temporal relationships between events to improve the noise problem.

There are three levels at which a temporal relationship can operate: (1) event level; (2) pattern level; and (3) queue level. We start with the event level temporal relationships. Here, events are considered as independent entities. Only the aging concept will be considered at this level. Three different types of aging techniques can be employed: (1) Threshold: an event will be active until a predefined value is reached when it is considered obsolete and should be removed from the queue. This behavior can be modeled as a step function; (2) Mathematical functions: an event is assigned a weight based on its age in the queue. Upon arrival in the queue, an event has a weight of 1. This weight decreases as time goes by until the weight becomes zero and the event is removed from the queue. The weight value follows the behavior of some function. Finding the best model for event aging is not simple and is system dependent. Simple functions such as linear or truncated exponential can be easily implemented and might be a good starting point.

If we treat all the different events within a queue equally, then the same techniques described for an event level can be applied to a queue. As an example, consider the

threshold level technique. Here, we assign a threshold value for a queue where any event older than that value will be considered obsolete and must be removed. The queue level temporal relationships can be seen as a simplified version of event level temporal relationships.

The last technique uses the pattern level temporal relationships. Here, every set of events that make up a pattern is treated differently. This is true of the same event belonging to different patterns. This level of temporal relationships is the most effective in dealing with the noise problems in communication systems because it treats the temporal constraints of every pattern differently.

Pattern level temporal relationships require more than a simple aging technique. There is a need for a temporal language to allow for the specification of a variety of temporal relationships between a set of events. In [23], an overview of existing temporal logic programming languages was given. While [24], provided composite event specification language that can be used to express complex temporal relationships between events. For now, we will define the following single operator: All events comprising a pattern must arrive within a specified period of time. This temporal constrained operator is useful in eliminating old and unrelated events.

So far we have been concerned with using temporal logic to reduce noise. Another important use of temporal logic is in the area of variable-length codes. Consider for example the following scenario: A periodic process runs every 30 minutes and if successful, it generates a pattern of 3 events: Ready, Start, and Complete. If the process fails, only one or two events are generated, i.e., Ready or Ready and Start are generated. The problem of distinguishing between the two patterns is that the failure pattern is a prefix of the success pattern; that is, the failure pattern is the same as the beginning of the success pattern. This problem is similar to the variable-length code problem – the receiver does not know when one pattern is over without looking further.

To illustrate the problem further, consider the following example which is outlined in [25]. Let's assume we have four symbols to be coded as follows: $s_1 = 0$, $s_2 = 01$, $s_3 = 011$, and $s_4 = 111$. Assume that the receiver receives the string $0111 \dots 1111$. It can only be decoded by first going to the end and then identifying groups of three 1s as each being an s_4 until the first symbol is reached. Only then can the first symbol be identified as s_1 , s_2 , or s_3 .

Waiting until the end of the string does not work in a noisy environment, i.e., with delayed, missing, or spurious events. Temporal logic can be used to address this problem more effectively. We can devise a pattern level temporal operator where once a pattern is detected, it is activated some time later if the same conditions still exist, i.e., the same events that triggered the pattern detection still in the queue. This operator is similar in concept to a watchdog timer activation and deactivation. Say we have two patterns p_1 and p_2 and assume that p_1 is a prefix of p_2 . When p_1 is detected, the watchdog timer is activated and is set to a predefined time. If p_2 is detected before the watchdog timer expires, it is disabled and p_2 is declared as the detected pattern. If, on the other hand, the watchdog timer expires, p_1 is then declared as the detected pattern. This concept can be extended to multiple patterns where each is a prefix of the other. Such an operator will help terminate an arriving pattern properly for positive identification.

5 Related Research

As we mentioned earlier, many researchers have investigated this problem applying a variety of techniques (see section 1). This section is intended to only describe previous research that we used to build our work on. We hope that this will give credit to previous work we used and highlight our contribution.

The work of [13] was based on fault detection using a Finite-State Machine (FSM) as a model of the system. In the system language section, we have modified some of the definitions used by them to fit our model.

In [8, 9], a graph theoretic approach was applied by utilizing the causal relations among events. The authors describe a distributed event correlation system and at the same time address some major issues in event correlation, i.e., scalability, resilience to noise, and generality.

The notation $p \rightarrow s$ is used to denote causality of event s by event p . Causality is defined as a partial order relation. The relation \rightarrow is described by a causality graph, which is a directed graph whose nodes represent events and whose edges represent causality. Each event in this causality graph is either a symptom event or a problem event. This graph is then pruned (all indirect symptoms and cycles are removed) to produce a correlation graph. Finally, a codebook with a predefined minimum Hamming distance is extracted from the correlation graph.

These graphs are generated automatically based on a specification model of the network at hand. The specification model is written in an object oriented modeling language called MODEL (Managed Object Definition Language). MODEL is an extension of the CORBA IDL language. It adds new syntactic constructs to specify semantics that cannot be specified in CORBA IDL: relationships, events, problems, and causal propagation.

6 Conclusion

We have introduced a framework for event correlation in communication systems. We showed how the concept of class in object-oriented methodology is used to provide scalability to our approach. Graph and coding theories are used for correlation. Finally, temporal reasoning was investigated for this framework.

We believe that this work has extended the research done by [8, 9] in many ways. We replaced the MODEL language with a simple free text causal language. We introduced a new decoder for pattern identification. Entities and their queues are created only if they generate events and are deleted once their queues are empty. This helps reduce memory requirements and decrease system complexity. Temporal relationships have been shown to be effective alongside the causal relationships between events.

References

1. ANSI T1.215, OAM&P - fault management messages for interface between operations systems and network elements, 1994.

2. Appleby, K., Goldszmidt, G., Stiender, M., "Yemanja - A Layered Event Correlation Engine for Multi-domain Server Farms," *Integrated Network Management VII*, pp. 329-344, May 2001.
3. Jakobson, G., Weissman, M., Brenner, L., Lafond, C., Matheus, C., "GRACE: Building Next Generation Event Correlation Services," *Proceedings of IEEE/IFIP Network Operations and Management Symposium*, pp. 701-714, April 2000.
4. Jakobson, G., Weissman, M., "Alarm Correlation," *IEEE Network*, Vol. 7, No. 6, pp. 52-59, Nov. 1993.
5. Sheers, K., "HP OpenView Event Correlation Services," *Hewlett-Packard Journal*, pp. 31-42, Oct. 1996.
6. Wietgreffe, H., Tuchs, K., Jobmann, K., Carls, G., Frohlich, P., Nejd, W., Steinfeld, S., "Using Neural Networks for Alarm Correlation in Cellular Networks," *International Workshop on Applications of Neural Networks in Telecommunications*, pp. 248-255, June, 1997.
7. Weiss, G., Eddy, J., Weiss, S., "Intelligent Telecommunication Technologies," in *Knowledge-Based Intelligent Techniques in Industry*, Jain, L., Johnson, R., Takefuji, Y., Zadeh, L., Editors, pp. 251-275, CRC Press, Boca Raton, FL, 1999.
8. Klinger, S., Yemini, S., Yemini, Y., Oshie, D., Stolfo, S., "A Coding Approach to Event Correlation," In A.S. Sethi, Y. Raynaud, and F. Faure-Vincent, editors, *Proceedings of the Fourth IEEE/IFIP International Symposium on Integrated Network Management*, pp. 266-277, Chapman and Hall, London, UK, May 1995.
9. Yemini, S., Klinger, S., Mozes, E., Yemini, Y., Oshie, D., "High Speed and Robust Event Correlation," *IEEE Communications Magazine*, Vol. 34, No. 5, pp. 82-90, May 1996.
10. Albaghdadi, M., Hood, C., Hamlen, M., "A Framework for Distributed Event Correlation," *Proceedings of the 17th IASTED International Conference - Applied Informatics*, pp. 467-470, Innsbruck, Austria, Feb. 1999.
11. Lo, C., Chen, S., "Robust Event Correlation Scheme for Fault Identification in Communications Network," *GlobeCom*, pp. 3745-3750, 1998.
12. Hasan, M., Sugla, B., Viswanathan, R., "A Conceptual Framework for Network Management Event Correlation and Filtering Systems," *Sixth IFIP/IEEE International Symposium*, pp. 233-246, 1999.
13. Wang, C., Schwartz, M., "Fault Detection with Multiple Observers," *INFOCOM*, pp. 2187-2196, 1992.
14. Bouloutas, A., Hart, G., Schwartz, M., "Simple Finite-State Detectors for Communication Networks," *IEEE Transactions on Communications*, Vol. 40, No. 3, pp. 477-479, March 1992.
15. Bouloutas, A., Hart, G., Schwartz, M., "Fault Identification Using a Finite State Machine Model with Unreliable Partially Observed Data Sequences," *IEEE Transactions on Communications*, Vol. 41, No. 7, pp. 1074-1083, July 1993.
16. Rouvellou, I., Hart, G., "Automatic Alarm Correlation for Fault Identification," *INFOCOM*, pp. 553-561, 1995.
17. Deng, R., Lazar, A., Wang, W., "A Probabilistic Approach to Fault Diagnosis in Linear Lightwave Networks," *IEEE Journal on Selected Areas in Communications*, Vol. 11, No. 9, pp. 1438-1448, Dec. 1993.
18. Dawes, N., Altoft, J., Pagurek, B., "Network Diagnosis by Reasoning in Uncertain Nested Evidence Spaces," *IEEE Transactions on Communications*, Vol. 43, No. 2/3/4, pp. 466-476, Feb./Mar./Apr. 1995.
19. Boubour, R., Jard, C., Aghasaryan, A., Fabre, E., Benveniste, A., "A Petri Net Approach to Fault Detection and Diagnosis in Distributed Systems. Part I: Application to Telecommunication Networks, Motivation, and Modeling," *Proceedings of the 36th Conference on Decision and Control*, pp. 720-725, San Diego, CA, Dec. 1997.

20. Aghasaryan, A., Fabre, E., Benveniste, A., Boubour, R., Jard, C., "A Petri Net Approach to Fault Detection and Diagnosis in Distributed Systems. Part II: Extending Viterbi algorithm and HMM Technique to Petri Nets," Proceedings of the 36th Conference on Decision and Control, pp. 726-731, San Diego, CA, Dec. 1997.
21. Stuttgart Neural Network Simulator (SNNS). University of Stuttgart, Institute for Parallel and Distributed High Performance Systems (IPVR), [cited 7 September 2000]. Available from <http://www.informatik.uni-stuttgart.de/ipvr/bv/projekte/snns/snns.html>.
22. Stremler, F., Introduction to Communication Systems, Third Edition, Addison Wesley, Reading, MA, 1990.
23. Orgun, M. A., Ma, W., "An Overview of Temporal and Modal Logic Programming," in Temporal Logic, First International Conference, ICTL 94, pp. 445-479, Springer-Verlag, Bonn Germany, July, 1994.
24. Liu, G., Mok, A. K., Yang, E. J., "Composite Events for Network Event Correlation," Integrated Network Management VI, pp. 247-260, May 1999.
25. Hamming, R., Coding and Information Theory, Second Edition, Prentice-Hall, Englewood Cliffs, NJ, 1986.

Policy-Based Management for Multimedia Collaborative Services

Hamid Harroud¹, Mouhsine Lakhdisi¹, Ahmed Karmouch¹, and Cliff Grossner²

¹ Multimedia & Mobile Agent Research Laboratory, School of Information Technology & Engineering (SITE), University of Ottawa,

161 Louis Pasteur St. Ottawa, ON, Canada K1N 6N5, Canada
{hharroud, mlakhdis, karmouch}@site.uottawa.ca
<http://deneb.genie.uottawa.ca>

² Nortel Networks, Ottawa, Canada

Abstract. Virtual team issue has emerged as an important new teamwork model, distinguished from the conventional way in which people work by its ability to transcend distance, time and organizational boundaries. A virtual team consists of a dynamic collection of individuals, a set of collaborative services and network facilities that ensure a flexible and secure coordinated resource sharing. In the Multimedia and Mobile Agent Research Laboratory we have developed the V-Team system, an agent-based multimedia collaborative environment to support virtual teams. V-Team aims to provide a set of team services for better collaboration between a virtual team participants, facilities for managing virtual teams with fully customized team services, and a simpler interface to network services. In this paper, we describe the main components of V-Team, the principles of context customization and the system monitoring approach via policies.

1 Introduction

With recent advances in network infrastructures and computing capabilities, groupware technology is becoming an important issue to offer an appropriate collaborative environment for distributed teams that transcend distance, time zones, and organizational boundary, known as *virtual teams* [1].

A virtual team consists of a dynamic collection of individuals, a set of collaborative services and network facilities that ensure a flexible and secure coordinated resource sharing. It requires agile and flexible groupware system that overcomes technical and organizational difficulties, without sacrificing the ability of its easy use and management from participants' point of view.

In the Multimedia and Mobile Agent Research Laboratory we are exploring different aspects of agent and groupware technologies in order to provide virtual teams with a system capable of supporting such requirements. The goal of the **V-Team** system is to develop a collaborative environment that would not only provide a set of team

services for better collaboration between a virtual team participants, but also facilities for managing virtual teams attributes and the context customization of their collaboration.

Our previous research experience has shown that agent paradigm combined with the use of policies is a promising approach for the development of flexible distributed systems [2][3]. Agents have the necessary autonomy to act on behalf of users or programs and to migrate from host to host on a network while performing their tasks. The use of policies carries out the ability to dynamically change the behavior of the system without altering its code. These characteristics highly reduce the system complexity, while permitting an efficient control of virtual team services at different levels.

Thus, we have designed an agent-based system that offers virtual teams an appropriate collaborative environment, with fully customized team services and flexible agent-based monitoring. Team services include multimedia conferencing, distance group meeting facilities and virtual teams' context management service. The context management service includes participants' attributes, their capabilities, the selection of applications and resources, and network mechanisms to be used.

The organization of the paper is as follows. The following section introduces the main components of V-Team. The basic principles of context customization are then described, with the policy-based approach and the system monitoring being presented in subsections. Then, basic team services that offer virtual team participants mechanisms to "meet" without being in face to face situation are presented. The benefits of our approach and related work are then discussed. Finally, we draw our conclusion and future work.

2 Agent-Based System for Collaborative Environment

In this section we first present fundamental requirements on an environment to permit collaborative work within a virtual team. Based on the identified requirements, we present a generic conceptual framework that supports synchronous collaboration between virtual team members. Then, we describe agents that compose the system architecture, the agent platform that supports them, and the inter-agent communication protocol.

2.1 System Requirements

In developing V-Team, we considered various requirements that could enable an efficient relationship among a recognizable team of people, a set of collaborative services, a set of network capabilities and a collection of rules binding these elements together such they behave in a consistent manner to satisfy team needs. For an environment supporting the integration of such entities, we have identified the following specific requirements:

- Ability to create and manage a virtual team context that enables a full customization of services.

- Automatic authentication of team members, and configuration of collaborative sessions without requiring their direct intervention.
- Enabling a seamless physical resource allocation of logical resources requested when creating the team.
- A separation between team services (applications) and network services (e.g. mobility location, CoS/QoS, multi-party, peer to peer, multimedia session control), so that a team service may dynamically be adapted to network conditions and team's context.
- Distributed control and management of dynamic changes that may occur during the team life cycle, either in its structure, its activity and planning, or in its members' locations.

2.2 Conceptual Approach

We have designed the V-Team environment as agent architecture. Agents show special promise in enabling the rapid construction of robust and reusable software [4]. Agents cooperate and communicate with each other, and have the ability to communicate with and directly control explicit applications. To monitor the behavior of an agent and its decision making, we attach to the agent a set of policies. A policy is a set of conditions that prohibit or permit an agent (called the subject) to perform actions on target entities. Conditions may concern the subject, the target or the state of the system. A policy may trigger events for performing some actions when conditions are applicable. The use of policies, in V-Team, highly reduces the system complexity, while permitting an efficient control of virtual team activities.

The key component of V-Team system is that of V-Team Context Agent (VTC). VTC agent manages information about one virtual team participants' attributes, their capabilities and roles, team services to be used by all or certain team members, logical resources requested when creating the team, network services and different forms of underlying transport mechanisms. By gathering information about a virtual team and its context, the VTC agent generates a set of policies that monitor the behavior of agents representing a virtual team's participants.

V-Team also features the network service agent (NSA) and the team control agent (TCA).

NSA provides a simple interface to network services, such as mobility management, transport classes-of-service, privacy of transport and multimedia session control. It exposes these, and other, services to teams and their team services in a manner that makes the services easy to access and use. NSA also permits the underlying mechanism to change dynamically according to the network conditions and the environment context as needed.

TCA supports collaborative session and controls interactions between active virtual team members. It manages the distributed events' serialization and dispatches them to each active participant. Management includes the reordering and the filtering of events.

These three agents establish a relationship between participants, team services or applications to collaborate with and the network links that enable participants' interactions (figure 1).

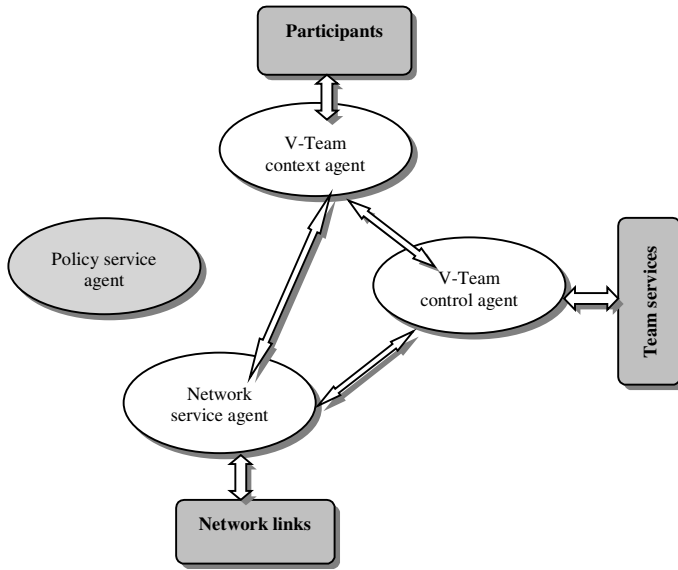


Fig. 1. V-Team system basic model

The Policy Service Agent (PSA) manages policies that govern the utilization of the system and network resources at each participant location. For instance, PSA may prohibit authorizations to a participant for obtaining a particular team service or resources at a specific location. Policies include the monitoring of the overall system configuration at different levels.

When an authorized authority (i.e. team leader) creates a virtual team, the system assigns a VTC agent to manage its specific environment and a NSA agent to support its network services. A TCA agent will then be associated to each collaborative session that would be activated by team participants.

2.3 System Components

The architecture of the system is depicted in figure 2. It is composed of several cooperating agents that work together to provide a collaborative environment for virtual teams.

Part of the V-Team context agent is the automatic authentication of participants, via their Personal Assistants (PAs). PAs are agents that represent participants and act on their behalf. The VTC agent provides a user interface that allows an authorized authority (i.e. team manager) to create a virtual team by specifying its members with

their privileges, logical resources assigned to the team, and the QoS assigned to different team services and media. The Team Participant Agent (TPA) customizes the user interface according to each user's privileges and the context. To setup a V-Team session, a negotiation process among PAs may then be engaged under the control of the VTC agent, which plays the role of a mediator.

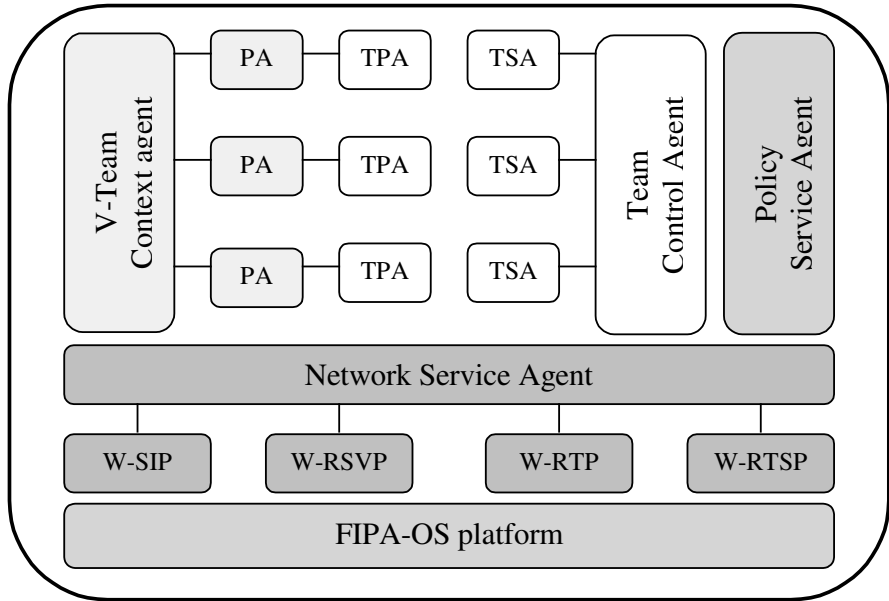


Fig. 2. Generic V-Team agent-based architecture

At the time of starting a V-Team session (e.g. audio conferencing), the NSA agent is required to locate the team members wherever they are and to control the session, using W-SIP and W-RTP agents that wrap SIP protocol [5] and Real-time Transport Protocol respectively.

Different network services become, in effect, integrated to the system by wrapping them through agents.

V-Team is intended to provide a basic set of team services that could be either a collaborative aware or unaware services. Any team service becomes “collaborative” by defining its corresponding Team Service Agent (TSA). TSA agents facilitate interaction among participants under the management of the team control agent (TCA).

V-Team is designed to be highly flexible and dynamically manageable through multi-level policies. Policies are maintained and distributed over V-Team agents through the PSA.

V-Team agents are executed under the control of FIPA-OS platform [6], which provides a framework for agent communication and management, including Agent

Communication Channel (ACC) using Agent Communication Language (ACL), Agent Management Service (AMS), Directory Facilitator (DF), and mobility services.

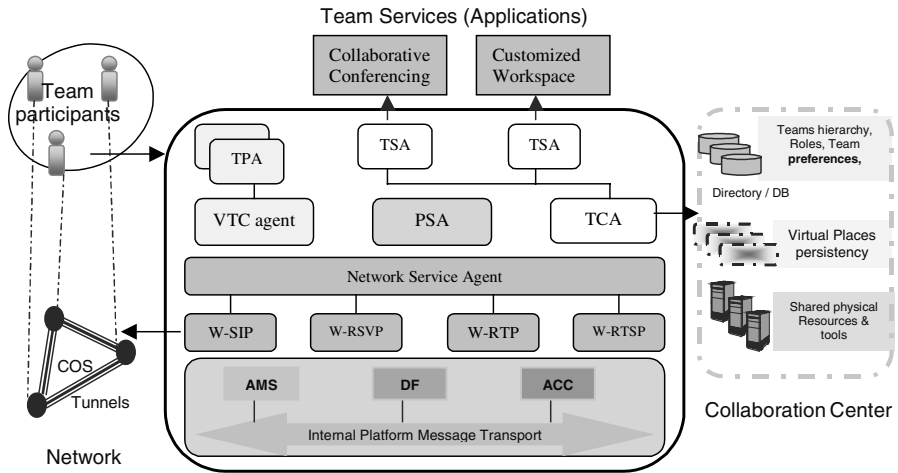


Fig. 3. V-Team System View

Figure 3 shows an example of V-Team system view. TSA provides linkage and execution environment necessary to control Collaborative Conferencing and Customized Workspace services. The NSA provides the control and resource reservation necessary to enable Class-of-Service and privacy in underlying transport mechanisms. Team participants are in different locations and may move from one location to another. Automatic locating of team participants is then required. Using SIP proxy server and location server, tunnels may be automatically established between participants. Participants “meet” at virtual places. The collaboration center permits places storage and playback, resource sharing, and information management.

3 Policy-Based Context Customization and Monitoring

A system architecture like that of the V-Team system described in the previous section requires specific mechanisms to collect and maintain virtual teams’ context, and to dynamically manage the overall behavior of the system. Those are the VTC and the PSA agents’ roles that are described in this section.

3.1 V-Team Context Agent

The VTC is a context-aware agent that uses context information to adapt the corresponding virtual team behavior to the current situation of use. Context information

includes any information that characterizes the team operating-environment, such as member’s identity, the role within the team, location, time, and availability of resources (e.g. bandwidth and device capabilities) to name but a few.

The VTC features are the following:

- To collect the entire context about a particular virtual team, by gathering information from the system entities such as the team manager, participants via their PAs, the network service agent and the PSA.
- To interpret context information by translating it from different representation formats to a set of context policies that could then be distributed to specific agents via the PSA (e.g. TPA, TCA) or used by the VTC agent itself.
- To evaluate context-triggered policies and to execute enabled actions accordingly when in a certain context. This includes the selection of information and services to be customized to the virtual team members. An example of customization would be that when a participant joins a collaborative session on a phone, he would use audio only and other materiel of the ongoing session (e.g. shared document) would be e-mailed to him or shown on a nearby terminal (e.g. fax machine). The VTC agent is also able to satisfy a participant specific needs by enabling a third party application (e.g. MS-Word), combined with a selection of network capabilities, as a customization of an authorized team service (e.g. join editing document).

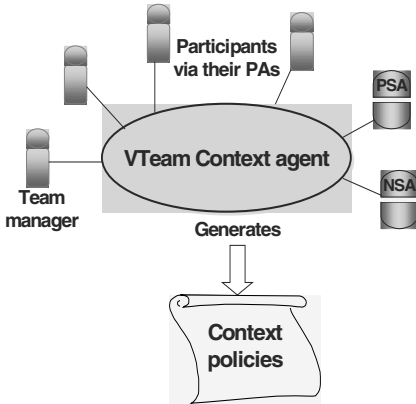


Fig. 4. VTC context information

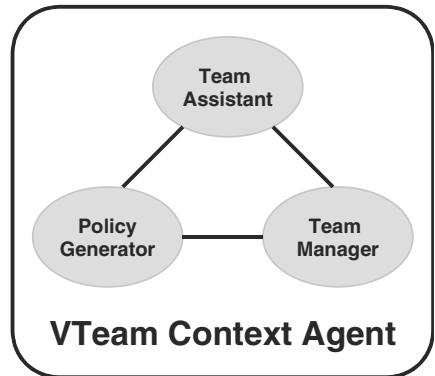


Fig. 5. VTC components

As shown in figure 5, the VTC agent is composed of three components: policy generator, team manager, and team assistant.

- The *Policy Generator* implements and interprets context information. The interpretation consists of translating received and caught notifications from different representation formats (e.g. user interface, ACL messages and events) to a set of policies, called **context policies**. Context policies maintain different types of information. These types are virtual team’s attributes (members and roles), team’s activities (team services), scheduling (time, day) and network services (location,

class of services, QoS). These policies are represented and stored in same way as the PSA does.

- The *Team Manager* provides the VTC agent with access to context information. It is responsible for context acquisition, which is achieved either by using user interfaces (e.g. team creation and personal preferences) or by interacting with agents that are involved in the virtual team activities.
- The *Team Assistant* uses context policies to trigger events that apply to specific agents according to their state and the context of their use. At setup phase of a collaborative session, the team assistant manages a negotiation process, that guides PAs agents to persuade each other for making decisions on users' behalf (e.g. when to "meet", duration, frequency, ...etc). The team assistant monitors the negotiation and helps agents to achieve a join agreement in conformance to enabled policies.

3.2 Policy Service Agent

3.2.1 Policy Definition

A policy is a set of rules reflecting an overall strategy or objective, affecting the behavior of agents and thus designed to help control and administer a system. A policy rule is a set of actions to be performed by a subject agent on a target agent providing some conditions are satisfied and/or some events are triggered.

The conditions, events and actions are all related to one or more agents of the system. Conditions are typically based on the state of an agent or a resource in the system. The events are triggered either by a time-period condition, a change in the state of an agent or as a result of an action (i.e. before/after events).

