NETWORK SUPPORT
FOR APPLICATIONS REQUIRING QUALITY OF SERVICE
IN HETEROGENEOUS ENVIRONMENTS

A Dissertation Presented

by

VICTOR FIROIU

Submitted to the Graduate School of the
University of Massachusetts Amherst in partial fulfillment
of the requirements for the degree of

DOCTOR OF PHILOSOPHY

September 1998

Department of Computer Science

NETWORK SUPPORT
FOR APPLICATIONS REQUIRING QUALITY OF SERVICE
IN HETEROGENEOUS ENVIRONMENTS

A Dissertation Presented

by

VICTOR FIROIU

Approved as to style and content by:

Donald Towsley, Chair

Weibo Gong, Member

Roch Guérin, Member

James F. Kurose, Member

Krithivasan Ramamritham, Member

James F. Kurose, Acting Department Chair
Department of Computer Science

Dedicated to

My wife Laura and my son Vlad

and

My parents, Radu Sebastian Firoiu and Elisabeta Firoiu

ACKNOWLEDGMENTS

It is with deep respect and appreciation that I acknowledge my advisors, Professors Don Towsley and Jim Kurose, for their contributions in the development of my research and their guidance in my professional growth. Under their leadership, I have learned how to do research and how to communicate it effectively to others. Their broad vision, deep insight, valuable advice and continual encouragement have made this work possible. I am also indebted to them for their caring personality, great concern and readiness to assist me in matters outside of academics.

I would like to thank Professor Krithi Ramaritham for his involvement on my dissertation committee and the many contributions he has made.

I would also like to thank Professor Weibo Gong for his involvement and guidance as a member of my dissertation committee.

I am deeply indebted to Dr. Roch Guérin for his guidance and collaboration during my visits at I.B.M. T.J. Watson Research Center and for his participation in my dissertation committee.

Many thanks are due to my fellow graduate students. Sambit Sahu and Jitendra Pahdye and I have worked closely together and I have received much enjoyment and benefit from our collaborations. I have had many helpful discussions with Zhi-Li Zhang on various research issues. I have enjoyed numerous discussions with fellow graduate students in the networking group: Supratik Bhattacharya, Timur Friedman, Sneha Kasera, Sue B. Moon, Erich Nahum, Ramachandran Ramjee, Dan Rubenstein, Jim Salehi, Subhabrata Sen, Maya Yajnik and David Yates.

Finally, I must mention the tremendous support I have received from my family. I wish to thank my wonderful wife Laura and son Vlad for their love and encouragements during my years of graduate school.

My deepest gratitude goes to my parents, Radu-Sebastian and Elisabeta Firoiu. Their love, sacrifice, support and care have always been the greatest inspiration in my life and in my pursuit for knowledge.

ABSTRACT

NETWORK SUPPORT
FOR APPLICATIONS REQUIRING QUALITY OF SERVICE
IN HETEROGENEOUS ENVIRONMENTS

SEPTEMBER 1998

VICTOR FIROIU

Diploma of Engineer, POLYTECHNIC UNIVERSITY, BUCUREŞTI, ROMÂNIA

M.Sc., UNIVERSITY OF MASSACHUSETTS, AMHERST

Ph.D., UNIVERSITY OF MASSACHUSETTS AMHERST

Directed by: Professor Donald Towsley

Group communication, be it one-to-many (such as TV broadcasting) or many-to-many (such as teleconferencing) is becoming increasingly important because it enables the widespread dissemination of information (such as in today's Word Wide Web) and the collaboration between remote groups. This kind of communication can be supported efficiently in digital networks through multicasting, a technique of non-redundant simultaneous data transmission from a sender to a set of receivers. Multicast applications such as voice and video require Quality of Service guarantees (such as maximum packet delay, packet loss probability), which can be provided by reserving network resources.

In this dissertation we propose solutions to several critical problems of multicasting in heterogeneous environments: differences in network resource availability, differences in

receiver Quality of Service requirements, differences in network resource availability and differences in resource reservation protocols.

In the first part of the dissertation we consider the problem of resource reservation for multicast sessions in the context of both network and receiver heterogeneity. We develop centralized and distributed algorithms that accommodate this heterogeneity by performing a differentiated per-link resource reservation. We apply these algorithms in the context of packetized voice and MPEG video multicast connections over wide area networks. We find that our algorithms enable a network to carry as much as a 50% more traffic compared to the case where the network does not accommodate heterogeneity.

In the second part of the dissertation we present algorithms for local (link) admission control and resource reservation at an Earliest Deadline First packet scheduler that provides heterogeneous packet delay guarantees at a link. When the data transmission is characterized by piecewise linear traffic envelopes, we show that the algorithms have very low computational complexity and thus, practical applicability.

In the third part of the dissertation we focus on resource reservation protocols in the heterogeneous environment of IP over ATM networks. We describe a method for establishing reservations in the ATM network for IP flows (named ATM shortcutting). This method provides better performance to IP flows by avoiding the IP processing of IP packets, and better utilization of ATM network resources.

In the last part of the dissertation we quantify the improvement in utilization of IP/ATM network when using ATM shortcutting. We present methods to evaluate this benefit given an IP/ATM network topology, link capacities and traffic patterns. We use this methods in simulation experiments using random networks. These experiments indicate that in many cases ATM shortcutting brings benefits in network utilization when it decreases the average length of network routes.

TABLE OF CONTENTS

LIST OF TABLES

Table **Page**

LIST OF FIGURES

Figure **Page**

CHAPTER 1

INTRODUCTION

1.1 Overview

Multi-party communication, such as audio or video broadcasting or teleconferencing, where many listeners can be addressed simultaneously, plays a central role in information dissemination and group collaboration. The Internet, due to its established status as a global data network, is the medium of choice for such communication. As the Internet's transmission capabilities have increased exponentially since its inception, an ever increasing part of this communication is Continuous Media, such as audio and video.

The possibility of multi-party Continuous Media transmission over large networks has enabled a variety of applications. Some of the application areas include distance learning, education and training, information and entertainment services, interactive virtual environments, desktop video conferencing and collaboration, and digital archives and libraries. In today's Internet, audio or video conferencing and radio/video broadcasting are becoming increasingly popular, although efficient support for multi-party Continuous Media transmission is lagging far behind the demand.

Figure 1.1 illustrates a tele-lecturing application. A teacher speaks, shows slides and draws on a board in front of a video camera. The audio and video is digitized and sent as data packets over a wide area network (e.g., the Internet) to a set of receivers, that are computers connected to the network. Students watch the lecture displayed on their computer screens. For a good quality display of CM on computer screen and speakers, the playout of CM must not be interrupted or have errors. This requires the network to transfer data packets in time for a continuous playout, and to have a low rate of packet loss. In

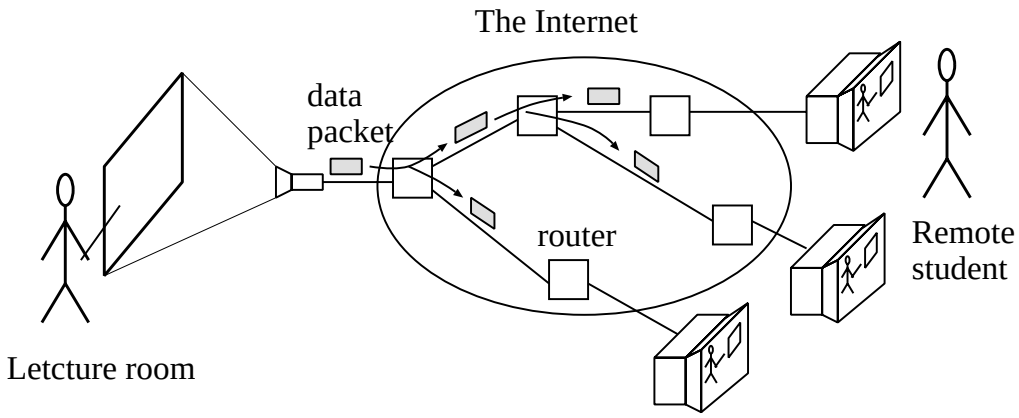


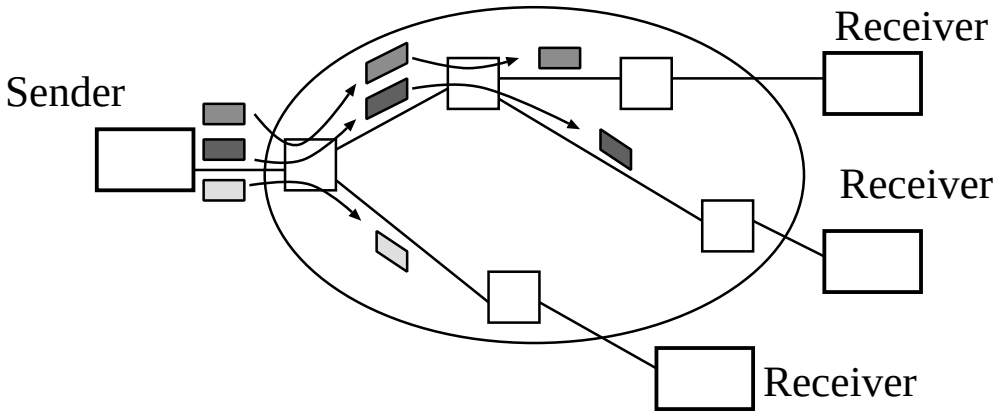
Figure 1.1. An example of tele-lecturing over the Internet

addition, interactive communication (for example when a student wishes to interrupt the exposition with a question or dialog) has an additional requirement of a very small delay of packet transfers.

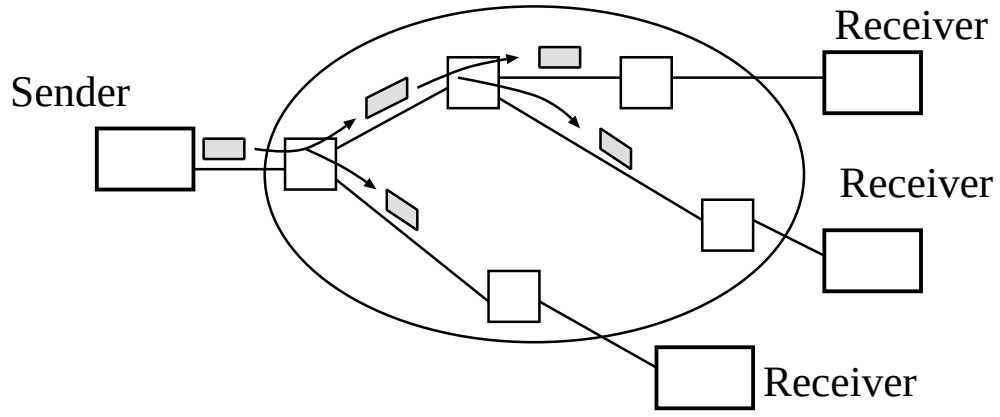
In this dissertation we are interested in network support for applications with Quality of Service requirements. These applications include, besides Continuous Media, other real-time applications such as industrial automation control. But Continuous Media applications are very pervasive, and thus they are our primary focus.

In order to guarantee the QoS requirements of an application, the network needs to reserve resources on behalf of the given application. Another aspect of the multi-party communication is the need for simultaneous transmission from a source to a set of receivers.

An essential technique for efficient multi-party digital communication is *multicasting*, see Figure 1.2. The one-to-many transmission can be achieved through multiple one-to-one transmission (a.k.a. multiple unicasting), as in Figure 1.2(a), where N copies of each data packet are created at the source and transmitted independently to each of the N receivers. Alternatively, through multicasting, Figure 1.2(b), one data packet is sent by the source and copies of the packet are made at the branching points (routers) inside the network. Clearly,



a). Multiple unicasting



b). Multicasting

Figure 1.2. Two methods for multi-party communication

multicasting has the potential to utilize network resources more efficiently than multiple unicasting.

In this dissertation we consider the problem of network support for multicast application with QoS requirements in heterogeneous environments. The goal of the work is

to provide algorithms and protocols for resource reservation in a wide area network that utilize network resources efficiently and accommodate network heterogeneity.

In particular we focus on the following problems:

- How to compute the amount of reservation needed at each link of a multicast distribution tree given a network with heterogeneous link resource availability and a set of heterogeneous receiver requirements.

- How to perform a fast admission control and resource reservation at an EDF scheduler, given a set of flows that are characterized by piecewise linear envelopes and having with heterogeneous maximum packet delay requirements.

- How to provide interoperability between reservation protocols in the heterogeneous environment of IP over ATM networks while utilizing network resources efficiently. Also, in this context, how to evaluate the benefit of using this interoperation.

Our contributions to the solutions of these problems are:

- We design multicast resource reservation algorithms that accommodate link heterogeneity through link load balancing, i.e., where resource reservation is done at each link in proportion to the link's resource availability. The algorithms also accommodate heterogeneity in receiver requirements by enabling a receiver with loose QoS requirement to take advantage of the resource reservation required by a more stringent QoS requirement of another receiver.

- We develop algorithms for admission control and resource reservation at an EDF scheduler. These algorithms have a very low computational time and result in an almost optimal utilization of link bandwidth.

- We propose a method for establishing ATM shortcuts, the transmission of IP packets through reserved ATM routes, which avoids the overhead of IP packet processing.

For resource reservation of ATM shortcuts, we propose an approach for interoperation between RSVP protocol and ATM signaling. Our solution requires minimum extensions to each protocol.

- We present a method to quantify the benefit in network utilization because of using ATM shortcuts for resource reservation, given an IP/ATM network topology and link capacities. We apply this method to a large population of random networks and we find the IP/ATM topological characteristics that indicate when there is most likely to benefit from ATM shortcutting.

In the rest of this chapter we overview the general network architecture needed to support multicast applications with Quality of Service guarantees, and use this overview to present the problems addressed in this dissertation. The chapter concludes with a summary of contributions and the structure of the dissertation.

1.2 Framework for Multicast Resource Reservation

In this section we present a general architecture of network support for multicast applications with Quality of Service guarantees.

We begin with a description of the general architecture for supporting applications that multicast data flows and have QoS requirements. We illustrate a multicast CM transmission over a wide area network in Figure 1.3. Audio/video signals produced by a camera are digitized and packetized according to a coding scheme (for example PCM for audio and MPEG for video). Data packets are sent by a computer to specific receivers through a network interface. The network transmits the packets through intermediate routers, that choose the packet's path according to their destination address. If there are multiple receivers, the packets have a multicast (or group) destination address, and the routers map this address to the proper receivers. For efficient data delivery, the routers establish a mul-

ticast (delivery) tree connecting the sender to its receivers and make copies of the packets only at the tree's branching points (as in Figure 1.2(b)).

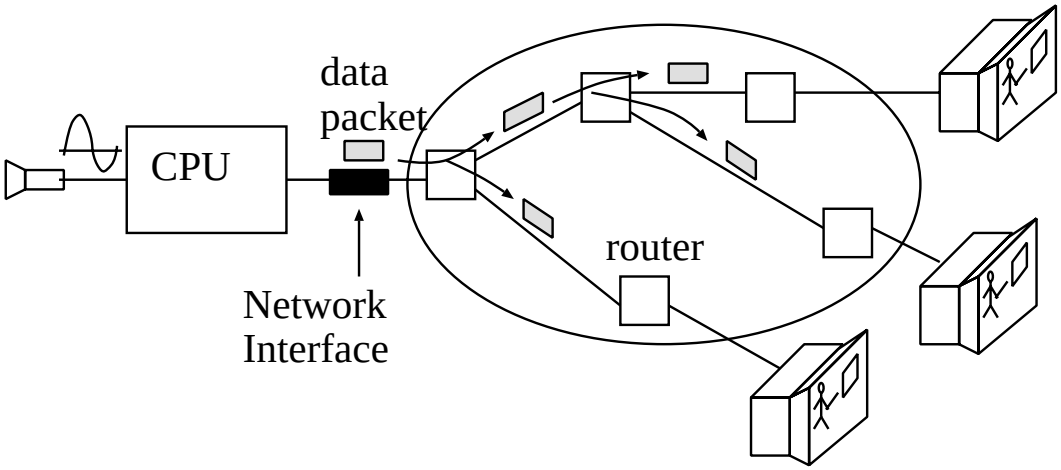


Figure 1.3. An example of a Continuous Media transmission

In addition, Continuous Media applications have performance requirements. For example interactive multicasting (e.g., where the users participate in a dialog) of CM requires a guarantee that the transmission between a sender and all its receivers is done within a small amount of time. Such QoS guarantees can be provided by reserving resources at each router on the transmission's path. In Figure 1.4 we illustrate the forwarding of packets inside a router. The packets are forwarded from input to output ports, and we observe that CM packets compete for transmission at the output port (link) with other packets coming from the same and other input ports. More precisely, the switching fabric of the router directs the incoming packets to the proper output ports according to their destination address. If the destination address is a multicast address and receivers subscribing to this address are reached through several output ports of the router, the switching fabric makes copies of the input packets and and sends a copy to each of the necessary output ports according to a multicast routing protocol (for example DVMRP [99] or MOSPF [74]).

At an output port (see Figure 1.5), packets coming from various input ports are queued at a buffer waiting to be transmitted on the output line. The waiting is necessary because

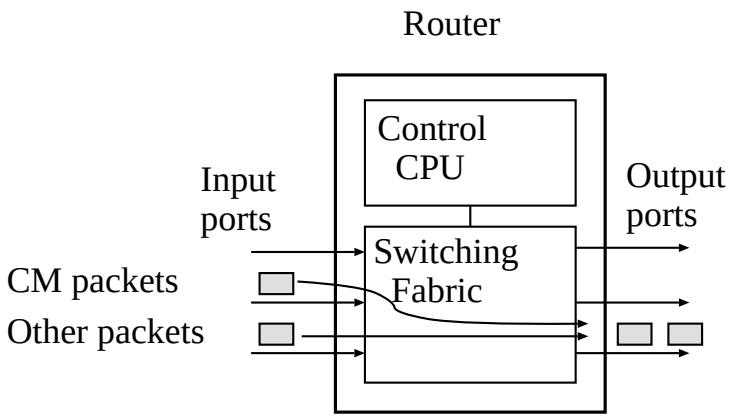


Figure 1.4. Packet transmission in a router

the aggregate packet arrival rate from input ports can surpass the capacity of the output line for a limited time. Thus, the CM packets can be delayed behind other packets waiting for transmission, or can dropped if the buffer is full upon their arrival. For a good quality CM play-out at the receivers, the CM packets need to have a low drop rate (packet loss probability). Also, for interactive CM, the packets need to have a small delay from sender to receiver. Maximum packet delay, packet loss probability, maximum packet delay jitter are collectively known as Quality of Service guarantees.