Policies could be applied through several levels of a system. From the low network level where policies monitor network devices to the organizational level where they define the role of members of a team. Policies are also useful for resource management and service monitoring.

3.2.2 Policy Management System (PMS)

In the architecture of the PMS (Fig.4), two agents are of particular interest, the Policy Management Agent (PMA) and the Policy Service Agent (PSA). These two agents are interacting and cooperating through the Policy Information Base.

- **Policy Management Agent (PMA):** the purpose of the PMA is to assist the administrator of the application through the policy editing process. PMA is also responsible for detecting static conflict between policies that may occur during the editing process. Policies are stored in the Policy Information Base under the management of the PMA. The PMA is application-independent. Information about the application is stored in the application profile within the PIB. Application profile is a set of attributes (e.g. application structure and components) used by the PMA to communicate with the applications.

- **Policy Service Agents (PSA):** each application (V-Team in our case) is associated with a Policy Service Agent that is responsible for locating the events and conditions likely to trigger a policy through the Event Listener and Constraint Manager. To enforce a particular policy, the PSA invokes a set of actions to be executed. For each triggered policy, the PSA keeps track, in a log file, of the result of policy evaluation and enforcement.
- **Policy Information Base (PIB):** a database of information about the application storing information about agents in the application, the services they provide and the resources they manage along with the policies monitoring their behavior.

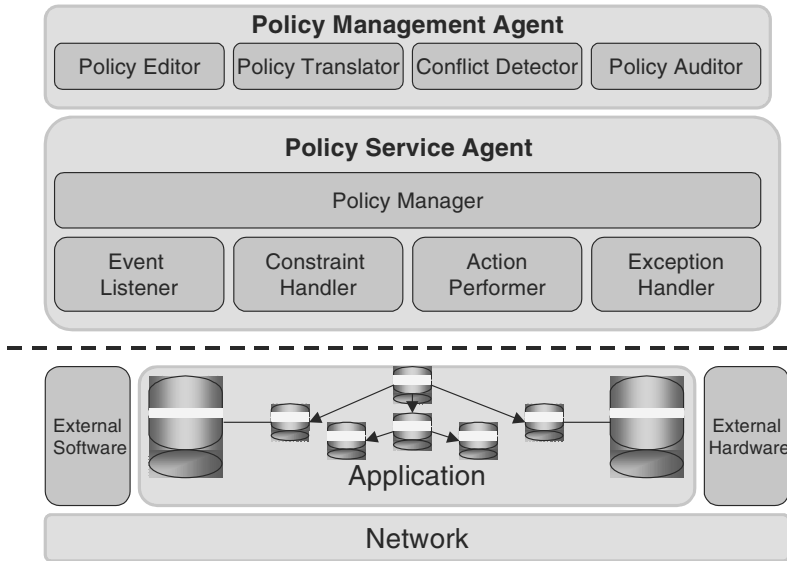


Fig. 6. Policy Management System Global Architecture

4 Basic Team Services Implementation

We have implemented V-Team components as Java-based agents on the top of FIPA-OS platform, and we have defined appropriate interfaces for linking and setting up existing basic services. Such services consist of SIP proxy and SIP User Agent (SIP-UA) developed by Columbia University [8] and RAT audio conferencing and WBD shared workspace tools developed by the Networked Multimedia Research Group at University College London [9].

SIP is a protocol for initiating interactive communication multimedia sessions between users. It provides mechanisms so that user agents and proxy servers can determine the location of participant and perform call setup including SDP for media description. We have developed a wrapper agent (W-SIP) that interfaces SIP-UA to the

NSA for automatic team participant's availability and invitation. An example of SIP-UA call invitation is shown in figure 7. *Karm* team member is invited, via the proxy server (i.e. *spica.uottawa.ca*), to participate in Advanced Networks team meeting (i.e. *subject*), with audio conferencing and shared workspace (i.e. *Media*). W-SIP sets up SIP-UA parameters and waits for the invitation status and SDP description that are then sent to the NSA using ACL messages.



Fig. 7. An example of SIP-UA call setup

In the following, we provide a scenario that describes the V-Team approach and, in particular, highlights features that have already been implemented as a prototype. Since the behavior of the system is aimed at being completely monitored by policies, the administrator, using the PMA, defines policies that determine who can create virtual teams and what logical resources are available to those team leaders.

The virtual team creation is customized according to the team leader privileges (e.g. resources assigned to the team, media and QoS to be assigned). The team leader plays the role of the team controller, but can assign other team members as team controllers. The team controller has access to the TCA agent, which provides collaborative session facilities.

When creating the team, the team leader can either set up scheduled team services (i.e. team meeting) or delay these to be done later by a team controller. Team management is performed by assigning policies to the system agents, as shown in figure 8.

In figure 9 one can see an example of customized TPA user interface (i.e. *Allen* participant). It displays, on the left, two virtual places (*Advanced Networks and Architecture Engineering*), since *Allen* participates in two teams. *Allen* can collaborate in

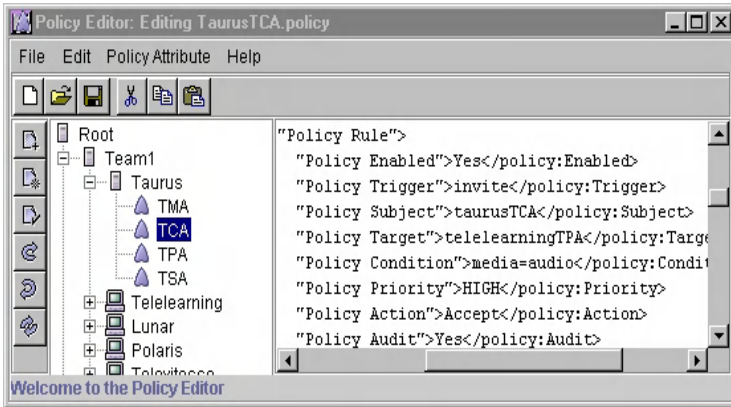


Fig. 8. A sample of editing TCA policies

one or another by selecting the corresponding virtual place. A virtual place corresponds to the pair virtual team and session. It represents a place where virtual team members can “meet” and work on a certain topic.

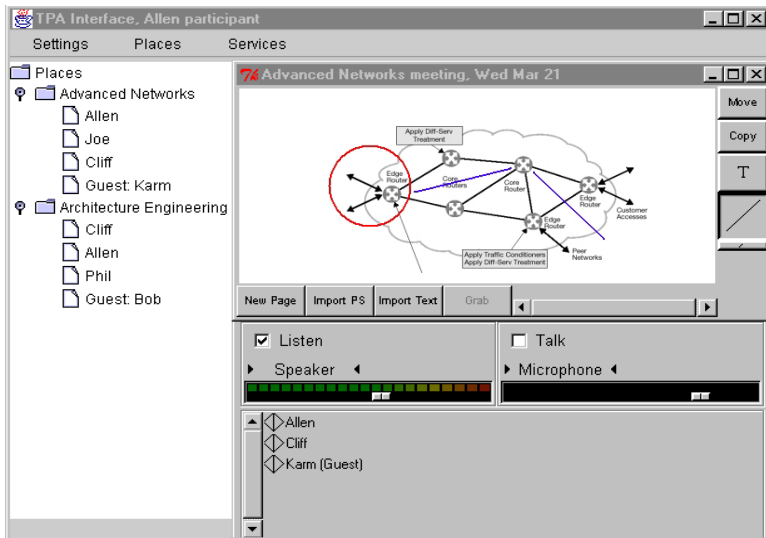


Fig. 9. TPA interface with white board and audio tools

The right part of figure 9 shows an on going session among *Advanced Networks* virtual team members (*Allen, Joe, Cliff* and *Karm*). In this example, members collaborate using integrated shared-workspace and audio-conferencing tools that are WBD and RAT respectively, however, there is one member (i.e. *Joe*) that has been invited but presently is not participating in the audio conferencing (e.g. *Joe*’s terminal capabilities, authorization policies).

TPA interface provides also other options, top of figure 9. A member uses these options to specify personalized preferences (*Settings* option), create new virtual places or enter existing ones (*Places* option) and access authorized team services such as virtual team creation, scheduling and support for joining or leaving an on going collaborative session (*Services* option).

5 Related Work

Many collaborative environments providing a shared team workspace with common tool-set and shared applications have been developed. These systems generally attempt to single-handedly provide necessary services to serve a large broad user base, and tend to use the network only for transport. In contrast, V-Team is strongly motivated by providing different users with different levels of reliability and awareness, and by enabling the integration of a variety of network services to satisfy the unique needs of diverse classes of virtual teams.

V-Team has certain similarities to mSTAR environment [10]. mSTAR features an agent-based architecture and separates application agents from the network agent, which makes the collaborative application more adaptable to existing network conditions. Despite these characteristics, mSTAR is primarily a framework that supports developers in creating distributed and real-time multimedia applications.

NCSA Habanero [11] is also an approach that supports the development of collaborative environments. Habanero lets developers create and convert Java applets into multi-user applications, known as “Hablets”. DISCIPLE [12] lies mainly in using Java event delegation model in conjunction with JavaBeans applications. DISCIPLE, like Java Collaborative Environment (JCE) [13], extends the Java-AWT to intercept events. These approaches have the cost of altering the application code in order to make it collaborative. The use of policies, combined with context-aware agents, enables V-Team to dynamically combine a collection of third party applications with a selection of network capabilities, and off-loads the need of dealing with these capabilities directly.

6 Conclusion and Future Work

In this paper, we have presented an agent-based multimedia collaborative environment to control and coordinate resource and service sharing in dynamic virtual teams. The proposed approach aims at more of a teamware approach, where the team context is a key factor that extracts, interprets and uses context information.

Since the structure and context of a virtual team may change significantly across time and space during its life cycle, the use of policies in managing dynamically the team behavior at different levels is of the highest relative value. We have introduced a policy management as a service that guides a team manager in the process of defining, enforcing and auditing policies.

The system uses the session initiation protocol as a basis for locating team members, and uses existing multicast-based media applications (i.e. RAT, and WDB) as basic tools for multimedia conferencing service. Future work includes integration of various network services, which will contribute to off-load collaborative services from large transport-related capabilities, resulting in a broader range and potential customization of these services.

Our current focus is on extending the virtual team context with more capabilities in collecting context information and adapting the system behavior accordingly. The use of policy-based agents that allows each agent to monitor its own policies is important in our work as well.

Acknowledgement. This work was supported partly by Nortel Networks and Natural Sciences and Engineering Research Council of Canada.

We would like to thank Allen Eddisford and Stephen Ng from Nortel Networks for their valuable feedback and technical suggestions on this work.

References

1. J. Lipnack and J. Stamps, "Virtual Teams, Reaching Across Space, Time and Organizations with Technology," John Wiley & Sons 1997.
2. T. Magedanz, A. Karmouch, "Mobile Software Agents for Telecommunication Applications," *Journal of Computer Communication*, Vol. 23, Issue. 8.
3. H. Harroud, M. Ahmed, A. Karmouch, "Agent-based personalized services for mobile users over a VPN," *International Conference on Enterprise Information Systems*, Setúbal, Portugal, July 2001.
4. M.N. Huhns, "Agent Teams: Building and Implementing Software," *IEEE Internet Computing*, February 2000.
5. IETF SIP Working Group. SIP: Session Initiation Protocol, Internet Draft. November 2000
6. FIPA-OS V1.3.3 Distribution Notes, Nortel Networks Corporation, 2000, available at <http://www.nortelnetworks.com/fipa-os>.
7. IETF Policy Framework Working Group, "Policy Core Information Model Specification," Internet Draft, October 2000.
8. sipd - SIP redirect, proxy and registration server, sipc - SIP user agent. Software licensing available at <http://www.cs.columbia.edu/~hgs/license/>
9. The UCL Networked Multimedia Research Group, Mbone Conferencing Applications, available at <http://www-mice.cs.ucl.ac.uk/multimedia/software/>
10. P. Parnes, K. Synnes, and D. Schefstrom, "mStar: Enabling Collaborative Applications on the Internet," *IEEE Internet Computing*, October 2000.
11. A. Chabert et al., "Java Object-Sharing in Habanero," *Communications of the ACM*, Vol. 41, No. 6, June 1998.
12. I. Marsic, "DISCIPLE: A Framework for Multimodal Collaboration in Heterogeneous Environments," *ACM Computing Surveys*, Volume 31, No. 2, Article No. 4, June 1999.
13. H. Abdel-Wahab et al, "An Internet Collaborative environment for Sharing Java Applications," *IEEE Computer Society Workshop on Future Trends of Distributed Computing Systems*, pp. 112-117, October 1997.

14. J. R. Nicol, Y. S. Gutfreund, J. Paschetto, K. S. Rush, and C. Martin, "How the internet Helps Build Collaborative Multimedia Applications," *Communications of the ACM*, Vol. 42, No. 1, January 1999.
15. J. Altmann, R. Weinreich, "An Environment for Cooperative Software Development: Realization and Implications," *Proceedings of the Hawaii International Conference on System Sciences*, Kona, Hawaii, January 1997.
16. K. Sikkell, "A Group-based Authorization Model for Cooperative Systems," *Proceedings European Conference on Computer-Supported Cooperative Work (ECSCW'97)*, Lancaster, September 1997.
17. T. Ishaia and L. Macaulay, "The Role of Trust in Virtual Teams," *Proceedings of the 2nd International VoNet-Workshop*, Zurich, September 1999.
18. V. Pham and A. Karmouch, "Mobile Software Agents: An Overview", *IEEE Communications, Special Issue on Mobile Agents and Telecommunications*, Vol.36, No.7, pp.26-36, 1998.
19. M. Sloman, J. Labo, E. Lupu, "Policies for Distributed Systems and Networks," *International Workshop*, Bristol, UK, Jan. 2001.
20. B.N. Schilit, N.I. Adams, R. Want, "Context-Aware Computing Applications," *Proceedings of the Workshop on Mobile Computing Systems and applications*, IEEE Computer Society, Santa Cruz, CA, pp. 85-90, 1994.

Agent-Enhanced Dynamic Service Level Agreement in Future Network Environments

David Chieng¹, Alan Marshall¹, Ivan Ho², and Gerard Parr²

¹ Advanced Telecommunication Systems Laboratory,
School of Electrical and Electronic Engineering, Ashby Bld, Stranmillis Road,
The Queen's University of Belfast, BT9 5AH Belfast, Northern Ireland, UK
{d.chieng, a.marshall}@ee.qub.ac.uk

² Telecommunication and Distributed Systems Group,
School of Information and Software Engineering,
University of Ulster at Coleraine, BT52 1SA Coleraine, Northern Ireland, UK
{wk.Ho, gp.Parr}@Ulst.ac.uk

Abstract. Current network infrastructures are experiencing rapid transformation from providing mere connectivity, to a wider range of flexible network services with Quality of Service (QoS). We propose an agent-enhanced system that facilitates dynamic Service Level Agreement (SLA) activities, such as end-to-end QoS specifications and service price negotiations in such an environment. A prototype system consisting of real-time Java-based agents that interacts with a simulated network was developed to demonstrate scenarios and enable analysis. The studies show that this form of dynamic SLA negotiation introduces many innovative ways on how network services can be provisioned. This paper also highlights the effects of implementing dynamic connection bandwidth pricing on traffic load and network provider's revenues.

1 Introduction

After years of rapid technical and standardization efforts, various Quality of Service (QoS) architectures such as Differentiated Services (DiffServ) [1], Multi Protocol Label Switching (MPLS) [2] and Resource Reservation Protocol (RSVP) [3] begin to take a foothold in today's networking environments. Over the next few years we are going to witness rapid transformation in current network infrastructures from providing mere connectivity to a wider range of tangible and flexible network services with QoS. Faced with increasingly complex network, to provision, manage and bill services are becoming some of the biggest challenges for the service, network and content providers today.

Issues regarding Service Level Agreement (SLA) arise due to the need to maximize customer satisfaction and service reliability. According to [4], many end users and providers are still unable to specify Service Level Agreements (SLAs) in a way that benefits both parties. Very often, the service or network providers would overprovision their services that leads to degradation or on the contrary, failed to provide serv-

ices to the best of their systems or networks capabilities. Hence, the abilities to decipher what the end users want from what they asked for and to respond effectively are critical as the competitiveness of future providers does not only rely on the diversity of the services they can offer, but also the ability to meet customers' requirements. As users are more discerned about QoS, it is believed that value-added services and applications are best to be delivered and billed on per session or per-transaction basis [5]. A well-defined pricing scheme is not only important as a tool to enable users to match and negotiate services as a function of their requirements, it can also be a mechanism itself, as the dynamic setting of prices can be used to control the volume of traffic carried [6]. In this paper we introduce an agent-enhanced system that facilitates dynamic SLA specifications between end users and various providers in order to setup end-to-end connections or Virtual Leased Lines (VLLs) with preferred QoS and prices. Here, agents are employed as autonomous negotiators on behalf of various parties involved.

The rest of the paper is organized as follows: Section 2 summarises some related work. Section 3 gives some background on the definitions of SLA and the metrics used in our work. Section 4 presents some advantages of employing agents as the SLA mediator. This is followed by the description of our agent-enhanced service brokering architecture in Section 5. Section 6 illustrates the components within the agent environment and their interactions. Since bandwidth is a shared finite resource, a fair distribution policy such as pay as much as you used is proposed. This is highlighted in section 7 where we demonstrated how dynamic bandwidth pricing strategy can affect the overall network utilization as well as the generated revenues if users are putting service quality and cheapest price as their priorities.

2 Related Work

Our current work is motivated by a number of research. First, the researchers in [7] proposed the implementation of bandwidth brokers to support SLA trading which includes resource allocation, path selection and pricing between DiffServ networks. The authors in [8] introduced Virtual Network Service (VNS) that uses a virtualisation technique to customize and support individual VPNs' QoS level. The authors in [9] proposed a QoS management architecture that employs distributed agents to establish and maintain the QoS requirements for various multimedia applications. Here, QoS specification is categorized into 2 main abstraction levels: application and system. The capability of current RSVP signalling is extended in [10] where users are allowed to reserve bandwidth or connections in advance so that blocking probability can be reduced. Authors in [11] offered a similar idea but use reservation agents to help clients to reserve end-to-end network resource. The Resource Negotiation and Pricing Protocol (RNAP) proposed by [12] enables network service providers and users to negotiate service availability, price quotation and charging information on per application basis. Researchers in [13] implemented adaptive bandwidth pricing and bidding at the lower level of granularity such as path or link via multi-agent systems (MAS).

3 SLA Metrics

SLA provides a means of quantifying service definitions. In networking environment, it specifies what an end user wants and what a provider is committing to provide such as the definitions for QoS, performance levels, etc [14]. The definitions of SLA vary at business, application or network level. Business level SLA involves the issues such as pricing schemes and contract. Application level SLA concerns the issues of server availability (e.g., 99% during normal hours and 97% during other hours). Network level SLA or often referred as Service Level Specification (SLS) involves lower layer parameters such as throughput, latency, packet loss and jitter. In our work we first focus on the basic SLA metrics involved in establishing Virtual Leased Line (VLL) on demand type of services which are described in table 1 as follows:

Table 1. SLA metrics and descriptions

SLA metrics	Description
Guaranteed BW (b_i) *for request i	The amount of guaranteed/reserved bandwidth allocated to VLL. We only consider this metric at present due the ease of its configuration. It is also the single most important factor (not always the case) that affects other lower level QoS parameters such as delay, jitter, etc. Guaranteed BW can be quantified in units of 1kb, 10kb, etc.
Start Time (T_s)	This is applicable to scheduled services such as VoD, video conferencing, news broadcast, MTV, etc. For instant access, this metric is simply assigned as the current time.
Session Length (T_i)	The duration required for this VLL. This is applicable to Video on Demand (VoD) or news broadcast type of services where the session time is known in prior. However the session time can be extended automatically depending on the availability, policy, etc.
Price (P_i)	This can be the maximum price a user is willing to pay for this service.
Option (Ω_i)	It consists preferences and priorities in the form of rules. This is useful at times when not all the requested SLA metrics can be granted. User can specify which parameter is the priority and which is tolerable.

* Other hardware and software specifications such as application types, IP/MAC addresses, port numbers and other resources though not explicitly mentioned, are assumed to be taken care of by lower level agents.

From the metrics described above a SLA request i can then be represented by:

$$\text{Request}_i(b_i, T_s, T_i, P_i, \Omega_i)$$

4 Why Agent-Mediated SLAs?

Software agents offer many advantages in this kind of environment. Agents can assist the end users, the service and network providers, to perform expertise brokering tasks. These include service selection, QoS specification, pricing negotiations, etc. Agents are particularly suited for the tasks where fast decision making is critical. These satisfy the two most important aspects of performance in an SLA; availability and responsiveness [14]. The following attributes highlight the capabilities of agents in carrying out brokering tasks [15].

- **Autonomy.** Agents can be either responsive or proactive. It is able to carry out tasks autonomously under pre-defined rules or constraints. The level of their intelligence depends on the given roles or tasks.
- **Communication.** With this ability, negotiations can be established between two agents. Agent Communication Language (ACL) from Foundation for Intelligent Physical Agent (FIPA) has been widely adopted as the de-facto language.
- **Cooperation.** Agents can cooperate to achieve a single task. Hence, agents representing end users, service providers and network providers are able to cooperate to setup an end-to-end VLL that spans across multiple domains.
- **Mobility.** Java-based agents can migrate across heterogeneous networks and platforms. This attribute differentiates mobile agents from the other forms of static agents. Mobile agents can migrate their executions and computation processes to remote hosts. This saves shared and local resources such as bandwidth and CPU usage compared to conventional client/server systems. Thus, intensive SLA negotiation processes can be migrated to the service provider or network provider's domain.

ObjectSpace™ Voyager ORB 3.0 [16] is used to implement of our agent system. Voyager Agent API offers the ability to construct remote objects in the remote host and a set of control mechanisms that offer more flexible instructions on how the agent should terminate itself. There are two types of communication mechanisms, namely Method Calling and ObjectSpace™ by which Voyager agents interact with each other. The former mechanism enables an agent to call methods of another agent. This is provided if the calling agent knows a-priori the method interface of the called agent. The latter mechanism enables voyager agents to multicast an event message to other agents for example advertising for new services or offers.

5 Agent-Enhanced Service Brokering Architecture

In this architecture, the network provider has the control over all the access points within its managed domain. To setup a VLL, an end-to-end path must be configured in advanced before the actual traffic can be admitted. Hence, the network provider must

maintain a global view of the network resources which can be realized using Internet Gateway Protocol (IGP) link state algorithm [17]. This is contrary to RSVP [3] peer-to-peer approach, where network nodes such as routers or switches can decide locally whether to accept or reject resource reservation. Distributed admission control complicates the service management task and especially when comes to billing. Figure 1 illustrates the open architecture:

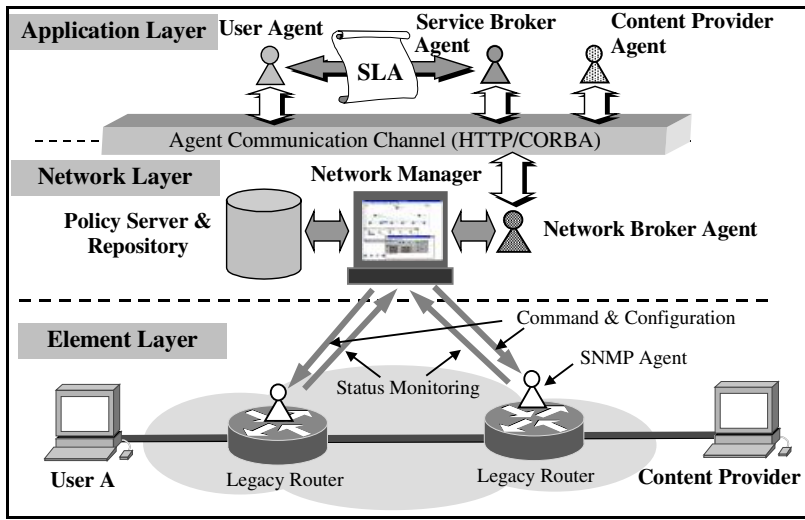


Fig. 1. Agent-Based Service Brokering Architecture [18]

Here, agents act as autonomous negotiators on the application layer. Conventional HTTP or CORBA can be utilized as the communication channel for agents. The role of an User Agent (UA) is to automate the service request procedures on the behalf of the end user. The Content Provider Agent (CPA) provides information or advertising facilities for its services such as news broadcast, Video on Demand (VoD), etc. The Network Provider Agent (NPA) acts as an access broker for its domain. Logically it is a component of the network manager but physically it may reside at the domain access router. The Network Manager has the direct access to a policy server which administers policies rules and actions for different services' SLAs. The policy repository stores defined policies as sets of rules for this domain. It may be a single physical site or replicated at several places in the form of databases, files, an administrative server or a directory server. Currently, Light Lightweight Directory Access Protocol (LDAP) directory is favoured by most vendors [19]. The Policy manager within the policy server validates policies in the policy repository so that it is mutually consistent in the network. The Network manager can also be an agent manager that assigns NPAs to serve the incoming agents. The Service Provider Agents (SPA) acts as a mediator between multiple parties involved. An UA first asks SPA for a service with various QoS preferences. The SPA then negotiates with respective CPA and NPA before setting up a VLL. The NPA then makes decisions based on the requested SLA and

propagates the required configurations down to the element layer. Conventional SNMP agents can be employed at the element layer to configure queues and monitor links and flows' states. The multi-party relationship model is illustrated in figure 2. If the route spans across multiple domains, the SPA may need to negotiate with different network providers and content providers in order to set up a VLL.

6 Agent Environment and Multi-party Interactions

A prototype multi-operator network model has been built using the Block Oriented Network Simulator (BONeS) [20] as a testbed for our proposed architecture. This allows functional and dynamic behaviour of the network under various agent-supported scenarios to be investigated. Figure 2 illustrates the interactions between various agent components. In our framework we have real-time agents running in LAN and virtual networks in simulation environment. Our agents use BONeS' Inter Process Communication (IPC) protocol to interface with the Sun Solaris-based Network Simulator via TCP/IP socket.

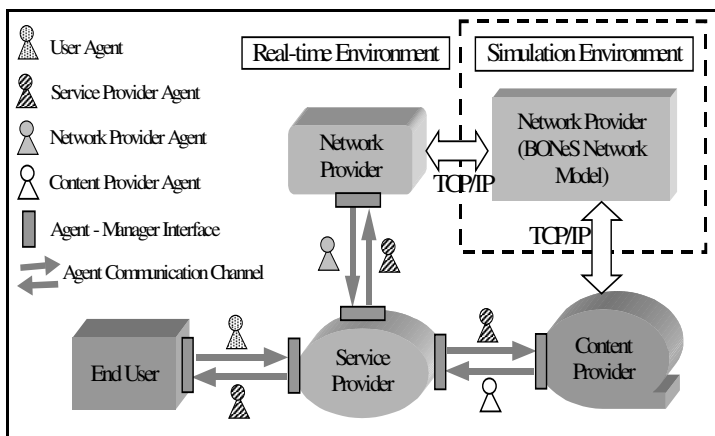


Fig. 2. Agent Components Interaction [15].

There are four main parties involved this scenario: the End User, Service Provider, Content Provider, Network Access Broker and the Network Provider. Each component consists of its own database, agent(s) and a manager. The manager's job is to provide service for any arriving agents, handle data transactions, storage retrieval, agent creation and task assignment. Figure 2 shows that when requesting a service, an end user can launch its agent to the central meeting point (SP) to interact with the local object (SP Manager). The manager then asks its server to create an agent with a task list. These agents can request for content listing from the CPA and then setting up a VLL with the NPA according to user SLA specification. At this stage, our agent security is based on 'trusted hierarchy' scheme where agents only communicate with trusted

remote hosts. However, it is anticipated that future development of the agent system will need to implement a much more rigorous security policy [15].

7 Demo – Agent Mediated SLA and Dynamic Bandwidth Pricing Scenario/Game

A demonstration on the prototype system was carried out during the technical visit session at Fujitsu Telecommunications Europe Ltd. headquarter in Birmingham in conjunction with World Telecommunication Congress/Integrated Signal Symposium (WTC/ISS2000). In each session, three volunteers were invited from the audience to assume the role of the future network operators. Their common goal was basically to maximize their network revenue by dynamically price the bandwidth under agent brokering environment. The scenario presents three individual networks owned by different operators; each is identical in terms of the number of devices, topology and available resources. Each network consists of nodes connected in a mesh topology. As a means of creating competition, all three networks offer the same access to a remote content provider that provides multimedia services as shown in figure 3 as follows:

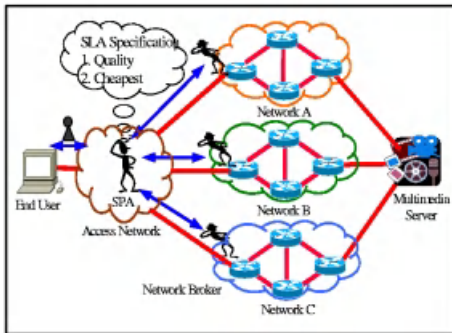


Fig. 3. Agent Brokering Scenario

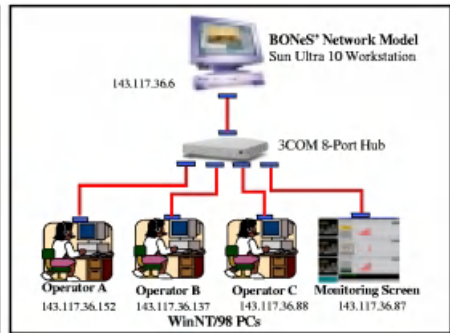


Fig. 4. System Set-up

A simple LAN was set-up for this demo as shown in figure 4. The multi-operator network model coupled with internal agent brokering mechanisms was run on the Sun Ultra 10 workstation. A few PCs were set-up for the competing network operators and a separate monitoring screen was provided (Fig. 6).

In this game we considered some universal customer preferences such as best QoS (guaranteed BW) and cheapest price. We assumed most users would want QoS as their first priority and the cheapest offering price as the second priority. Hence the options for User SLA request are:

$Q_i(b_i, Ts_i, T_i = \text{NOT negotiable AND First Priority}, P_i = \text{Cheapest AND Second Priority})$

If the cheapest network provider could not provide the required guaranteed VLL BW, the service provider would opt for the second cheapest one. During the game, SPA continually requested NPAs to setup VLLs to the multimedia server. In order to show the effects of dynamic pricing, we allowed the invited audience/acting network operator to manually change the BW price. For this demo we did not provide different pricing schemes for different user classes. The table below describes various billing parameters for the network operators in this demonstrator.

Table 2. Billing Parameters Description

Parameters	Description
BW selling price, P_i	The selling price for a BW Unit per minute for VLL i .
Cost Price, θ_i	The cost price for a BW Unit per minute for VLL i that changes according to link's reservation load status. This loosely represents the management overhead cost.
Operation Cost, σ	This represents the overall maintenance cost, hardware cost, labour cost etc per minute.
Guaranteed BW, b_i	This is the size of Guaranteed unit bandwidth allocated to VLL i .
No. of links, ℓ_i	The number of links used by VLL i . Here all the possible paths are pre-computed. Since no. of links is considered in the charging equation, the shortest available path is therefore preferred.
Session, T_i	The session length in minute subscribed by VLL i .

*Note: In this game, price and bandwidth were simply given in term of units. Some attributes are specifically created for this game.

In this game, users were charged at the end of their sessions. The calculation for gross revenue earned by a network operator from each VLL i is based on the following equations:

$$\text{Rev}_i = (P_i - \theta_i) \cdot b_i \cdot \ell_i \cdot T_i \quad (1)$$

Therefore the total gross revenue, $\text{Rev}_{\text{gross}}$ after period t hence:

$$\text{Rev}_{\text{gross}}(t) = \sum_{i=1}^{n(t)} \text{Rev}_i \quad (2)$$

Where

t = simulation time elapsed in minute.

$n(t)$ = total number of VLLs sold after t .

Total net revenue Rev_{net} after t hence:

$$\text{Rev}_{\text{net}}(t) = \sum_{i=1}^{n(t)} \text{Rev}_i - \sigma \cdot t \quad (3)$$

Where

σ = operating cost per minute.

Each player or acting network operator could monitor his/her competitors' offered BW prices and set their own price at the console shown in figure 5. Network links' reservation load, cost and network topology showing current network utilisation were also displayed. Users and network blocking statistics were also reported. For this game, the operators' revenues were solely generated from VLL sales. The monitoring window (figure 6) displayed the total revenues (profit/lost) generated by each network operator. We associated the link QoS level in terms of Gold, Silver and Bronze by referring to link's reservation load of 0-50%, 50-75% and 75-100% respectively. This is different from per-user's QoS as each user's VLL was already assigned an amount of guaranteed BW. Therefore link reservation load is the aggregation of all the independent VLLs' guaranteed BW.



Fig. 5. Network Operator Console



Fig. 6. Monitoring Window

The users' service request arrival rates were generated according to different Poisson arrival distributions. Table 3 shows the characteristics set for the three basic classes of VLL subscribers.

Table 3. User Characteristics

User Classes	Mean Request Arrival Rate (per hour)	Mean b (Guaranteed BW unit)	T (Session in mins)	Example Applications
1	70	2	3-10	VOIP/Live Music
2	15	30	10-60	VoD/Conferencing
3	28	20	1-10	File Transfer

Figure 7 shows the accumulated bandwidth request (offered load) profile with this specification. The profile shows that Class 1 users (e.g. audio) requested for relatively

low BW but arrived very frequently. Class 2 users (e.g. VoD) required high bandwidth VLLs over long periods of time and they were the dominant traffic contributors. Class 3 users requests produced bursty type of traffic profiles that suited applications such as FTP.

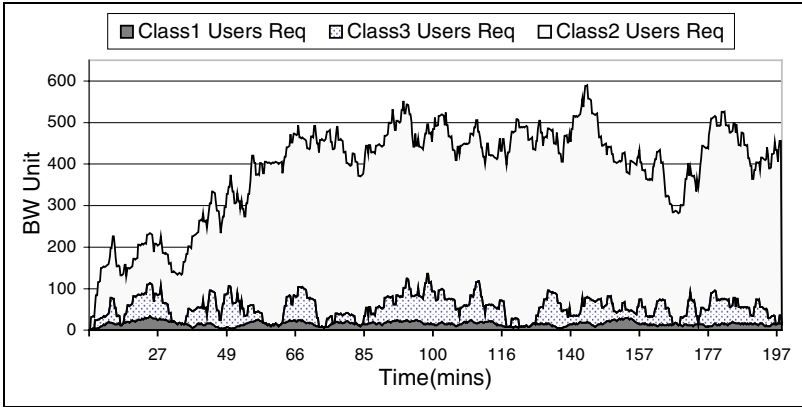


Fig. 7. Multimedia Users Bandwidth Request Profile (Accumulated Bandwidth)

8 Result of Dynamic Pricing

The results from one of the sessions were collected and analyzed in figures 8 through figure 11. Figure 8 shows the pricing history of the three acting network operators.

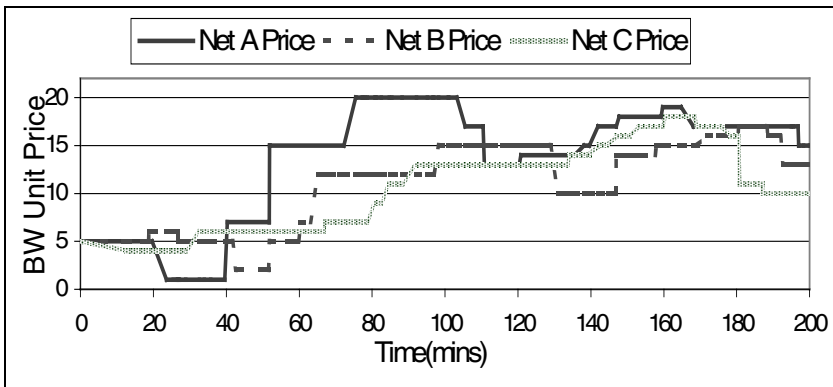


Fig. 8. Bandwidth Price bidding vs. Time (minutes)

Here, the three network operators were trying to maximise the revenues by setting different BW prices dynamically. At $t > 20$ mins, network A lowered its bandwidth

price to 1 and caused a sharp increase in load over the measured link (see figure 9). At $t > 50$ mins, network A increased its price dramatically, and soon became much more expensive than the others. As a result, a significant drop in traffic was observed after $t > 75$ mins. This was most likely due to class 2 subscribers leaving the network. At another time instant, $t > 110$ mins when network B's price remained constant, network A beat network B in price and attracted traffics to its network.

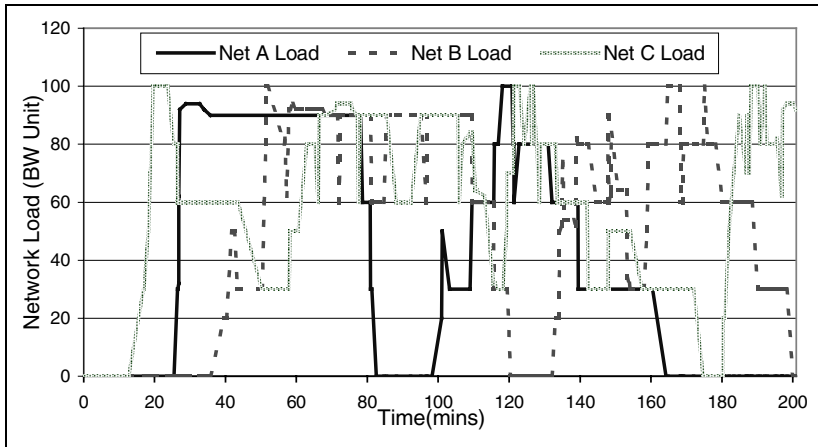


Fig. 9. Network load measured at one of the links(link2) vs. Time

Notice that at time $t > 100$ mins, when network A was still the most costly network, traffic was coming into the network because the other two networks were saturated and were unable to provide the required bandwidth. This earned network A a sharp rise in revenue (see figure 10) and a short lead in revenue race as a result of users buying high cost connections at bulk volume. In figure 11, we can observe a close relationship between load and price. In this case, it seemed that the cheapest provider earned the most revenues. However if we look at figure 10, network B was just marginally lost to network A. This means network B can actually bid a higher average price and win the game because network C had a significantly higher average bandwidth price compared to network B. Nevertheless, this strategy is only applicable for this scenario.

Figure 12 shows the importance of setting the right price at the right time. Network A made a loss at 20-40 minutes interval due to low offer price. However much of the loss was compensated at 100-120 interval due to bulk bandwidth sales at high price. On the whole, network C managed to maintain a good stream of revenue generated throughout the session. It is also observed from this simple experiment that it is more profitable to get revenue from high BW, long session stream such as video conferencing.

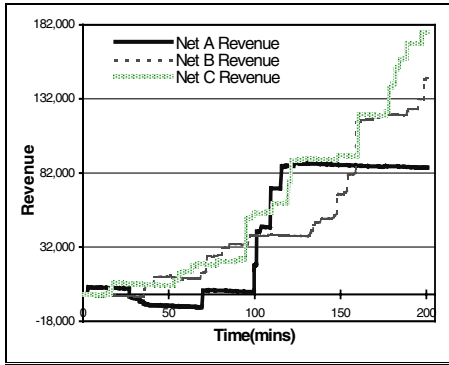


Fig. 10. Revenue Generated vs. Time

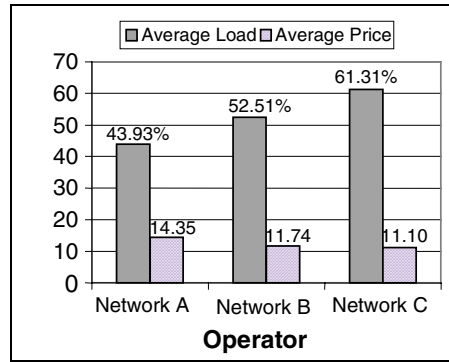


Fig. 11. Average Load Vs. Price

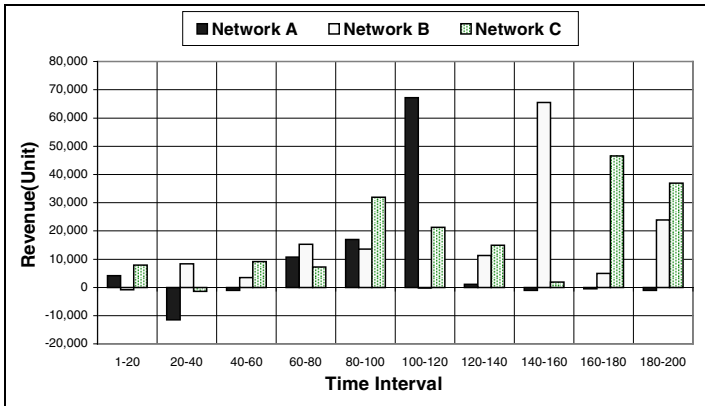


Fig. 12. Revenue Generated Per Time Interval vs. Time

9 Conclusions and Future Work

In this paper we demonstrated a futuristic scenario where agents can be employed to support dynamic SLA specification for network services. Various aspects of SLA especially regarding the guaranteed BW and pricing were investigated. A demonstrator consisting of real-time agents and simulated networks was built to support our analysis. The studies show that dynamic SLA negotiation introduces many innovative ways on how the future network services can be provisioned and priced. At this stage, our agents only exercise the simplest form of negotiation such as resource query and pricing comparison. Hence, BW consumption during the negotiation process is negligible whether client-server based agents or mobile agents were used. Nevertheless it is

believed that when agents have acquired a higher level of negotiation capabilities and intelligence, this issue must be further addressed.

During the demonstration, we allowed the audience to manually set the BW price. In the future, operator agent can potentially take over this task where it can set the right price at the right time based on a more sophisticated pricing model or mechanism i.e. different pricing schemes for different service classes. Advance reservation option and various tolerance parameters can be incorporated as part of the SLA metrics. Links segregation can be also implemented to facilitate lower granularity BW resource control such as preventing high BW services e.g. VoD or FTP flows to starve all the bandwidth.

Acknowledgement. The authors gratefully acknowledge Fujitsu Telecommunications Europe Ltd for funding this project. Special thanks to Professor Collin South, Dr. Dominic Greenwood, Dr. Keith Jones and Dr. Desmond Maguire for all their invaluable feedbacks and support.

References

1. S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, and W. Weiss. "An Architecture for Differentiated Services", RFC 2475, Network Working Group, IETF, December 1998. (www.ietf.org)
2. E. C. Rosen, A. Viswanathan and R. Callon. "Multiprotocol Label Switching Architecture", Internet Draft, Network Working Group, IETF, July 2000. (www.ietf.org)
3. R. Braden, L. Zhang, S. Berson, S. Herzog, and S. Jamin. "Resource ReSerVation Protocol (RSVP) -- Version 1 Functional Specification", RFC 2205, Network Working Group, IETF, September 1997. (www.ietf.org)
4. Fred Engel, Executive Vice President and CTO of Concord Communications. "Grasping the ASP means service level shakedown", Communications News, Aug 2000, pp. 19-20.
5. Roland Courtney, founder of ViewGate Networks, "IP QoS: Tracking the different levels", Telecommunications Magazine International Edition, January 2001, pp. 58-60. (www.telecommagazine.com)
6. Roch Guerin et.al., "Quality-of-Service in the Internet", Guest Editorial, IEEE Journal on Selected Areas in Communications, December 2000, Vo. 18, No. 12, pp 2485-2487.
7. G. Fankhauser, D. Schweikert, and B. Plattner, "Service Level Agreement Trading for the Differentiated Services Architecture", Swiss Federal Institute of Technology, Computer Engineering and Networks Lab, Technical Report No. 59. November 1999.
8. L.K. Lim, J.Gao, T.S. Eugene Ng, P. Chandra, "Customizable Virtual Private Network Service with QoS", Computer Networks Journal, Elsevier Science, Special Issue on "Overlay Networks", to appear in 2001.
9. N. Agoulmine, F. Nait-Abdesselam and A. Serhrouchni, "QoS Management of Multimedia Services Based On Active Agent Architecture", Special Issue: QoS Management in Wired & Wireless Multimedia Communication Networks, ICON, Baltzer Science Publishers, Vol 2/2-4, ISSN 1385 9501, Jan 2000.

10. M. Karsen, N. Beries, L. Wolf, and R. Steinmetz, "A Policy-Based Service Specification for Resource Reservation in Advance", Proceedings of the International Conference on Computer Communicatons (ICCC'99), September 1999, pp. 82-88.
11. O. Schelen and S. Pink, "Resource sharing in advance reservation agents", Journal of High Speed Networks: Special issue on Multimedia Networking, vol 7, no. 3-4, pp. 213-28, 1998.
12. X. Wang, , H. Schulzrinne, "RNAP: A Resource Negotiation and Pricing Protocol", Proc. International Workshop on Network and Operating System Support for Digital Audio and Video (NOSSDAV'99), New Jersey, Jun. 1999.
13. M.A.Gibney, N.R.Jennings, N.J. Vriend and J.M.Griffiths, "Market-Based Call Routing in Telecommunications Networks using Adaptive Pricing and Real Bidding", IATA'99, Stockholm, Sweden, 1999.
14. Dinesh Verma, "Supporting Service Level Agreements on IP Networks", Macmillan Technical Publishing, ISBN: 1-57870-146-5.
15. David Chieng, Alan Marshall, Ivan Ho, Gerald Parr, "A Mobile Agent Brokering Environment for The Future Open Network Marketplace", Seventh International Conference On Intelligence in Services and Networks (IS&N2000), Athen, 23-25 February 2000, pp 3-15. (Springer-Verlag LNCS Series, Vol. 1774, ISBN 3-540-67152-8).
16. www.objectspace.com/products/voyager
17. William A. Shay, "Understanding Data Communications & Networks" 2nd Ed, International Thomson Publishing Inc., ISBN 0-534-95054-X.
18. A. Marshall, S. A. Hussain, D. Chieng, and Q. Gu, "Dynamic Network Adaptation Techniques In An Open Network Environment", International Conference on IT and Communications at the dawn of the New Millennium (AIT2000), August 2000, Bangkok, Thailand
19. R. Rajan, D.Verma, S.Kamat, E.Felstaine, S.Herzog, "A Policy Framework for Integrated and Differentiated Service in the Internet", Technology Articles, Allot Communications Inc., Aug 15, 2000. (www.allot.com/technology/PBN.htm)
20. BONEs DESIGNERâ Ver 4.01, Alta GroupTM of Cadence Design Systems, Inc.

WEBARM: Mobile Code Based Agent for Web Application Response Measurement – Software Implementations and Analysis

Joseph V. Elarde and Gregory B. Brewster

DePaul University
jelarde@cs.depaul.edu

Abstract. WEBARM enables end-to-end response time measurement for Web applications through Web page instrumentation. WEBARM-agents are a form of mobile code capable of moving with the Web application to monitor response time. WEBARM includes the software components responsible for interception and processing of the WEBARM API calls including support for response time measurement, collection, and communication. This work provides an assessment of WEBARM instrumentation impact on application response times as well as insights into the design issues involved. We first provide a general overview of alternative approaches to Web response time measurement. We then consider alternative WEBARM agent software designs, focusing on techniques to store timers across Web page references and communicate logged data to a server. Next, we present measurement results for the implementation alternatives to provide insight into the overhead involved with instrumentation and the developed designs. Finally, we present conclusions and a description of future work.

1 Introduction

The dramatic expansion in use of the Internet represents one of the major technological revolutions of recent history [2]. As organizations deploy Web applications over the Internet, we have entered an era where direct customer contact with an organization's Web applications is commonplace. The quality of this experience in terms of functionality, performance, and availability will ultimately affect the customer's view of the organization. Thus, maintaining a high performance Web application is directly related to customer satisfaction [1] and ultimately an organization's success.

The solution for any performance issue begins with a program of objective measurements - knowing what service is delivered can lead to the necessary root cause identification and corrective action implementation processes. Not knowing the service delivered to the end-customer is dangerous - equivalent to not knowing an organization's financial results. So given that service (response time) measurement is important - how should a comprehensive Web response-time measurement program

be implemented? The alternatives we consider in this paper are server-based estimation, site monitoring, and client agent based measurement.

- **Server-based measurement:** Web Server logs or network monitors local to the Web server are used to estimate end-to-end response time delivered to the end-customer (Passive Monitoring).
- **Site monitoring:** Utilizes strategically deployed workstations called "probes" to execute scripted transactions at various locations within the network. The probe machines capture and analyze the scripted transaction responses to estimate overall application availability and performance (Active Monitoring).
- **Client agent:** Apply client based agent technology to monitor client Web browser and system activity to derive end-to-end response time or utilize application instrumentation to measure response time.

Knowing that a response time problem exists is an important first step in implementing a corrective action. For Web applications some examples of possible corrective actions are as follows:

- An installation may decide to relocate parts of the Web server equipment (Proxy servers) nearer to problem areas to reduce network latency [3].
- Capacity and/or technology upgrades along the transaction path may be driven based upon the service delivered.
- The installation may decide to develop adaptive applications [4] capable of using the recorded response time information to vary resource demands and consequently mitigate the impact of network or server latency.

But for any or all of these and other corrective actions to occur, it must begin with measurement. WEBARM represents an easy to implement, minimal overhead, and reduced maintenance cost approach to Web application response time measurement. It provides a set of instrumentation APIs, similar in nature to the ARM Version 1.0 APIs, but without the complexity. Deployment, often a barrier to agent implementation, is also addressed through WEBARM's mobile code based agent architecture.

This paper is organized as follows. Section 2 provides an overview of various Web response time monitoring techniques commonly used to capture or estimate response time, and introduces our focus technique, the Client Based Mobile code agent technique (WEBARM) which is evaluated in the remainder of this paper. In section 3 we provide an in depth description of the WEBARM technique used to capture and record Web application response time. Section 4 presents the results of measurement experiments designed to assess the overhead involved with the response time measurement technique presented in section 3. And finally, a summary and conclusion are offered in section 5.

2 Web Application Response Time Measurement

In this section, we examine commonly used approaches to Web application response time measurement. Considering an application's life cycle stage, measuring response time under various loads during development is an essential part of application per-

formance engineering. However, equally if not more important and sometimes neglected is measurement of applications during the production stage. It is at this point where application users including customers are directly affected by application performance. It is at this point where knowing and adapting to changing infrastructure conditions begins with knowing the actual service delivered. We have identified three Web Application Response Measurement techniques: Server based measurement, Site Monitoring, and Client Agent based measurement. Each of the presented techniques measures overall response time and the components thereof in varying degrees of completeness and accuracy.

2.1 Server Based Measurement Techniques

A common approach to Web application response time measurement is to use the Web Server logs to measure Web page references; bytes transferred, and server time [5]. However, the measurements are server focused and consequently do not capture the entire end-to-end response time. Server logs typically include per page or object measurements, so where multiple browser objects are requested server response time estimations must consider the first to last object timings. Further, only objects not cached at the client browser are included in the server logs complicating the estimation of response time.

If the server application is instrumented, server focused measurements can be enhanced with a *ping* or *traceroute* measurement to estimate the network latency. By applying a fixed client time the overall end-to-end transaction response time is estimated. Alternatively, where server measurement is not possible, network "Sniffers" or monitors may be used to measure Web server response time calculating the difference between connect and disconnect times or page service time. Web server measurements captured by operating system facilities and/or by network monitoring collect a wealth of information regarding the performance, availability, and reliability of the Web server and hosted applications. Useful measurements, but Server Based Measurements do not provide a true assessment of end-to-end response time.

2.2 Site Monitoring Techniques

As commercial enterprises become more dependent upon the Internet, external site monitoring services have emerged as an approach to Web application response time and availability monitoring. This involves the strategic placement of a set of application monitoring probes designed to periodically execute a scripted reference to a Web site's home page or, in a more involved procedure, invoke a series of Web site transactions[6][7]. The script timings are recorded for response time analysis, availability trending, and problem alerting. Some implementations involve placement of probes connected to specific locations throughout the Internet to enable assessment of regional variations. This approach facilitates the understanding of how accessible /available the specific Web site and how responsive the site is likely to be. This technique is also useful for Intranet sites within and organization's firewall. However, it does not capture the true end customer experience. End user PC processor capacity, memory, hardware/software platform, and modem access speeds vary widely and may

play a significant role in the end user performance. Regarding response time, what is measured from a probes perspective is the probes view of the Web site or application and this may or may not be representative of actual end-user equipment and experience. While clearly a useful and valuable approach to performance and availability measurement, site monitoring is not a complete solution since it does not capture the end-customer's experience.

2.3 Client Agent Monitoring Techniques

This technique involves the application of client agent technology to locally monitor Web browser activity and calculate response time. Two forms are prevalent, installed agents [8] and mobile code/agents [9]. Installed agents exist as services on the end-user's equipment. The client agent monitors Web browser activity for transaction start/stop events and then logs the transaction response time information either locally at the client or to an external server. The difficulty is that an agent must be installed and active to monitor the Web application. Since the Web client machine is often beyond the control of the organization, the ability and justification involved in deploying agents on the customer's machine represents a significant implementation challenge. Moreover, the Client agent must be parameterized to intercept the transaction start and stop events; application changes may disturb the agent's understanding of transaction events and thus, it is likely the agent must be retrained or parameterization for each application change.

Mobile client agent code moves with the Web application to the target machine [9]. The mobile agent code is directly integrated into the application's Web page. Since the agent code is integrated with the application, interception of transaction start and stop events is accomplished through instrumentation. In addition, since a "light-weight" end-customer presence is desired, recorded response time information is logged directly to an external server. The advantages are that no formal agent deployment is needed and consequently remote maintenance costs are reduced. The disadvantage is that the mobile code must be transmitted at least once to be cached locally at the end-customer's workstation, and depending how frequently the local browser cache is purged impacts how frequently this overhead is incurred.

3 WEBARM – Mobile Code Based Web Response Time Measurement

In this section, we provide an in-depth examination of the client based mobile code agent (WEBARM) technique introduced in section 2. As indicated, the client based agent alternatives are to deploy an agent service/daemon to capture transaction start and stop events or integrate the response time measurement instrumentation into the application's Web pages to capture transaction response time.