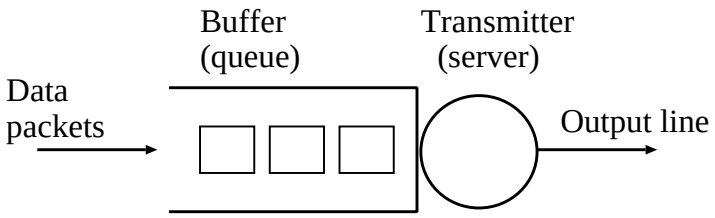


Figure 1.5. Packet processing at an output port

To provide QoS guarantees to the packets of a flow, there is a need for resource reservation. A very simple example of reservation is peak rate reservation. For example, if a CM flow has a rate that is less than r, a reservation at an output port of capacity c can be achieved by setting aside a capacity r for the flow and leaving $c \Leftrightarrow r$ for the rest of the

traffic (see Figure 1.6). In this way the delivery of CM packets can be guaranteed without queuing delay and without loss, at the price of reducing service to the rest of the traffic. In Chapter 3 we will examine other, more efficient methods for providing QoS guarantees.

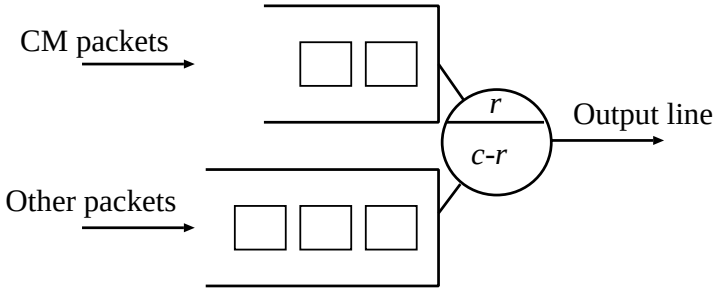


Figure 1.6. Rate reservation at an output port

Observe that reservations require specific router support. Before reserving resources, the router must verify that the resources needed to meet a QoS guarantee are available. This operation is called (link or local) admission control. After the reservation, a packet classifier needs to be installed to separate the packets of the reserved flow from the rest of traffic and have them served by the reserved transmission capacity. The information about the flow's traffic characterization (for example its rate) and the required Quality of Service (for example maximum packet delay) needs to be communicated from the multicast application to the routers on the flow's path, and the outcome of the link admission control and resource reservation must be communicated back to the application.

This is accomplished by a reservation protocol. In Figure 1.7 we give an example of a reservation protocol. A source sends a $Setup$ message to the multicast address subscribed by three receivers. The message contains information about the flow characteristics. The message is intercepted by each router in the multicast tree (route). Each router records in the message the minimum packet delay that it can guarantee. When a $Setup$ message reaches a receiver, the receiver compares the minimum packet delay accumulated along the path with its own maximum packet delay requirement. If the requirement is larger than the cumulative delay, it means that the requirement can be guaranteed end-to-end, and a $Reserve$

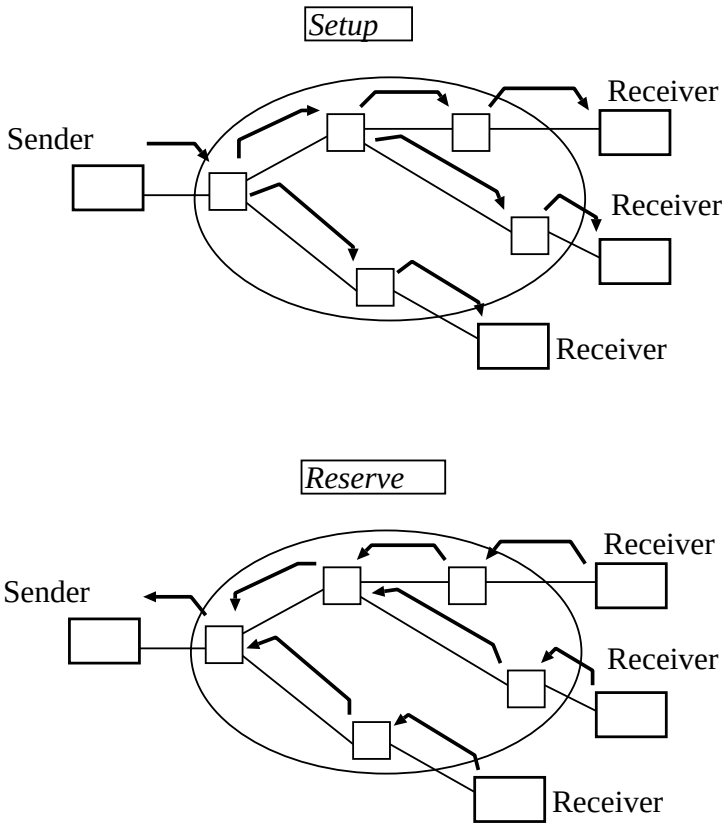


Figure 1.7. A reservation protocol

message is sent back on the same path. The $Reserve$ message contains a reservation request with a maximum packet delay objective. Again each router intercepts the message, performs a reservation, computes the local delay thus guaranteed, subtracts it from the end-to-end objective in the $Reserve$ message and sends it on the path to the source. The source is thus informed about the outcome of the admission control and resource reservation, and can transmit CM data on the reserved multicast route.

1.3 Problems Addressed in the Dissertation

The goal of the work presented in this dissertation is to design and evaluate methods for efficient resource reservation for multicast sessions with QoS requirements in heterogeneous environments, methods that result in high network utilization and that make efficient

use of network resources. We now introduce the problems in multicast resource reservation algorithms and protocols that we address in this dissertation.

1.3.1 Admission Control and Resource Reservation for Multicast Sessions

As mentioned earlier, multicast communications with QoS requirements need per-link resource reservations. In principle, many different combinations of per-link reservations can satisfy a given end-to-end QoS requirement. We are interested in finding a combination of such reservations that accommodates the different link resource availability and is aimed at balancing the load on the links accross the network. We also investigate how a reservation for one receiver can be used to guarantee QoS to another receiver. The algorithms developed should have low computation time and provide high network utilization (support a high number of admitted flows).

1.3.2 Efficient Admission Control of Piecewise Linear Traffic Envelopes at Earliest Deadline First Schedulers

For providing per-link delay guarantees, we consider the Earliest Deadline First packet scheduling discipline at an output port of a router. The CM data flows are characterized by an upper bound on the amount of data transmitted during a time interval. This upper bound is called an envelope. The computation of admissibility and resource reservation is efficient if the envelope is a piecewise linear function. We are interested in fast algorithms for computing the admissibility and resource reservation of flows that are characterized by piecewise linear envelopes and require a maximum packet delay guarantee. The high speed of the algorithm is essential in providing resource reservation with low latency, and thus providing short waiting times to users. Furthermore, the computation time should not increase noticeably as the number of flows admitted at a link is large (order of $10^3 \Leftrightarrow 10^4$ are typical for today's networks).

1.3.3 Support for RSVP-based Services over ATM networks

Today's wide area networks such as the Internet span many administrative domains, some of them implementing Asynchronous Transfer Mode [51] as data link and network layer technology for high speed networking. IP networks spanning ATM networks are known as IP over ATM (or IP/ATM) networks. Since most applications today originate in the Internet, they use the IP protocol stack which has RSVP [14] as a possible resource reservation protocol. In the ATM networks, reservations are performed through ATM signaling [7]. Reservations in IP/ATM networks can be done through reservations exclusively at the IP layer using RSVP, without involving the support of the underlying ATM network. Alternatively, the reservation can be done using RSVP in the IP network and ATM signaling in the IP/ATM network. The potential benefit of integrated IP and ATM reservations is a higher utilization of ATM resources and a reduction in the overhead of IP processing of IP packets. We investigate the issues related to the interoperation between RSVP and ATM signaling: the mechanism of initiating ATM signaling messages from RSVP messages and vice-versa, and the translation of traffic information between the two protocols.

1.3.4 Performance Evaluation of ATM Shortcut Connections in Overlaid IP/ATM Networks

The methods to interface RSVP and ATM signaling come also with some tradeoffs. For example, the interfacing complicates the code in the control module of the routers and also ATM signaling can have a significant overhead. But the benefits in higher utilization of network resources can outweigh the above disadvantages. We are interested in developing methods to quantify these benefits. These methods can be used by network engineers to assess when, given a IP/ATM network, it is beneficial to configure RSVP interaction with ATM signaling.

1.4 Contributions of the Dissertation

- In the first part of the dissertation, we focus on admission control and resource reservation for a multicast application, while accommodating heterogeneity in receiver requirements and link resource availability. We develop centralized resource reservation algorithms, where the QoS computation is done at a single place (for example the multicast source node), and distributed algorithms, where the QoS computations and resource reservation are done at the nodes in the multicast tree. These algorithms accommodate heterogeneity on link resource availability and heterogeneity in receiver QoS requirements. We evaluate these algorithms in two different network settings: one having packet loss probability as the QoS metric and the other, maximum packet delay. Our algorithms prove to be highly effective. Our simulation results show that when our algorithms are used, the network can accept about 10% more sessions for the loss probability model and about 50% more sessions for the delay model compared to the case where the network does not accommodate heterogeneity.

- As mentioned before, end-to-end maximum packet delay can be guaranteed eficiently when the packet scheduling discipline at each link is Earliest Deadline First (EDF). EDF can also accommodate efficiently heterogeneous packet delay guarantees required by different flows at the same link. In the second part of the dissertation, we consider admission control at EDF schedulers. We apply the admission control conditions put forward by [66] to flows characterized by multiple-segment envelopes. We develop a first set of algorithms with a computation complexity of $O(KN)$, where N is the number of flows admitted in the EDF scheduler at the time of algorithm invocation and K is the number of segments per envelope. A second set of algorithms places the horizontal position of concave points of flow envelopes into a predefined set of values (discretization points), thus reducing the computational complexity of admission control to $O(K + L)$, where L is the number of predefined

discretization points. A set of simulation experiments show that the improvement in execution time achieved by the discrete admission control is indeed very important (240 times faster for an OC12 link) and that the algorithm's execution time is independent of the number of flows admitted. Moreover, we find that the link performance degradation of the discrete admission control relative to the exact admission control is less that 5%, while using a small number of discretization points (15). Taken together, these results suggest that these algorithms form the basis of a practical and highly efficient solution for the problem of admission control of real-time flows at EDF schedulers.

- In the third part of the dissertation we consider the problem of establishment of QoS connections in the heterogeneous environment of IP over ATM networks. For sessions whose path extends across ATM networks we investigate the interplay between RSVP flows at the network layer and ATM calls at the subnetwork layer. We use an approach, classical RSVP over ATM with shortcuts, which is intended to leverage the strengths of the ATM technology in support of applications with QoS requirements. Establishing shortcuts through an ATM network avoids the performance penalty associated with layer 3 processing in a classical IP over ATM approach. We provide solutions applicable to both unicast and multicast flows.

- ATM shortcuts have the potential of providing a better utilization of IP/ATM network resources. In the last part of the dissertation, we consider the problem of evaluation of benefit of ATM shortcuts in IP/ATM networks, i.e., the benefit of ATM routing of IP flows in ATM networks. We propose to measure the benefit of ATM shortcuts with the Network Load Ratio, that expresses the increase in the number of flows accepted by an IP/ATM network for the same network blocking probability. We develope a low complexity computation for Asymptotic Load Ratio, which estimates the Network Load Ratio relatively well (maximum relative error less than 0.12) for

underload network conditions (network blocking probability less than 0.01). An important result is a methodology for comparing network performance, which can be used to evaluate the benefit and tradeoff of ATM shortcuts, and in the more general context of network design. We use this method in simulation experiments using random networks. These experiments indicate that in many cases the utilization of an IP/ATM network increases proportionally to the decrease in the average path length when ATM shortcuts are used.

1.5 Structure of the Dissertation

In this introduction we have provided the context and motivation for four problems in the area of network support for multicast applications with QoS guarantees in heterogeneous environments. In the next four chapters we examine each of the problems in detail.

In Chapter 2 we consider the problem of resource reservation for multicast sessions in the context of both network and receiver heterogeneity. We present a solution that accommodates this heterogeneity by performing a differentiated per-link resource reservation. We present centralized and distributed algorithms for computing per-link reservations and we evaluate their computational and communication complexity. We present an application of these algorithms in the context of packetized voice and MPEG video multicast connections over wide area networks. We evaluate through simulation the algorithms' effectiveness in utilizing network resources.

In Chapter 3 we present algorithms for local (link) admission control and resource reservation at an Earliest Deadline First packet scheduler when the data transmission is characterized by piecewise linear traffic envelopes. We show that the algorithms have very low computational complexity and thus, practical applicability. We further decrease this complexity through *discrete* admission control, whereby the range of positions for the end points of linear segments of the traffic envelopes is restricted to a finite set. We present a

set of simulation experiments to evaluate their computation time and effectiveness in link resource utilization.

In Chapter 4 we present a solution for interoperation of resource reservation protocols in the heterogeneous environment of IP over ATM networks. We describe a method for establishing reservations in the ATM network for IP flows (named ATM shortcutting) and the necessary extensions to the RSVP protocol and ATM signaling.

In Chapter 5 we present methods to evaluate the benefit of ATM shortcutting given an IP/ATM network topology, link capacities and traffic patterns. We perform simulation experiments to validate these methods. In another set of simulation experiments we seek to find the IP/ATM network topologies which are most likely to benefit from ATM shortcutting.

Finally, we summarize the dissertation in Chapter 6 and suggest several possible avenues for future work.

CHAPTER 2

ADMISSION CONTROL AND RESOURCE RESERVATION FOR MULTICAST SESSIONS

2.1 Introduction

The increasing accessibility of IP-multicast on the Internet has spurred a rapid growth in the number of multicast applications using the Internet. These include audio (vat [52], NeVoT [93]) and video (nv [39]) which, being constrained to provide smooth play-out at the receiver, require connection-oriented services and the reservation of sufficient network resources in order to guarantee the desired play-out quality. Unfortunately, such services are currently not provided by the Internet and the play-out quality for such applications has been quite variable.

In this chapter we address the problem of connection setup for such multicast applications that require quality of service (QoS) guarantees on end-to-end delay or loss probability. A solution to this problem requires establishing a multicast tree (routing), checking whether or not the application can be admitted on that tree (admission control), and reserving resources. We present a general framework for developing algorithms for performing admission control and reserving the resources required to provide a desired quality of service (QoS) to a multicast connection and, within this framework, develop algorithms to solve these problems. The approach to resource reservation is based on the division of the end-to-end QoS requirement for each receiver into local QoS requirements at each of the links on the path from the source to that receiver. Important contributions of our work are algorithms for multicast resource reservation that accommodate heterogeneity in receiver requirements and link resource availability.

Last, two very different applications are given to illustrate the benefits of the algorithms developed in this chapter. The first application is to admission control of packetized voice where the QoS metric is packet loss probability. Admission control and resource reservation/reclamation algorithms based on *effective bandwidths* coupled with *generalized processor sharing (GPS)* are described and their performance evaluated. The second application is to admission control of MPEG video where the QoS metric is end-to-end delay. In this application, the algorithms are based on the *deterministic guarantee model* coupled with *Earliest Deadline First (EDF)* scheduling discipline [35, 65]. As a consequence of these applications, we observe that the choice of resource reservation algorithm has a large impact on network performance. Specifically, accommodating heterogeneity in both receiver requirements and link resource availability can increase the traffic load carried by a network by as much as 50% for a given network blocking probability.

Much of the previous work on admission control has focussed on the development of mathematical models for providing performance bounds, with applications to admission control of unicast connections (see[27, 33, 80]). Resource reservation has been another topic of research in recent years and various protocols have been developed: RSVP ([108]), ST-II ([98]), (see also [35, 3]). Although RSVP and ST-II support multicast applications, they merely provide mechanisms, but not policies, for performing resource reservation.

Route establishment is an important part of connection establishment. However, we do not consider it for several reasons. First, it has been studied extensively in recent years (see [28, 8, 31, 32, 76, 101]). Second, we believe that a good solution to the combined problem of routing and admission control is one that focuses on each separately. Several papers have considered the combined problem of route establishment and admission control of multicast sessions, [76, 101]. However, these works have focussed on solving a *static optimization* problem where all of the multicast sessions are known. A solution to this problem then provides the routes and resource reservation that minimizes some cost function that includes link cost and delay. The solution is usually obtained through integer program-

ming. This problem reduces to the well known Steiner tree problem that has proven to be NP-complete, so the combined problem is too complex to be solved in practice. Our conclusion that the two problems should be decoupled is also supported by observations in [101] that the cost of the multicast tree constructed from shortest path routes (as done in the present Internet multicast routing) is close to the minimum cost solution (Steiner tree). Similar observations were also made in [31].

Our approach to the problem of multicast connection establishment is as follows: the routing of the multicast connection (determining the multicast tree) is done first using one of the existing algorithms, followed by resource reservation (with admission control). We address this latter problem. We first develop an algorithm for "static" multicast sessions (where all receivers are known before the session's setup). Then, we extend the algorithm to handle "dynamic" multicast sessions (where receivers can join or leave anytime during the life of the session). Finally, we develop a distributed algorithm for receiver oriented multicast connection establishment.

In Section 2.2 we give a formal statement of the problem. A decomposition of the problem is found in Section 2.3 along with a number of alternative solutions to each sub-problem. In Section 2.4 we present centralized and distributed algorithms for multicast resource reservations. Sections 2.5 and 2.6 illustrate the effectiveness of our algorithms in the context of packetized voice and MPEG video. Section 2.7 concludes this chapter.

2.2 Problem Description

In this section we formalize the problem of admission control and resource reservation for multicast sessions. We begin with some notation:

- A network is represented by a directed graph $G(V, E)$. A link l from A to B in G is denoted by $A \stackrel{l}{\Leftrightarrow\!\!\rightarrow} B$. We associate with each link $l \in E$ a measure of resource availability on that link, $A_l \geq 0$. For example, A_l can be the amount of unreserved

bandwidth on link l, or the minimum delay the link can guarantee to the given session.

- $\mathcal{M} = (S, \mathcal{D}, (Q_{(S,D)})_{D \in \mathcal{D}})$ denotes a multicast session where S is the source node, $\mathcal{D} = \{D^1, ..., D^N\}$ is a set of N receivers and $Q_{(S,D)}$ is the end-to-end QoS requirement for the transmission from S to $D \in \mathcal{D}$. For example, $Q_{(S,D)}$ can be the maximum packet delay requirement between S and D.

- $\mathcal{T} \subseteq G(V, E)$ is a directed tree in G (the multicast tree) connecting S with all $D \in \mathcal{D}$.

- If a path exists from A to B in $\mathcal{T}$, denoted by $P = (A, B)$ then $\mathcal{L}(P)$ is the set of links, and $\mathcal{N}(P)$ is the set of nodes (including A and B) on that path. Note that if the path exists in $\mathcal{T}$, it is unique. In the rest of this chapter we will consider only paths in $\mathcal{T}$. We also use $\mathcal{D}_A = \{D \in \mathcal{D} | A \in \mathcal{N}((S, D))\}$ and $\mathcal{D}_l = \{D \in \mathcal{D} | l \in \mathcal{L}((S, D))\}$ for denoting the set of receivers in the subtree of $\mathcal{T}$ rooted in node $A \in \mathcal{T}$ and link $l \in \mathcal{T}$ respectively.

We consider the following problem of admission control and resource reservation of static multicast sessions:

> *Given a network G, a multicast session $\mathcal{M}$, and a multicast tree $\mathcal{T}$, based on the available network resources $(A_l)_{l \in E}$, determine whether $\mathcal{M}$ can be admitted on route $\mathcal{T}$ and, if so, reserve the necessary network resources.*

The dynamic multicast problem is a variation of the above where receivers can join or leave anytime during the existence of the session.