3.1 WEBARM Relationship to the ARM API Standard

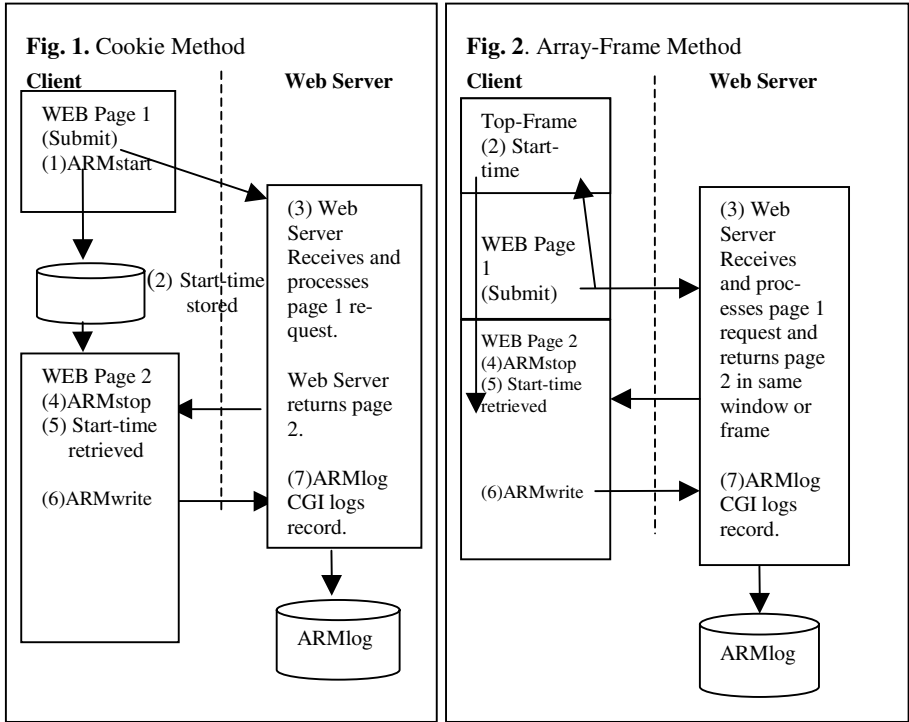
For instrumentation we use a set of Application Programming Interface (API) functions that are a simplified adaptation of the ARM Application Response Measurement

API [10]. We define three API functions *ARMstart*, *ARMstop*, and *ARMwrite*. The *ARMstart* API function records the start or beginning of a Web transaction by storing the start time in a frame based array variable or a cookie. The *ARMstart* API, as we present it here, combines the ARM version 1.0 *arm_init*, *arm_getid*, and *arm_start* API calls into a single function call [11]. *ARMstart* is typically invoked in reaction to a transaction event such as a "mouse click" or "function key". The *ARMstop* API function records the transaction stop or end-time and calculates the response time. *ARMstop* emulates the *arm_stop* ARM API to the extent that it records the stop time and computes the response time; however, the response time logging process is built as a separate function to allow the Web application the freedom to select when to log the response time information. *ARMstop* is invoked at a point in a Web page representing a transaction end point. The last WEBARM API defined is *ARMwrite*. While there is no direct ARM equivalent, the function is normally embedded in the *arm_stop* API. *ARMwrite* directs the recorded transaction information to a logging server and should be coded at a point in the Web page processing that overlaps the log processing with user think time.

3.2 Timer Storage Alternatives

To facilitate response time determination, an important consideration is the storage of timer information between the *ARMstart* and *ARMstop* API calls. Two alternative techniques are presented: the Cookie method and the Array-Frame method.

- **The Cookie Method:** Client Web browser "cookies" are typically used to maintain state information locally at the client. For example, a Web application may collect some information in a session to be used in subsequent sessions. To avoid re-requesting the information from the end user, the entered data is stored in temporary storage, referred to as a "cookie", and later retrieved as needed. Rather than store end-user information, the WEBARM Cookie method uses cookies to store the transaction start time as part of the *ARMstart* API (Figure 1). *ARMstart* is normally called in response to a "submit button" or other event designating the start of a transaction. The *ARMstop* API retrieves the transaction start time from a specific cookie, calculates and returns the response time. The response time is then logged to a server using the *ARMwrite* API. The advantages of this method exist predominately in that a cookie can store information across browser sessions and instances. The main disadvantage is the slightly higher overhead involved in comparison to the Array-Frame technique.
- **The Array-Frame Method:** As opposed to using cookies to store the transaction start time, the Array-Frame method utilizes a "top-frame" JavaScript array variable to store the transaction start time and identification information. Similar to the cookie technique, the *ARMstart* API saves the transaction start time, but rather than in a "cookie" an array variable is used (Figure 2). *ARMstop* retrieves the start time from the array, calculates, and returns the response time. The *ARMwrite* API is then invoked to log the ARM data to a server. The advantage is the lower overhead and simplicity of implementation. The main disadvantage is that access to the stored time is not persisted across sessions and a separate frame is needed to store the start time value.



3.3 The WEBARM Instrumented Web Page

The WEBARM APIs and supporting functions are included and referenced in the Web page by the programmer during development, as a set of embedded functions or as a separate object on the Web page. However, using a separate object allows the WEBARM functions to be cached locally at the client, and subsequent references can avoid the network transfer time for each reference from the cache.

The HTML statements below illustrate the separate object approach to instrumentation, in a basic two-page Web application. Page 1 (Figure 3) is an example Web page including a basic HTML form and the associated "submit" button to initiate the form processing. When activated, "submit" generates the onlick event and invokes the *ARMstart* function to store the current time in a timer identified as *ARM_Timer*. Next, our sample application opens Page2.html (Figure 4) in the same window to complete the transaction.

The JavaScript object *ARMn.js* contains the WEBARM APIs, supporting functions, and the in-line code to invoke the *ARMstop/ARMwrite* APIs to compute, log, and display the response time. Note each page must reference the same *ARMn.js* object to facilitate browser caching, and "n" represents the method employed, for example, *ARMCookie.js* or *ARMArray.js*.

Fig. 3. Sample Web Page 1

```

Filename: Page1.html

<html><head>
<title>Test Web App</title>

<script lan-
guage="Javascript1.2"
SRC="/ARMn.js"> </script>

</head><body>

<FORM>
<INPUT TYPE=button
VALUE="Submit"
on-
Click="ARMstart('ARM_Timer');
window.name='test';
win-
dow.open('Page2.html','test')"
>
</FORM>

<p> The submit button records
the start time and opens page2
in the same window.
</body>
</html>

```

Fig. 4. Sample Web Page 2

```

Filename: Page2.html

<html><head>
<title>Test Web App</title>

<script lan-
guage="Javascript1.2"
SRC="/ARMn.js"> </script>

</head><body>

<p> Web page returned, in-
cluding response time.
</body></html>

```

Fig. 5. Frames Example

```

Filename: Page0.html

<HTML><HEAD>
<TITLE>ARM Test Frame
</TITLE></HEAD>

<FRAMESET ROWS="1,*">
  <FRAME SRC="dummy.html"
NAME="FR1">
  <FRAMESET >
    <FRAME SRC="Page1.html"
NAME="FR2">
  </FRAMESET>
</FRAMESET>
</HTML>

```

The Array-Frame technique requires the use of frames to persist the recorded timers across Web page references. Without the "top frame" storage area, the timer information is cleared when a subsequent HTML document is loaded. Figure 5 provides an example page using frames to reference the sample Web application. Note the frames page (Page0.html) must be loaded first which in turn references and loads Page1.html.

3.4 WEBARM Transaction Logging Alternatives

Once the response time data has been collected and communicated to the Web server, the information must be logged to disk to facilitate analysis and reporting. The logging alternatives considered for WEBARM are as follows:

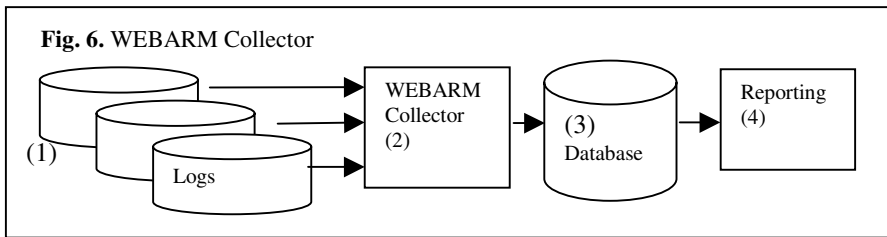
1. **Web server logs:** Web servers typically log information for each server access, including HTML page references, image loads, and script executions. In the case of (IIS) Internet Information Server (Microsoft's Web Server), the invocation of

the CGI logging script (ARMweblog), called during the *ARMwrite* API, is logged including all the parameters used to invoke the CGI – notably response time.

2. **CGI Script logging:** Since the *ARMwrite* API invokes a script (ARMweblog) to communicate the response time information to the Web server, the passed CGI parameters are then logged to disk under the control of the ARM Web logging script. We evaluate the processing impact associated with alternatives 1 and 2 in section 4.3.

3.4.1 WEBARM Log Collection

For many mission critical Web application implementations, parallel scalable Web clusters[12] are configured to enhance availability, performance, and scalability. However, this creates a design challenge for WEBARM, the separate Web server logs must be merged to develop a complete assessment of the application's performance.



Consequently, we developed a separate asynchronous process, known as the WEBARM Collector (Figure 6), to periodically collect the logged data from the clustered servers, into a consolidated database for analysis and reporting purposes. Moreover, the WEBARM Collector enables the periodic issuance of performance threshold alerts based upon the collected response time information.

Selecting a flexible log format and database architecture is an important related design consideration; [13] [14] describe some resent log format research. WEBARM utilizes a keyword based log communication format and a variable length delimited log record when stored in the WEBARM database.

3.5 WEBARM JavaScript Implementation

The mobile agent/code implementation technique described herein utilizes a set of JavaScript functions to implement the ARM APIs and related supporting subroutines packaged in a browser cacheable object. For the Cookie method we have adapted and renamed some commonly available cookie management functions [15]: *setCookie* and *getCookie*, to support timer storage and access (*setTimer* and *getTimer*). All the functions are contained, in the Web page included *ARMCookie.js/ARMArray.js* object (see section 3.1) depending upon which alternative timer storage method is employed.

The "*setTimer*" function, as its name implies, is used to set a named cookie or array element to a specific value. For WEBARM, we store the Web application's transaction start time. "*setTimer*" requires two input parameters: timer-name, and the as-

sociated timer value. The other support function -"*getTimer*" is used to access the stored timer value i.e., the transaction start time after the transaction completes processing or when needed by the *ARMstop* API. "*getTimer*" requires that the timer-name be passed as an input argument.

The individual API functions, support subroutines, and in-line code included are presented below:

1. ***ARMstart***: This API (Figure 7.) stores the current time in a specifically named timer using the *setTimer* function. By default the name *ARM_Timer* is used; however, it is possible to use any unique name.
2. ***ARMstop***: This API computes the transaction response time by, obtaining the current time, accessing the stored cookie or array start time value, and then computing the response time. Note the *setTimer(timerName, 0)* effectively deletes the timer. It was discovered, when using the Cookie method, that by removing a cookie through the cookie delete function increased the overhead involved. Cookie functions that employ the expiration date processing "expires=" experience significantly higher overhead than functions avoiding this parameter (see section 4.1.1).

<p>Fig. 7. ARMstart</p> <pre>function ARMstart(timerName) { setTimer (timerName, new Date ().getTime());}</pre>	<p>Fig. 9. ARMwrite</p> <pre>function ARMwrite(value){ log_url="http://" + document.domain + "/scripts/armweblog.exe?"; cgi= new Image(); cgi.src=log_url+" "+escape(value); return true;}</pre>
<p>Fig. 8. ARMstop</p> <pre>function ARMstop(timerName) { endtime = new Date().getTime(); starttime=getTimer(timerName) If (starttime==0 endtime == starttime){ resp = 0} else{ resp = (endtime - starttime)/1000} setTimer (timerName, 0); return resp;}</pre>	

3. ***ARMwrite***: This API logs the measure transaction response time to the initiating Web server using a technique that passes the response time information along with an Image object CGI request. This technique eliminates the requirement to have a signed script or a Java Applet to return log data to the Web server. Since the response time information is normally logged in the Web server log, the CGI program can be designed to simply return a null image pointer, or for more sophisticated implementations special processing can be performed within the CGI to store additional measurements. For example, the IP address of the sending workstation or client could be translated to the corresponding machine name or a network *traceroute* command could be executed to record network round trip times to assess network conditions near the time of the transaction execution.

Fig. 10. getTimer Cookie Method

```
function getTimer (name) {
  var result=0;
  var myCookie= " " + document.cookie + ";";
  var searchName = " " + name + "=";
  var startOf-
  Cookie=myCookie.indexOf(searchName);
  var endOfCookie;
  if (startOfCookie != -1){
    startOfCookie += searchName.length; endOf-
    Cookie=myCookie.indexOf(";",startOfCookie);
    result=unescape(myCookie.substring(
    startOfCookie, endOfCookie)); }
  return result;}

```

Fig. 11. getTimer Array Method

```
function getTimer (name) {
  return(getTimers()[timerName]);}

function getTimers() {
  var result = top.ARMTimers ;
  if (result == null) {
    top.ARMTimers = new Array() ;
    result = top.ARMTimers ; }
  return result ;}

```

4. **getTimer:** The *getTimer* function is used to access and return the value associated with a specifically named timer. The *ARMstop* API calls *getTimer* to access the necessary start time to compute the transaction response time. The cookie and array-frame methods are illustrated in figure(s) 10 and 11 respectively.
5. **setTimer:** *setTimer* stores a value in a specifically named cookie. Three parameters are used: the cookie name, the value to set, and an expiration date. As indicated above, we avoid the use of expiration date to delete a cookie to reduce overhead

Fig. 12. setTimer Cookie Method

```
function setTimer (name, value, expires) {
  if (!expires) {document.cookie = name + "=" +
  escape (value) + "; path=/";}
  else{document.cookie = name + "=" + escape
  (value) + ";
  expires=" + expires.toGMTString() + ";
  path=/";} }

```

Fig. 13. setTimer Array Method

```
function setTimer (name, value, expires) {
  getTimers()[timerName] = value;}

```

Fig. 14. In-line Code

```
resp=ARMstop("ARM_Timer");
if (resp>0){
  document.writeln("Response Time:" +
  resp);
  ARMwrite("resp=" + resp + ";" +
  "tran=" + document.title + ";" +
  "etime=" + (new Date()).getTime() + ";");}

```

6. **In-line code:** Optionally this program code can be placed with in the *ARMn.js* object (as shown) or maintained separately in the Web page. This code is placed within the Web page to reflect the end of the Web transaction, normally at the top of the document. First, *ARMstop* is called to compute the last transactions response time. Second, if the response time returned is valid the *ARMwrite* function is called to log the information at the Web server. Note the format of the logged information consists of a string of **KEYWORD[x]=value;** pairs - similar to the Universal Logging Message format described in [13]. The [x] optionally defines

a unique index number to enable multiple transactions to be logged in a single communication.

4 WEBARM Instrumentation Performance Analysis

To assess the impact of the WEBARM instrumentation, a series of benchmark experiments were conducted on the following platform configurations to evaluate the processing delays an application may experience due to instrumentation.

1. Intel Pentium II 266MHz, NT 4.0 SP6, Internet Explorer 4.0
2. Intel Pentium II 266MHz, NT 4.0 SP6, Internet Explorer 4.0, via dial-up 33.6bps modem.
3. Intel Pentium II 350MHz, NT 4.0 SP6, Internet Explorer 4.0
4. Intel Pentium II 450MHz, NT 4.0 SP6, Internet Explorer 4.0
5. Intel Pentium II 450MHz, NT 4.0 SP6, Internet Explorer 5.0

With the exception of configuration 2 above, a 100Mbps Ethernet LAN connection to the Web server was used. The Web server configuration consisted of a dual 466MHz Celeron processor configuration running NT 4.0 and IIS 4.0 at SP6. Configurations 3-5 were connected to a dual 450MHz Intel Xeon processor configuration running NT 4.0 IIS 4.0 SP4 via a 100Mbps Ethernet LAN. Performance measurements and analysis are presented for the "Cookie" and "Array-Frame" techniques (described in section 3) in 4.1 and 4.2 respectively. And finally, we present a volume stress test benchmark of the "ARMweblog" program in 4.3 to assess logging impact and efficiency.

4.1 Performance Analysis of the Cookie Technique

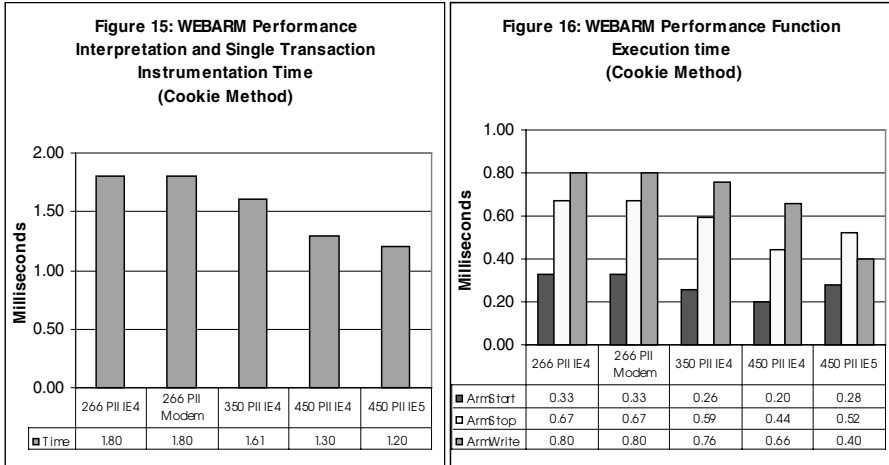
To measure overhead of the Cookie technique a simple form based, test Web application was developed and instrumented. Measurements include the overall impact on a set of instrumentation API calls to assess impact per transaction, individual function call timings, and the recorded test application response times.

4.1.1 Cookie Technique Individual Transaction Overhead Assessment

The objective of this experiment is to measure a complete set of instrumentation calls for a single transaction including the JavaScript interpretation time. Assuming the script (ARMn.js) is loaded from the browser cache, this benchmark assesses the expected impact each instrumented page should experience. Figure 15 below illustrates the various measurements observed across the configurations evaluated.

The 450Mhz Pentium II processor NT 4.0 IE5 configuration recorded the lowest instrumentation impact at 1.20ms and the 266 MHz Pentium II the highest measured impact. One result in particular is interesting. The initial *ARMstop* API included a function to delete the stored cookie by setting the expiration date to a specific past date. This is a frequently used approach to cookie cleanup; however, initially a significant 61% greater path-length overhead was observed for the 450MHz IE4 configuration in comparison to the 450MHz Pentium II running IE5. By removing the "ex-

pires=" cookie parameter and changing the *ARMstop* function to recognize a zero as a deleted cookie, path-length overhead was reduced to - a 7.6% differential between configurations 4 and 5. This suggests that performance improvements have been implemented in IE5 from the IE4 browser.



4.1.2 Cookie Technique Individual API Measurements

This benchmark scenario involves execution of each API (*ARMstart*, *ARMstop*) function 1000 times for 10 iterations (figure 16). As indicated, the functions measured include the *ARMstart* function, which stores the transaction start time in a cookie, and the *ARMstop* function, which reads the cookie and calculates the response time. The *ARMwrite* sub-function measures the path-length cost of logging the recorded response time to the Web server. This function was also invoked 1000 times and the mean path-length computed.

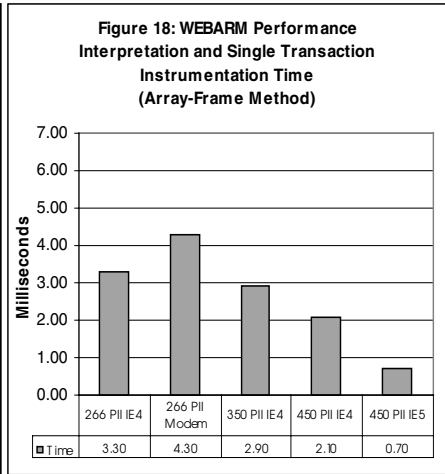
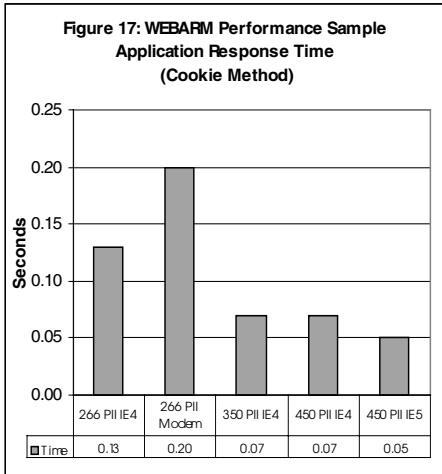
The Pentium II 450MHz IE4 configuration exhibits the best *ARMstart* and *ARMstop* performance while the PII 450MHz IE5 recorded the most favorable *ARMwrite* timing. The 40% reduction in path-length time for the *ARMwrite* function from the PII 450MHz IE4 to the IE5 configuration is also significant. Note that the reduction in path-length between IE4 and IE5 occurs predominantly in the *ARMwrite* function.

4.1.3 Cookie Technique Test Application Response Time

The test Web application used to evaluate the performance impact of the Cookie technique consists of two Web pages. The first page, page 1, contains a form with a button control to initiate the response time measurement (*ARMstart* function) and then open a second page in the same window. The second page, page 2, calls the *ARMstop* function and displays the response time and function path-length timings, demonstrating the ability to persist timer information across pages. Figure 17 examines the relative performance of all the configurations executing the test Web application.

Results are generally consistent with the single transaction impact, except that the 266MHz modem configuration is showing at least a 53% increase in response over the 266MHz LAN configuration.

An important design consideration is that the ARM program logic was configured as a separate script object (ARMn.js) in the test Web application. The overhead to access the script initially (non-cached) is .25 seconds for the LAN based configurations and .7 seconds for the dial-up 33.6bps modem. The alternative to the "separate script entity" is to integrate the (ARMn.js) script into the application's Web page. However, while the initial Web-page response time is reduced, subsequent arm script references on different pages do not benefit from (ARMn.js) script caching. In addition, if the script is integrated into the application page, script changes will require all pages with the integrated script to be updated; whereas, if the separate entity (ARMn.js) is changed no application page updates are required. Next, we review the performance of the "Array-Frame" technique.



4.2 Performance Analysis of the Array-Frame Technique

To measure overhead of the "Array-Frame technique" a "frames" based, test Web application was developed and instrumented. As with the Cookie technique analysis, measurements include the overall impact on a set of instrumentation API calls to assess impact per transaction, individual function call timings, and the recorded test application response times.

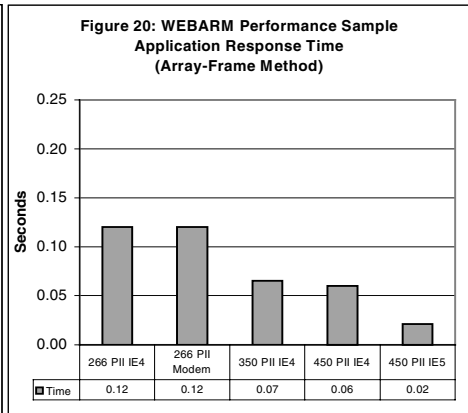
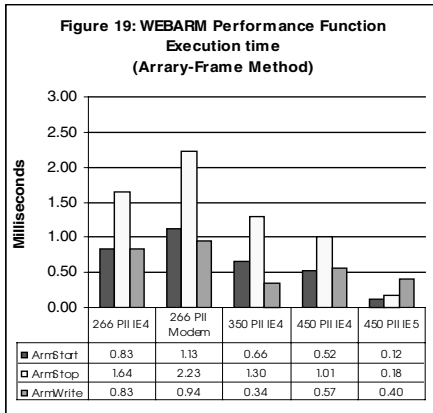
4.2.1 Array-Frame Technique Individual Transaction Overhead Assessment

As with 4.1.1, this measurement describes the expected impact (using the Array-Frame technique) each instrumented page should experience given that the script is loaded from the local cache. Figure 18 illustrates the various measurements observed across the subject configurations. The 450MHz Pentium II IE5 configuration recorded

the lowest instrumentation impact at (.7ms) and the 266MHz Pentium II the highest measured impact (4.3 ms). The main difference between the Cookie and Array-Frame techniques is the location where the measurement timings are stored, i.e., cookie versus memory array.

4.2.2 Array-Frame Technique Individual API Measurements

Similar to the experiment described in 4.1.2, this benchmark scenario (Figure 19) involves execution of each API (*ARMstart*, *ARMstop*) function 1000 times for 10 iterations. As indicated, the functions measured include the *ARMstart* function, which stores the transaction start time in an array variable, and the *ARMstop* function, which access the array variable and calculates the response time. The *ARMwrite* sub-function measures the path-length cost of logging the recorded response time to the Web server. Although the Array-Frame technique supports multiple concurrent timer measurements, a single response was used for our sample application.



4.2.3 Array-Frame Technique Test Application Response Time

The Array-Frame test application employs a Web page consisting of three frames:

1. Frame 1: The Top frame is used to store the ARM measurement array.
2. Frame 2: A Web page is loaded in frame 1 that contains a form, similar to the "cookie" test Web application, with a submit button. The "submit" button invokes the *ARMstart* function and opens an instrumented page in frame 3.
3. Frame 3: When the Web page is loaded a call is made to the *ARMstop* and *ARMwrite* functions to display and record the measured response time, function path-length timings, and demonstrate the ability to persist timer information across frames.

Figure 20 examines the relative performance of all the configurations executing the test Web application. As with the Cookie technique, the 450MHz Pentium II configuration exhibited the superior performance with a .02 seconds response time.

Comparing the Array-Frame to the Cookie technique, in general, response times are equal or better (for the Array-Frame technique), ranging between .0 and .12 seconds difference. The Cookie technique path-length timings are consistently less than the Array-Frame for configurations 1 to 4, and the Array-Frame shows the improvements for the Pentium II 450MHz configuration.

4.3 Web Logging Capacity Requirements

To assess WEBARM logging capacity requirements, we developed a benchmark program to repeatedly execute the "ARMweblog" CGI script at various rates while recording the corresponding Web server processor utilization. The results of the "ARMweblog" benchmark for the dual 466MHz Celeron processor configuration indicate that in the range of 1 to 25 logs per second, processor usage increases linearly from 1.9% to 52%. The alternative, IIS logging technique, extracts the response time information directly from the IIS (Web Server) logs. The results obtained indicate that by increasing the transaction rate from 1 to 25 logs per second processor impact varied from .1% to 3.5%. Note the significant difference in overhead required to spawn and execute the CGI process in comparison to basic Web server logging. Web server logging is clearly the low overhead alternative if additional processing or metric collection is not required.

5 Summary

Web application performance is an important concern for businesses implementing mission critical applications over the Internet, since the level of performance delivered directly affects the customer. In order to understand the level of service an application is providing measurement is essential. We presented three forms of Web application performance measurement methodologies: server based response time measurement, site monitoring, and client agent/based measurement.

We provided a detailed design and software description of WEBARM a mobile code based agent application instrumentation tool used for capturing end-to-end response times. Although the application designer must be concerned with the additional API calls required, the end result is a more manageable application through service measurement.

We have analyzed several important design aspects of the WEBARM agent software. In particular, WEBARM agent architecture, Web page instrumentation, logging considerations, and log collection techniques were studied. We proposed two methods of storing timer information across Web page references: the Cookie and Array-Frame methods, and provided a detailed description of the WEBARM APIs and supporting functions. We proposed, developed, and benchmarked the WEBARM agent techniques and studied the impact of the WEBARM logging CGI process.

WEBARM is an adaptation of the ARM Version 1.0 API set; however, WEBARM has been optimized and simplified for Web applications. While ARM Version 3.0 incorporates JAVA support, its emphasis is more on JAVA applications than Web page instrumentation [16].

Our future work includes continued monitoring of the ARM APIs evolution, investigation into response-time component analysis for Web applications, and network based designs for automating server instrumentation.

References

- [1] Nelson, M., "Fast Is No Longer Fast Enough", *Information Week*, June 2000, <http://www.informationweek.com/789/web.htm>
- [2] "Web Commerce Doubles in Year 2000 As Global Growth Continues," ActiveMedia Research, August 2000, http://www.activmediaresearch.com/body_free_newsroom.html
- [3] Caceres, R., et al., "Web Proxy Caching: The Devil is in the Details", *ACM SIGMETRICS Performance Evaluation Review*, Vol. 26 No. 3, (December 1998): pp 11-15.
- [4] Gross, T., Steenkiste, P., Subhlok, J., "Adaptive Distributed Application on Heterogeneous Networks", *Proceedings of the Eight Heterogeneous Computing Workshop*, (1998).
- [5] Hochheiser, H., Shneiderman B., "Understanding Patterns of User Visits to Web Sites: Interactive Starfield Visualization of WWW Log Data", *Institute for Systems Research Technical Report*, (March 1999).
- [6] Barford, P.,Crovella, M., "Measuring Web Performance in the Wide Area", *ACM SIGMETRICS Performance Evaluation Review*, Vol. 27 No. 2, (September 1999): pp 37-47.
- [7] Norris, C., "Understanding Web Site Speed and Reliability", *Keynote Systems, Inc.*, June 2000, http://www.keynote.com/services/downloads/product_lib.html
- [8] Kitay, D., "Understanding End-To-End Monitoring", *Candle Computer Report*, Nov-Dec 1997, Vol.19 No. 9, Candle Corporation
- [9] Pham, V., Karmouch, A., "Mobile Software Agents: An Overview", *IEEE Communications*, July 1998, pp. 26-37.
- [10] "ARM 2.0 SDK User's Guide", Computer Measurement Group, November 1997, <http://www.cmg.org/regions/cmgarmlw/sdk/armguide.html>
- [11] Elarde, J., Brewster, G., "Performance Analysis of Application Response Measurement (ARM) Version 2.0 Measurement Agent Software Implementations", *Proceedings from the 19th IEEE International Performance, Computing, and Communications Conference*, 2000
- [12] Microsoft TechNet, "System Redundancy", April 2000, <http://www.microsoft.com/technet/e-commerce/sysredund.asp>
- [13] Tierney, B., Johnston, W., et al., "The NetLogger Methodology for High Performance Distributed Systems Performance Analysis", *Proceedings from the Seventh International Symposium on High Performance Distributed Computing*, 1998.
- [14] Park, D., "XLF: The Extensible Log Format", XLF Working Group, July 1998, <http://www.docuverse.com/xlf/NOTE/xlf-19980721-all.html>
- [15] JavaScript Unleashed, Richard Wagner, et al., Sams.net Publishing, 1997, Second Edition
- [16] Johnson, M., " ARM 3.0 – Enabling Wider Use of ARM in the Enterprise", December 1999, <http://www.cmg.org/regions/cmgarmlw/ARM30.html>

The Dynamics of Price, Revenue, and System Utilization

Srinivasan Jagannathan and Kevin C. Almeroth

Department of Computer Science, University of California, Santa Barbara,
CA 93106-5110

{jsrini,almeroth}@cs.ucsb.edu

Abstract. Content delivery is a growing enterprise in the Internet. Critical to the management of content delivery systems is understanding customer behavior, its impact on the consumption of system resources and how together, these affect revenue. We believe that price dictates overall customer behavior and that understanding the relationship between price and customer behavior is the key to effective system management. To this end, we study the impact of price on revenue and system utilization. Our service model is based on customers being able to refuse content based on their capacity to pay and their *willingness* to pay the price quoted. We quantify the effects of such customer behavior on revenue and system utilization. Since customer behavior and characteristics are highly varying and not known *a priori*, we develop an adaptive pricing model which tracks user behavior as well as the arrival process to maximize revenue. We validate it using simulation. Our simulation results indicate that the adaptive pricing scheme generates nearly the same revenue as the theoretical expectation under very dynamic workloads.

1 Introduction

The volume of multimedia traffic has been steadily growing over the past few years. It is very likely that in the future, it will constitute the largest percentage of data transmitted in the Internet. The sheer volume of data involved makes content delivery a lucrative business proposition, so much so, that specialized overlay networks called *Content Delivery Networks (CDNs)* are being deployed to meet the demand. Some of the most important issues for managing content delivery networks include: (1) capacity planning, (2) controlling system utilization, and (3) maximizing revenue. Capacity planning is essential to meet future demand. System utilization is a measure of the popularity of the enterprise as well as an indicator of the short term availability of resources. Control over system utilization allows the content provider to provision resources in an efficient manner. At the same time, revenue maximization is the central goal of any business enterprise. Therefore, an important task is to develop mechanisms that control system utilization as well as maximize revenue. In this paper we investigate pricing and price adaptation as mechanisms to achieve these goals.

A fundamental contribution of this paper is to recognize the probabilistic nature of user behavior and quantify its impact on the system. Let us consider

an example to illustrate the probabilistic nature of customer behavior. Consider a teenager with \$15 as pocket money at a video-game parlor. The latest release of a hit video-game is very attractive to him, but whether or not he chooses to play the game depends on the price associated with the game and the money he has with him. He may be very likely to play for \$5, but not for \$14. He may decide to wait for another month when the game is not so new and the price falls. There is a probability associated with his decision to play based on the price and his capacity to pay. This probability is typically modeled using a utility function. In this paper, we choose a different but analogous model to represent this behavior. We can see a direct correlation between the example described here and purchasing content in the Internet. In general, the probability that a customer buys the service decreases with increasing price and increases with his or her capacity to pay. We try to capture this behavior in our work. Under specific assumptions of user behavior and a system model, we analyze pricing mechanisms which maximize *expectation* of revenue. We use the term *expectation* in the statistical sense, because the revenue generated depends on a probabilistic user behavior model. Our work is based on a video-on-demand server, but it is sufficiently general to be applied to other forms of content.

Delivery of content depends on three factors—resource availability, customer capacity to pay and customer willingness to pay. In this paper, we analyze pricing mechanisms for a system with limited resources, a *Pareto* distribution of customer capacity to pay and a probabilistic model for user willingness to pay the quoted price. We argue that charging a constant price will maximize the expected revenue for any user willingness model in which user willingness decays with increasing price. We derive the constant price for the user willingness models we use in this paper. We also derive a relationship between system utilization and price. Using this relationship, one can control system utilization in predictable ways by varying the price. Since the parameters of the customer capacity distribution will not be known to the service provider, we develop an adaptive pricing model which estimates these parameters. Our algorithm adapts to the user behavior as well as to dynamic workloads. We show, using simulations, that revenue using the adaptive pricing scheme matches closely with the expectation of revenue, given the probabilistic customer willingness to pay.

Pricing mechanisms have been suggested earlier for managing resource allocation and congestion control [1,2]. Researchers have also suggested differentiated pricing schemes based on service and quality [3,4]. These differentiated schemes propose creation of service classes with different quality guarantees. In this paper, we assume one single service class for content delivery. Goldberg et al. [5] suggest flat rates for pay-per-view movies. But they argue that finding the optimal price to maximize profit is difficult. Our adaptive algorithm is a mechanism to overcome this problem. Basu and Little[6] consider the impact of price on the request arrival process for video-on-demand systems. They formulate pricing strategies assuming that a good user behavior model can be found using marketing analyses. In our work we take a different approach. We develop an

analytical model that describes general user behavior and use an adaptive algorithm to learn the actual values of the parameters governing the user behavior.

The rest of the paper is organized as follows. We describe our basic system model used in this paper in Section 2. We formulate the theoretical expectation of revenue and the relationships between system utilization and price in Section 3. In Section 4, we develop the adaptive pricing scheme. We validate it using simulations in Section 5. We conclude the paper in Section 6.

2 System Model

We consider a system where requests are satisfied if resources are available and the customer agrees to pay the quoted price. Resources are modeled as *logical channels*. Every request which is satisfied occupies a channel for some finite amount of time. For a video-on-demand server we can think of the channels as the number of movies that can be served simultaneously. In this paper we do not focus on how the channel is allocated or how an allocated channel is managed. These issues have been treated in detail in earlier work [7,8,9,10,11]. We mainly focus on the interaction between the system and the customer before a channel is allocated.

Economic theory has established that there are a large number of customers with a small income and a very small number of customers with a very large income [12]. It is reasonable to assume that customers’ capacities to spend will follow a similar behavior. In this paper, we use a *Pareto* distribution[12] to represent the capacity to spend. Every customer has the capacity to pay based on a Pareto distribution with two parameters—shape α and scale b . All customers have capacities at least as large as b . The shape α determines how the capacities are distributed. The larger the value of α , the fewer the people with a very large capacity to pay. The Pareto density function is defined as $f_{\varphi}(x) = \frac{\alpha b^{\alpha}}{x^{\alpha+1}}$, for $x \geq b$. Figure 1 illustrates the Pareto density function for different values of shape α , and scale $b = 67^1$.

Even though customers *can* spend, they may not be *willing* to do so. To adequately describe the willingness of customers to pay, we define a family of probability functions. Consider an arbitrary customer with capacity χ . We denote his/her decision to purchase the service, by the random variable Υ which can take two values—1 for accept and 0 for reject. As discussed in the example in the previous section, the probability that the customer accepts the price ψ , denoted by $P\{\Upsilon = 1 \mid \psi\}$ depends on his/her capacity χ , and the price ψ .

$$P\{\Upsilon = 1 \mid \psi\} = \begin{cases} 1 - \left(\frac{\psi}{\chi}\right)^{\delta} & , \quad 0 \leq \psi \leq \chi \\ 0 & , \quad \psi > \chi \end{cases} \tag{1}$$

In this paper, we work with a simple model, where $P\{\Upsilon = 1 \mid \psi\}$ is defined as shown in Equation 1. By varying the parameter δ , we can make the willingness

¹ For, $b=67$ and $\alpha = 3$, the mean of the Pareto distribution is 100.

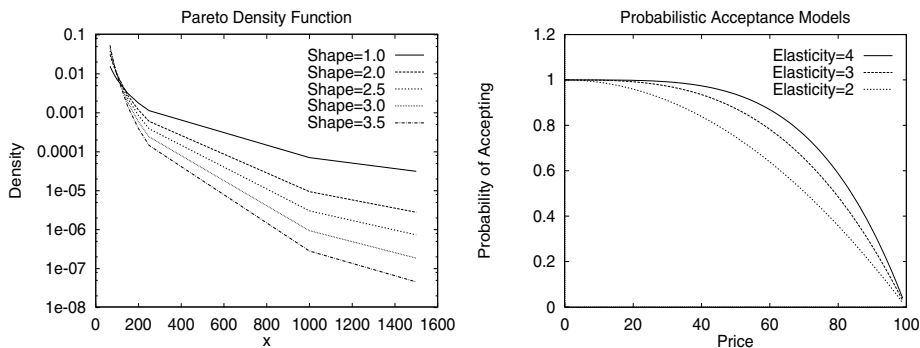


Fig. 1. Pareto Density (left) and Probabilistic User Willingness (right)

as *elastic* as desired. The higher the value of δ , the more willing are customers to spend money. We show three different willingness models for a customer having capacity 100, with δ values 2, 3 and 4 respectively in Figure 1. As can be seen, the model with $\delta = 4$ makes the customer much more willing to spend money than in the case of the other two models. In fact, as δ increases to around 4.0 or greater, the willingness begins to resemble a “step-function”. To make our model easier to analyze, we make a simplifying assumption that all customers will conform to one single model (as opposed to different customers obeying models with different values for δ). As we shall show later, this does not affect the adaptive pricing algorithm that we develop in Section 4.

3 Price, Revenue, and System Utilization

In this section, we discuss how customer capacities and their willingness to pay affects revenue and system utilization. We then derive expressions for the optimal price which maximizes the expectation of revenue under different workload conditions.

Even though the overall distribution of capacities is assumed to be Pareto, there is no way to know the capacity or willingness of any given customer. Hence, to maximize revenue, it makes more sense to charge every customer a constant price. By choosing a constant price we maximize the chances that they accept. We have proven this intuition correct using probability theory. The proof hinges on the fact that the expectation(mean) of a function(in this case revenue) is maximized when the probability that the function takes the maximum value is 1, i.e., the function is a constant. The actual value of the price for which expectation of revenue is maximized depends on the user willingness model. We state the following theorems without proof (owing to reasons of space):

Theorem 1. *If the shape parameter $\alpha > 1$ (i.e., a finite mean for the Pareto distribution exists), and willingness $P\{\mathcal{Y} = 1 \mid \psi\}$ decreases monotonically with respect to ψ and tends to 0 as ψ approaches ∞ , then the expectation of revenue, $E[\gamma]$, is maximized when ψ is a constant.*

Theorem 2. For the user willingness defined in Equation 1, the expectation of the variable Υ given price ψ , $E[\Upsilon | \psi]$ is as follows:

$$E[\Upsilon | \psi] = \begin{cases} 1 - \frac{\alpha}{\alpha + \delta} \left(\frac{\psi}{b}\right)^\delta, & 0 \leq \psi \leq b \\ \frac{\delta}{\alpha + \delta} \left(\frac{b}{\psi}\right)^\alpha, & \psi > b \end{cases} \quad (2)$$

Theorem 3. For the user willingness defined in Equation 1, the expectation of revenue, $E[\gamma]$, is maximized when the price $\psi_{max} = \left[\frac{\alpha + \delta}{(\delta + 1)\alpha}\right]^{\frac{1}{\delta}} b$. The expectation of Υ given price ψ_{max} is $\frac{\delta}{\delta + 1}$.

According to Theorem 1, the content provider should charge a flat rate to maximize revenue for the system model used in this paper. Theorem 2 gives an estimate on the mean rate at which customers accept a quoted price ψ . Theorem 2 tells us that the mean rate of acceptance is at least $\frac{\delta}{\alpha + \delta}$ if the price is less than b , and at most $\frac{\delta}{\alpha + \delta}$ if the price is greater than b . Theorem 3 suggests what price should be charged to maximize revenue.

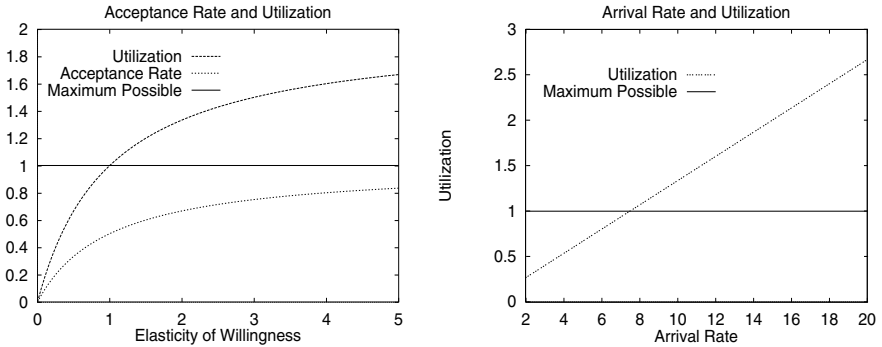


Fig. 2. Price, Arrival Rate, Acceptance Rate and System Utilization

Consider a system with n channels. If requests arrive at a rate λ , and the mean duration to service the request (e.g., the play-out time for a movie) is d , then the system utilization, ρ , when we charge price ψ is given by:

$$\rho = \frac{\lambda E[\Upsilon | \psi] d}{n} \quad (3)$$

According to Theorem 3, when the content provider charges a price $\left[\frac{\alpha + \delta}{(\delta + 1)\alpha}\right]^{\frac{1}{\delta}} b$, the rate at which customers accept the price is $\frac{\delta}{\delta + 1}$. Hence, the system utilization ρ_{max} which yields maximum expectation of revenue is given

by $\frac{\lambda\delta d}{(\delta+1)n}$. Figure 3 shows how the acceptance rate, $E[\mathcal{Y} | \psi_{max}]$, and ρ_{max} vary with the elasticity of willingness, δ , for a system with 500 channels, a mean play-out duration of 100min and an arrival rate of 10 requests/min. Also shown is how ρ_{max} varies with the arrival rate for the same system when the elasticity, δ , is 2.0. Note that system utilization cannot be greater than 1. This implies that resource constraints may prevent achieving the maximum expectation of revenue when either the arrival rate is high, or willingness to pay is very high. Therefore in an environment with large fluctuations in arrival rate or one in which customer willingness is high, the price charged must adapt itself to achieve maximum revenue. Note that in a system with n channels, and mean play-out duration d , on average, not more than $\frac{nt}{d}$ requests can be served in a time duration t . Also note that according to Theorem 3, the maximum expectation of revenue is achieved over time t , when $\frac{\lambda\delta t}{\delta+1}$ requests are served. Thus, when $\frac{\lambda\delta t}{\delta+1} > \frac{nt}{d}$, resources are insufficient to achieve maximum expectation of revenue. It is intuitive to increase the price at such a juncture to such an extent that the acceptance rate declines to equilibrium, i.e., only as many customers accept the price, as can be accommodated by the system. We have proved using calculus that this does indeed achieve maximum revenue. We state the following theorem without proof:

Theorem 4. *Let customer capacities be Pareto distributed with shape α , $\alpha > 1$, and scale b . Let their willingness to pay be as defined in Equation 1. Consider a system with n channels serving content with mean play-out duration d . Let λ be the request arrival rate. The expectation of revenue for the system is maximum when the content provider charges a price ψ_{MAX} defined as follows:*

$$\psi_{MAX} = \begin{cases} \left[\frac{\alpha+\delta}{(\delta+1)\alpha} \right]^{\frac{1}{\delta}} b & , \quad \frac{\delta}{\delta+1} \leq \frac{n}{d\lambda} \\ \left[\left(\frac{\alpha+\delta}{\alpha} \right) \left(1 - \frac{n}{d\lambda} \right) \right]^{\frac{1}{\delta}} b & , \quad \frac{\delta}{\delta+1} > \frac{n}{d\lambda} \geq \frac{\delta}{(\alpha+\delta)} \\ \left[\frac{\lambda d \delta}{n(\alpha+\delta)} \right]^{\frac{1}{\alpha}} b & , \quad \frac{\delta}{\delta+1} > \frac{\delta}{(\alpha+\delta)} > \frac{n}{d\lambda} \end{cases} \quad (4)$$

Theorem 4 gives the content provider a mechanism to maximize revenue based on arrival rate and observed willingness elasticity δ . We delve into this in greater detail in the next section.

4 Adaptive Pricing Scheme

The theory developed in the previous section tells us how the user acceptance rate varies with price and what price to charge to maximize the expected revenue. However, the optimal price is dependent on the Pareto distribution parameters (shape α , and scale b) and the willingness elasticity parameter, δ . These parameters will not be known to the content provider. Moreover, they may not even be observable, because a customer will not disclose his capacity to pay. The only observable events are the customer's acceptance of the quoted price (denoted by the binary variable \mathcal{Y}), and the request arrival rate λ .

We have developed an adaptive pricing algorithm that learns from the rate at which customers accept a given price. We use the equations developed in Theorems 2 and 3 to relate the rate of acceptance and the price charged. There are three unknown variables, α , b and δ and only two observable events, \mathcal{Y} and λ . Of these two observables, λ cannot always be used. Hence, we must assume some reasonable value for two of the unknowns to be able to predict the third:

1. We assume an arbitrary value for α . This is because the expected revenue is not very sensitive to α even for moderately large values of α . And we expect α to be large for the customer capacities. We shall show later using simulations that mis-estimating α does not significantly alter our results.
2. We now have to assume some reasonable value for one other unknown. Instead of assigning a single estimate we identify a set of possible values and then compute the other unknown. The parameter that we choose to identify a set for is the willingness elasticity parameter, δ . We choose the set Δ , consisting of feasible values of δ , such that it covers a wide range of elasticity of willingness. For instance, if we constrain δ to belong to the set $\{0.7, 1.0, 1.3, 1.6, 2.0, 2.4, 2.7, 3.0, 3.4, 3.8\}$, any actual value of willingness elasticity can be approximated to one of the elements in the set. Now, the problem of prediction is slightly more tractable.

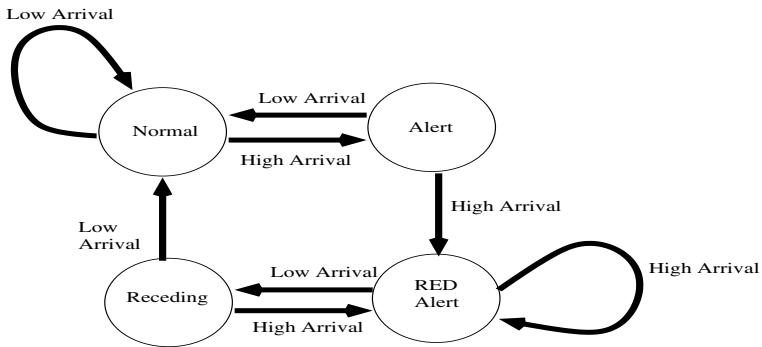


Fig. 3. State Diagram to Monitor Arrival Rate

Our algorithm adapts after observing a *round* of requests. Each round consists of 100 customer requests. Note that we chose 100 arbitrarily. We want to observe acceptance rate over a reasonable number of customers. A constant price ψ is charged in each round. This allows us to observe the rate of acceptance for price ψ . This observed rate is then equated to the formula for $E[\mathcal{Y} | \psi]$ derived in Theorem 2. At the same time, the arrival rate is also monitored. The time elapsed for one round to complete gives an estimate of the current arrival rate. For each feasible value of elasticity δ , (i.e., elements of set Δ), we compute a possible value for b . We choose the appropriate equation to use from Equation 2 based on whether or not the observed rate of acceptance is greater or less than

$\frac{\delta}{\alpha+\delta}$. Once we have a set of feasible values for b , we perform one more round of experiments. We choose an arbitrary price and compute the expected rate of acceptance for each of the feasible values of b . After this second round, we compute which feasible value of b most closely predicted the observed acceptance rate. We use that value for b , and the corresponding δ in Theorem 4 to get the price to be charged in the next round. Since the price to be charged depends on the arrival rate, we maintain a state based on the state diagram shown in Figure 3. If $\frac{\lambda\delta}{\delta+1} \leq \frac{n}{d}$, then we term the arrival rate as “low”. Otherwise, we call it “high”. Since a high arrival rate in one single round of 100 requests can be a false alarm, we change state to “Red-Alert” only when the arrival rate is high enough in two successive rounds. Similarly, when in a “Red-Alert” state, there is a transition to “Normal” only after two successive rounds of low arrival rate. Since state transitions are dependent on elasticity value δ , for each $\delta_i \in \Delta$, we maintain a separate state S_i . Thus, depending on the current value of δ being used, we charge a price based on that δ and the corresponding b and state S . For states “Normal”, “Alert” and “Receding”, we charge $\left[\frac{\alpha+\delta}{(\delta+1)\alpha}\right]^{\frac{1}{\delta}} b$ while for the state “Red-Alert”, we charge either $\left[\left(\frac{\alpha+\delta}{\alpha}\right)\left(1 - \frac{n}{d\lambda}\right)\right]^{\frac{1}{\delta}} b$ or $\left[\frac{\lambda d\delta}{n(\alpha+\delta)}\right]^{\frac{1}{\alpha}} b$ based on the value of $\frac{nd}{\lambda}$ as shown in Theorem 4. The algorithm is presented in Figure 4.

5 Simulations

In this section, we validate the theory and algorithm developed in the previous sections using simulations. We have implemented a simulator to model the content delivery system. Our simulations can be divided into two broad categories. One set of simulations validates the theoretical model developed in Section 3. We verified that Theorem 2 accurately predicts the acceptance rate and that the price suggested by Theorem 3 maximizes the revenue. We validated Theorem 4 through simulations. The second category of simulations deals with the adaptive pricing algorithm. The adaptive algorithm assumes some value for the shape of the customer capacity distribution. We show that the revenue earned is fairly insensitive to this assumed value. We compare the revenue earned using our algorithm with that earned by a prescient algorithm that knows all the parameters of the customer distribution. We show that the adaptive algorithm earns revenue very close to the predicted maximum expectation for a wide range of customer willingness elasticity and highly varying workloads. We also show simulation results that indicate that the adaptive algorithm is robust in dealing with situations not modeled by the theoretical framework developed in Section 3.

The following is a description of parameters that we used for our analysis. We performed simulations with 500 logical channels. We chose a fixed number of channels because the system capacity typically does not change very often. We chose request service times from a uniform distribution between 90 and

1. Choose an arbitrary price ψ_0 . and a value for α
2. Choose a set of elasticity values $\Delta = \{\delta_1, \dots, \delta_n\}$
3. For the next 100 arrivals charge ψ_0 .
4. Compute the observed acceptance rate p_0 and the arrival rate λ_0 .
5. $\forall \delta_i \in \Delta$, initialize state S_i to “Normal” or “Alert” based on δ_i and λ_0 .
6. $\forall \delta_i \in \Delta$, compute scale b_i using Theorem 2.
7. Choose another arbitrary price ψ_1 .
8. $\forall \delta_i \in \Delta$, compute expected acceptance rate for price ψ_1 using scale b_i and Theorem 2.
9. $k \leftarrow 1$
10. Repeat forever
 11. For the next 100 requests, charge a price ψ_k
 12. Compute the acceptance rate p_k and arrival rate λ_k .
 13. $\forall \delta_i \in \Delta$ update state S_i based on λ_k .
 14. For each $\delta_i \in \Delta$, compute scale b_i using Theorem 2.
 15. Compare the acceptance rates predicted in round $k - 1$ with the observed acceptance rate p_k .
 16. Identify the δ_{opt} whose predicted acceptance rate most closely matches the observed acceptance rate p_k . Let b_{opt} be the scale computed in this round using δ_{opt} .
 17. Set price ψ_{k+1} using δ_{opt} , b_{opt} and Theorem 4
 18. For each $\delta_i \in \Delta$, compute expected acceptance rate for price ψ_{k+1} using scale b_i and Theorem 2.
 19. $k \leftarrow k + 1$.
20. End loop

Fig. 4. Our adaptive pricing algorithm

110 minutes. This closely models the typical length of movies. Channels were allocated based on a FCFS policy. Requests arriving when there are no free channels are rejected. There is no waiting queue. The capacities of individual customers were chosen from a Pareto distribution with scale 67 and shape 3.0. This distribution has a mean value of 100. We chose only one representative Pareto distribution of capacities because of two reasons. First, the actual shape of the distribution does not affect the results. It is the relative error between the assumed and actual values that will affect the simulation results. Second, the actual value of b is more related to the service being sold and its perceived value. Hence it is largely independent of our prediction algorithm. We chose values for customer willingness (δ) from the set $[0.5, 5.0]$. When δ is small, the customer is extremely unwilling to purchase the content. For δ values around 5.0, the probability that the customer will purchase the content is nearly 1.0 (if the price is less than his capacity to pay).

The workload models we used in our simulations are shown in Figure 5. These models are adapted from the work on arrival-rate based scheduling by Almeroth et al. [9]. The workloads are modeled based on a 24 hour period beginning from 8.00am of one day and running to 8.00am of the next. “Prime time” periods see

a surge in demand. We have used a steady baseline workload, with no surges in demand, and three non-steady workloads. The arrival rates during prime time for the non-steady workloads was around five times greater than the normal rate, based on statistics reported by Little and Venkatesh [11]. We simulated both gradual as well as sudden increases in arrival rate. We also used a workload with hourly spikes in arrival rate during primetime. This type of workload is based on the belief that the workload for some systems may be synchronized with an external event like wall-clock time.

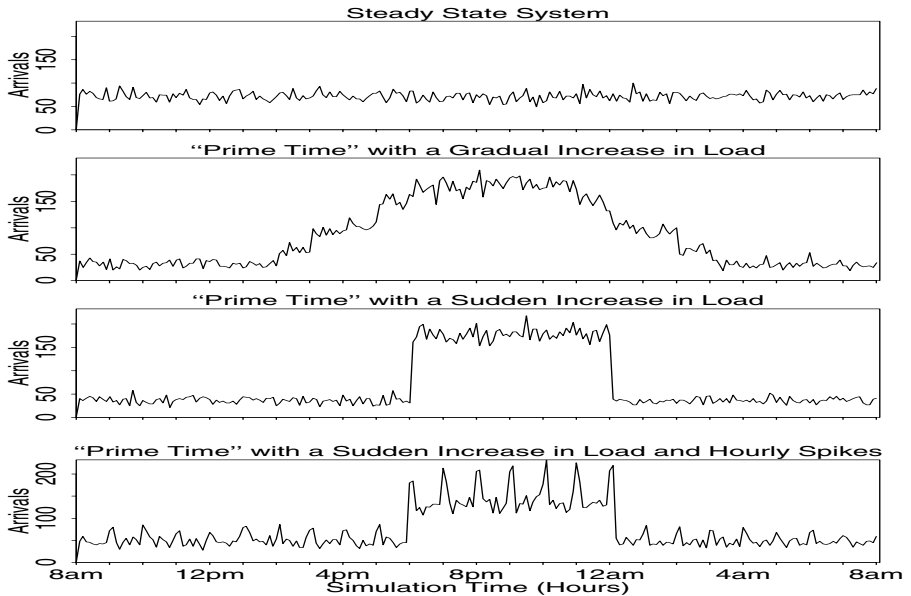


Fig. 5. Workloads

5.1 Validating the Analytical Framework

To validate Theorems 2 and 3, we ran different sets of simulations, charging a constant price from each customer and varying the price across simulations. We chose arrival rates which would not overload the system. We found that the theorems accurately predict the acceptance rate and mean revenue, given the price. The results of these simulations are described in greater detail in our earlier work [13].