In solving the above problem we make the following assumptions:

- The multicast tree $\mathcal{T}$ is known.

- The QoS guarantees are additive: the end-to-end QoS guarantee on a path is equal to the sum of the link QoS guarantees on that path.

- The minimum amount of resources required on a link to guarantee a given QoS requirement is an increasing function of that QoS requirement.
- Each node (router) includes a switch which is responsible for packet forwarding on all outgoing links, and a control processor which is responsible for resource bookkeeping and admission control.

2.3 Computing per-link QoS Requirements

In Section 2.4 we present several algorithms for multicast resource reservation. These algorithms are based on the following primitives for computing per-link reservations:

1. A mechanism to map the local QoS requirement to the minimum amount of resource required at the link to guarantee the QoS (presented in Section 2.3.1);
2. A policy to map the end-to-end QoS requirement for a single receiver into local QoS requirements (presented in Section 2.3.2).

2.3.1 Mapping Local QoS Requirements to Link Resources

The mapping between a session's link QoS requirement and the amount of resources necessary to be reserved in order to guarantee that QoS depends on the nature of the QoS metric (loss probability, delay), the workload characteristics of the session (e.g. linear bounded arrival process (LBAP) [80], exponentially bounded burstiness (EBB) [105], on/off Markov fluid [48, 33]) and the link scheduling policy (First Come First Served (FCFS), Generalized Processor Sharing (GPS) [80], Earliest Deadline First (EDF) [35]). It is possible that no explicit mapping between local QoS requirement and resources exists in the case of some link scheduling policies (for example Rate Proportional Porcessor Sharing [80]). In such cases procedures are required to map the end-to-end QoS requirements directly to the local resources needed at the links.

In the case that it is possible to map local QoS requirements to resources, we assume that the link scheduling policy exhibits the following two properties. First, it supports sessions requiring different local QoS guarantees at a link. Not all policies permit this. For instance, per session QoS requirements cannot be guaranteed by a FIFO link scheduling policy. However, policies such as Generalized Processor Sharing (GPS) do permit this (see for example [80]). A second, highly desirable (but not mandatory) property is that a newly accepted session does not affect any of the pre-existing sessions' QoS guarantees at a link. The absence of this property implies the need to recompute the resource reservations of all the affected sessions as a new session is admitted. This is likely to be computationally intractable. This occurs for example, in the case of deterministic worst-case GPS (as analyzed in [80]). A stochastic Generalized Processor Sharing (GPS) and Earliest Deadline First (EDF) link scheduling policies exhibit both properties, and will be part of our examples in Section 2.5 and 2.6 .

Henceforth, we assume the existence of the functions $F_l : \mathbb{R} \to \mathbb{R}$ and $H_l : \mathbb{R} \to \mathbb{R}$,

$$F_l(Q) \mapsto R, \quad H_l(R) \mapsto Q$$

where $F_l(Q)$ is the amount of resources needed at l to guarantee a local QoS Q, and H_l is the inverse function of F_l, $H_l(R) = F_l^{-1}$. Such a function can be determined for a given combination of QoS requirement, session arrival process characteristics, and link scheduling policy via performance analyses (for example [48]). In Chapter 3 we derive such a mapping for Earliest Deadline First packet scheduling.

2.3.2 Mapping End-to-End QoS to Local QoS

In order to reserve resources at the links given an end-to-end QoS requirement, we need to first divide the end-to-end QoS requirement into local QoS requirements (such that if a session is guaranteed every local QoS requirement, it is also guaranteed the end-to-end QoS requirement) and then determine the resources required at each link in order to achieve the

local QoS. In the remainder of this section we describe several end-to-end QoS division policies along with a property that they exhibit.

There are several issues related to this division of end-to-end QoS such as how to compose the local QoS requirements to reconstruct the end-to-end QoS requirement and how to distribute the QoS requirement to links. These issues are considered in the following subsections.

2.3.2.1 Examples of QoS Division

In Section 2.2 we have stated the assumption that the QoS guarantees are additive. Here we give two examples of additive QoS guarantees that we use in the applications in Sections 2.5 and 2.6.

Let $A \overset{l}{\Leftrightarrow} B \overset{m}{\Leftrightarrow} C$ be two adjacent links in network G. If the QoS is packet loss probability, then an upper bound on the end-to-end loss probability is:

$$\begin{aligned}
\Pr(\text{loss on } (A,C)) &= \Pr((\text{loss on } (A,B)) \vee (\text{loss on } (B,C))) \\
&= \Pr(\text{loss on } (A,B)) + \Pr(\text{loss on } (B,C)) \Leftrightarrow \\
&\quad \Pr((\text{loss on } (A,B)) \wedge (\text{loss on } (B,C))) \\
&< \Pr(\text{loss on } (A,B)) + \Pr(\text{loss on } (B,C))
\end{aligned}$$

In the case that $\Pr(\text{loss on } (A,C))$ is small ($< 10^{-2}$), $\Pr(\text{loss on } (A,B)) + \Pr(\text{loss on } (B,C))$ is a tight bound for $\Pr(\text{loss on } (A,C))$. In this case we write $Q_{(A,C)} \approx Q_l + Q_m$.

If the QoS is a constraint on packet delay, then

$$\text{max\_delay}(A,C) = \text{max\_delay}(A,B) + \text{max\_delay}(B,C)$$

and we write $Q_{(A,C)} = Q_l + Q_m$.

2.3.2.2 End-to-end QoS Division Policies

A policy for dividing end-to-end QoS requirements among the links of a path P can be defined as follows. Define $C_l : \mathbb{R} \to \mathbb{R}$ to be a non-increasing function of the measure A_l of available resources at link l. The QoS assigned to link l given the end-to-end QoS Q for path P and $(A_l)_{l \in \mathcal{L}(P)}$ is

$$Q_l = \frac{QC_l(A_l)}{\sum_{m \in \mathcal{L}(P)} C_m(A_m)} \quad \forall l \in \mathcal{L}(P) \ , \tag{2.1}$$

under the assumption that the link l has sufficient resources to guarantee Q_l. In other words, the end-to-end requirement Q is divided into $(Q_l)_{l \in \mathcal{L}(\mathcal{P})}$ shares according to the weights $(C_l(A_l))_{l \in \mathcal{L}(\mathcal{P})}$. We address later the case where there are insufficient resources.

We consider the following policies.

Even policy. This policy allocates equal shares of the end-to-end QoS among links on a path, $C_l(A_l) = 1$.

Relative Proportional policy. This policy allocates shares of the end-to-end QoS to links in proportion to their utilization. This results in lower utilized links getting assigned more stringent QoS requirements over higher utilized links. More specifically, $C_l(A_l) = (T_l \Leftrightarrow A_l)/T_l$ where T_l denotes the total amount of resources at link l.

Absolute Proportional policy. This policy assigns local QoS in inverse proportion to the available resources, i.e., $C_l(A_l) = 1/A_l$.

Threshold policy. Under this policy, C_l takes one of two values according to the available resources. For example:

$$C_l(A_l) = \begin{cases} c_1, & A_l < \text{threshold}, \\ c_2, & \text{otherwise} \end{cases}$$

where $c_1, c_2 > 0$.

Resource-limited policies. We consider now the case where one or more links has an insufficient amount of resources to guarantee the QoS given by (2.1), but the path P

has sufficient resources to guarantee the given end-to-end QoS. More precisely, for some $l \in \mathcal{L}(P)$, $Q_l < Q_l^{min}$ (where $Q_l^{min} = H_l(A_l)$ is the most stringent QoS requirement that link l can support), but $\sum_{m \in \mathcal{L}(P)} Q_m^{min} \leq Q$.

The problem is then:

Given Q, $(Q_l^{min})_{l \in \mathcal{L}(\mathcal{P})}$ and $(C_l(A_l))_{l \in \mathcal{L}(\mathcal{P})}$, find Q_{free}^P such that

$$Q_l = \max(Q_l^{min}, Q_{free}^P C_l(A_l)) \tag{2.2}$$

It is easy to see that the problem has a solution if $\sum_{l\mathcal{L}(\mathcal{P})} Q_l = Q$.

The QoS division algorithm first assigns to each link n that is resource limited its corresponding Q_n^{min}, and then assigns Q_l to each link l that is not resource limited, where

$$Q_l = \frac{Q \Leftrightarrow \sum_{\substack{n \in \mathcal{L}(P) \\ n \text{ is res. lim.}}} Q_n^{min}}{\sum_{\substack{m \in \mathcal{L}(P) \\ m \text{ is not res. lim.}}} C_m(A_m)} C_l(A_l)$$

We present the algorithm formally in Figure 2.1. The set of links in $\mathcal{L}(P)$ is processed in descending order of $\frac{Q_l^{min}}{C_l(A_l)}$. For each link $l \in \mathcal{L}(P)$, a check is made to see if, by dividing the end-to-end QoS according to (2.1) among all the unprocessed links, the resulting share is larger than the largest Q^{min} of the unprocessed links. If so, all these links can accommodate this share and the algorithm terminates. If not, assign to the current link l, its Q_l^{min}, subtract it from the QoS to be divided, advance to the next link in the list and repeat the above for the remaining links.

To analyze the computational complexity of DIVIDE_QOS_LIM we observe that the number of iterations in loop 3-12 in Figure 2.1 is $O(|\mathcal{L}(P)|)$ and that there is an implicit sort having complexity $O(|\mathcal{L}(P)| \log |\mathcal{L}(P)|)$, giving a total computational complexiy of $O(|\mathcal{L}(P)| \log |\mathcal{L}(P)|)$.

DIVIDE_QOS_LIM(input: P, the path
$(A_m)_{m\in\mathcal{L}(P)}$, link resource availability
$(Q_m^{min})_{m\in\mathcal{L}(P)}$, link QoS limitations
Q; end-end QoS requirement
output: $(Q_m)_{m\in\mathcal{L}(P)}$, link QoS requirements
Q_{free}^P) unrestricted link QoS share

1 $Q_{rest} \leftarrow Q$
2 $V_{rest} \leftarrow \sum_{m\in\mathcal{L}(P)} C_m(A_m)$
3 **for** $l \in \mathcal{L}(P)$ in descending order of $\frac{Q_l^{min}}{C_l(A_l)}$ **do**
4 **if** $\frac{Q_l^{min}}{C_l(A_l)} \leq Q_{rest}/V_{rest}$
5 **then** $Q_{free}^P \leftarrow Q_{rest}/V_{rest}$
6 **for** $m \in \mathcal{L}(P)$ in descending order of $\frac{Q_m^{min}}{C_m(A_m)}$ starting with l **do**
7 $Q_m \leftarrow Q_{free}^P C_m(A_m)$
8 **return**
9 **else** $Q_{free} \leftarrow Q_l^{min}$
10 $Q_l \leftarrow Q_l^{min}$
11 $Q_{rest} \leftarrow Q_{rest} - Q_l^{min}$
12 $V_{rest} \leftarrow V_{rest} - C_l(A_l)$

Figure 2.1. QoS division with resource limitations

Division policies for QoS classes.

The previous QoS division policies are based on the assumption that a QoS guarantee can take an arbitrary value within an interval on the real line. There are systems (as in the examples in Section 2.5 and 2.6) where the QoS guarantee can only take one of a finite set of values. Consider the case where Q_l can take the values $0 < q_{1,l} < \cdots < q_{L,l}$ at link l, each value associated with a QoS class $j = 1, .., L$. Any division policy (including all policies described above) can easily be modified to accommodate a finite number of QoS values by "truncating" each QoS result to the closest smaller value (tighter QoS value) in the set (and thus guaranteeing the end-to-end QoS). More precisely, if Q_l is the value obtained by a division policy as defined by (2.1) and if $q_{i,l} \leq Q_l < q_{i+1,l}$ for some i, then $q_{i,l}$ is the QoS to be guaranteed on link l.

2.3.2.3 The Uniformity Property

Under certain conditions, the QoS division policies presented in the previous section produce "uniform" assignments of QoS requirements to links. This property is not required by our resource reservation algorithms. However, when a QoS division policy is uniform, it is possible to improve the efficiency of reservation algorithms (we present this in Section 2.4.2.2).

Consider two paths to receivers D^1 and D^2 in a multicast tree that share two or more links. An assignment is uniform if the QoS requirement for D^1 is either greater than or less than that for D^2 at all links on their common part. This is illustrated in Figure 2.2 where we observe that for each link on the common path (S, A) the local QoS ($= 1$) from the first path is smaller than the corresponding local QoS ($= 3$) from the second path.

The formal definition of the uniformity property follows:

Definition 2.1 *Let $\mathcal{P}$ be a QoS division policy, $P^i = (S, D^i)$, $(i = 1, 2)$ be two paths sharing at least one link, $P^1 \cap P^2 \neq \emptyset$. Let $Q_{(S,D^i)}$, $(i = 1, 2)$ be their end-to-end QoS requirements and $(Q_l^i)_{l \in \mathcal{L}(P^i)}$, $(i = 1, 2)$ the results of applying policy $\mathcal{P}$ on P^1 and P^2. $\mathcal{P}$ is said to be uniform if:*

$$\begin{aligned} &\textit{either} \quad Q_l^1 \leq Q_l^2, \quad \forall l \in \mathcal{L}(P^1 \cap P^2) \\ &\textit{or} \quad Q_l^1 \geq Q_l^2, \quad \forall l \in \mathcal{L}(P^1 \cap P^2) \ . \end{aligned} \tag{2.3}$$

In the following we assume that the values for $C_l(A_l)$ that are used for QoS division computation for both P^1 and P^2 are the same:

$$C_l(A_l^{P^1}) = C_l(A_l^{P^2}), \quad \forall l \in P^1 \cap P^2 \ .$$

This assumption is valid, for example, in the case that the multicast session is established at the same time for all receivers (static multicast). At the end of this section we examine the

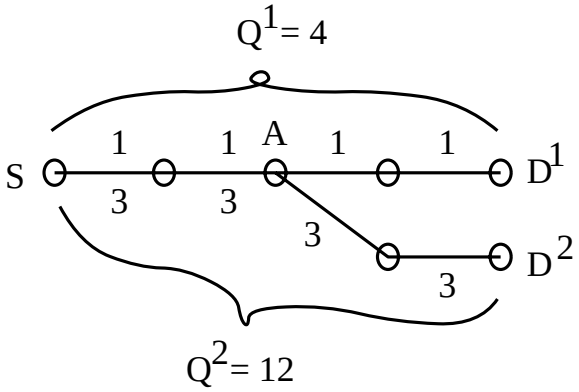


Figure 2.2. An example of uniform QoS division

case where this assumption does not hold, as is the case when receivers can independently join or leave independently the multicast session (dynamic multicast). We establish the following theorem in Appendix A.1.

Theorem 2.1 *Any division policy defined by (2.1) is uniform.*

The class-based QoS division policies are also uniform. To prove this, it is sufficient to extend the proof in Appendix A.1 by replacing any Q_n with the closest smaller value in the finite set (i.e., with $q_{i,n}$ where $q_{i,n} \leq Q_n < q_{i+1,n}$) and observing that this operation preserves the inequalities.

Suppose now that P^1 and P^2 are not established simultaneously. Then the Even QoS division without resource limitations is still uniform because $C_l(A_l) = 1$ at any time and the relations in (2.3) are verified. However any policy that allows C_l to depend on the available resources (e.g Proportional and Threshold) is not uniform because the values $(A_l)_l$ are time dependent. Any division policy with resource limitations is not uniform because the values $(Q_n^{min})_n$ are time dependent.

The above uniformity property allows us to introduce an ordering among paths.

Definition 2.2 *In the context of a uniform division policy, P^1 is* less *than P^2, denoted $P^1 \prec P^2$ if*

$$\textit{there exists} \quad l \in \mathcal{L}(P^1 \cap P^2) \quad \textit{such that} \quad Q_l^1 < Q_l^2$$

Given a set of paths, the path in the set that is minimum with respect to the relation $\prec$ yields the most stringent QoS requirement on their common part. We refer to this path as *critical* and we use this concept in Section 2.4.2.2. We examine next how this ordering can be established for the proposed QoS division policies.

In the case of a division policy without resource limitations we can associate with each path sharing a link l the QoS requirement on l obtained from dividing the end-to-end QoS requirements for each path P^D $D \in \mathcal{D}_l$. The paths are thus ordered based on $(Q_l^D)_{D \in \mathcal{D}_l}$.

In the case of a division policy with resource limitations we can associate with each path sharing a link l the "normalized" QoS requirement to be assigned to links on that path that do not have resource limitations:

$$Q_l^1 < Q_l^2 \Leftrightarrow \frac{Q_l^1}{C_l(A_l)} < \frac{Q_l^2}{C_l(A_l)} \Leftrightarrow Q_{free}^{P^1} < Q_{free}^{P^2}$$

The paths are thus ordered based on $(Q_{free}^{P^D})_{D \in \mathcal{D}_l}$.

2.4 Reservation Algorithms

In this section we consider the problem of multicast resource reservation as introduced in Section 2.2 and present several multicast resource reservation algorithms. First, we distinguish between static multicast sessions, where all receivers are known at the start of reservation, and dynamic multicast sessions, where receivers can join or leave the multicast session at any time.

Second, we distinguish between centralized and distributed multicast reservation algorithms. A centralized algorithm first collects link information (such as resource availability) in a single place, computes per-link QoS requirements for all links in the multicast tree, requirements that are then communicated to the links, where the local reservations are finally done.

A distributed algorithm collects data incrementally through messages circulating hop-by-hop in the multicast tree, and performs local QoS requirement and resource reservation computations locally at each link it traverses. Such a distributed algorithm can be sender-initiated or receiver-initiated.

In Section 2.4.1 we present an outline of link QoS computation for a multicast session. This computation is common to all reservation algorithms presented next. In Section 2.4.2 we present centralized reservation algorithms for static multicast sessions. In Section 2.4.3 we present distributed, sender-initiated reservation algorithms for static multicast sessions. In Section 2.4.4 we present distributed, receiver-initiated reservation algorithms for dynamic multicast sessions.

2.4.1 Outline of Multicast QoS Computation

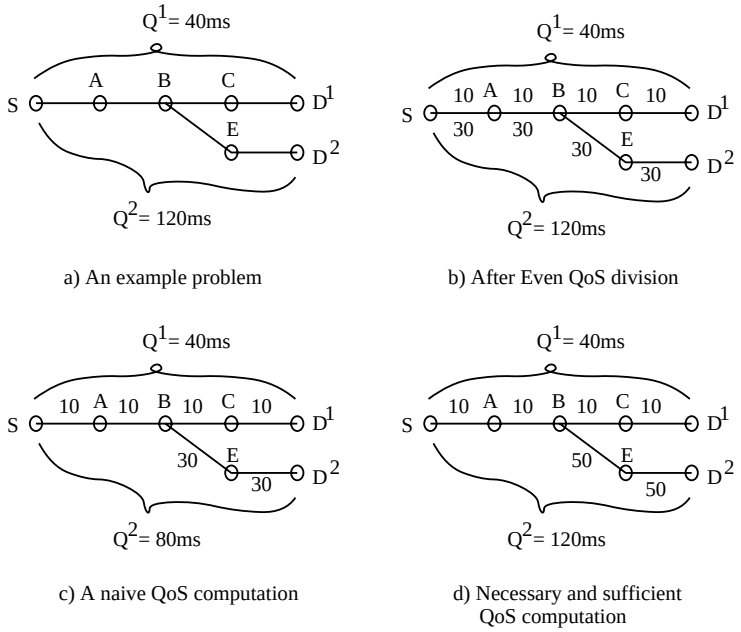


Figure 2.3. An example of resource reservation and reclamation

In order to motivate the algorithms that we will present, we begin with a simple example. We consider a multicast session with two receivers as in Figure 2.3(a), and assume that

we use the Even QoS division. A reservation algorithm needs to determine the QoS requirement for each link on the multicast tree in order to guarantee the given end-to-end requirement of $Q^1 = 40ms$ and $Q^2 = 120ms$. If we consider the two requirements independently we obtain per-link requirements of $10ms$ and $30ms$ respectively, as in Figure 2.3(b). Since the links (S, A) and (A, B) are common to both receivers, the tightest delay requirement, $10ms$, is assigned to these links, as in Figure 2.3(c), otherwise the end-to-end requirement of D^1 would not be met. Observe that this computation guarantees receiver D^2 a maximum packet delay of $80ms$, which is more stringent than it has requested and translates in the reservation of more resources at links (B, E) and (E, D^2) than are required. In Figure 2.3(d) we show a set of local QoS guarantees that give the exact amount of end-to-end guarantees for all receivers. Observe that the guarantees on (B, E) and (E, D^2) are less stringent than in Figure 2.3(c), and thus require less amount of resources. This example motivates the benefit of resource reclamation following the independent allocation of resources on paths from the sender to all receivers.

The reservation algorithms presented in the next sections compute necessary and sufficient per-link QoS requirements as presented above.

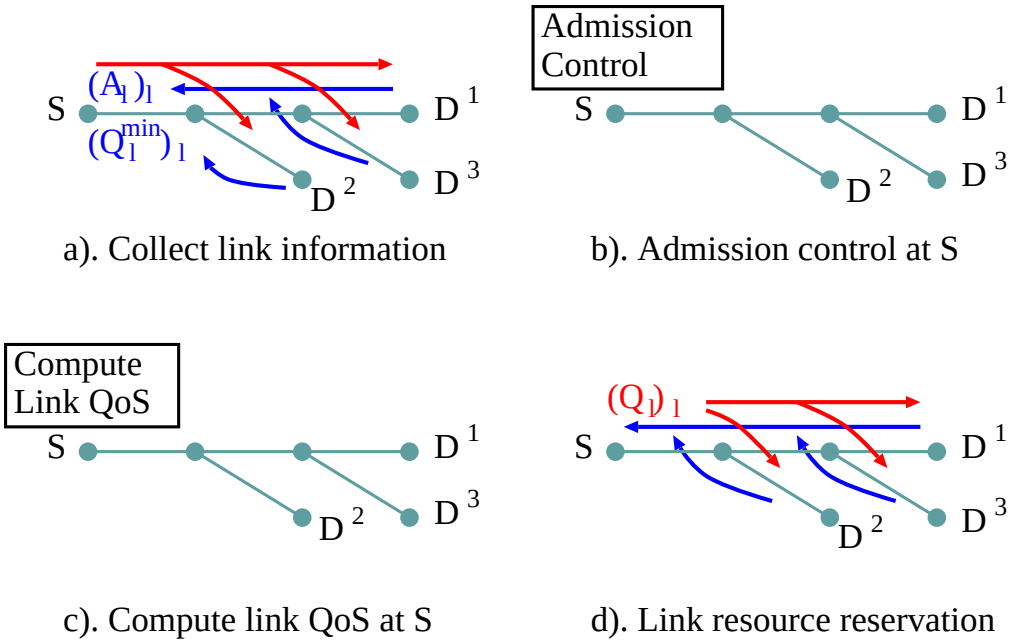
2.4.2 Centralized Reservation Algorithms for Static Multicast

In this section we present two centralized algorithms for resource reservation of static multicast sessions. For these algorithms all computation is performed in a single place, which we take to be the sender node of the multicast session.

The outline of the centralized reservation algorithms is as follows (see Figure 2.4):

1. The algorithm is given the set of receivers, $\mathcal{D}$, and their respective end-to-end QoS requirements, $(Q_{(S,D)})_{D \in \mathcal{D}}$.

2. The sender S collects information on link resource availability $(A_l)_{l \in \mathcal{L}(\mathcal{T})}$ and QoS limitations, $(Q_l^{min})_{l \in \mathcal{L}(\mathcal{T})}$, for all links in the multicast tree. This can be done by



a). Collect link information b). Admission control at S

c). Compute link QoS at S d). Link resource reservation

Figure 2.4. The mechanism of centralized resource reservation

sending messages to the receivers, which are reflected back to the sender, and the information is collected from all links traversed (Figure 2.4(a)).

3. The sender performs multicast admission control: given the collected link QoS limitations, $(Q_l^{min})_{l\in\mathcal{L}(\mathcal{T})}$, it computes the most stringent end-to-end QoS that can be guaranteed for each path (S, D), $Q_{(S,D)}^{min} = \sum_{l\in\mathcal{L}(S,D)} Q_l^{min}$. The multicast session is accepted if each end-to-end QoS requirement can be guaranteed, $Q_{(S,D)} \geq Q_{(S,D)}^{min}$ for all $D \in \mathcal{D}$, and rejected otherwise. Alternatively, other admission control policies can be implemented; for example the multicast session is admitted if a majority of receivers can be satisfied (Figure 2.4(b)).

4. In the case admission control has succeeded, per-link QoS requirements, $(Q_l)_{l\in\mathcal{L}(\mathcal{T})}$, are computed, as outlined in Section 2.4.1 (Figure 2.4(c)). We present in detail two variants of QoS computation algorithms in Sections 2.4.2.1 and 2.4.2.2.

5. The link QoS requirements $(Q_l)_{l \in \mathcal{L}(\mathcal{T})}$ are communicated to the links in the multicast tree using for example the same messaging mechanism used for collecting link information. As each link receives its link QoS requirement Q_l, it reserves a corresponding amount of resources, $F_l(Q_l)$. After all messages have returned to the sender confirming all reservations, the multicast session can start transmitting data (Figure 2.4(d)).

2.4.2.1 A First QoS Computation Algorithm

MCAST_QOS(input: S, $\mathcal{D}$, source, receivers
$(Q_{(S,D)})_{D \in \mathcal{D}}$, e-e QoS requirements
$(A_l)_{l \in \mathcal{L}(\mathcal{T})}$, link availability
$(Q_l^{min})_{l \in \mathcal{L}(\mathcal{T})}$; link limitations
output: $(Q_l)_{l \in \mathcal{L}(\mathcal{T})}$) link QoS requirements

1 $Q_{(S,S)} \leftarrow 0$
2 $V_{S,S} \leftarrow 0$
3 **for** $l \in \mathcal{L}(\mathcal{T})$ in order of Breadth First Search **do**
4 **let** A, B such that $A \xrightarrow{l} B$
5 $V_{S,B} \leftarrow V_{S,A} + C_l(A_l)$
6 **for** $l \in \mathcal{L}(\mathcal{T})$, in order of Breadth First Search **do**
7 **let** A, B such that $A \xrightarrow{l} B$
8 **if** $B \in \mathcal{D}$
9 **then** $Q_l \leftarrow Q_{(S,B)} - Q_{(S,A)}$
10 **else for** $D \in \mathcal{D}_l$ **do**
11 $Q_l^D \leftarrow \frac{C_l(A_l)(Q_{(S,D)} - Q_{(S,A)})}{V_{S,D} - V_{S,A}}$
12 $Q_l \leftarrow \min_{D \in \mathcal{D}_l} Q_l^D$
13 $Q_{(S,B)} \leftarrow Q_{(S,A)} + Q_l$

Figure 2.5. Centralized multicast QoS computation

In this section, we present a centralized algorithm MCAST_QOS (Figure 2.5) that implements the link QoS computation outlined in Section 2.4.1. The algorithm finds a solution $(Q_l)_{l \in \mathcal{L}(\mathcal{T})}$ to the following recursive equation

$$\text{for link} \quad A \overset{l}{\Leftrightarrow} B, \quad Q_l = \min_{D \in \mathcal{D}_l} \frac{C_l(A_l) Q_{(A,D)}}{\sum_{m \in \mathcal{L}(A,D)} C_m(A_m)}$$

We define $Q_{(X,Y)}$ to be the computed QoS guarantee between nodes X to Y. Also, we define $V_{X,Y}$ to be the sum of link wights $C_l(A_l)$ on path (X, Y)

$$V_{X,Y} = \sum_{l \in \mathcal{L}((X,Y))} C_l(A_l)$$

The algorithm MCAST_QOS (Figure 2.5) computes the QoS requirement for each link in the multicast tree, from sender to receivers, in Breadth First order (lines 6-13). For a given link l it computes a link share Q_l^D of the QoS requested by each receiver D in the subtree below l, and the tightest of these shares is guaranteed for l. The procedure is repeated for the links in its subtree.

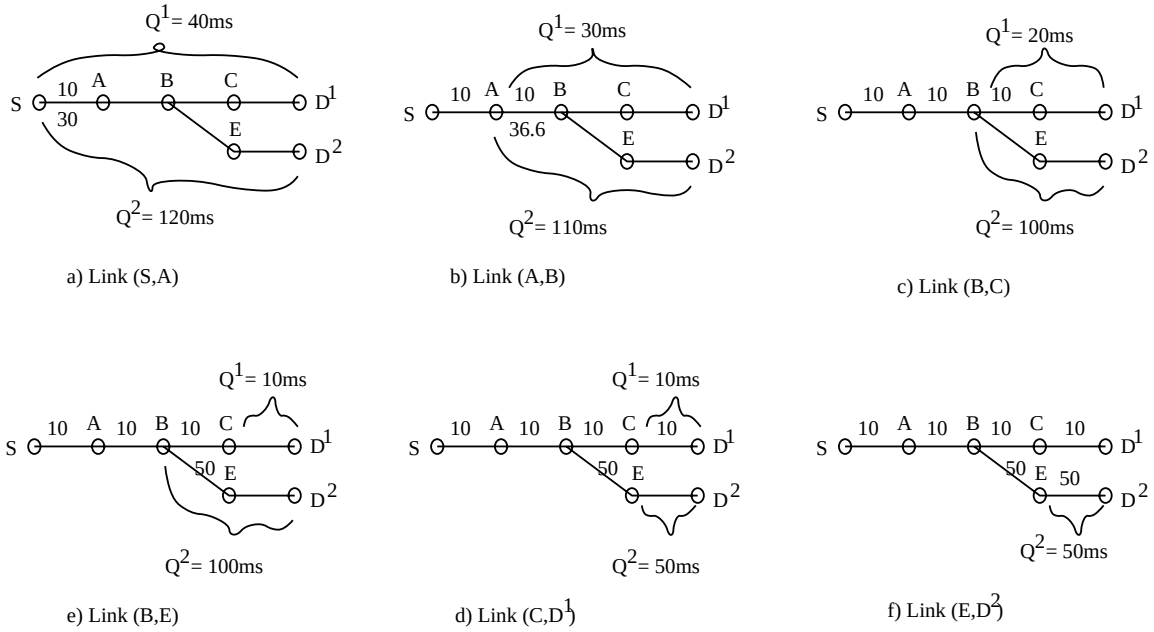


Figure 2.6. An example of centralized multicast QoS computation

Example. We illustrate our algorithm with an example in Figure 2.6. The algorithm is given a multicast session with two receivers. In our example we use Even QoS division, and thus, we have $C_l(A_l) = 1$ for any link l, that yields $V_{S,D^1} = V_{S,D^2} = 4$. The algorithm starts by considering link (S, A) (Figure 2.6(a)). The QoS share on link (S, A), given receivers D^1's requirement, results from the QoS division formula (2.1)

$$Q^{D^1}_{(S,A)} = \frac{Q_{(S,D^1)} C_{(S,A)}(A_{(S,A)})}{\sum_{m \in \mathcal{L}(S,D^1)} C_m(A_m)} = \frac{Q_{(S,D^1)}}{V_{S,D^1}} = \frac{40}{4} = 10ms$$

Similarly, the QoS share on the same link given receiver D^2's requirement is

$$Q^{D^2}_{(S,A)} = \frac{Q_{(S,D^2)}}{V_{S,D^2}} = \frac{120}{4} = 30ms$$

The QoS requirement to be guaranteed on (S, A) is then

$$Q_{(S,A)} = \min(Q^{D^1}_{(S,A)}, Q^{D^2}_{(S,A)}) = 10ms$$

The next iteration of our algorithm considers link (A, B) (Figure 2.6(b)). In a similar it computes the QoS shares on (A, B) derived from D^1 and D^2 requirements reduced by the QoS allocated to (S, A)

$$Q^{D^1}_{(A,B)} = \frac{Q_{(A,D^1)} C_{(A,B)}(A_{(A,B)})}{\sum_{m \in \mathcal{L}(A,D^1)} C_m(A_m)} = \frac{Q_{(A,D^1)}}{V_{A,D^1}} = \frac{Q_{(S,D^1)} \Leftrightarrow Q_{(S,A)}}{V_{S,D^1} \Leftrightarrow V_{S,A}} = \frac{30}{3} = 10ms$$

Similarly, the QoS share on link (A, B) given receiver D^2's requirement is

$$Q^{D^2}_{(A,B)} = \frac{Q_{(A,D^2)} C_{(A,B)}(A_{(A,B)})}{\sum_{m \in \mathcal{L}(A,D^2)} C_m(A_m)} = \frac{Q_{(S,D^2)} \Leftrightarrow Q_{(S,A)}}{V_{S,D^2} \Leftrightarrow V_{S,A}} = \frac{110}{3} = 36.6ms$$

and the QoS requirement to be guaranteed on (A, B) is then $10ms$. The rest of the algorithm's computation is shown in Figure 2.6(c)-(f).

For QoS division policies with resource limitations, line 11 of the QoS computation algorithm (Figure 2.5) is replaced by:

$$\text{DIVIDE\_QOS\_LIM}((A, D), (A_m)_{m \in \mathcal{L}(A,D)}, (Q^{min}_m)_{m \in \mathcal{L}((A,D))}, Q_{(S,D)} \Leftrightarrow Q_{(S,A)}; \\ (Q^D_m)_{m \in \mathcal{L}((A,D))}, Q^{(A,D)}_{free})$$

where DIVIDE_QOS_LIM is described in Figure 2.1.

Complexity Analysis. To analyze the complexity of this algorithm, we introduce the folowing notation

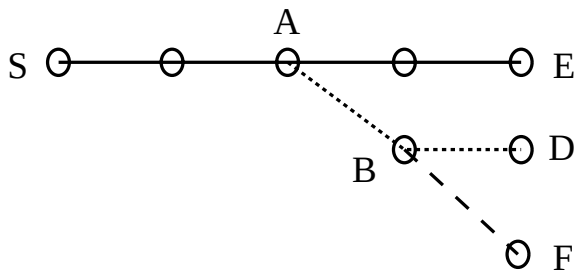
- $N = |\mathcal{D}|$ is the number of receivers;
- $M = |\mathcal{L}(\mathcal{T})|$ is the number of links in the multicast tree.

We first consider the case of QoS division without resource limitation, and we derive the worst case running time of the algorithm in Figure 2.5. Lines 1-2 are executed once. The loop 3-5 is executed once for each $l \in \mathcal{L}(\mathcal{T})$, i.e., $O(M)$ times. The loop 6-13 is executed once for each $l \in \mathcal{T}$, for a total of $O(M)$ times. Inside this loop, lines 7,8,9 and 13 are executed once, the loop 10-11 a maximum of $O(N)$ times and line 12 is $O(N)$. Thus, the loop 6-13 is $O(NM)$ and then MCAST_QOS has a worst case running time of $O(M + NM) = O(NM)$.

In the case of QoS division in the presence of resource limitations, a similar analysis applies, with the only difference being the complexity of DIVIDE_QOS_LIM in line 11. We assume that the set $(Q_m^{min})_{m\in(S,D)}$ is sorted (complexity $O(|\mathcal{L}((S,D))|\log|\mathcal{L}((S,D))|)$) only once for the entire run of MCAST_QOS, and the ordered list is stored for the duration of MCAST_QOS. Then, this list is scanned (complexity $O(|\mathcal{L}((S,D))|)$) in line 3 Figure 2.1 for each call to DIVIDE_QOS_LIM on the path (A, D) with $A \in \mathcal{N}((S,D))$. We have seen that the loop 3-12 in DIVIDE_QOS_LIM (Figure 2.1) has a worst case running time of $O(|\mathcal{L}((A,D))|)$. Hence, the aggregate complexity of line 10 in MCAST_QOS is:

$$\sum_{D\in\mathcal{D}}(O(|\mathcal{L}((S,D))|\log|\mathcal{L}(S,D)|)+\sum_{A\in\mathcal{N}((S,D))}O(|\mathcal{L}((S,D))|+|\mathcal{L}((A,D))|))=$$
$$\sum_{D\in\mathcal{D}}O(|\mathcal{L}((S,D))|^2)=O(NM^2)$$

Consequently, MCAST_QOS has a worst case running time $O(NM + NM^2)$ which is $O(NM^2)$.



Critical paths: (S,E), (A,D), (B,F)

Figure 2.7. A tree partitioned into critical paths

2.4.2.2 A Faster Algorithm for QoS Computation

We have seen that the most computationally intensive part of MCAST_QOS are lines 10-12, where, for each link in the tree, the algorithm performs a comparison of the QoS requirements imposed by each path containing that link. The complexity can be reduced in the case where the QoS division policy exhibits the *uniformity* property described in Section 2.3.2.3. In this case, it is not necessary to compute and compare Q_l^D for each link $l \in \mathcal{L}(\mathcal{T})$. Let a *critical receiver* for a link be the receiver that requires the most stringent QoS on that link. The *uniformity* property ensures that, if D is the critical receiver for the link $A \overset{l}{\Leftrightarrow} B$, then D is the critical receiver for all links on (A, D). If A is the node closest to S with the above property, then we call (A, D) a *critical path*. It is easy to see that critical paths begin only at S or a branching node, and end at a receiver. Furthermore, the tree $\mathcal{T}$ is partitioned into critical paths (see Figure 2.7 for an example of a partitioned tree).