To confirm our intuition that during peak-hours, we must charge as much money as will result in highest predicted system utilization, we ran different sets of simulations with very high arrival rates. In each set of simulations, we varied the price across simulations. The price charged in each particular simulation was kept constant. We show the system utilization, acceptance rate and revenue

earned in two such sets of simulations in Figure 6. The arrival rate in these sets of simulations were 10 per minute and 30 per minute respectively. Henceforth we shall refer to these sets as *Set 1* and *Set 2* respectively. The elasticity δ for both the sets was 2. Note that according to Theorem 3, a price of 49.93 will yield maximum revenue and result in an acceptance rate of 0.667. However, this means that on average we have 6.67 requests (for Set 1) and 20 requests (for Set 2) being serviced per minute. But the system cannot sustain more than $\frac{500}{100}$ requests per minute. Thus we cannot achieve the maximum predicted by Theorem 3 due to lack of resources. Theorem 4 was derived for such situations. Theorem 4 predicts that highest expectation of revenue is achieved at price 61.16 for Set 1 and at price 89.70 for Set 2. We observe that this prediction is indeed true for both Set 1 as well as Set 2. Furthermore, the acceptance rate predicted by our theorems are also accurate. The observed system utilization also closely matches the predictions. We found Theorem 4 to be accurate in all the sets of simulations we ran.

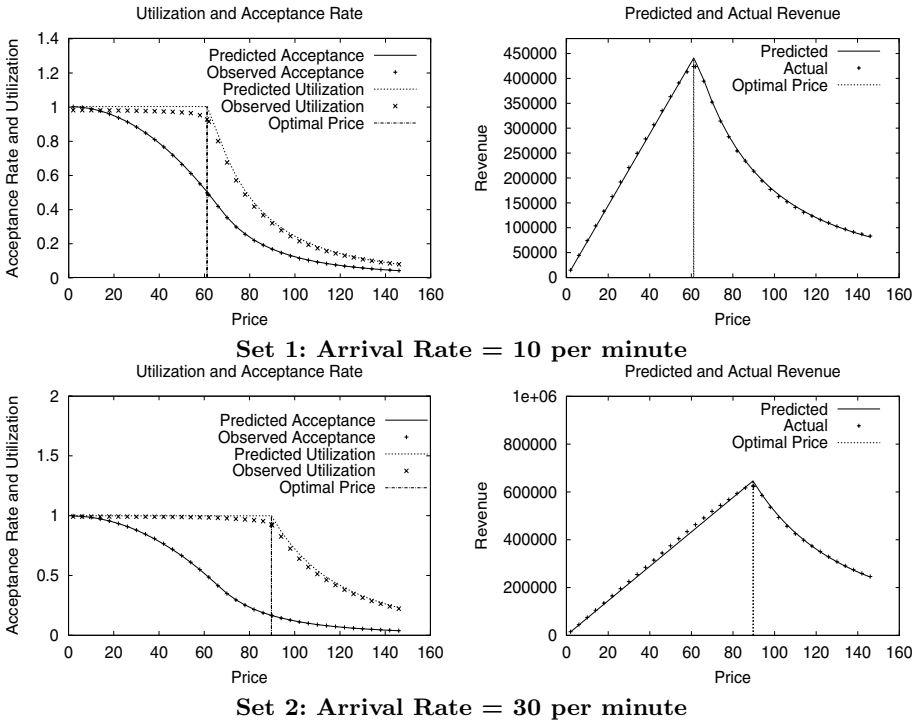


Fig. 6. Verification of Theorem 4

5.2 Validating the Adaptive Pricing Algorithm

Having validated our theoretical framework, we performed simulations to validate our adaptive algorithm. We ran simulations to: 1) evaluate how much rev-

enue is generated using the adaptive algorithm, 2) study how the assumed value of shape impacts revenue, 3) study the impact of the starting price and, 4) evaluate the robustness of the algorithm. To evaluate the performance of the adaptive algorithm, we compared the revenue generated by it to the predicted expectation as well as to the revenue generated by a prescient algorithm. The prescient algorithm has knowledge of all the parameters in the simulation except the exact knowledge of the capacity and willingness of each individual customer. The revenue generated by the prescient algorithm represents a physical upper-bound on the expected revenue. Since the predicted revenue will vary with willingness elasticity δ and the workload, we use the ratio of actual revenue and predicted revenue as a metric. We shall call this ratio as the *Revenue-ratio*. The greater this ratio, the better the algorithm. Note that this ratio can be greater than 1.0 because we are predicting the maximum expectation of revenue. We also use the predicted and observed system utilization as a measure of the performance of the algorithm. The closer the predicted and observed utilization, the better the algorithm. The system utilization achieved by the prescient algorithm gives us an idea of the physical bounds even when we have complete knowledge.

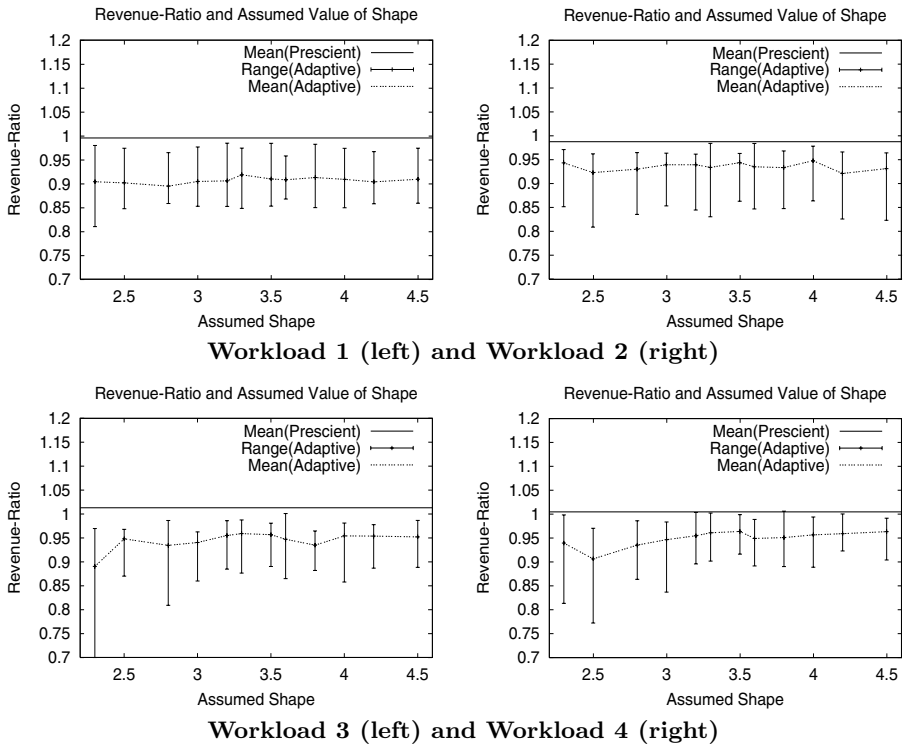


Fig. 7. Revenue-Ratio for Different Workloads

We ran a wide range of simulations varying the assumed value of shape as well as the elasticity of willingness δ . Figure 7 depicts the the observed range of Revenue-ratio for different assumed values of shape. For each assumed value of shape, we obtained this range by running simulations with different values of willingness elasticity δ . The values of δ used in the simulations was different from the set Δ used by the adaptive algorithm. Also shown is the mean revenue-ratio of the prescient algorithm for those values of δ . The starting price, ψ_0 , was 20 and ψ_1 was 40, for all these simulations. As can be seen, the adaptive algorithm performs very well for each kind of workload irrespective of the assumed value of shape.

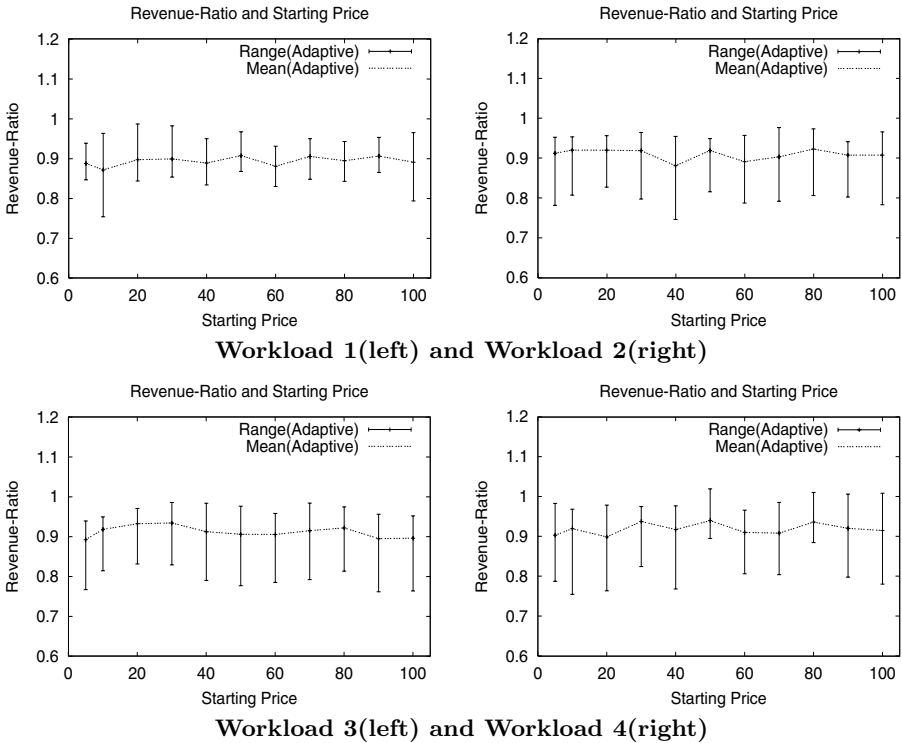


Fig. 8. Impact of Starting Price on Different Workloads

To study the impact of starting price on the revenue generated by the algorithm, we ran a number of simulations with different starting values. We varied ψ_0 in these simulations and always chose ψ_1 to be twice the starting price, i.e., $2 \times \psi_0$. Our simulations indicate that the performance of our algorithm is independent of the starting price. This behavior is expected because, the algorithm computes the scale for a wide range of feasible values of δ . Thus, it is able to adapt to the correct scenario no matter what the starting price. We present a

representative plot of the results for each kind of workload in Figure 8. Shape, α , was assumed to be 2.0 (by the adaptive algorithm) for this set of simulations. The elasticity δ was varied. As before, we present the range of revenue-ratios we observed, when we varied δ for a given starting price ψ_0 . Due to reasons of space we are unable to present a representative plot of the impact of starting price on system utilization. Our simulations indicate that, like revenue, system utilization too is relatively independent of the starting price.

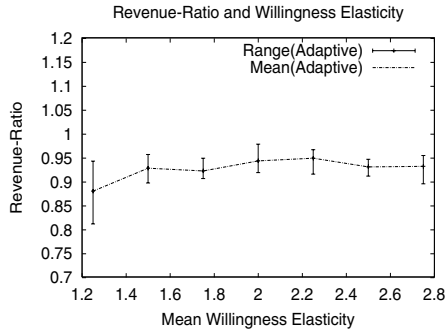


Fig. 9. Impact of Variable Willingness Elasticity

In this paper, we made an assumption that all the customers conform to a single elasticity parameter δ . However, in reality, this parameter may vary from person to person. Our adaptive algorithm will be able to adapt to these variations. It will choose the closest δ which approximates the overall customer behavior as a whole. To confirm that this is indeed true, we ran simulations in which the parameter δ for a customer was chosen from a uniform distribution. Note that for such customer behavior we do not know the maximum expectation of revenue. Therefore it is difficult to evaluate the performance of the algorithm. However, if the workload does not exceed the available resources, according to Theorem 1, the maximum expectation of revenue is achieved when we charge a constant price. For such workloads, we can exhaustively search for the optimal price using simulations. We chose a steady workload (arrival rate of 6 per minute) such that resources would be available even when δ for all customers is the maximum possible. We ran simulations to exhaustively search for the price which generates maximum revenue. We then ran simulations with our adaptive algorithm for the same customer behavior. We found that the algorithm performed remarkably well. The ratio of the revenue generated by the algorithm to the maximum revenue found using simulations is shown in Figure 9. We present the range of ratios generated when we used different assumed values for the shape, α , in the adaptive algorithm. Though the performance in the graph does not “prove” that the algorithm is robust, it does indicate that the algorithm adapts fairly well to customer behavior.

6 Conclusions and Future Work

In this paper, we studied the impact of price on the revenue and utilization of FCFS content-delivery systems. We developed an analytical framework for maximizing revenue and quantified the relationship between price and system utilization. This relationship can be used to manage system utilization by controlling the rate at which customers purchase the content. We have also developed an adaptive pricing scheme that tracks observable customer behavior to suggest the price. We performed simulations to validate our analytical framework and to evaluate the performance of the adaptive pricing scheme under dynamic workloads. Our simulations indicate that our algorithm is robust and generates revenue close to the maximum theoretical expectation. Therefore, with an effective dynamic pricing scheme, we now have a powerful mechanism to affect system utilization and revenue. Both of these characteristics are critical to successfully managing a content delivery service.

Our future work is to focus on using price to manage systems which use sophisticated scheduling mechanisms like batching. Also, the effects of content popularity and temporal changes in customer behavior need to be studied in greater detail. Content popularity has a significant impact on its *perceived value* and hence the revenue. An eventual goal of our work is to leverage content popularity and maximize revenue by allowing customer and provider to negotiate the price for the content.

References

1. D. F. Ferguson, C. Nikolaou, and Y. Y., "An economy for flow control in computer networks," in *Conference on Computer Communications*, (Ottawa, Canada), April 1989.
2. A. Ganesh, K. Laevens, and S. R., "Congestion pricing and user adaptation," in *IEEE Infocom*, (Anchorage, Alaska), April 2001.
3. J. Altman, B. Rupp, and P. Varaiya, "Internet user reactions to usage-based pricing," in *Proceedings of the 2nd Internet Economics Workshop*, (Berlin), May 1999.
4. J. Altman and K. Chu, "A proposal for flexible plan that is attractive to users and internet service providers," in *IEEE Infocom*, (Anchorage, Alaska), April 2001.
5. A. Goldberg, J. Hartline, and A. Wright, "Competitive auctions and digital goods," Tech. Rep. STAR-TR-99-01, InterTrust Technologies Corporation, November 2000.
6. P. Basu and T. Little, "Pricing considerations in video-on-demand systems," in *ACM Multimedia Conference*, (Los Angeles, California), November 2000.
7. S. Jagannathan, K. C. Almeroth, and A. Acharya, "Topology sensitive congestion control for real-time multicast," in *Workshop on Network and Operating System Support for Digital Audio and Video (NOSSDAV)*, (Chapel Hill, North Carolina, USA), June 2000.
8. S. Jagannathan and K. C. Almeroth, "Using tree topology for mutlicast congestion control," in *International Conference on Parallel Processing*, (Valencia, Spain), September 2001.
9. K. Almeroth, A. Dan, D. Sitaram, and W. Tetzlaff, "Long term channel allocation strategies for video applications," in *IEEE Infocom*, (Kobe, JAPAN), April 1997.

10. A. Dan, D. Sitaram, and P. Shahabuddin, "Scheduling policies for an on-demand video server with batching," in *ACM Multimedia*, (San Francisco, California, USA), October 1994.
11. T. Little and D. Venkatesh, "Prospects for interactive video-on-demand," *IEEE Multimedia*, pp. 14–23, Fall 1994.
12. B. C. Arnold, *Pareto Distributions*. Burtonsville, Maryland: International Co-operative Publishing House, 1983.
13. S. Jagannathan and K. C. Almeroth, "An adaptive pricing scheme for content delivery systems," in *Global Internet Symposium*, (San Antonio, Texas, USA), November 2001.

Developing Pattern-Based Management Programs

Koon-Seng Lim and Rolf Stadler

Center for Telecommunications Research and
Department of Electrical Engineering
Columbia University
New York, NY 10027-6699
{kooenseng, stadler}@ctr.columbia.edu
<http://www.comet.columbia.edu/adm>

Abstract. We have recently proposed a novel approach to distributed management for large-scale, dynamic networks. Our approach, which we call pattern-based management, is based on the methodical use of distributed control schemes. From an engineering point of view, our approach has two advantages. First, it allows for estimating the performance characteristics of management operations in the design phase. (This was the subject of an earlier paper.) Second, it reduces the complexity of developing scalable, distributed management programs, by promoting the re-usability of key software components. In this paper, we demonstrate how pattern-based management programs can be designed and implemented. We propose an object model for these programs and give a concrete example of how a management task, such as obtaining the current link load distribution of the entire network, can be realized. The example illustrates that our model facilitates writing the key software components in a compact and elegant way.

1 Introduction

In our recent research, we have proposed a novel approach to distributed management, which applies particularly well to large-scale networks and to networks that exhibit frequent changes in topology and network state [5][6]. We call this approach pattern-based management. It is based on the methodical use of distributed control schemes. The key concept is that of *navigation patterns*, which describe the flow of control during the execution of a (distributed) management program. A navigation pattern determines the degree of parallelism and internal synchronization of a distributed management operation.

The concept of navigation patterns has two main benefits. First, it allows for the analysis of management operations with respect to performance and scalability [6]. Second, from a software engineering point of view, navigation patterns allows us to separate the semantics of a management operation from the control flow of the operation. This means that a typical management operation can be realized using different navigation patterns, which can be chosen according to different performance objectives. Also, since a pattern is generic and does not encapsulate any management-specific semantics, the same pattern can be used for different management tasks. We believe that, as a consequence, our approach will free the management application programmer from developing distributed algorithms, allowing him/her to focus on the

specific management task and to select a navigation pattern that captures the requirements for that task.

Table 1: Management paradigms with different control schemes and programming models

Manager-Agent	Centralized control	Program runs on management station, operates on 'global' MIB
Management by Delegation, Remote Evaluation	Centralized control Dynamic delegation of subtasks	Program runs on management station, subprogram runs on local MIBs
Mobile Agent	Decentralized control, realized by intelligent autonomous decisions	Agent program controls code migration and operations on local MIBs
Pattern-based Management	Decentralized control, realized by network algorithms	Distributed program runs on topology graph whose nodes are local MIBs

Table 1 shows how pattern-based management relates to other management paradigms. The manager-agent model, based on the client-server paradigm, realizes a centralized control scheme. This control scheme can be characterized by a management program running on a management station and operating on the Management Information Base (MIB), which is distributed among agents in the network. The manager-agent model serves as a basis for the management standards behind SNMP and CMIP [15][9]. The main limitation of this model is its poor scalability, since it can lead to a large amount of management traffic, a large load on the management station, and long execution times for management operations, especially in large networks [17]. To overcome this drawback, the concepts of *management by delegation* (MbD) and *remote evaluation* (REV) were proposed [1][16]. These concepts allow for delegating tasks from the management station to the agents in the network via the downloading of scripts. A further step towards decentralized control was taken with the introduction of *mobile agents* for management tasks [11]. Mobile agents can be characterized by self-contained programs that move in the network and act on behalf of a user or another entity. Mobile agents are generally complex, since they exhibit some kind of intelligent behaviour, which allows them to make autonomous decisions. In contrast to the mobile agent approach, our paradigm, pattern-based management, stresses the use of simple, dedicated components, which run in parallel on a large number of network nodes. All of the above paradigms, except for the manager-agent model, necessitate an execution environment on the network nodes, which executes scripts, mobile agents or pattern programs, respectively.

Our work is close to Raz and Shavitt [12], which advocates the use of distributed algorithms in combination with active networking for network management. We differ from [12] in our emphasis on the separation of the flow of control of the management operation from its semantics through an explicit interface.

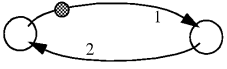
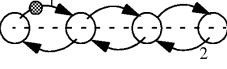
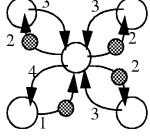
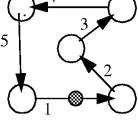
operation	typical application	navigation pattern
type 1: node-to-node	1 node control/monitor (e.g. get/set of variables)	
type 2: visit all nodes along a path/flow	1 flow/path control (e.g. traceroute, bottleneck detection, signalling, VPN operation)	
type 3: distribute agents to all nodes in subnet (parallel control)	subnet control, message broadcast (e.g. congestion location detection)	
type 3: visit all nodes in subnet (sequential control)	subnet control (e.g. topology detection)	

Fig. 1. Examples of simple navigation patterns

Figure 1 shows simple examples of navigation patterns. The most basic pattern is the type 1 pattern, where control moves from one node to another and returns after triggering an operation. This pattern exemplifies a typical manager-agent interaction. A type 2 pattern illustrates the case where control moves along a path in the network, triggers operations on the network nodes of this path, and returns to the originator node along the same path. A possible application of this pattern is resource reservation for a virtual path or an MPLS tunnel [13]. In a type 3 pattern, control migrates in parallel to neighbouring nodes, triggers operations on these nodes, and returns with result variables. This pattern can be understood as a parallel version of the type 1 pattern. Finally, in a type 4 pattern, control moves along a circular path in the network. All these examples illustrate that navigation patterns can be defined independently of the management semantics.

To fully exploit the advantages of pattern-based management, we envision the use of patterns with more inherent parallelism than those given in Figure 1. For instance, in [6] we have introduced the echo pattern, which triggers local operations on all nodes of the network and aggregates the local results in a parallel fashion.

As discussed in [6], navigation patterns can be defined using asynchronous parallel graph traversal algorithms, also known as network algorithms [8]. Note that the concept of a navigation pattern is very different from that of a design pattern as used in software engineering. While a navigation pattern captures the flow of control of executing a distributed operation, a design pattern describes communicating objects and classes for the purpose of solving a general design problem in a particular context [4].

A pattern-based management system can be realized in various ways. One possibility is through a mobile agent platform that is installed on the network nodes (or on control processors that are attached to the network nodes) and provides access to the

nodes' control and state parameters through local interfaces [5]. On such a platform, management operations are coded as mobile agent programs, and navigation patterns define the migration, cloning and synchronization of the agents, which are executed on network nodes, perform local operations, and aggregate results. Another possibility for realizing the concept of navigation patterns is to program patterns on an active management platform, such as ABLE [12]. Such a platform includes commercial routers that have been enhanced by an active management node, which processes active packets that have been filtered out by the router. In ABLE, such packets contain Java code that is executed on the management node. An interesting, third possibility is the case whereby the code for performing a management operation is not transported to the nodes as part of this operation. (The code has either been installed beforehand or is loaded on demand from a code server.) In this case, pattern-based management can be realized by a system that exchanges messages between network nodes, which simply contain part of the execution state of the management operation.

This paper reports on our research in how to program navigation patterns and how to develop pattern-based management programs. Section 2 discusses the structure of a pattern-based management program. Section 3 introduces the three main object classes in a management program-- navigation pattern, operator and aggregator. Section 4 gives a concrete example of a management program, using the echo pattern to compute the current load distribution of all links in the network. This example also shows how the migration of control, defined by the pattern, can be decoupled from the management semantics of the program. Finally, section 5 summarizes the results of this paper and gives an outlook on our future work.

2 The Structure of a Pattern-Based Management Program

Figure 2 illustrates the structure of a pattern-based management program. It is comprised of three abstract object classes [14]. The first class, called navigation pattern, specifies how the flow of control of the program migrates from node to node during the course of its execution. The second class, called aggregator, specifies the operation to be performed on each node and how its results are to be aggregated. The third class, called operator, provides a (management) platform independent interface to state and control variables of a network node.

The relationship between the instances of a pattern, an aggregator and an operator is as follows. The navigation pattern controls the execution of the aggregator on the network nodes. The aggregator implements the management semantics of a particular management operation. It runs in an execution environment and accesses node functions through the operator. A navigation pattern can also access the operator to retrieve node information in order to make traversal decisions.

The execution of a pattern-based management program involves asynchronously creating and maintaining a set of distributed states throughout the network. We distinguish among three different abstract classes for these states. The pattern state class contains information used by navigation patterns to traverse the network, while the aggregator state class holds the state of the distributed management operation. Finally, the node state class represent the operational state of a network node that is accessible to management operations.

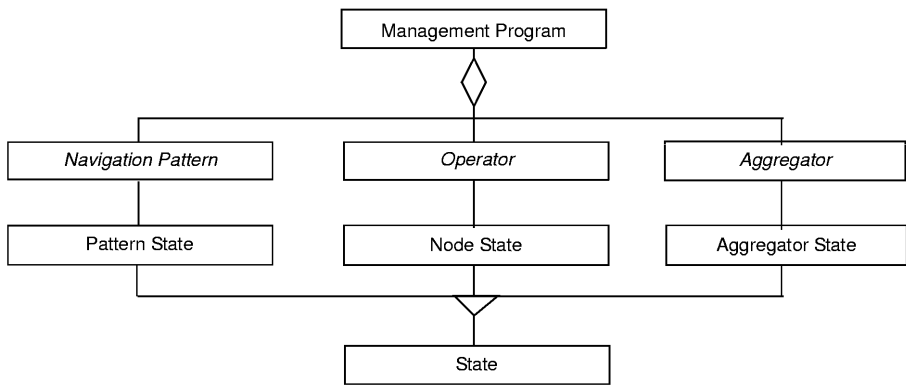


Fig. 2. Class diagram of a pattern-based management program. Abstract classes are in italic.

Pattern and aggregator state can exist in two forms. State that is stored on individual nodes for the lifetime of the management program is called fixed state. Sometimes, it is required that data is carried from node to node as the program executes on the network, in which case we refer to it as mobile state. For example, an aggregator that counts the number of leaf nodes in a network may store sub-totals as fixed states on certain nodes during program execution. In contrast, a navigation pattern that “backtracks” to nodes previously visited may carry its return path in form of a mobile state. While both fixed and mobile states have the same generic structure, the mobile state implements additional functionality, such as data serialization which it requires for state migration.

3 Patterns, Operators, and Aggregators

The abstract class navigation pattern is specialized into concrete classes, each of which defines a particular graph traversal algorithm. These algorithms distribute the flow of control of the management program in a particular manner. For example, the echo pattern shown in the Figure 4 represents a navigation pattern that distributes control according to the wave-propagation algorithm [3][6].

For each concrete class of a navigation pattern, we define a corresponding abstract aggregator class. A pattern and an aggregator must fit together in order to function properly. Therefore, as shown in Figure 3, an abstract echo aggregator class is defined for the echo pattern class. This aggregator class in turn must be specialized into concrete application-specific classes, one for each type of management operation to be implemented. For example, the echo aggregator in Figure 3 is specialized into two different concrete aggregator classes--the leaf counting aggregator, and the aggregator for computing the connectivity distribution of the network topology.

In Figure 3, we also see that the abstract operator class is specialized into classes that interface with specific technologies for node access. In this example, three types of operators are given-- interfaces for SNMP, CMIP and GSMP [10].

As we will show in the following section, we define and implement a navigation pattern using the model of a Finite State Machine (FSM). When a management

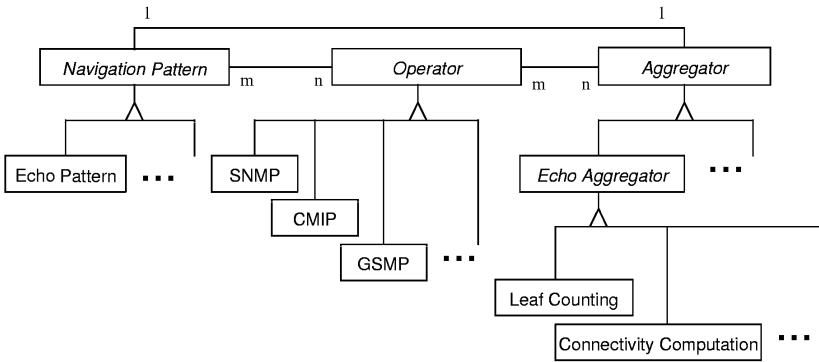


Fig. 3. Specialization of patterns, aggregators and operators. Abstract classes are in italic.

program is launched, control is transferred from the management station to the program’s pattern. The execution of a pattern on a network node involves two actions. First, the FSM state of the pattern is determined via the internal logic of the pattern and, second, a function corresponding to that state is triggered in the program’s aggregator. Note that for each FSM state, there is exactly one function in the aggregator corresponding to that state. This function defines the local actions or operations to be performed by the aggregator when the pattern is in the associated state. In this sense, the interface between pattern and aggregator can be modelled as an FSM that represent the operation of the pattern and the navigation pattern can be seen as controlling the execution of the management operation (via the aggregator) through those states.

4 Writing a Pattern-Based Management Program

Writing a pattern-based management program is a two-step process. In the first step, a navigation pattern which drives the execution of the management operation, is developed (or selected from a catalogue of well-known patterns). In the second step, an aggregator, which performs the actual management operation, is written. In the following subsections, we discuss both steps in detail.

Consider the development of a management program that obtains the link load statistics and computes the link load distribution across a large network in near real-time. Management operations of this nature provide a network operator with an immediate snapshot about the state of the network and allow decisions to be made in a responsive manner.

The traditional approach to this problem, using a centralized management paradigm, calls for writing a management program that polls every node in the network and then processes the results to compute the histogram. While feasible for small networks, this approach does not scale for most network topologies, due to the increasing processing time and the induced traffic overhead. A different, more sophisticated approach, includes a hierarchy of management entities-- a solution that is not feasible for dynamic networks with frequent changes in topology.

In contrast, a pattern-based management system is distributed in nature and requires no global topology information in order to accomplish the above task. A pattern-based approach to the problem involves selecting a navigation pattern that traverses all nodes in the network and implementing an associated aggregator that computes the required histogram during the pattern's traversal. As we have shown in [6] with the example of running the echo pattern on the Internet, a pattern-based management program can be significantly more efficient in a large network for this task than a centralized system.

4.1 Programming the Echo Pattern

In this section, we show how to implement the echo pattern, a navigation pattern we have introduced in [6]. Management operations based on this pattern do not need knowledge of the network topology, can dynamically adapt to changes in the topology and scale well in very large networks.

The defining characteristic of the echo pattern is its two-phase operation, which works as follows. In the first or expansion phase, the flow of control emanates from the node attached to the management station (start node) via messages called explorers. When an explorer arrives at a node for the first time (unvisited node), it sends out copies of itself on all links, except for the one it arrived from (explorer link). It then marks the node as being visited. When an explorer arrives on an already visited node, it triggers the second phase of the pattern, the contraction phase, by sending a message called an echo out on its explorer link. If an explorer arrives at a leaf node, (one whose only link is an explorer link), it is returned as an echo. When all the explorers of originating from a node have returned as echoes, a new echo is generated and sent out on the node's explorer link. Therefore, for every explorer that is sent over a link, there is a corresponding echo that returns over the same link. The pattern terminates when the last echo returns to the management station.

The execution of echo pattern can be visualized as a wave that propagates out from the start node to every node in the network, before converging back to the start node.

A navigation pattern is implemented as a distributed, asynchronous program that runs identical copies of itself on a set of nodes. On each of these nodes, its execution can be described by an FSM. Upon invocation on a node, the local copy determines its FSM state by examining the fixed and mobile pattern states (Section 2) and triggers a function within the aggregator that corresponds to its current FSM state. When that function returns, it signals to a subset of adjacent nodes (including itself) to schedule its further execution on those nodes.

In the case of the echo pattern, its local program can be described by an FSM with the following seven states:

- ***OnBegin***
This state is the first state entered when the pattern is launched from the management station. It is triggered exactly once on the start node.
- ***OnTerminate***
This state corresponds to the termination of the navigation pattern. It is entered on the node where the management program has been launched, after the aggregator triggered by ***OnLastEcho*** state has returned.

- ***OnFirstExplorer***
This state is entered when an explorer reaches an unvisited node. It is triggered exactly once on each node in the network.
- ***OnSecondaryExplorer***
This state is entered when an explorer reaches an already visited node.
- ***OnFirstEcho***
This state is entered when the first echo returns to a node. It is triggered exactly once on each node in the network.
- ***OnSecondaryEcho***
This state is entered when an echo, other than the last or first echo, arrives at a node.
- ***OnLastEcho***
This state is entered when the last echo arrives at a node. It is triggered exactly once on each node in the network.

The seven states listed above can be further classified into three broad categories, namely, (a) *initialization and termination states* (consisting of the ***OnBegin*** and ***OnTerminate*** states), (b) *expansion states* (consisting of the ***OnFirstExplorer*** and ***OnSecondaryExplorer*** states) and (c) *contraction states* (consisting of ***OnFirstEcho***, ***OnSecondaryEcho*** and ***OnLastEcho*** states).

The initialization and termination states are used by the aggregator to initialize mobile state variables and to return results back to the management station respectively. The expansion states are typically used to disseminate information, initialize fixed state variables or initiate local operations in aggregators. The contraction states are used to incrementally aggregate results or to propagate the results of operations back to the start node.

Since the echo aggregator specifies the local actions/operations to be performed for each of the possible seven states that the echo pattern can be in at any point in time, it is defined by seven abstract functions corresponding to these states.

Figure 4 shows the implementation of the `run()` method of the echo pattern in C++. Note that the interface definition of this method, restricts access to the operator (`op`), the pattern state (`ps`), the aggregator (`agg`) and the aggregator state (`as`). The program begins by testing if it was just launched by checking the mobile state variable `begin_m` (line 2). If the variable does not exist (meaning the current node is the start node), the program creates it and triggers its aggregator by calling its `OnBegin()` function (lines 3 and 4).

Next, the program checks if it is an echo or an explorer (line 5). If it is an explorer, it checks if the current node has been visited (line 6). If not, the program triggers its aggregator, by calling its `OnFirstExplorer()` function, flags the node as visited and obtains a list of its neighbours from its operator (lines 7 to 10). The program also initializes a fixed state variable `first_echo_f` to indicate that the first echo has not yet returned (line 9).

The program then checks, if a mobile (pattern) state variable `path_m` has been created (line 11). If not, it creates such a variable to store the return path for use during the contraction phase (line 12). If `path_m` has been created before, the program checks, if the current node has any neighbours other than its parent, which would result in further expansion (line 13). (The term ‘parent’ refers to the neighbour

node that is connected via the node's explorer link.) If not, the pattern has encountered a leaf node, in which case, the program marks itself as an echo by creating a mobile state variable `is_echo_m` (line 14). It then creates a fixed state variable `num_explorers_f` with value 1 and schedules itself to run on the same node (lines 15 and 16). If the current node is not a leaf node, the program saves its path in `path_m`, creates a fixed state variable `num_explorers_f` to count the number of outstanding explorers, and proceeds by passing control to its neighbour nodes (lines 17 to 19).

If the executing instance of the pattern is an explorer and the current node has already been visited, then the program triggers its aggregator, by calling its `OnSecondaryExplorer()` function (line 20), marks itself as an echo (line 21) and returns to its parent node, whose id is stored in `path_m` (line 22).

If the instance of the pattern program is an echo, it determines if it is the first echo by checking `first_echo_f` (line 23). If so, it triggers its aggregator by calling its `OnFirstEcho()` function and resets `first_echo_f` to indicate that the first echo has returned (lines 24 and 25).

The program then decrements `num_explorers_f` (line 26) and checks the result for zero (line 27). If this is the case, this means that all outstanding explorers have returned as echoes, and the program calls `OnLastEcho()` of its aggregator (line 28). Furthermore, if `path_m` is empty, which would indicate that the pattern has completed its traversal (line 29), the pattern calls `OnTerminate()` of its aggregator (line 30). Otherwise, the pattern calls `OnSecondaryEcho()` of its aggregator (line 32).

As can be seen from the description above, programming patterns can be a non-trivial exercise, requiring a background in distributed algorithms. As such, a key benefit of the pattern-based management approach lies in the re-usability of navigation patterns to solve classes of management problems.

4.2 Programming an Echo Aggregator

Programming the aggregator for computing the link load distribution involves (a) deriving a new class from the abstract class echo aggregator (Figure 3), and (b) defining the semantics of each of its seven functions. Briefly, its operation can be described as follows. While the pattern expands, a histogram is created as a fixed (aggregator) state variable on each node, which is populated with the link utilization of the outgoing links. In the other case, when the pattern contracts, a node's histogram is copied into a mobile (aggregator) state variable and propagated to back its parent in the echo. When the echo arrives at the parent, this histogram is aggregated with the parent's histogram. When all echoes have arrived at the parent node, the parent sends its aggregated histogram to its parent which repeats this process recursively.

```

1 HopList EchoPattern::run(Operator op, PatternState ps, AggregatorState as,
  Aggregator *agg){
2     if (!ps.has_state("begin_m")) {
3         ps.create_mobile_state("begin_m", 0);
4         agg->OnBegin(as, op);
5     }
6     if (!ps.has_state("is_echo_m")) {
7         if (!ps.has_state("visited_f")) {
8             agg->OnFirstExplorer(as, op);
9             ps.create_fixed_state("visited_f", 1);
10            ps.create_fixed_state("first_echo_f", 0);
11
12            NodeList nl = op.get_neighbors();
13
14            if (!ps.has_stack("path_m"))
15                ps.create_fixed_stack_state("path_m", 0)
16            else if (nl.num_nodes() == 1) {
17                ps.create_mobile_state("is_echo_m", 1);
18                ps.set_state("num_echoes_f", 1);
19                return make_hop_list(op.node_id());
20            }
21
22            ps.push_stack("path_m", op.node_id());
23            ps.create_fixed_state("num_explorers_f", nl.num_nodes() - 1);
24            return make_hop_list_except_parent(nl, ps.peek_stack("path_m"));
25        } else {
26            agg->OnSecondaryExplorer(as, op);
27            ps.create_mobile_state("is_echo_m", 1);
28            return make_hop_list(ps.pop_stack("path_m"));
29        } else {
30            if (ps.get_state("first_echo_f") == 0) {
31                agg->OnFirstEcho(as, op);
32                ps.set_state("first_echo_f", 1);
33            }
34
35            ps.set_state("num_explorers_f", ps.get_state("num_explorers_f")--);
36            if (ps.get_state("num_explorers_f") == 0) {
37                agg->OnLastEcho(as, op);
38                if (ps.peek_stack("path_m") == 0) {
39                    agg->OnTerminate(as, op);
40                    return NULL;
41                }
42            } else agg->OnEcho(as, op);
43        }
44    }
45 }

```

Fig. 4. C++ implementation of the echo pattern

Figure 5 gives the C++ implementation of this aggregator. When the pattern triggers the `OnFirstExplorer()` function of the aggregator (line 2), an empty histogram is created (line 3). The program then calls the operator to read the link utilization statistics of each link, quantizes them and updates the histogram (lines 4 to 6). Finally, it stores the results in a fixed state variable `histogram_f` (line 7). When the `OnSecondaryEcho()` function of the aggregator is triggered (line 10),

```

1 void LinkLoadDistributionAggregator::OnBegin(AggregatorState as,
  Operator op) {
  }

2 void LinkLoadDistributionAggregator::OnFirstExplorer(AggregatorState as, Operator
  op) {
3   histogram *h = new histogram();
4   Links l = op.get_links();
5   for(int i = 0; i < l.count(); i++)
6     h->add(op.get_utilization_quantized(l.link_id(), 1000000), 1);
7   as.create_fixed_state("histogram_f", h);
  }

8 void LinkLoadDistributionAggregator::OnSecondaryExplorer(AggregatorState as,
  Operator op) {
  }

9 void LinkLoadDistributionAggregator::OnFirstEcho(AggregatorState as, Operator op)
  {
  }

10 void LinkLoadDistributionAggregator::OnSecondaryEcho(AggregatorState as, Operator
  op) {
11   histogram *h;
12   if (!as.has_state("histogram_m")) {
13     h = as.get_state("histogram_f");
14     histogram *h2 = as.get_state("histogram_m");
15     h->sum(h2);
16     as.set_state("histogram_f", h);
  }
  }

17 void LinkLoadDistributionAggregator::OnLastEcho(AggregatorState as, Operator op) {
18   histogram *h;
19   if (!as.has_state("histogram_m")) {
20     h = as.get_state("histogram_f");
21     as.create_mobile_state("histogram_m", h);
22   } else {
23     h = as.get_state("histogram_m");
24     histogram *h2 = as.get_state("histogram_f");
25     h->sum(h2);
26     as.set_state("histogram_m", h);
  }
  }

27 void LinkLoadDistributionAggregator::OnTerminate(AggregatorState as, Operator op)
  {
28   histogram *h = as.get_state("histogram_m");
29   cout << "Histogram is " << h.print();
  }

```

Fig. 5. C++ implementation of the link load distribution aggregator

the program checks if a histogram is being returned in the echo through a mobile state variable `histogram_m` (line 12). If so, it aggregates its histogram with the received histogram using the `sum()` method (line 15) and re-saves it (lines 16). Similarly, when the `OnLastEcho()` function is triggered (line 17), the program checks if a histogram is being returned through `histogram_m` (line 19). If not, it creates the histogram from a copy of `histogram_f` (line 21) and returns. Otherwise, it aggregates its histogram with the received histogram (lines 23 to 25) and returns it to its parent as `histogram_m`. Finally, when the `OnTerminate()` function is triggered (line 27), the program prints out the histogram (line 29).

Note that there is generally more than one way to implement an aggregator, namely, by associating aggregator functions with different states of the pattern. For example, in the code of Figure 5, the local operation of reading a node's link utilization statistics is performed in the `OnFirstExplorer()` function, i.e. the function associated with the *OnFirstExplorer* state, while the `OnFirstEcho()` function is empty. An alternative implementation for the same operation would be to reverse the associations of these two functions by performing the local operation in the `OnFirstEcho()` function instead. The difference between these two implementations lies in the performance characteristics of the operation. An example showing this difference can be found in [7]. Network managers can take advantage of this effect by choosing implementations according to their performance objectives.

Moreover, although the we distinguished five states--excluding initialization and termination--as part of the echo pattern, one state is sufficient to realize many operations. This state can be understood as combining the three contraction states, namely *OnFirstEcho*, *OnSecondaryEcho* and *OnLastEcho*, into a single state. Consider the task of computing the average load of all network nodes. An aggregator implementing this task can be written as a function that incrementally aggregates the partial averages delivered via echoes and the local load. In general, functions, such as sum, product, average, min, and max, can be implemented using this approach. These functions are based on operators that are commutative, i.e., , and associative, i.e., .

$$Op(x_i, x_j) = Op(x_j, x_i) \forall (i, j)$$

$$Op(x_i, Op(x_j, x_k)) = Op(Op(x_i, x_j), x_k) \forall (i, j)$$

As explained above, using more than one state allows the implementor to create operations with different performance profiles, e.g., operations with shorter completion times. For example, if a local read operation takes a long time, it is often more efficient to perform the read operation in one of the expansion states and use the contraction states for aggregation only. Therefore, such an implementation requires (at least) two states, in addition to initialization and termination.

5 Discussion

One of the primary benefits of pattern-based management is that it allows pattern programs to be reused to solve a large class of management problems. This is possible because navigation patterns are generic programs and do not encapsulate the semantics of a management operation. So far, we have successfully designed echo aggregators that compute network load statistics and distributions, calculate global topological properties and perform network initialisation.

In general, aggregators are also significantly less complex than navigation patterns. Patterns are distributed programs that must be designed operate correctly in any given network topology. Aggregators, on the other hand, are local function calls. Aggregator programmers do not need to contend with issues related to network topology, concurrency, synchronization or termination. Therefore, we envision a pattern-based management system to contain a catalogue of a few carefully designed patterns developed by distributed systems specialists and a wide variety of aggregators written by management application programmers.

In our approach, we model the interface between a pattern and an aggregator as a FSM. This design hides the distributed nature of the algorithm implementing the

navigation pattern from the programmer of the aggregator. In many cases, the number of states of this state machine can be reduced even further. The cost of this simplification, as we have shown in section 4.2, is that the programmer has less of a choice in selecting specific performance profiles. However, as the example also clearly shows, even this one-state FSM can be used to implement a large class of useful computations, all without requiring any understanding of how the echo pattern works.

In order to support the development of pattern-based management programs and to study their scalability properties, we have built a software workbench for constructing, testing and simulating a pattern-based management system [7]. The tool, called SIMPSON, is written in Visual C++ and runs on Microsoft Windows platforms (Win 95, 98, NT, 2000). Currently, it allows for the interactive simulation and visualization of patterns on networks as large as 10,000 nodes.

We believe that this work opens up interesting avenues for further research. Currently, we focus on the following aspects. First, we want to explore the applicability of our approach to provisioning and management of Internet services, such as DiffServ. Second, we want to analyse navigation patterns in terms of survivability properties. For example, the echo pattern in this paper is not resilient to node or link failures and needs to be enhanced to increase its robustness. Further, we plan to investigate which classes of management problems can be solved in an asynchronous, symmetrical and distributed way, which makes them ideal candidates for implementation on a pattern-based management system. Above all, we want to develop an inventory of navigation patterns that are applicable to key management tasks, analyse them regarding performance, scalability, and survivability, and implement them as software components that can be embedded in management platforms.

Acknowledgments. The work described in this paper has been supported by a grant from British Telecom.

References

- [1] M. Baldi and G.P. Picco, "Evaluating the Tradeoffs of Mobile Code Design Paradigms in Network Management Applications," in Proceedings of ICSE '98, Kyoto, Japan, April 1998, pp. 146-155.
- [2] J. Banks (Ed), J. S. Carson, B. L. Nelson and D. M. Nicol, Discrete Event System Simulation, Prentice-Hall, August 2000.
- [3] E. J. H. Chang, "Echo Algorithms: Depth Parallel Operations on General Graphs," IEEE Transactions on Software Engineering., vol. 8, no. 4, pp. 391-401, July 1982.
- [4] E. Gamma, R. Helm, Ralph Johnson, and John Vlissides: Design Patterns—Elements of Reusable Object-Oriented Software, Addison Wesley, 1995.
- [5] R. Kawamura and R. Stadler: "A middleware architecture for active distributed management of IP networks," in Proceedings of IEEE/IFIP NOMS 2000, Honolulu, Hawaii, April 2000, pp. 291-304.
- [6] K.S. Lim and R. Stadler, "A Navigation Pattern for Scalable Internet Management," in Proc. of 7th IFIP/IEEE IM'01, Seattle, USA, May 2001, pp. 405-420.

- [7] K.S. Lim and R. Stadler, "Developing Pattern-Based Management Programs," CTR Technical Report 503-01-01, Columbia University, New York, July 2001.
- [8] N. Lynch, *Distributed Algorithms*, Morgan Kaufmann Publishers, August 1997.
- [9] R. Marshall, *The Simple Book*, Prentice Hall, 1994.
- [10] P. Newman, W. Edwards, R. Hinden, E. Hoffman, C. F. Liaw, T. Lyon and G. Minshall, "Ipsilon's General Switch Management Protocol Specification Version 2.0," RFC 2297, March 1998.
- [11] V. A. Pham and A. Karmouch, "Mobile Software Agents: An Overview," *IEEE Communications Magazine*, July 1998, Vol. 36 No. 7, pp. 26-37.
- [12] D. Raz and Y. Shavitt, "An active network approach for efficient network management," in *Proc. of IWAN'99*, (LNCS 1653), Berlin, Germany, June/July 1999, pp. 220-231.
- [13] E. Rosen, A. Viswanathan and R. Callon, "Multiprotocol Label Switching Architecture," RFC 3031, March 1998.
- [14] J. Rumbaugh, M. Blaha, W. Premerlani, F. Eddy and W. Lorenson, *Object-Oriented Modeling and Design*, Prentice Hall, January 1991.
- [15] W. Stallings, *SNMP, SNMPv2, SNMPv3, and RMON 1 and 2*, Addison-Wesley Publishers, January 1999.
- [16] Y. Yemini, G. Goldszmidt, and S. Yemini, "Network Management by Delegation," in *Proceedings of 2nd IFIP/ISINM '91*, Washington, DC, USA, April 1991.
- [17] Y. Zhu, T.M. Chen and S.S. Liu, "Models and Analysis of Trade-offs in Distributed Network Management Approaches," in *Proceedings of 7th IFIP/IEEE IM'01*, Seattle, USA, May 2001, pp. 391-404.

Supermedia in Internet-Based Telerobotic Operations^{*}

Imad Elhajj¹, Ning Xi¹, Wai Keung Fung², Yun hui Liu², Wen J. Li²,
Tomoyuki Kaga³, and Toshio Fukuda³

¹ Dept. of Electrical and Computer Engineering, Michigan State University, East Lansing, MI 48824, U.S.A.

² Dept. of Automation and Computer-Aided Engineering, Chinese University of Hong Kong, Shatin; N.T., Hong Kong

³ Center for Cooperative Research in Advanced Science and Technology, Nagoya University, Nagoya 464-8603, Japan

Abstract. In the past decade robotics and the Internet, fed by the great advances in computing and networking, matured into giant interdisciplinary scientific fields. Therefore, it is not surprising that many are trying to merge these two technologies and develop Internet-based robotic teleoperation. More interestingly, Internet-based bilateral teleoperation, where supermedia is fed back to the operator in order to increase efficiency and achieve telepresence. Supermedia is the collection of multimedia (video, audio, ...), haptic and other sensory information. This paper studies supermedia enhanced teleoperation via the Internet, concentrating on the real-time control of such systems and their performance. The potential of this technology and its advantages will be explored. In addition, key issues, such as stability, synchronization and transparency, will be analyzed and studied. Specifically, event-based planning and control of Internet-based teleoperation systems is presented with experimental results of several implemented system scenarios.

1 Introduction

When it comes to robotics, the lack of applications has never been an issue. The same can be said for Internet-based teleoperation, where the potential for applications is just limited by our imaginations. One of those applications is telemedicine, where several successful experiments have been done in this field. The main use of tele-medicine is in remote checkup and surgery, home care (elderly and disabled remote assistance) and physical therapy [1] [2]. Another application is operations in hazard environments, most commonly hazard material manipulation [3], space and deep sea operations [4] [5]. In addition e-services, such as remote maintenance and monitoring of manufacturing processes, equipment and products. Also the ability to use sophisticated and expensive manufacturing facilities by several users around the world is very cost effective [6]. Other

^{*} Research Partially supported under NSF Grant IIS-9796300 and IIS-9796287.

applications relate to law enforcement. Government agencies are already using remotely operated machines in their operations to reduce human risk. Also Internet-based teleoperation can be applied to entertainment. There are already on the web teleoperated games, where users can remotely login and operate a robot. This application will experience great advance in the coming few years once businesses realize its potential.

However, before these applications become widely accepted and used, several things have to be supplied and ensured. Most importantly is telepresence, which is the transfer of human senses to remote locations by feeding back sensory information from the remote environment. This would closely couple the operator with the remote environment and thus gives a more realistic feeling of remote presence. In addition to realism, telepresence significantly increases the efficiency of teleoperation. To achieve this, sensory information has to be presented to the operator, the most common of which is visual feedback. However this is not sufficient for some applications; therefore, additional types of feedback are required. Supermedia feedback, which can correspond to different sensory information, considerably increases operators' efficiency and makes some tasks feasible [7] [8]. The general structure of such a system is shown in Fig.1, where the operator sends velocity commands and receives supermedia (visual and haptic) information.

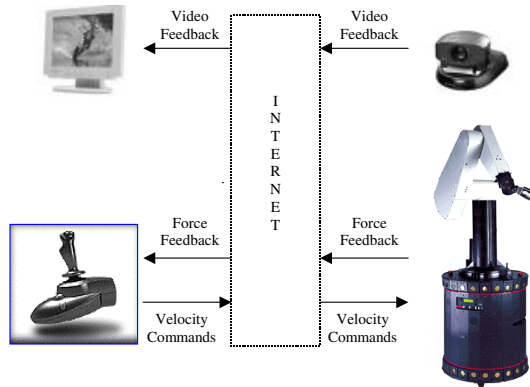


Fig. 1. General structure of an Internet-based teleoperation system with supermedia feedback.

Examining the literature, Internet-based robots can be divided into three categories according to the method used to send commands and receive haptic feedback and according to the nature of feedback supplied. These three categories are referred to here as: teleprogrammed, telesimulated, real real-time teleoperated.

Teleprogrammed Internet-based robots are ones that require the operator to upload a plan or set of commands for it to execute. This uploaded program is

executed by the robot either autonomously or semi-autonomously. This category can be further divided into “program and watch” robots or “program, watch and intervene” robots. Typically, teleprogrammed robotic systems include only visual feedback [9]-[11].

Telesimulated Internet-based robots include systems that feed forward commands in real-time but the feedback in simulated. This simulated or predicted feedback can be one of two forms; either it is completely simulated based on a robot and environment model, or it is partially simulated based on a model and corrected by actual feedback. Generally feedback is in the form of predicted display and/or predicted force [12] [13].

Real real-time teleoperated Internet-based robotic systems feed forward commands in real-time and feedback real real-time sensory information. The feedback comes in several forms the most typical of which are video and force [14] [15].

As in any control system, in order for Internet-based bilateral control to be acceptable it has to satisfy several performance criteria. Stability is the main one since it relates to feasibility and safety of the operation. The other two are, synchronization and transparency, which relate to the efficiency of the operation and its realism.

However, when it comes to the Internet, ensuring those performance characteristics becomes a significant challenge. The random time delay, network buffering effects and disconnections experienced over the Internet present major difficulties. Those difficulties have been extensively studied in the literature, especially resolving time delay effects in teleoperation systems with haptic feedback [7] [16]-[19]. But all those studies have several limitations since they assume time delay to be either fixed, the same in both directions or has an upper bound, none of which applies to Internet type delays. So there is a need for a planning and control method that can achieve stability, synchronization and transparency of Internet-based teleoperation with real-time supermedia feedback regardless of time delay and its variance. This paper presents such a method, as well as theoretical and experimental results.

2 Supermedia in Closed-Loop Control of Internet-Based Teleoperation

Initially most teleoperation done was open-loop, meaning the only form of feedback was video. Once haptic feedback started being considered, the control loop was closed between the robot and operator. Haptic information is that relating to the sense of touch, in other words tactile information. This information can be fed back, where it is reproduced to the operator using force generating devices. The haptic information does not always correspond to actually physical forces but can also correspond to other sensory information, such as heat, radiation and distance to obstacles. The need for haptic feedback is to increase the efficiency of teleoperations. However, it does not substitute the need for visual feedback, which is still an important form of feedback. Actually any type of multimedia (audio, video ...) can be fed back to increase efficiency and telepresence.

To capture the notion of all these feedback streams the term supermedia is presented. Supermedia is the collection of multimedia, haptic and any other sensory information. So the interest of this paper is in studying the effect of varying time delay on Internet-based teleoperation systems that include supermedia. And to develop a new planning and control methodology that would ensure stability, synchronization and transparency regardless of the time delay faced. This time delay, where in the Internet can not be predicted and is random with a very complex stochastic model.

Several attempts have been made to model delay over the Internet, and several complex models have been derived [20]. In [20] a wavelet-based model was developed, where delay was shown to be self-similar [21] in a statistical sense. However, estimating the arrival time of packets is still difficult to achieve. Fig.2 gives an example of the round trip time (RTT) between Michigan State University and Hong Kong, where the randomness of RTT is clearly shown.