We describe the faster QoS computation algorithm through an example shown in Figure 2.8. It first considers the link(s) directly connected to the source, which in our example is (S, A), and determines its critical path. By applying the Even division policy on each of the three paths including (S, A), we find that (S, D^1) is critical since (Figure 2.8(a))

$$Q_{(S,A)}^{D^1} = 2 < Q_{(S,A)}^{D^2} = 8 < Q_{(S,A)}^{D^3} = 9 \quad .$$

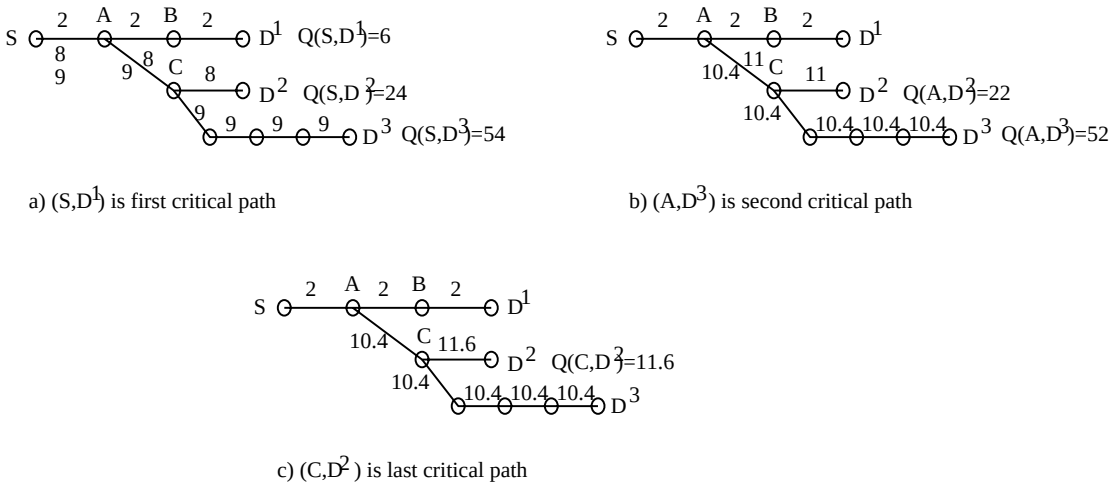


Figure 2.8. An example of a faster centralized multicast QoS computation

Thus, we can compute the QoS allocation on all links on (S, D^1). Next, the algorithm determines the critical path on link (A, C), which is (A, D^3) since (Figure 2.8(b))

$$Q^{D^3}_{(S,A)} = 10.4 < Q^{D^2}_{(S,A)} = 11 \quad .$$

The last critical path is (C, D^2) (Figure 2.8(c)).

The QoS division algorithm that relies on the uniformity property is FASTER_MCAST_QOS (Figure 2.9). It starts with the critical paths containing S, and computes the QoS requirements for all their links. Also, all links branching out of these paths are marked. The process is repeated by determining the critical path in each subtree rooted in a marked link and by computing the QoS requirement for all links on these critical paths. The procedure is repeated until all of $\mathcal{T}$ is processed.

More precisely, the **for** loop in lines 8-19 determines, for each outgoing link l from source S or branching node A, the critical path that contains it (lines 9-13) by determining the path that gives the minimum of the set $(Q^D_l)_{D \in \mathcal{D}_l}$. For this critical path, it computes the QoS requirement for each of its links (line 15), computes the QoS for the upstream parts of the path (line 16-17), and marks the links branching out of this critical path (lines 18-19).

FASTER_MCAST_QOS(input: S, $\mathcal{D}$, source, receivers
$(Q_{(S,D)})_{D\in\mathcal{D}}$, e-e QoS requirements
$(A_l)_{l\in\mathcal{L}(\mathcal{T})}$, link availability
$(Q_l^{min})_{l\in\mathcal{L}(\mathcal{T})}$; link limitations
output: $(Q_l)_{l\in\mathcal{L}(\mathcal{T})}$) link QoS requirements

1 $Q_{(S,S)} \leftarrow 0$
2 $V_{S,S} \leftarrow 0$
3 **for** $l \in \mathcal{L}(\mathcal{T})$ in order of Breadth First Search **do**
4 **let** A, B such that $A \xrightarrow{l} B$
5 $V_{S,B} \leftarrow V_{S,A} + C_l(A_l)$
6 **for** $l \in \mathcal{L}(\mathcal{T})$, outgoing link from S **do**
7 mark l
8 **for** $l \in \mathcal{L}(\mathcal{T})$ marked **do**
9 unmark l
10 **let** A, B s.t. $A \xrightarrow{l} B$
11 **for** $D \in \mathcal{D}_l$ **do**
12 $Q_l^D \leftarrow \frac{C_l(A_l)(Q_{(S,D)}-Q_{(S,A)})}{V_{S,D}-V_{S,A}}$
13 **let** E such that $Q_l^E = \min_{D\in\mathcal{D}_l} Q_l^D$
14 **for** $m \in \mathcal{L}(A, E)$ **do**
15 $Q_m \leftarrow \frac{C_m(A_m)(Q_{(S,E)}-Q_{(S,A)})}{V_{S,E}-V_{S,A}}$
16 **let** X, Y s.t. $X \xrightarrow{m} Y$
17 $Q_{(S,Y)} \leftarrow Q_{(S,X)} + Q_m$
18 **for** $n \in \mathcal{L}(\mathcal{T})$, outgoing from X, $n \neq m$ **do**
19 mark n

Figure 2.9. Faster centralized multicast QoS computation

The algorithm terminates when there are no more marked links and thus all the critical paths in $\mathcal{T}$ have been processed.

For QoS division policies with resource limitations, the lines 12-15 in FASTER_MCAST_QOS are replaced by:

12 DIVIDE_QOS_LIM($(C, D), (A_m)_{m\in\mathcal{L}(\mathcal{T})}, (Q_m^{min})_{m\in\mathcal{L}((C,D))}, Q_{(S,D)} - Q_{(S,C)}$;
$(Q_m^D)_{m\in\mathcal{L}((C,D))}, Q_{free}^{(C,D)}$)
13 **let** E such that $Q_{free}^{(C,E)} = \min_{D\in\mathcal{D}_l} Q_{free}^{(C,D)}$
14 **for** $m \in \mathcal{L}(A, E)$ **do**
15 $Q_m \leftarrow Q_m^E$

Here the critical path is the one yielding the smallest value in $(Q_{free}^{(C,D)})_{D\in\mathcal{D}_l}$. Also, in line 15 the QoS requirement on link m given by the critical path is Q_m^E, which was already computed by DIVIDE_QOS_LIM on line 12.

Complexity Analysis. We first address the efficiency of QoS division policies when there are no resource limitations. The following is a worst case running time analysis for the algorithm FASTER_MCAST_QOS (Figure 2.9). The loops 3-5 and 6-7 are $O(M)$. Defining a branching link in a tree to be a link outgoing from a branching node, we observe that the links marked by this algorithm are exactly all the branching links in $\mathcal{T}$. Thus, the loop 8-19 is executed for each branching link in $\mathcal{T}$. Defining $\mathcal{B}(\mathcal{T})$ the set of branching links in $\mathcal{T}$, we have $|\mathcal{B}(\mathcal{T})| = O(N)$ since the number of branching nodes in a tree is less than the number of leaves in that tree. Then, the aggregate running time for lines 11-13 is $O(N^2)$:

$$\sum_{l\in\mathcal{B}(\mathcal{T})}\sum_{l\in\mathcal{D}_l} 1 \leq |\mathcal{B}(\mathcal{T})| * |\mathcal{D}| \leq N^2$$

The aggregate complexity of loop 14-19 is $O(M)$ since each link m in $\mathcal{T}$ is considered only once because a link is contained only in one critical path. Consequently, FASTER_MCAST_QOS has a worst case running time $O(N^2 + M)$.

In the case that the QoS division policy is applied to a network with resource limitations, a similar analysis applies, with the only difference in the complexity of DIVIDE_QOS_LIM in line 12. We assume that the set $(Q_m^{min})_{m\in\mathcal{T}}$ is sorted (complexity $O(M\log M)$) and then scanned (complexity $O(M)$) in line 3 of algorithm DIVIDE_QOS_LIM (Figure 2.1) for each call to DIVIDE_QOS_LIM. We have seen that the loop 3-12 in DIVIDE_QOS_LIM has a worst case running time of $O(|\mathcal{L}((C,D))|)$. So, the aggregate complexity of line 12 of FASTER_MCAST_QOS is:

$$O(M\log M) + \sum_{C\in\mathcal{B}(\mathcal{T})}\sum_{D\in\mathcal{D}_C}(O(M) + O(|\mathcal{L}((C,D))|)) = O(M\log M + N^2 M)\ .$$

Consequently, FASTER_MCAST_QOS has a worst case running time of $O(M \log M + N^2 M)$.

We conclude that, in the case of QoS division without resource limitation, FASTER_MCAST_QOS reduces the worst case running time to $O(N^2 + M)$ from $O(NM)$ for MCAST_QOS. In the case of QoS division with resource limitation, FASTER_MCAST_QOS reduces the worst case running time to $O(N^2 M + M \log M)$ from $O(NM^2)$ for MCAST_QOS. These reductions are important since, in general, the number of leaves (N) of a tree is significantly smaller than the number of links (M) in that tree.

2.4.3 Distributed, Sender-Initiated Reservation Algorithms for Static Multicast

So far, we have presented a set of centralized algorithms for computing link QoS requirements and for reserving corresponding link resources. Such a centralized approach, despite the advantage of being conceptually simple, may impose excessive computational burdens on a single node.

In this section we propose a distributed version of the algorithm in which each multicast node computes the QoS requirement for its outgoing links in the multicast tree. The outline of the sender-initiated, distributed algorithm is as follows (see also Figure 2.10).

1. The algorithm starts in the sender where it is given the set of receivers $\mathcal{D}$ and their respective end-to-end QoS requirement, $(Q_{(S,D)})_{D \in \mathcal{D}}$.

2. The sender starts the first phase, named "Setup", by sending *Setup* messages to receivers (Figure 2.10(a)). In this phase, the messages accumulate information regarding QoS limitations, $(Q_l^{min})_{l \in \mathcal{L}(S,D)}$, and resource availability $(A_l)_{l \in \mathcal{L}(S,D)}$ on the links traversed, and store part of it at the nodes visited. We present the details of this phase in Section 2.4.3.1.

3. The Setup phase ends when the *Setup* messages have reached the receivers (Figure 2.10(b)). Each receiver performs a separate admission control: a local accept is

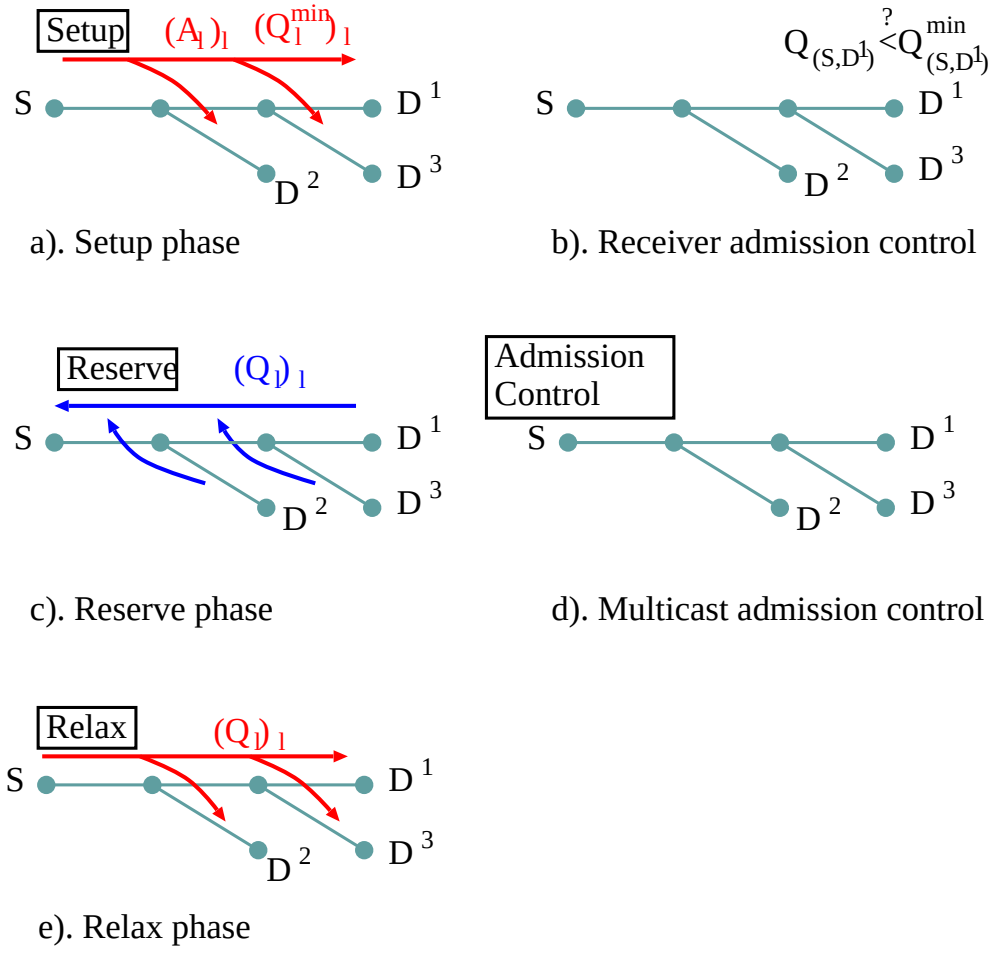


Figure 2.10. The mechanism of distributed, sender-initiated resource reservation

generated if the end-to-end QoS requirement for this receiver is looser than the tightest QoS guaranteeable on this path, $Q^{min}_{(S,D)} \leq Q_{(S,D)}$. In case the receiver admission control fails, a *Reject* message is returned to the source.

4. In case a receiver chooses to admit a session, it returns a *Reserve* message to the sender, marking the beginning of the second, "Reservation" phase (Figure 2.10(c)). This message traverses the multicast tree upstream, triggers link QoS computation and resource reservation on traversed links. We present in detail an algorithm for this QoS computation in Section 2.4.3.1.

5. After receiving *Reserve* or *Reject* messages from all receivers, the source decides on whether to accept or reject the multicast session (Figure 2.10(d)). In case it rejects the multicast session, the source sends a *Release* message to all nodes for releasing all reserved link resources. Otherwise, the multicast session can start transmitting data.

6. During the Reservation phase, some QoS requirements may have been guaranteed in excess of the end-to-end requirements (see for example Figure 2.14). To relax these QoS guarantees and reclaim the corresponding resources, a third, "Relaxation" phase is initiated by the source by sending *Relax* messages to receivers (Figure 2.10(e)). The QoS guarantees are relaxed on some of the traversed links, as described in detail in Section 2.4.3.1. At the end of this phase, all end-to-end QoS guarantees are equal to the end-to-end QoS requirements.

2.4.3.1 A Distributed, Sender-Initiated QoS Computation Algorithm

PROCESS_SETUP_MSG($SetupMsg$={	X,	source of message
	A,	destination of message
	$(Q_{(S,D)})_{D\in\mathcal{D}_A}$,	end-to-end QoS requirements
	$Q^{min}_{(S,A)}$,	minimum guaranteeable QoS on (S, A)
	$V_{(S,A)}$})	resource availability on (S, A)

1 Store $(Q_{(S,D)})_{D\in\mathcal{D}_A}$, $V_{S,A}$ at A
2 **for** l outgoing from A in $\mathcal{T}$ **do**
3 **let** B such that $A \xrightarrow{l} B$
4 Get Q^{min}_l from link l
5 $Q^{min}_{(S,B)} \leftarrow Q^{min}_{(S,A)} + Q^{min}_l$
6 $V_{S,B} \leftarrow V_{S,A} + C_l(A_l)$
7 $SetupMsg \leftarrow \{A, B, (Q_{(S,D)})_{D\in\mathcal{D}_B}, Q^{min}_{(S,B)}, V_{S,B}\}$
8 Send $SetupMsg$ to B

Figure 2.11. Processing of Setup message for distributed, sender-initiated protocol

PROCESS_RESERVE_MSG($ReserveMsg$=$\{B$, source of message
A, destination of message
$Q_{(S,B)}$, QoS requirement for (S, B)
$V_{(S,D)}\}$) resource availability on (S, D)

1 Store $V_{(S,D)}$ at A
2 **let** l such that $A \xrightarrow{l} B$
3 $Q_l^{new} \leftarrow \frac{C_l(A_l)Q_{S,B}}{V_{S,B}}$
4 **if** Q_l defined **and** $Q_l > Q_l^{new}$
5 **then** Release $F_l(Q_l)$
6 **if** Q_l undefined **or** $Q_l > Q_l^{new}$
7 **then** $Q_l \leftarrow Q_l^{new}$
8 Store Q_l
9 Reserve $F_l(Q_l)$
10 $Q_{(S,A)} \leftarrow Q_{(S,B)} - Q_l$
11 **let** X and m such that $X \xrightarrow{m} A$ in $\mathcal{T}$
12 $ReserveMsg \leftarrow \{A, X, Q_{(S,A)}, V_{S,D}\}$
13 Send $ReserveMsg$ to X

Figure 2.12. Processing of Reserve message for distributed, sender-initiated protocol

PROCESS_RELAX_MSG($RelaxMsg$=$\{X$, source of message
A, destination of message
$Q_{(S,A)}\}$) QoS guaranteed for (S, A)

1 **if** (A is not a receiver)
2 **then for** l outgoing from A in $\mathcal{T}$ **do**
3 **let** B such that $A \xrightarrow{l} B$
4 **if** B is a receiver
5 **then** $Q_l^{new} \leftarrow Q_{S,B} - Q_{S,A}$
6 **else for** $D \in \mathcal{D}$ **do**
7 $Q_l^D \leftarrow \frac{C_l(A_l)(Q_{S,D} - Q_{S,A})}{V_{S,D} - V_{S,A}}$
8 $Q_l^{new} \leftarrow \min_{D \in \mathcal{D}_l} Q_l^D$
9 **if** $Q_l^{new} > Q_l$
10 Release $F_l(Q_l)$
11 Reserve $F_l(Q_l^{new})$
12 $Q_l \leftarrow Q_l^{new}$
13 Store Q_l
14 $Q_{(S,B)} \leftarrow Q_{(S,A)} + Q_l$
15 $RelaxMsg \leftarrow \{A, B, Q_{(S,B)}\}$
16 Send $RelaxMsg$ to B

Figure 2.13. Processing of Relax message for distributed, sender-initiated protocol

In this section we present a sender-initiated distributed algorithm for QoS computation and resource reservation. The algorithm starts with a Setup phase, where a Setup message is passed hop-by-hop downstream the multicast tree. Its form is

$$SetupMsg = \{X, A, (Q_{(S,D)})_{D \in \mathcal{D}_A}, Q^{min}_{(S,A)}, V_{S,A}\}$$

where

X	is the source of message
A	is the destination of message
$(Q_{(S,D)})_{D \in \mathcal{D}_A}$	is the end-to-end QoS requirements
$Q^{min}_{(S,A)}$	is the minimum guaranteeable QoS on (S, A)
$V_{(S,A)}$	is the resource availability on (S, A)

The algorithm starts when the source S of the multicast session sends the following Setup message to all its successor nodes A in the multicast tree:

$$SetupMsg = \{S, A, (Q_{(S,D)})_{D \in \mathcal{D}_A}, Q^{min}_{(S,S)} = 0, V_{S,S} = 0\}$$

The message triggers the computation execution of PROCESS_SETUP_MSG (Figure 2.11) at each node A that it traverses. It accumulates the minimum QoS guaranteeable on the path traversed so far, $Q^{min}_{(S,A)}$, and the resource availability on the same path, $V_{S,A}$. $Q^{min}_{(S,A)}$ is needed at the receivers for local admission control, and $V_{S,A}$ is needed at the current node for QoS computation during the Reservation phase. Setup messages are generated and transmitted to all of A's successor nodes in $\mathcal{T}$.