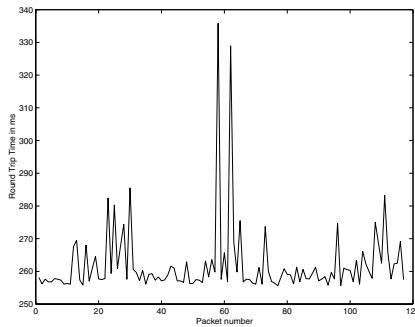


Fig. 2. Round trip delay in ms for packets transmitted between the robot and Hong Kong station.

This interest placed on time delay is a result of the instabilities it causes. Since delays on the order of a tenth of a second can destabilize the teleoperator [7]. And when haptic feedback is being used the system is stabilized only when the bandwidth was severely reduced. In addition to instability, de-synchronization can occur between the different supermedia streams. Desynchronization is a result of the buffering effect delay creates, where the network behaves as a FIFO buffer. This results in desynchronization between the feedback and feed forward streams thus destabilizing the system, and between the feed back streams themselves thus resulting in operator confusion and efficiency reduction. As for transparency, delay causes high frequency haptic changes to be filtered thus reduces the transparency of the operation.

3 Event-Based Planning and Control of Internet-Based Teleoperation

The performance effects discussed in the previous section are a result of using time as a reference for the different system entities. So intuitively if a different non-time based reference can be used these effects can be eliminated or at least reduced significantly. In traditional control systems the dynamics of the system is modeled by differential equations in which the free variable is the time variable t . And usually the trajectory is a function of time, but if we do not constraint the trajectory by time we would allow the system to be at any point at any time. So the general idea is to model the system and the trajectory with parametric equations. The parameter is called *motion reference or action reference*, and usually denoted by s [22].

This planning and control method is referred to as event-based, which was extended to bilateral teleoperation applications in [23]. This extension had to make some alterations to the initial event-based theory in order to be fit for teleoperation applications. The first main thing that had to be considered was the fact that in teleoperation there is no predefined path. The path is generated in real-time by the operator based on the feedback received. The other issue is that the operator has limited computational power so a simple event or reference has to be chosen. In addition, the method had to be independent of the operator and environment model, and it should not require significant overhead.

Therefore, the *event*, s , had to be carefully chosen in a way that no overhead is required from the operator and it had to be intuitive. In addition, to maintain the synchronization between the operator and the remote machine regardless of time delay the event had to be designed in a way to eliminate the buffering effect of delay. This effect is shown in time based part of Fig.3, where it is clear in the time based approach that the delay will cause many signals to be flowing within the network. So by the time a new command, V_{t+m} , that was generated as a result of a force, F_{t+k} , arrives at the destination, the state of the machine would have changed due to the commands that were already buffered in the network, $V_{t+1}, V_{t+2}, \dots, V_{t+m-1}$. This implies that when the robot gets the command V_{t+m} at time $t+m+n$, it has already changed status and is being subject to the force F_{t+n+m} ; therefore, the command would be a wrong one since it was a result of F_{t+k} and not F_{t+n+m} .

In order to ensure that the right command is being received and executed the buffering effect should be eliminated. So the communication and control model should be similar to the one shown in the event based part of Fig.3. As shown, the transmission of new commands and forces is not specified by time. A new command, V_{n+1} , will not be generated and sent until the most up-to-date status of the robot, F_n , is received. At the same time, a new force, F_{n+1} , will not be fed back until a new velocity, V_{n+1} , is received. This implies that there are no commands or forces within the network flowing thus the buffering effect is eliminated.

This setup is achieved by using event-based planning and control, with the event taken as the command number. So if currently the operator is sending the

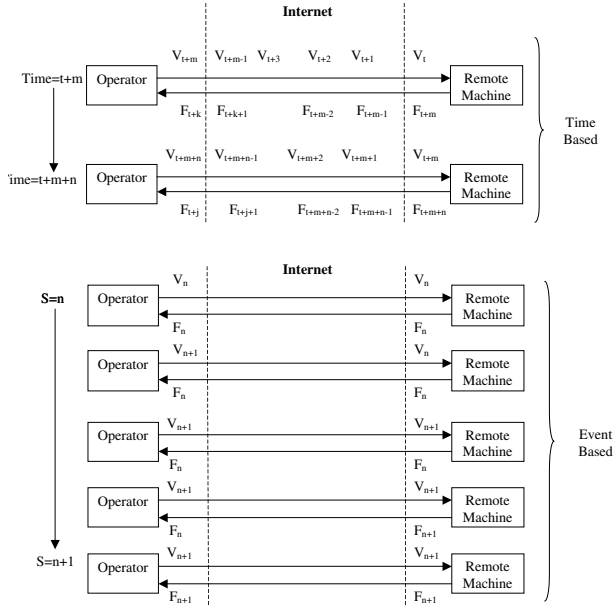


Fig. 3. Illustration of the buffering effect of delay, and the elimination of this effect in event-based planning and control.

n^{th} command, then the event is n . This choice of the event is intuitive and does not require the operator to keep track of it, since the communication procedure will take care of ensuring this specific behavior of the system regardless of the operator’s command. This is an additional advantage to this approach since the whole procedure mentioned is transparent to the operator.

4 Performance Analysis of Internet-Based Teleoperation

This section will analyze the stability, synchronization and transparency of the event-based approach. Concerning the stability of the system under event-based referencing, the following theorem was proven in [22]:

Theorem 1. *If the original robot dynamic system (without remote human / autonomous controller) is asymptotically stable with time t as its action reference; and the new non-time action reference, $s = \prod(y)$ is a (monotone increasing) nondecreasing function of time t , then the system is (asymptotically) stable with respect to the new action reference s .*

The only assumption needed is that the robot is a stable system, which means that the original robot dynamic system (without remote human operator) is asymptotically stable with t as its action reference. This would allow the use of Theorem 1 and proves the (asymptotical) stability of the system with respect

to the new action reference s , simply by proving that the new non-time action reference is (monotone increasing) non-decreasing function of time t . The advantage of this approach is that stability is proven independent of the human model or the statistics of time-delay. Clearly the choice of event used in this approach gives an event that is a non-decreasing function of time, since the command numbering can only increase. Therefore, the event-based planning and control for teleoperation with haptic feedback results in a stable system despite random time delay.

Examining the event-based case in Fig.3, it is clear that the update of signals in both directions is not triggered by time. Also since the buffering effect of delay is eliminated then the haptic force felt by the operator is the most up-to-date one; there could not have been any change in the system status meanwhile because there are no velocity commands flowing in the network. The same thing applies to the velocity received by the robot, it is the most up-to-date one since there were no new forces flowing in the network that could have generated new commands. This implies that the operator and the robot are always synchronized in event regardless of time delay and its variation. These two different entities in the system can not be at different events; in a closed loop system, the feedback obtained has to correspond to the most up-to-date status of the plant being controlled.

From the analysis in the previous section it is clear that the frequency of events is a function of time delay. Since we have an event per round trip delay then the frequency by which the haptic information is fed back is a function of time delay. This implies that the force will be sampled once for each round trip delay and since this delay is variable then the sampling of force is variable. From Shannon's sampling theorem we know that in order to reconstruct a signal perfectly from a sampled one, the sampling rate should at least be twice F_{max} , where F_{max} is the highest frequency component of the original signal. So as long as the round trip delay is less than the inverse of the highest frequency component of the force applied on the robot, then the fed back haptic force can be used to regenerate the actual one almost perfectly and thus the system transparency is achieved. Once the time delay increases beyond that value the transparency will decrease accordingly. So this approach does not ensure perfect transparency regardless of time, but it does ensure that the best transparency will be achieved for a certain time delay.

5 Experimental Implementation and Results

The details of several experimental setups will be given with their results. These results will confirm the analysis done in the previous section. The first scenario is the teleoperation of a mobile robot with haptic feedback. The feedback in this case corresponds to the distance to obstacles detected in the environment. So the operator is able to sense the obstacles in front of the robot before hitting them. The general model of such a system is shown in Fig.4.

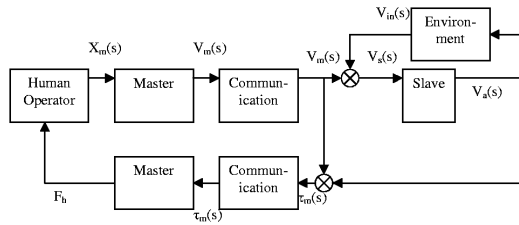


Fig. 4. The system architecture of the mobile base teleoperation system.

This model shows that the operator is sending velocity commands and receiving haptic information corresponding to sensory information from the environment. This system was experimented with an operator in Hong Kong controlling the robot at Michigan State University in real-time. The results in Fig.5 are showing the plot of time versus s , the event, in the first row. The other plot in the first row shows the desired rotational velocity. The second row shows the desired velocities in x and y directions, which is called V_m . The third row displays the actual velocities in both directions, which is called V_a . Next we illustrate the force that is played by the joystick in both directions, called τ_m . The last row displays the same plot, which is the closest detected distance to obstacles in the environment.

As seen, the actual velocities are tracking the desired ones as long as the distance to obstacles is less than a critical distance. Therefore, the system output stability is illustrated. In addition, the change of the desired and actual velocity direction is occurring at the same event, thus the system is synchronized.

The second system implemented was a mobile manipulator that is controlled in real-time, where the operator sends a velocity command that is divided by the local controller between the arm and the mobile base. This division is done to achieve an acceptable posture for the arm, so if the robot arm is almost fully extended or extracted, the mobile platform will be repositioned to achieve a better posture. The haptic feedback in this case is actual force detected by the force/torque sensor mounted on the gripper.

The experimental procedure was to have the operator in Hong Kong move the robot in x and y directions randomly. Meanwhile, a person will subject the robot gripper to forces in the x and y directions randomly. The average frequency of communication was $3.1Hz$, which implies that on average we had $3.1events/sec$.

The main interest was in how close the actual velocity of the robot tip was following the desired velocity specified by the operator, and in how close was the force felt by the operator close to the one detected by the force/torque sensor. In other words, how were the operator's intentions executed by the mobile manipulator and how the environment was felt by the operator, despite time delay and time delay variance.

The results are shown in Fig.6 and Fig.7. In Fig.6 the first row gives the operator's desired velocity in the x and y directions with respect to time. The second row is the actual velocity of the arm tip in the x and y directions with

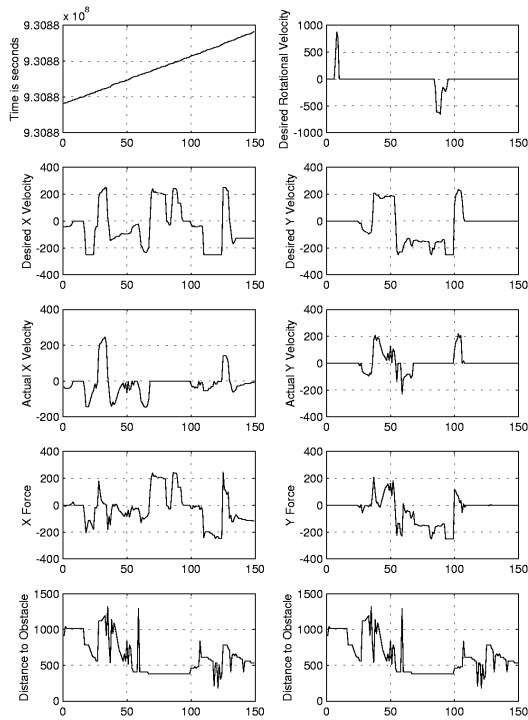


Fig. 5. The behavior of the system during the control from Hong Kong.

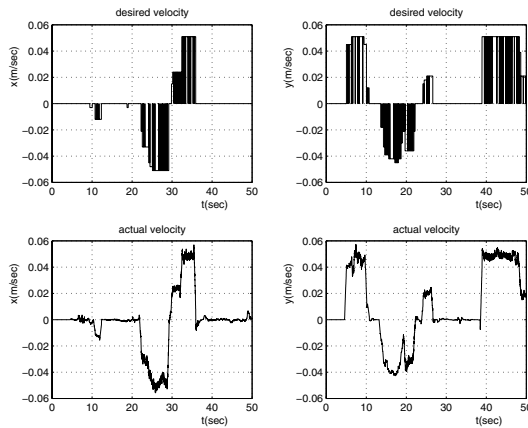


Fig. 6. The velocity performance of the system.

respect to time. The first thing to note is that the desired velocity is a step function since it is a sampled version of the continuous motion of the joystick, where the sampling rate is a function of the time delay faced and the advance

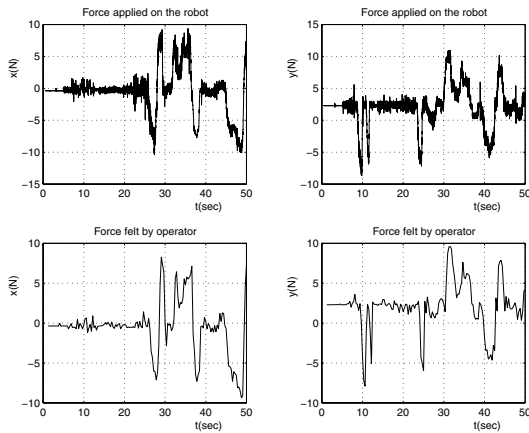


Fig. 7. The force performance of the system.

of the event. More importantly we see that the actually velocity is almost a continuous version of the desired one and it is tracking it very closely.

As for Fig.7, the first row is the force detected by the force/torque sensor and the second row is the force felt by the operator with respect to time. Note that the sensor is detecting vibrations all the time, this is due to its sensitivity. As for the force felt, it is clear that it is a step function that is almost tracking the actual one. The only slight deformation shown is due to the variable time delay faced between the two machines.

It is clear from these results that the system is stable and event-synchronized since the actual behavior of the robot is tracking the desired one and the force felt by the operator is very close to the actual one the robot is detecting. In addition, the system has high transparency revealed by the fact that the force felt by the operator is very close to the one experienced by the robot. Importantly all those results were achieved under variable time delay with no assumptions regarding it.

The third experiment done was a multi-operator at multi-site mobile manipulator teleoperation with haptic feedback. In this experiment the mobile manipulator (Robotics and Automation lab, Michigan State University), operator1 (Robot Control lab, Chinese University of Hong Kong) and operator2 (Nagoya University, Japan) were connected via the Internet. As seen in the model of the system in Fig.8, the operators send velocity commands and receive haptic information that correspond to the velocity sent by the other operator. This implies that the force felt by each operator corresponds to the intentions of the other one. This setup increases the efficiency of collaboration since it reduces the amount of contradicting commands sent and reflects the intentions of the other operator faster than visual feedback alone.

One of the operators was requested to follow the force felt. This implies that the slave would eventually track the motion of the master. The results of one

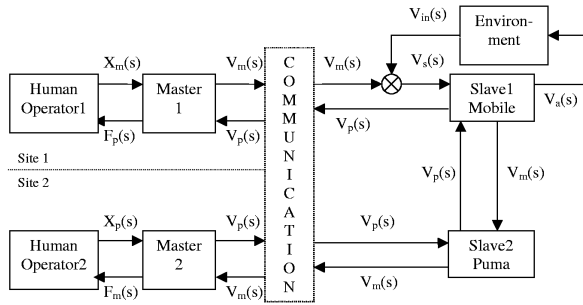


Fig. 8. The system architecture of the multi-operator mobile manipulator teleoperation system implemented.

such experiment are seen in Fig.9, where the operator in Japan is controlling the mobile (slave) and the operator in Hong Kong is operating the puma (master).

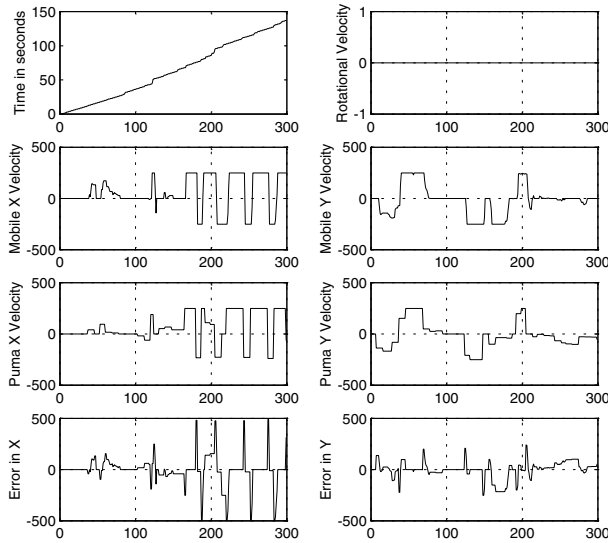


Fig. 9. System Behavior while being controlled from Hong Kong and Japan.

The results show that the desired velocity of the mobile is tracking that of the puma in real-time. The top row of Fig.9 shows a plot of time versus s , the event. The other plot in the top row shows the desired rotational velocity. The second row shows the desired velocities of the mobile in x and y directions, V_m . The Third row plots the desired velocities of the puma in x and y directions, V_p . The plots of V_m and V_p also correspond to the forces fed back to operator2

and operator1 consecutively. The last row is the error between the mobile and the puma desired velocities in both directions.

The main points to note, are the synchronization and fast response. It is clear that both robots are event synchronized since the shift in direction occurs almost at the same s. Fast response is clear from the sharp decrease in the error between the velocities of the two robots.

Another experiment, that reflects the generality of the approach, is a micro manipulator teleoperation. A PVDF (Polyvinylidene fluoride) micro-tip was used as a piezoelectric sensor for the Internet force-feedback experiment [24]. This tip is about 2.5mm long with about 0.8mm at the triangular base. The output from this sensor is amplified using an inverted amplifier with feedback gain of 50. Its signal is then feed to a 8255 analog-to-digital conversion (ADC) card connected to a PC for signal transmission to the Internet. This experimental setup is housed in the Advanced Microsystems Laboratory (AML) of The Chinese University of Hong Kong.

Sensor tip is attached to an x-y computer-control positioning table, which can be control via the Internet by a force reflection joystick in the Robotics and Automation Laboratory (RAL) at Michigan State University. A cantilever is attached to a vibration drum and has a tip vibration of 100mm to 1mm from the frequency range of 1 to 120 Hz. The AML sensor tip position can be manipulated by the RAL joystick to contact the vibrating cantilever. The RAL operator observes the AML tip position using a video conferencing software. The force of the vibrating cantilever sensed by the tip is sent to RAL via the Internet. Once the force is received the force feedback joystick plays it. After that the operator generates a new movement command to be sent to the sensor via the Internet.

The experimental results presented here relate to the testing done between Hong Kong and Michigan State. During this experiment the operator (RAL) sends position increment commands and receives force feedback from the sensor (AML). The position increments are sent for both x and y axes while the force is sensed only in the y axis. The commands sent are random, which is typical of a teleoperation scenario. This makes approaches based on prediction of forces or virtual forces non-realistic. Therefore, actual force had to be sensed and fed back. Fig.10 presents plots of the force felt by the operator, the force sampled for the sensor and the error between them. As seen the force felt is closely following the one sampled from the sensor. Although this is not occurring at the same time instant, since both plots are with respect to local and not global time, the system is still stable and event synchronized. Despite the random time delay experienced between Hong Kong and Michigan State, the system performance is stable as seen from the error, which is constantly converging to zero and has a small value at all times.

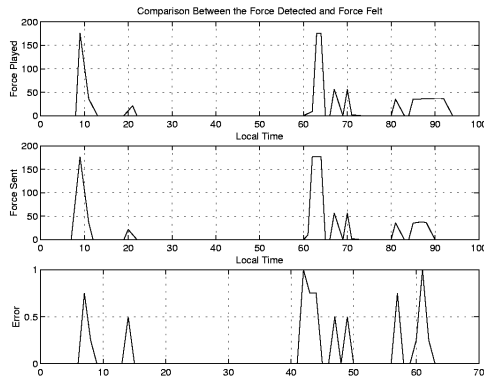


Fig. 10. Comparison between the forces felt and the ones sent.

6 Conclusions

This paper examined several issues relating to supermedia in teleoperation over the Internet. The effects of random time delay faced over the Internet were studied. Stability, synchronization and transparency were analyzed and a new planning and control method that ensures those features was presented. The event-based approach was theoretically and experimentally shown to be adequate for the planning and control of teleoperation with haptic feedback.

Several experimental setups were examined and their results presented. All of which showed that the event-based approach resulted in a stable, synchronized and transparent system regardless of time delay and its variance. Other advantages of this approach is that it is independent of the system model, human model and environment model. Therefore, it can be easily adopted to any teleoperation system that includes haptic feedback.

References

1. D. Kwon, K. Y. Woo, H. S. Cho, "Haptic Control of the Master Hand Controller for a Microsurgical Telerobot System", IEEE Int. Conf. on Robotics and Auto., 1999.
2. M. Tanimoto, F. Arai, T. Fukuda, and M. Negoro, "Force Display Method to Improve Safety in Teleoperation System for Intravascular Neurosurgery", IEEE Int. Conf. on Robotics and Automation, Vol. 3, pp. 1728-1733, May 1999.
3. S. E. Everett, R. V. Dubey, "Model-Based Variable Position Mapping for Telerobotic Assistance in a Cylindrical Environment", IEEE Int. Conf. on Robotics and Automation, Vol. 3, pp. 2197-2202, May 1999.
4. L. Hsu, R. Costa, F. Lizarralde, J. Soares, "Passive Arm Based Dynamic Positioning System for Remotely Operated Underwater Vehicles", IEEE Int. Conf. on Robotics and Automation, Vol. 1, pp. 407-412, May 1999.

5. G. Hirzinger, B. Brunner, J. Dietrich, and J. Heindl, "Sensor-Based Space Robotics-ROTEX and Its Telerobotic Features", *IEEE Trans. on Robotics and Automation*, Vol. 9, No. 5, October 1993.
6. R. Luo, W. Z. Lee, J. H. Chou, and H. T. Leong, "Tele-Control of Rapid Prototyping Machine Via Internet for Automated Tele-Manufacturing", *IEEE Int. Conf. on Robotics and Automation*, Vol. 3, pp. 2203-2208, May 1999.
7. R. Anderson, M. Spong, "Asymptotic Stability for Force Reflecting Teleoperators with Time Delay", *The Int. Journal of Robotics Research*, Vol. 11, April 1992.
8. R. Anderson, M. Spong, "Bilateral Control of Teleoperators with Time Delay", *IEEE Trans. on Automatic Control*, Vol. 34, No. 5, May 1989.
9. K. Brady, T. J. Tarn, "Internet-Based Remote Teleoperation", *Proceedings of the 1998 IEEE Int. Conf. on Robotics and Automation*, Leuven, Belgium, May 1998.
10. D. Pai, "ACME, A Telerobotic Measurement Facility for Reality-Based Modelling on the Internet", *IROS Workshop on Robots on the Web*, Canada, 1998.
11. R. Simmons, "Xavier: An Autonomous Mobile Robot on the Web", *IROS Workshop on Robots on the Web*, Canada, 1998.
12. N. Y. Chong, T. Kotoku, K. Ohba, K. Komoriya, "Remote Coordinated Controls in Multiple Telerobot Cooperation", *IEEE Int. Conf. on Robotics and Auto.* 2000.
13. L. F. Penin, K. Matsumoto, S. Wakabayashi, "Force Reflection for Time-delayed Teleoperation of Space Robots", *IEEE Int. Conf. on Robotics and Auto.*, 2000.
14. M. Stein, "Painting on the World Wide Web: The PumaPaint Project", *IROS Workshop on Robots on the Web*, Canada, 1998.
15. P. Saucy, F. mondada, "KhepOnTheWeb: One Year of Access to a Mobile Robot on the Internet", *IROS Workshop on Robots on the Web*, Canada, 1998.
16. W. Kim, B. Hannaford, and A. Bejczy, "Force-Reflection and Shared Compliant Control in Operating Telemanipulators with Time Delay", *IEEE Trans. on Robotics and Auto.*, Vol. 8, April 1992.
17. G. Leung, B. Francis, J. Apkarian, "Bilateral Controller for Teleoperators With Time Delay via μ -Synthesis", *IEEE Trans. on Robotics and Automation*, Vol 11, No. 1, February 1995.
18. M. Otsuka, N. Matsumoto, T. Idogaki, K. Kosuge, T. Itoh "Bilateral Telemanipulator System With Communication Time Delay Based on Force-Sum-Driven Virtual Internal Models", *IEEE Int. Conf. on Robotics and Automation*, 1995.
19. G. Niemeyer, J. Slotine, "Stable Adaptive Teleoperation", *IEEE Journal of Oceanic Engineering*, Vol 16, No. 1, January 1991.
20. R. Riedi, M. Course, V. Ribeiro, R. Baraniuk, "A Multifractal Wavelet Model with Application to TCP Network Traffic", *IEEE Trans. on Info. Theory* April 1999.
21. W. Leland, M. Taquq, W. Willinger, D. Wilson, "On the Self-Similar Nature of Ethernet Traffic ", *IEEE/ACM Trans. on Networking*, Vol. 2, No. 1, February 1994.
22. T. J. Tarn, N. Xi, A. Bejczy, "Path-Based Approach to Integrated Planning and Control for Robotic Systems", *Automatica*, Vol. 32, No. 12, pp. 1675-1687, 1996.
23. I. Elhajj, N. Xi, Y. Liu, "Real-Time Control of Internet Based Teleoperation with Force Reflection", *IEEE Int. Conf. on Robotics and Automation*, 2000.
24. King W. C. Lai, Carmen K. M. Fung, Wen J. Li, Imad Elhajj, Ning Xi, "Transmission of Multimedia Information on Micro Environment via Internet", *International Conference of the IEEE Industrial Electronics Society*, Nagoya Japan, 2000.

Author Index

- Ahmed, T. 110
Albaghdadi, M. 271
Almeida, M.J.B. 222
Almeroth, K.C. 256, 329
Atmaca, T. 204
- Baras, J.S. 227
Bauer, M. 142
Borsos, T. 85
Boutaba, R. 57
Braun, T. 160
Brewster, G.B. 313
Briley, B. 271
Buridant, G. 110
- Campbell, A.T. 43
Chahed, T. 204
Chieng, D. 299
- Elarde, J.V. 313
Elbiaze, H. 204
Elhadj, I. 359
Evens, M. 271
- Feng, W.-c. 1
Fukuda, T. 359
Fung, W. K. 359
- Gao, J. 71
Granville, L.Z. 222
Grossner, C. 285
Gu, G. 193
- Hamlen, M. 271
Harroud, H. 285
He, Y. 210
Hébuterne, G. 204
Ho, I. 299
- Iraqi, Y. 57
- Jagannathan, S. 329
Jorgenson, J. 241
- Kaga, T. 359
Kammoun, K. 96
Kang, C.-H. 198
Karmouch, A. 285
Katchabaw, M. 142
Khalil, I. 160
- Lakhdissi, M. 285
Lecuire, V. 96
Lee, S.-H. 198
- Lee, S.-J. 198
Lelescu, D. 128
Lepage, F. 96
Li, H. 227
Li, W. J. 359
Lim, K.-S. 345
Liu, Y. h. 359
Lutfiyya, H. 142
- Manikopoulos, C. 241
Marshall, A. 299
Mehaoua, A. 110
Messerschmitt, D.G. 175
Miyabayashi, M. 29
Miyahara, H. 29
Molenkamp, G. 142
Murata, M. 29
- Parr, G. 299
Petiwala, M. 271
- Reason, J.M. 175
Ribeiro, M.B. 222
Rubin, I. 71, 215
- Saraç, K. 256
Schonfeld, D. 128
Seelam, N. 1
Seok, S.-J. 198
Sethi, P. 1
Son, S. 16
Stadler, R. 345
Sukkar, R. 271
- Tarouco, L.M.R. 222
- Vicente, J. 43
Vincent, P. 215
- Wakamiya, N. 29
Wang, Q. 210
- Xi, N. 359
- Yang, S. 210
Yuan, Y. 193
- Zhai, M. 193
Zhang, Z. 241
Zhao, L. 210
Zhong, Y. 210