At the end of the Setup phase, each receiver D receives a Setup message. The local admission control test calculates the value of the predicate $Q^{min}_{(S,D)} < Q_{(S,D)}$ where both $Q^{min}_{(S,D)}$ and $Q_{(S,D)}$ were transported by the Setup message. If the above inequality is false,

the receiver returns a Reject message to S, otherwise a Reserve message. In the Reserve phase a Reserve message is passed hop-by-hop upstream the multicast tree. Its form is

$$ReserveMsg = \{B, A, Q_{(S,B)}, V_{S,D}\}$$

where

B	is the source of message
A	is the destination of message
$Q_{(S,B)}$	is the QoS requirement for (S, B)
$V_{(S,D)}$	is the resource availability on (S, D)

If the admission control succeeds, receiver D sends a Reserve message to its predecessor A with the content:

$$ReserveMsg = \{D, A, Q_{(S,D)}, V_{S,D}\}$$

where $V_{S,D}$ is taken from the Setup message. The Reserve phase is thus started.

For each Reserve message received, a node performs the algorithm PROCESS_RESERVE_MSG (Figure 2.12). It first computes a candidate Q_l^{new} for the QoS to be guaranteed on l, using a QoS division formula derived from (2.1). If Q_l^{new} is a tighter requirement than Q_l, a QoS guaranteed by a preceding Reserve message, then Q_l^{new} becomes the current QoS guarantee Q_l and resources are reserved accordingly. The rest of the not-yet-guaranteed QoS budget, $Q_{(S,A)}$, is sent to A's predecessor node X in $\mathcal{T}$. The end-to-end path resource availability $V_{S,D}$ is also stored at each node for use in the Relaxation phase.

After the source performs the multicast admission control and accepts the session, the Relaxation phase is started. In the Relaxation phase a Relax message is passed hop-by-hop downstream the multicast tree. Its form is

$$RelaxMsg = \{X, A, Q_{(S,A)}\}$$

where

X	is the source of message
A	is the destination of message
$Q_{(S,A)}$	is the QoS guaranteed for (S, A)

The Relaxation phase is started when the source S sends Relax messages to each of its immediate successors A, with the following content

$$RelaxMsg = \{S, A, Q_{(S,S)} = 0\}$$

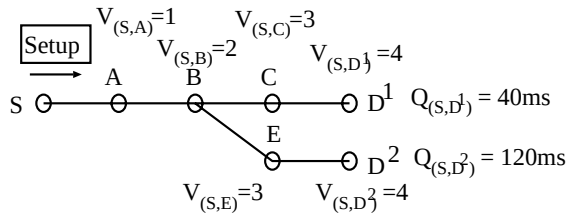
Each node A executes PROCESS_RELAX_MSG (Figure 2.13) upon receiving a Relax message. A new QoS requirement is computed at each node A, considering all receivers $D \in \mathcal{D}_l$ downstream the current link l outgoing from A. When the computed value Q_l^{new} is larger than the current guarantee Q_l, there exists an opportunity for QoS relaxation. The relaxation is done by reserving resources to guarantee Q_l^{new} instead of Q_l. Finally, a Relax message is sent to A's successors B, with a relaxed QoS guarantee of $Q_{(S,B)}$.

Example. We illustrate the sender-initiated algorithm in Figure 2.14. Setup messages collect link information in Figure 2.14(a). We use the Even QoS division policy, thus $C_l(A_l) = 1$ for any link l, and $V_{S,A}$ is equal to the hop count of the path (S, A).

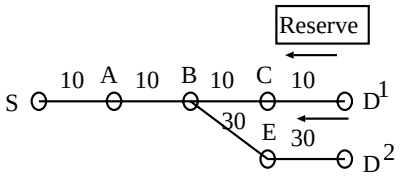
The result of the Reservation phase (Figure 2.14(b)) is an equal distribution of QoS shares: $10ms$ for each link on (S, D^1) and $30ms$ for each link on (S, D^2). For (S, A) and (A, B), a QoS of $10ms$ is guaranteed since it is the more stringent requirement between $10ms$ and $30ms$. Observe that the resulting end-to-end QoS guarantee on (S, D^2) is $80ms$, which is more stringent than the QoS requirement of $120ms$.

Thus, the opportunity for relaxation occurs at node B, where

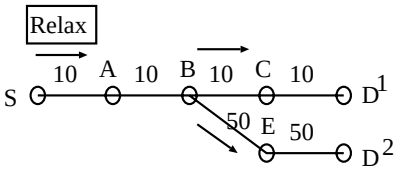
$$Q_{(B,E)}^{new} = \frac{Q_{(S,D^2)} \Leftrightarrow Q_{(S,B)}}{V_{S,D} \Leftrightarrow V_{S,B}} = \frac{100}{2} = 50ms > Q_{(B,E)} = 30ms$$



a) Setup phase: compute $(V_{(S,X)})_X$



b) Reservation phase:compute $(Q_l)_l$



c) Relaxation phase: relax $(Q_l)_l$

Figure 2.14. An example of the distributed, sender-initiated QoS computation algorithm

At the end of the Relaxation phase, the QoS guarantees on links (B, E) and (E, D^2) are relaxed from $30ms$ to $50ms$.

Complexity Analysis. In the follwing we determine the computational complexity of our distributed algorithm at an arbitrary node A in the multicast tree, and the communication complexity (size of the messages) needed by our reservation algorithm.

We denote by $N_A = |\mathcal{D}_A|$ the number of receivers downstream A. We consider first PROCESS_SETUP_MSG (Figure 2.11). Line 1 is $O(|\mathcal{D}_A|) = O(N_A)$, and the loop 2-8 has $|\mathcal{B}_A| \leq |\mathcal{D}_A| = N_A$ iterations. Thus the computational complexity of Setup processing is $O(N_A)$ at node A.

PROCESS_RESERVE_MSG (Figure 2.12) is $O(1)$, and there are N_A such messages input to node A (one from each receiver $D \in \mathcal{D}_A$). Thus, the complexity of the Reservation phase is $O(N_A)$ at node A. In PROCESS_RELAX_MSG (Figure 2.13) the loop 2-16 is executed for each $l \in \mathcal{B}_A$ ($\mathcal{B}_A$ is the set of links outgoing from A in $\mathcal{T}$) with an inner loop 6-7 and 8 of

$D \in \mathcal{D}_l$. The aggregate complexity is

$$\sum_{l \in \mathcal{B}_A} \sum_{D \in \mathcal{D}_l} 1 = \sum_{D \in \mathcal{D}_A} 1 = N_A$$

Thus, the Relaxation phase is also $O(N_A)$. We conclude that the complexity of the distributed algorithm is $O(N_A)$ for node A. Observe that the total complexity of the distributed algorithm for the whole tree is $O(NM)$ since

$$\sum_{A \in \mathcal{N}(\mathcal{T})} N_A \leq \sum_{A \in \mathcal{N}(\mathcal{T})} N = NM$$

This is equal to the complexity of the centralized algorithm MCAST_QOS (Figure 2.5).

The communication complexity of a distributed algorithm is given by the size of its data messages (see for example [71]). The size of the Setup message incoming to a node A is $O(N_A)$ since it includes $(Q_{(S,D)})_{D \in \mathcal{D}_A}$. The size of a Reserve message is $O(1)$, and since there are N_A Reserve messages incoming to A, the communication complexity of Reserve is $O(N_A)$. The size of Relax messages is $O(1)$. Thus, the three-phase distributed algorithm has communication complexity of $O(N_A)$ at a given node A in the multicast tree.

2.4.4 Distributed, Receiver-Initiated Reservation Algorithms for Dynamic Multicast

In this section we present a distributed algorithm for dynamic multicast sessions, where receivers join and leave the multicast session at any time.

This algorithm is similar to the distributed algorithm presented in Section 2.4.3 in that it consists of three phases: link information gathering, reservation and relaxation phases. It differs from the sender-initiated distributed algorithm in that the reservations are initiated independently by each receiver joining the multicast session. Its design is inspired by the RSVP protocol [14]. The outline of the receiver-initiated, distributed algorithm is as follows (see also Figure 2.15).

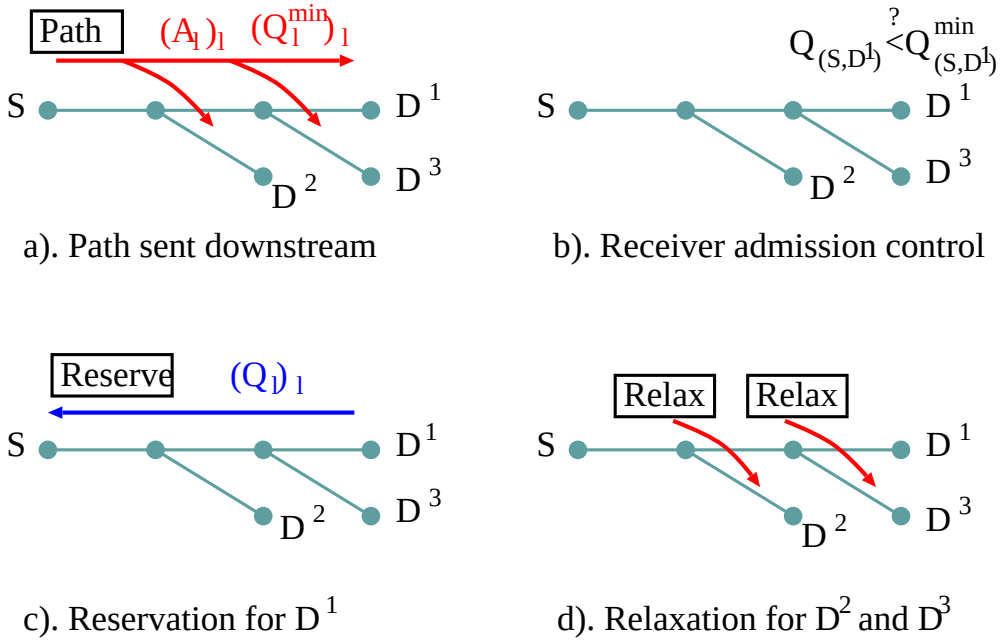


Figure 2.15. The mechanism of distributed, receiver-initiated resource reservation

1. The multicast source sends periodic *Path* messages to its current receivers, irrespective of their having resources reserved or not. (Figure 2.15(a)). The messages accumulate information regarding QoS limitations, $(Q_l^{min})_{l\in\mathcal{L}(S,D)}$, and resource availability $(A_l)_{l\in\mathcal{L}(S,D)}$ on the links traversed, and store parts of this information at the nodes visited. We present the details of these operations in Section 2.4.4.1.

2. When a *Path* message reaches a receiver D^1, the receiver can perform a local admission control (Figure 2.15(b)): if the end-to-end QoS requirement for this receiver is larger than the minimum QoS that can be guaranteed on this path, $Q^{min}_{(S,D^1)} \leq Q_{(S,D^1)}$, then a reservation can be initiated.

3. If there are sufficient resources, the receiver sends a *Reserve* message to the sender (Figure 2.15(c)). This message traverses the multicast tree upstream, triggers link QoS computation and resource reservation on traversed links. We present in detail an algorithm for this QoS computation in Section 2.4.4.1.

4. During resource reservation, some QoS requirements may be guaranteed in excess of the end-to-end requirements for receivers other than D^1. To relax these QoS guarantees and reclaim the corresponding resources, each branching node detecting a relaxation opportunity sends *Relax* messages on its subtree (to D^2 and D^3 in Figure 2.15(d)). The QoS guarantees are relaxed on the traversed links, as described in detail in Section 2.4.4.1.

2.4.4.1 A Distributed, Receiver-Initiated QoS Computation Algorithm

PROCESS_PATH_MSG($PathMsg$={X,		source of message
	A,	destination of message
	$Q^{min}_{(S,A)}$,	minimum guaranteeable QoS on (S, A)
	$V_{(S,A)}$})	resource availability on (S, A)

1 Store $V_{S,A}$ at A
2 **for** l outgoing from A in $\mathcal{T}$ **do**
3 **let** B such that $A \xrightarrow{l} B$
4 Get Q^{min}_l from link l
5 $Q^{min}_{(S,B)} \leftarrow Q^{min}_{(S,A)} + Q^{min}_l$
6 $V_{S,B} \leftarrow V_{S,A} + C_l(A_l)$
7 $PathMsg \leftarrow \{A, B, Q^{min}_{(S,B)}, V_{S,B}\}$
8 Send $PathMsg$ to B

Figure 2.16. Processing of Path message for distributed, receiver-initiated protocol

In this section we present a receiver-initiated distributed algorithm for QoS computation and resource reservation. The source S of the multicast session periodically sends $Path$ messages that are passed hop-by-hop downstream the multicast tree. Their general form is

$$PathMsg = \{X, A, Q^{min}_{(S,A)}, V_{S,A}\}$$

where

X	is the source of message
A	is the destination of message

```
PROCESS_RESERVE_MSG(ReserveMsg={B,        source of message
                                A,        destination of message
                                Q_(S,B),  QoS requirement for (S, B)
                                V_(S,D)}) resource availability on (S, D)

1  Store V_(S,D) at A
2  let l such that A --l--> B
3  let X and m such that X --m--> A in T
4  Q_l^new ← C_l(A_l)Q_(S,B) / V_(S,B)
5  Q_(S,A)^new ← Q_(S,B) − Q_l^new
6  if Q_(S,A) undefined
7     then Q_l ← Q_l^new
8          Q_(S,A) ← Q_(S,A)^new
9          Store Q_l, Q_(S,A)
10         Reserve F_l(Q_l)
11         ReserveMsg ← {A, X, Q_(S,A), V_(S,D)}
12         Send ReserveMsg to X
13    else if Q_(S,A) < Q_(S,A)^new
14            then Relax on l given Q_(S,A): execute lines 4-16 of Figure 2.18
15                 return
16            else Q_(S,A) ← Q_(S,A)^new
17                 Store Q_(S,A)
18                 if Q_l defined and Q_l > Q_l^new
19                   then Release F_l(Q_l)
20                 if Q_l undefined or Q_l > Q_l^new
21                   then Q_l ← Q_l^new
22                        Store Q_l
23                        Reserve F_l(Q_l)
24                 Q_(S,A) ← Q_(S,B) − Q_l
25                 ReserveMsg ← {A, X, Q_(S,A), V_(S,D)}
26                 Send ReserveMsg to X
27                 for n outgoing from A in T, n ≠ l do
28                     let C such that A --n--> C
29                     RelaxMsg ← {A, C, Q_(S,A)}
30                     Send RelaxMsg to C
```

Figure 2.17. Processing of Reserve message for distributed, receiver-initiated protocol

$Q^{min}_{(S,A)}$ is the minimum guaranteeable QoS on (S, A)

$V_{(S,A)}$ is the resource availability on (S, A)

The source S of the multicast session periodically sends the following $Path$ message to all its successor nodes A in the multicast tree:

```
PROCESS_RELAX_MSG(RelaxMsg={X,        source of message
                            A,        destination of message
                            Q_(S,A)}) QoS guaranteed for (S, A)

1  if (A is not a receiver)
2    then for l outgoing from A in T do
3           let B such that A --l--> B
4           if B is a receiver
5             then Q_l^new ← Q_(S,B) − Q_(S,A)
6             else for D ∈ D do
7                       Q_l^D ← C_l(A_l)(Q_(S,D) − Q_(S,A)) / (V_(S,D) − V_(S,A))
8                  Q_l^new ← min_(D∈D_l) Q_l^D
9           if Q_l^new > Q_l
10            Release F_l(Q_l)
11            Reserve F_l(Q_l^new)
12            Q_l ← Q_l^new
13            Store Q_l
14          Q_(S,B) ← Q_(S,A) + Q_l
15          RelaxMsg ← {A, B, Q_(S,B)}
16          Send RelaxMsg to B
```

Figure 2.18. Processing of Relax message for distributed, receiver-initiated protocol

$$PathMsg = \{S, A, Q^{min}_{(S,S)} = 0, V_{S,S} = 0\}$$

For each node A that this message traverses, the message triggers the execution of PROCESS_PATH_MSG (Figure 2.16). It accumulates the minimum guaranteeable QoS on the path traversed so far, $Q^{min}_{(S,A)}$, and the resource availability on the same path, $V_{S,A}$. $Q^{min}_{(S,A)}$ is needed at receivers for local admission control, and $V_{S,A}$ is needed at the current node A when processing Reserve messages.

After receiving a Path message, a receiver D having a QoS requirement $Q_{(S,D)}$, first computes the predicate $Q^{min}_{(S,D)} < Q_{(S,D)}$, where $Q^{min}_{(S,D)}$ was transported by the Path message. If the above inequality is false, the receiver's request cannot be guaranteed, otherwise a Reserve message. The Reserve messages are passed hop-by-hop upstream the multicast tree. Their general form is

$$ReserveMsg = \{B, A, Q_{(S,B)}, V_{S,D}\}$$

where

B	is the source of message
A	is the destination of message
$Q_{(S,B)}$	is the QoS requirement for (S, B)
$V_{(S,D)}$	is the resource availability on (S, D)

If the admission control succeeds, receiver D sends a Reserve message to its predecessor A with the content:

$$ReserveMsg = \{D, A, Q_{(S,D)}, V_{S,D}\}$$

where $V_{S,D}$ is taken from the Path message.

When node A receives a Reserve message, A computes the necessary resources for this receiver and performs the corresponding reservation if it was not done already for another receiver. The algorithm for this is PROCESS_RESERVE_MSG (Figure 2.17). It first computes (lines 4-5) the link requirement Q_l^{new} derived from D's end-to-end requirement, and the rest $Q_{(S,A)}^{new}$ of D's requirement for the path (S, A). If there is no previous reservation on (S, A), then (lines 7-12) resources are reserved on link l to guarantee Q_l^{new}, and a Reserve message is sent to A's predecessor node.

In the case that a reservation exists for (S, A) and it is more stringent than the requirement $Q_{(S,A)}^{new}$, then the current end-to-end guarantee on (S, D) is more stringent than D's requirement. A QoS relaxation is thus initiated (lines 14-15) on the path (A, D).

On the other hand, if the new requirement $Q_{(S,A)}^{new}$ is more stringent than the existing guarantee $Q_{(S,A)}$, then (lines 16-26) a new reservation is done for link l (if Q_l^{new} is more stringent than the existing one, if any) and a reservation message is sent upstream to A. Also, since the new requirement $Q_{(S,A)}^{new}$ is more stringent than the existing one, a relaxation is initiated (lines 27-30) for the subtrees rooted at A, that do not contain the (A, D) path.

The QoS relaxation procedure PROCESS_RELAX_MSG (Figure 2.18) is identical to the one presented for the sender-initiated algorithm.

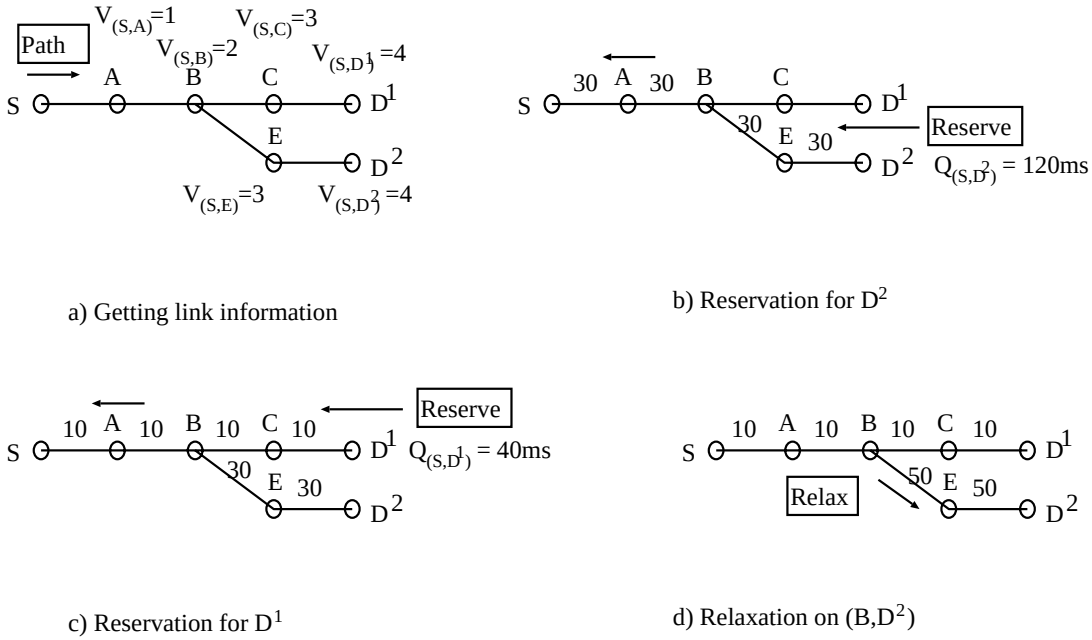


Figure 2.19. An example of the distributed, receiver-initiated QoS computation algorithm

Example. We illustrate our receiver-initiated algorithm in Figure 2.19. Path messages collect link information in Figure 2.19(a). In our example we use Even QoS division policy, thus $C_l(A_l) = 1$ for any link l, and $V_{S,A}$ is equal to the hop count of the path (S, A).

Receiver D^2 is the first to request reservation for an end-to-end requirement $Q_{(S,D^2)} = 120ms$. Shares of $30ms$ are guaranteed on all links from D^2 to S. When D^1 requests an end-to-end guarantee $Q_{(S,D^2)} = 40ms$, reservations are done on (C, D^1) and (B, C) since there are no preexisting reservations for this sesson. On (A, B) and (S, A), reservations are made to tighten their respective guarantees to $10ms$.

Also, node B detects the opportunity for relaxation on (B, D^2) because

$$Q_{(S,B)} = 60ms > Q^{new}_{(S,B)} = 20ms$$

A Relax message is sent on (B, D^2) and the QoS guarantees on links (B, E) and (E, D^2) are relaxed from $30ms$ to $50ms$.

Complexity Analysis. In the following we determine the computational complexity of the receiver-initiated algorithm at an arbitrary node A in the multicast tree, and the communication complexity (size of the messages) needed by our reservation algorithm.

We denote by $N_A = |\mathcal{D}_A|$ the number of receivers downstream A. We consider first PROCESS_PATH_MSG (Figure 2.16). The loop 2-8 requires $|\mathcal{B}_A| \leq |\mathcal{D}_A| = N_A$ iterations. Thus the computational complexity of Path processing is $O(N_A)$ at node A. PROCESS_RESERVE_MSG (Figure 2.17) is $O(N_A)$ since the loop 27-30 has $|\mathcal{B}_A|$ iterations. PROCESS_RELAX_MSG (Figure 2.18) is also $O(N_A)$, as shown in Section 2.4.3.1. The Path, Reserve and Relax message are all fixed size.

2.5 An Application with Packet Loss Probability

In our first application, the QoS metric is packet loss probability. Some of the computations that follow are based on results presented in [48]. Through simulation, we evaluate the effectiveness of the Even, Proportional and Threshold division policies with and without resource reclamation.

2.5.1 The Network and the Link Scheduling Policy

The network consists of nodes that use Generalized Processor Sharing (GPS) as the link scheduling policy among L classes of sessions. A class is characterized by a packet loss probability that is guaranteed for all sessions that are admitted in that class. All packets belonging to sessions in a given class are transmitted in FIFO order. Given link l, the GPS class weights at link l are $\{\phi_{l,k}\}_{k=1,..,L}$ and are chosen such that $\sum_{k=1}^{L} \phi_{l,k} = 1$. All links have identical characteristics (service rate, QoS of each class, etc.) but not necessarily the same GPS weights $\{\phi_{l,k}\}_k$. These GPS coefficients will be dynamically adjusted to

accommodate various class loads, as described in Section 2.5.4.

2.5.2 The Local QoS, Composition and End-to-End Division

All sessions within class k will share a buffer of capacity B_k. Due to the use of FIFO for scheduling within a class, one can show (see [48]) that the loss probability is the same for all the sessions in this class, and that it has the following assimptotic behavior

$$\lim_{B_k \to \infty} \frac{1}{B_k} \log \Pr\left(X_k > B_k\right) \leq \Leftrightarrow \delta_k$$

which yields an exponential upper bound (valid for large values of B_k):

$$\Pr(X_k > B_k) \leq e^{-\delta_k B_k} \quad . \tag{2.4}$$

where X_k is the backlog (queue length) and δ_k is the exponential decay rate that links the loss probability to the effective bandwidth condition in (2.6). From now on we will assume that the set $(\delta_k)_{k=1,..,L}$ is fixed and identical for all links. Also we will assume $\delta_k > \delta_{k+1}, \quad k = 1,..,L \Leftrightarrow 1$, i.e. class 1 has the lowest loss probability. We will also assume that the end-to-end loss probabilities are $\ll 1$. In this case, given a path $P = (l_1,..,l_n)$, a bound for the end-to-end loss probability is given by (Section 2.3.2.1):

$$\Pr(\text{loss on } P) \approx \sum_{i=1}^{n} \Pr(X_{i,k_i} > B_{k_i}) < \sum_{i=1}^{n} e^{-\delta_{k_i} B_{k_i}} < Q_{(A_0,A_n)} \quad . \tag{2.5}$$

The reverse operation, the division of QoS, can thus be stated as follows: given an end-to-end packet loss probability $Q_{(A_0,A_n)}$ for the path P, choose a set $(k_i)_{i=1,..,n}$ (the class in which to admit the session at each link) such that equation (2.5) is satisfied. The QoS division is done as described in Section 2.3.2.2.

2.5.3 Session Arrival Process

We take packetized voice as the source traffic for a session. This traffic can be modeled as i.i.d. on/off Markov fluids (see [4] for a description of this model). Such a traffic model captures the behavior of bursty traffic and has been used in the literature [15] to model packetized voice. In this model a source can be in one of two states, "on" or "off". While in the "on" state, the source sends data at a constant rate of r, and while in the "off" (silence) period, the source sends nothing. The sojourn times in the "on" and "off" states are exponential random variables with means $1/\mu$ and $1/\lambda$ respectively. The admission control and resource reservation presented in Section 2.5.4 uses the theory of effective bandwidths taken from [48, 27]. The expression for the effective bandwidth of an on/off Markov fluid source is:

$$\alpha(\delta) = \frac{\delta r \Leftrightarrow \mu \Leftrightarrow \lambda + \sqrt{(\delta r \Leftrightarrow \mu + \lambda)^2 + 4\lambda\mu}}{2\delta} \quad . \tag{2.6}$$

where δ is a parameter that characterizes the QoS (loss probability) to be guaranteed, as described in Section 2.5.2.

Also we know (e.g. from [27]) that, because we will take $r < c$ (c is the maximum processing rate (capacity) of the server (link)), the effective bandwidth of the departure process is equal to that of the arrival process (on/off Markov fluid). Consequently, the session is characterized by the effective bandwidth given in (2.6) at all of the links in the multicast tree. In conclusion, combining formulas (2.4) and (2.6), and denoting $Q_k = \Pr(X_k > B_k)$ as the local QoS (loss probability), we derive a closed-form expression for the function F introduced in Section 2.3.1 :

$$F(Q_k) = \frac{\frac{-\log Q_k}{B_k} r \Leftrightarrow \mu \Leftrightarrow \lambda + \sqrt{(\frac{-\log Q_k}{B_k} r \Leftrightarrow \mu + \lambda)^2 + 4\lambda\mu}}{2\frac{-\log Q_k}{B_k}} \quad . \tag{2.7}$$

2.5.4 Admission Control and Resource Reservation

To simplify admission test computations we will use a conservative approximation of resource utilization in a server using the GPS queue discipline. We will assume that the GPS server with bandwidth c is divided into L sub-servers, one for each of the L classes, each having a bandwidth of $(\phi_k c)_{k=1,..,L}$. This assumption is conservative in the sense that the bandwidth of a class that is idle at one moment is assumed not to be available by the other classes. The session admission criterion for each sub-server k is:

$$\Pr(X_k > B_k) \leq e^{-\delta_k B_k} \quad \text{iff} \tag{2.8}$$

$$N_k \alpha(\delta_k) < \phi_k c \ , \tag{2.9}$$

where N_k is the number of sessions in class k. We take as the session admission criterion for the GPS server

$$\sum_{k=1}^{L} N_k \alpha(\delta_k) < c \ . \tag{2.10}$$

It is clear that, if (2.10) is satisfied, then

$$\exists (\phi_k)_{k=1,..,L} \ , \ 0 \leq \phi_k \leq 1 \ , \ \sum_{k=1}^{L} \phi_k = 1$$

such that (2.9) is satisfied for all $k = 1, .., L$ and so the QoS is guaranteed for all L classes as stated in (2.8).

2.5.5 A Numerical Example

We present an example of admission control, resource reservation and reclamation for a simple network and three classes of loss probability. The sessions' arrival process is an on/off Markov fluid modeling packetized voice having: mean "on" time $1/\mu = 0.352s$, mean "off" time $1/\lambda = 0.650s$, and peak rate $r = 32Kb/s$, values often used in the literature [15]. Each class has a buffer capacity of $B = 30Kb$. The loss probabilities and their corresponding effective bandwidths from (2.7) are:

P_j		0.001	0.005	0.017
$F(P_j)$	$[Kb/s]$	22.431	20.602	18.818

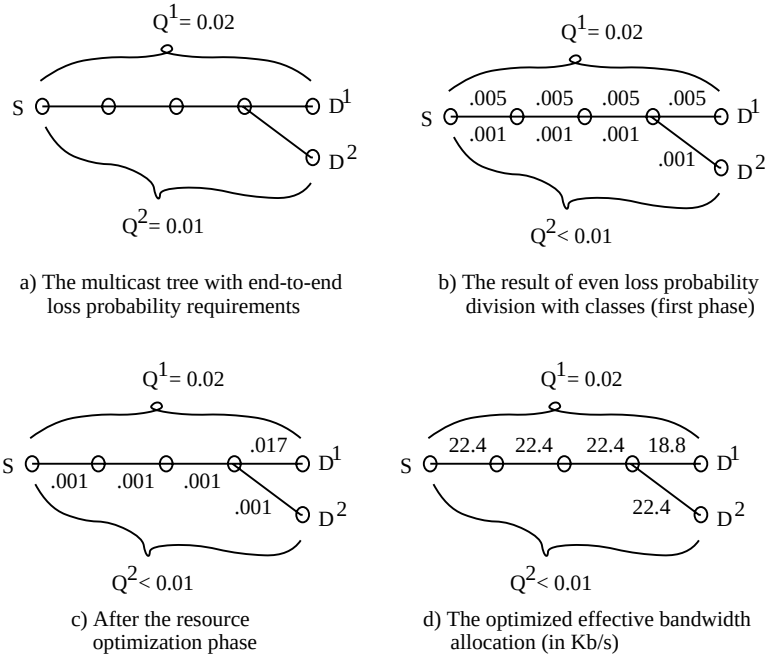


Figure 2.20. A numerical example

Let us now consider the network in Figure 2.20(a) where we would like to establish a multicast voice session from S to D^1 and D^2, with the loss probability requirements $Q_{(S,D^1)} = 0.02$ and $Q_{(S,D^2)} = 0.01$. Clearly the multicast session can be admitted using for example the lowest loss probability class in all links ($4 * 0.001 < Q_{(S,D^2)} < Q_{(S,D^1)}$). Now applying the Even division policy for QoS classes as in Section 2.3.2.2 we get the loss class allocation as in Figure 2.20(b). Finally, using the reclamation algorithm, we find that we can relax the loss class in D^1 from 0.005 to 0.017 as shown in Figure 2.20(c), thus allowing a reduction in resource reservation from $20.6Kb/s$ to $18.81Kb/s$. The final resource reservation is shown in Figure 2.20(d).

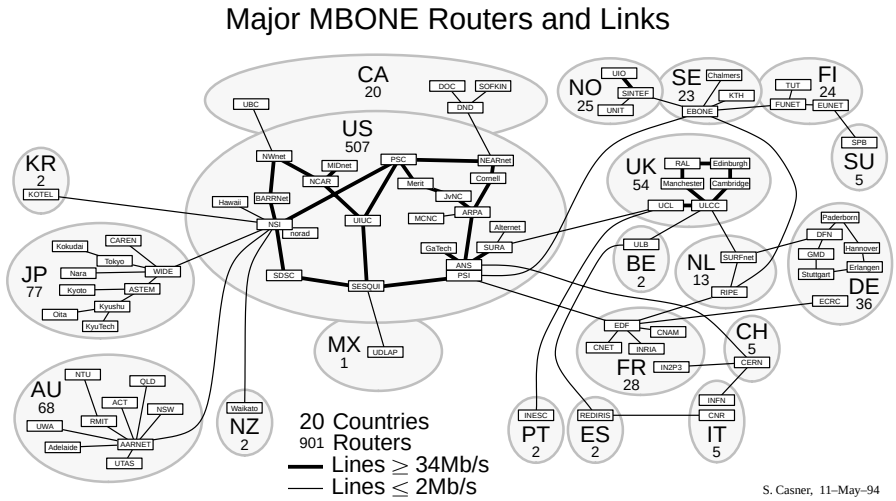


Figure 2.21. The extended MBone

2.5.6 Simulations

The voice multicast sessions operate on an *idealized* version of the Internet multicast backbone (MBone) as it existed on January 1995 (Figure 2.21) [20]. We assume that the links depicted with thick lines are T3 links (with capacity $45Mb/s$) and the remaining links are T1 (with capacity $1.544Mb/s$). Multicast sessions are generated according to a Poisson process with parameter α and their durations are exponentially distributed with mean $1/\beta$. The parameter $\rho = \alpha/\beta$ characterizes the load offered to the network, i.e. the average number of multicast sessions that would exist at any time in an infinite resource network.

Each multicast session has its source and receivers chosen with equal probability from the nodes of the network. The number of receivers is uniformly distributed in the range $[1, 83]$ (the total number of nodes in this network is 84). The arrival process for each session is described by an on/off Markov fluid modelling packetized voice with a mean "on" time $1/\mu = 0.352s$, mean "off" time $1/\lambda = 0.650s$, and peak rate $r = 32Kb/s$, values often used in the literature [15]. The end-to-end loss probability for each source-receiver pair is 10^{-a} where the exponent a is uniformly distributed in the range $[1, 3]$.

We take the number of classes at each node, L, to be 10 and the buffer capacity of each class to be $B_k = 30Kb$, $k = 1, \ldots, 10$. The QoS metrics associated with these classes are $q_k = F^{-1}(x)$ where $x = 26, 25. \ldots, 17$Kb/s. These QoS metrics cover the range of loss probabilities $[10^{-7}, 10^{-1}]$. We have observed in our study that the relative over-reservation of bandwidth due to admission in the next more stringent QoS class is less than 6%.

We use the method of independent replications to generate 90% confidence intervals for the performance metrics of interest to us. We simulate the Even division policy, the two

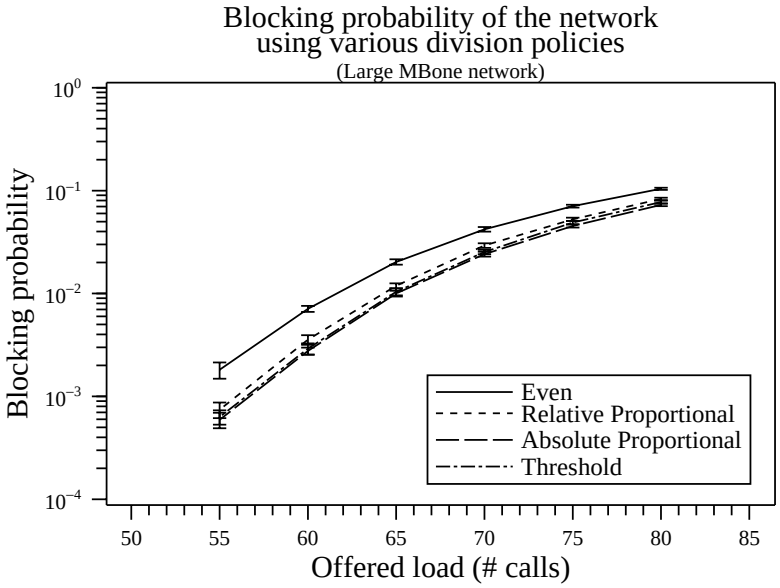


Figure 2.22. Absolute comparison of QoS division policies

Proportional division policies, and the Threshold policy, using the same offered workload,

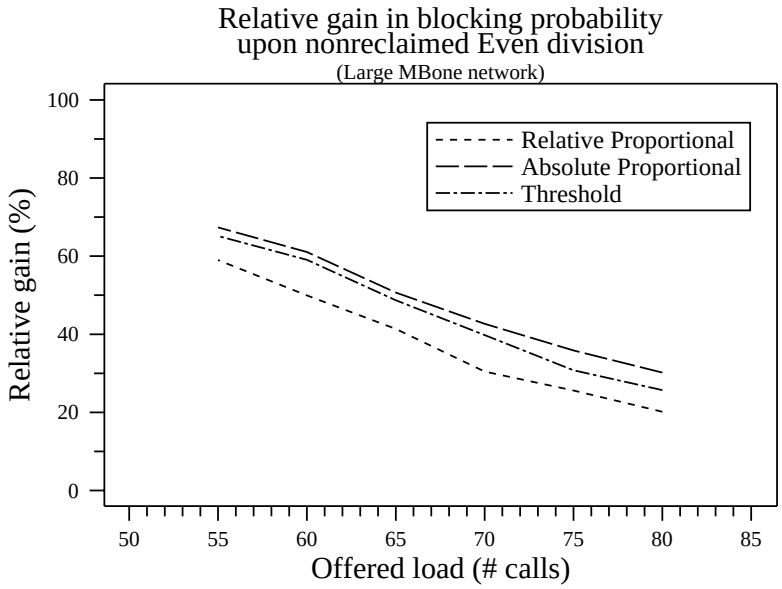


Figure 2.23. Relative comparison of QoS division policies

with and without the reclamation phase. Figure 2.22 shows the network's session blocking probability as a function of the multicast flow offered load ρ for the four policies without resource reclamation. Figure 2.23 illustrates the relative difference in the performance of Relative Proportional, Absolute Proportional and Threshold (with parameters $c_1 = 100$ and $c_2 = 1$) with respect to Even as a function of offered load. First, we observe that the Proportional and threshold policies perform significantly better than Even, with the Absolute performing the best (30% to 70% relative gain). We observe also that Threshold performs nearly as well as Absolute. Next, in Figure 2.24 and 2.25 we focus on the Even and Absolute policies and examine the benefits of adding the reclamation phase. We observe that resource reclamation provides a significant performance improvement to the Even policy. The Absolute policy also benefits from resource reclamation but to a lesser extent. Thus, it is desirable to include resource reclamation, especially in conjunction with the Even policy. Figure 2.24 can also be interpreted in another way. If the requirement is that the network blocking probability be $\sim 10^{-3}$ then the network can handle an offered load under the Absolute Proportional policy with reclamation which is $\sim 10\%$ greater than under the Even

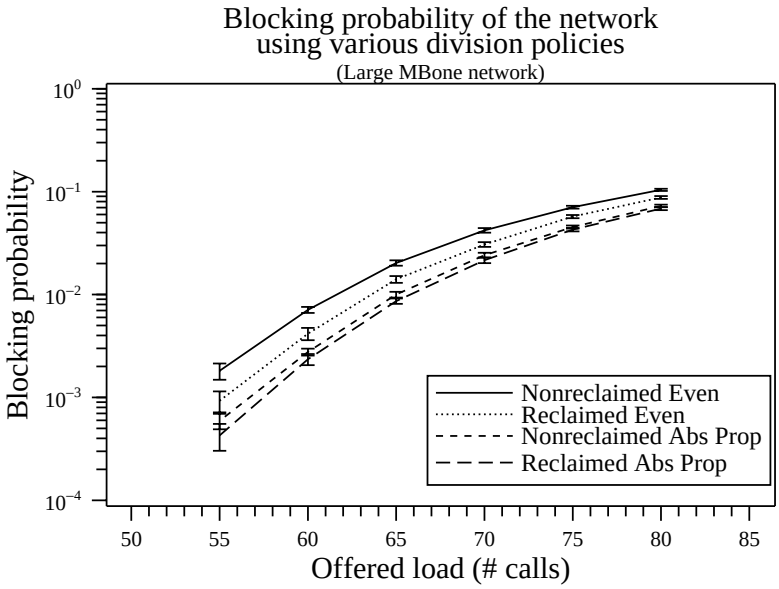


Figure 2.24. Absolute gain using proportional policy and reclaiming

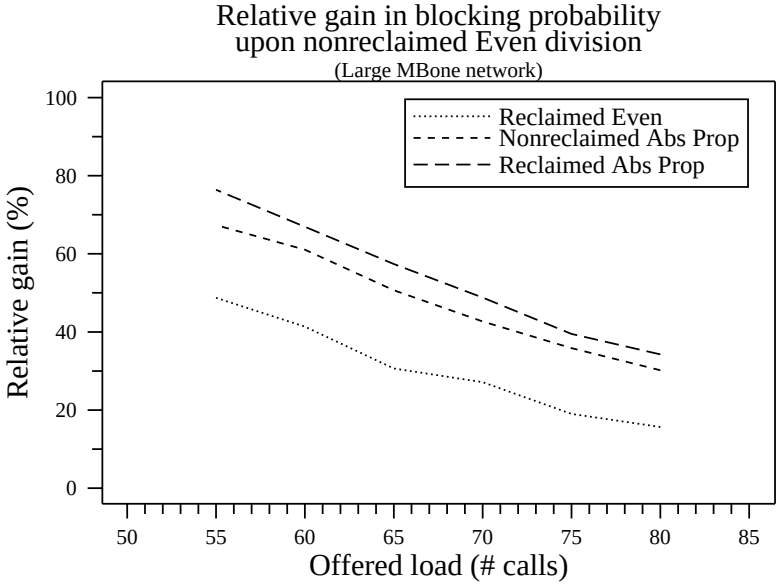


Figure 2.25. Relative gain using proportional policy and reclaiming

division policy without reclamation. Last, we study the dependence of the network blocking probability on the end to end QoS requirement (Figure 2.26) and on the multicast group size (Figure 2.27). The greatest differences between the policies occurs when the end-to-

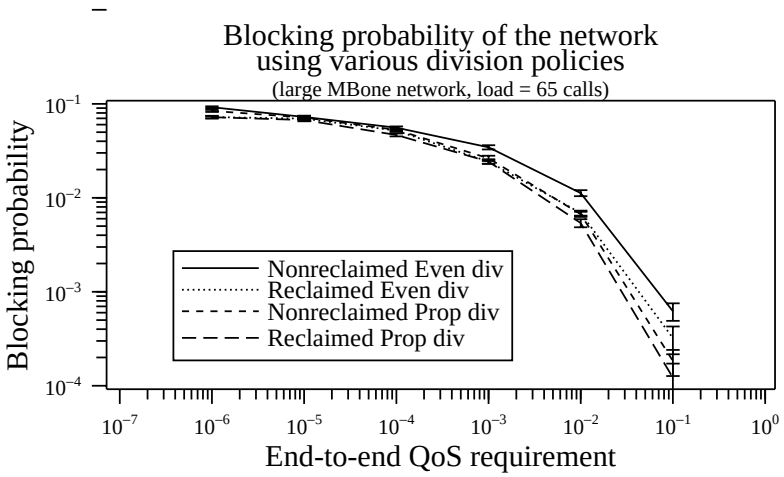


Figure 2.26. Blocking probability function of QoS requirement

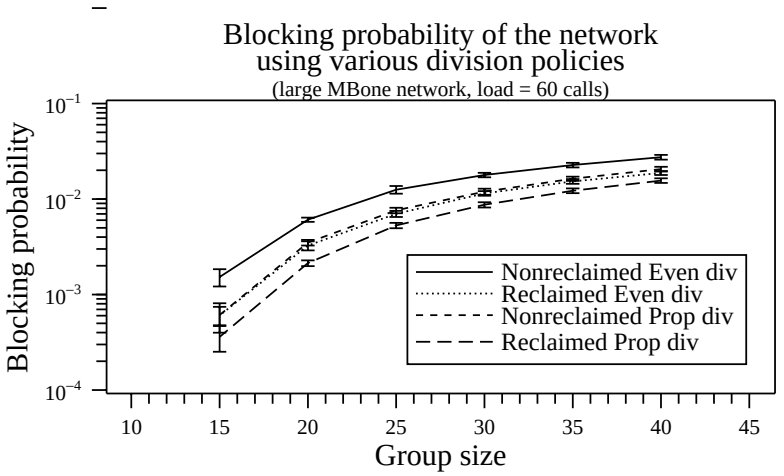


Figure 2.27. Blocking probability function of multicast group size

end QoS is high, say in the range $[10^{-2}, 10^{-1}]$. This is most likely due to the fact that the bandwidth requirement (given by (2.7)), for a session is much more sensitive to the QoS requirements when they are loose. Hence any opportunity to reserve and reclaim resources can result in more effective use of the resources. We also observe a highly nonlinear dependence of network blocking probability on group size (Figure 2.27). This is explained by the strong dependence of the amount of resources reserved on the number of receivers in a multicast session.

2.6 An Application with Packet Delay

In this second application, the network consists of nodes that use Rate Controlled [107] Earliest Deadline First (EDF) [35, 65] as the link scheduling policy among L classes of sessions (the same set of classes is used for all links in the network). Class k is characterized by maximum packet delay d_k guaranteed to all sessions admitted within that class. The source traffic for multicast sessions is taken to be MPEG video characterized by a leaky bucket with bucket size $\sigma = 576Kb$ and bucket rate $r = 518.4Kb/s$, values derived from the empirical envelope of a video trace presented in [61]. We show in Chapter 3 (based on [65]) that, in order to guarantee the delays for all sessions at a link (there are n_k sessions in class k which guarantees the delay d_k), it is necessary and sufficient to have $Lr < T$ and

$$d_k \geq \frac{\sigma \sum_{i=1}^{k} n_i \Leftrightarrow r \sum_{i=1}^{k} n_i d_i}{T \Leftrightarrow r \sum_{i=1}^{k} n_i} \quad k = 1, .., L \tag{2.11}$$

where T is the bandwidth of the link.

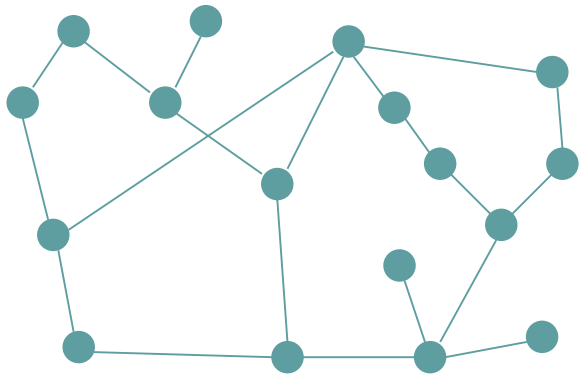


Figure 2.28. The T3 MBone

The video sources operate on an idealized version of the Internet multicast backbone (MBone) (as of January 1995), illustrated in Figure 2.28. This differs from the network used in the previous application in that we include only its T3 (45Mb/s) bidirectional links. Multicast sessions are generated according to a Poisson process with parameter α and their

durations are exponentially distributed with mean $1/\beta$. As before, $\rho = \alpha/\beta$ characterizes the load offered to the network. Each multicast session has its source and receivers chosen with equal probability from the nodes of the network. The number of receivers is uniformly distributed in the range $[1, 16]$. The end-to-end delay for each source-receiver pair is uniformly distributed in the range $[0.25, 2]$ seconds. Each node has 13 QoS classes corresponding to delays in the range $50ms$ to $1.5s$, where the delays within adjacent classes differ by a multiplicative factor of 1.3. As before, we use the method of independent replications to generate 90% confidence intervals.

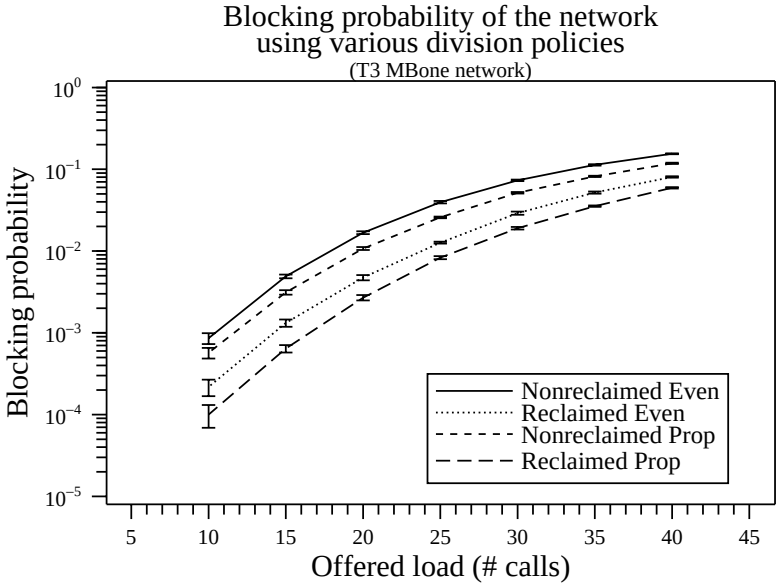


Figure 2.29. The blocking probability v.s. offered load

The same series of multicast sessions are simulated under the Even division and Proportional division policies with and without the reclamation phase. The Proportional policy has been modified so that C_i corresponds to the smallest delay $d_{min,i}$ that the link's EDF scheduler can guarantee, given its present load. Figure 2.29 shows the network blocking probability of a multicast session as a function of the offered load ρ, and Figure 2.30 shows the relative difference in the performance of Even division policy with resource reclamation, Proportional policy without resource reclamation and Proportional policy with re-

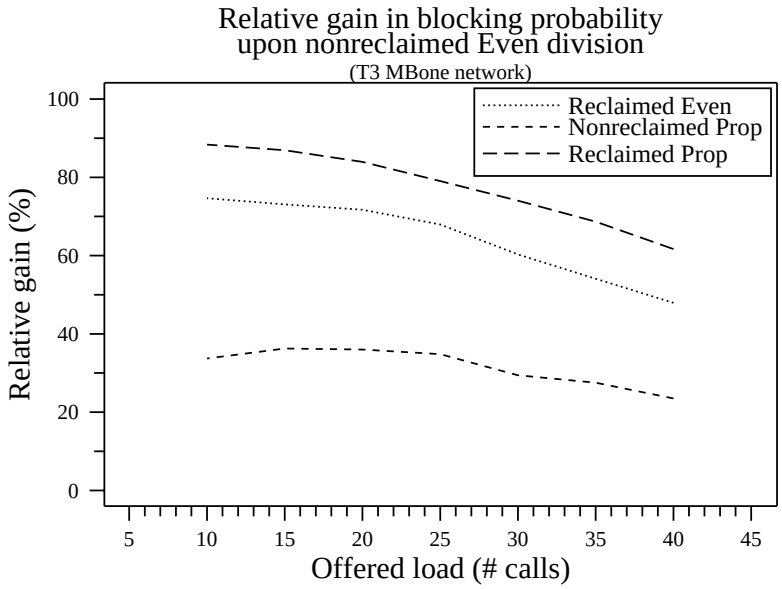


Figure 2.30. The relative gain

source reclamation with respect to Even policy without resource reclamation, as a function of offered load. First we observe that resource reclamation reduces blocking probabilities by almost an order of magnitude for both the Even and the Proportional policies. This improvement is considerably greater than in our previous application. This is due to the fact that the dependence of resources on QoS requirement is linear in the case of EDF scheduling coupled with delay guarantees, and logarithmic (2.7) in the case of packet loss probability guarantees and effective bandwidth reservation.

One can also use the results in Figure 2.29 to determine the increase in offered load that can be supported by a network by adding resource reclamation. In this application, resource reclamation permits the network to accept an additional 50% greater load while providing the same network blocking probability ($\sim 10^{-3}$).

2.7 Conclusion

In this chapter we have developed solutions to the problem of admission control and resource reservation for a multicast application once a route (multicast tree) has been chosen.

We have developed centralized resource reservation algorithms, where the QoS computation is done at a single place (for example the multicast source node), and distributed algorithms, where the QoS computations and resource reservation are done at the nodes in the multicast tree.

We presented a set of computationally efficient multicast reservation algorithms. A preliminary evaluation of these algorithms was presented in two different network settings: one having packet loss probability as QoS metric and the other, packet delay. We have demonstrated through simulation that our algorithms are highly effective: the network blocking probability decreases by a factor ranging from 2-3 in the case of loss probability and an order of magnitude in the case of delay. Wiewed from another perspective, our simulation results show that, by applying the resource reclamation in combination with the Proportional policy, the network can accept about 10% more sessions in the case of a loss requirement, and about 50% more sessions in the case of end-to-end delay requirement, while providing the same network blocking probability.

Although our applications focussed on packet loss probability and maximum packet delay as the QoS metrics, all of the algorithms apply equally to other additive QoS metrics such as the probability that the end-to-end delay exceeds a known quantity or a bound on delay jitter. An interesting direction of study is the behavior of various QoS division policies in the context of session correlations as is the case of filters proposed by RSVP.

CHAPTER 3

EFFICIENT ADMISSION CONTROL OF PIECEWISE LINEAR TRAFFIC ENVELOPES AT EDF SCHEDULERS

3.1 Introduction

The demand for real-time communication services in data networks such as the Internet has grown rapidly in recent years. Two important applications requiring the timely delivery of data packets are voice and video. To be able to guarantee the delay requirements of these applications, the network has to reserve resources at the links on the paths of the corresponding real-time flows. Several flow setup protocols that convey end-to-end user delay requirements to the links have been proposed and are in the process of standardization; these include RSVP [14] for the Internet, and ATM signaling [7] for ATM networks.

The problem of providing delay guarantees at a network link is the focus of much current research. Much of this work focuses on the issue of packet scheduling – determining the order in which queued packets are forwarded over outgoing links at switches and routers. This order determines the packet waiting times in the link's queue and, ultimately, the delays that the link scheduler can guarantee. A variant of Weighted Fair Queuing (WFQ) [30] (also known as Generalized Processor Sharing [80]) was proposed in [94] to guarantee a maximum queuing delay by reserving a certain amount of link bandwidth for the given flow. Although simple, this policy is known to be sub-optimal. Another discipline, Earliest Deadline First (EDF) [68] associates a per-hop deadline with each packet and schedules packets in the order of their assigned deadlines. EDF has been proven to be optimal in the sense that if a set of tasks is schedulable under any scheduling discipline (i.e., if the packets can be scheduled in such a way that all of their deadlines are met), then the

set is also schedulable under EDF. Also, Rate-Controlled EDF [107] was shown to outperform GPS in providing end-to-end delay guarantees in a network [42]. In the present work we use the Rate-Controlled EDF framework, where the end-to-end, delay-based admission control is reduced to performing EDF schedulability verifications at each link.

Sufficient conditions for the EDF schedulability of flows have been proposed for some particular cases of flow characterizations [55, 109]. Recently, a set of necessary and sufficient conditions for flow schedulability has been put forward in [66, 40, 90], using a general characterization of flows. The optimality of EDF and the existence of necessary and sufficient conditions for schedulability makes EDF an attractive choice for providing delay guarantees for real-time flows. There are, however, two important concerns about the practicality of EDF scheduling. First, an implementation of EDF scheduling requires a search of $O(\log Q)$ time in the list of packets (ordered by their deadlines) waiting in the queue of length Q for transmission. This issue has been addressed in [103, 66], where the search time is reduced to constant ($O(1)$) time in an approximate implementation where the range of packet deadline values is discretized. The second issue is that, although the EDF schedulability conditions in [66] can be expressed simply, the algorithms to perform these schedulability tests can be computationally very complex.

In this chapter we address the second issue, and present simple and computationally efficient algorithms for performing flow admission at links that use a EDF scheduler. That is, rather than considering various flow characteristics and associated admission control procedures [62, 35, 47], we take a specific traffic characterization (envelopes) and scheduling discipline (EDF), and examine the computational aspects of performing admission control in this setting.

Our proposed flow admission control algorithms are compatible with the emerging standards for flow setup protocols specified by the Internet Integrated Services [94] and the ATM signaling [7]. In the case that flows are described by envelopes characterized by piecewise linear functions with K segments, we find that our algorithms have low com-

plexity $O(KN\log(KN))$ where N is the number of admitted flows at the EDF scheduler at the moment of the algorithm's invocation. We further simplify these algorithms significantly by restricting the range of positions for the flexion points (i.e., the points attaching two linear segments) of traffic envelopes to a finite set of values. Simulation demonstrates that we obtain a very significant improvement in the run time (two orders of magnitude speedup for the examples we consider), with the additional benefit that the execution time is independent of the number of flows N. We examine the relative performance degradation (in terms of the number of flows admitted) incurred by the discretization and find it to be small.

In this chapter, we characterize flows by multiple-segment envelopes that bound the amount of generated traffic over an interval of length t. The motivation for this general characterization is that recent studies (for example [61]) have shown that, in order to achieve high link utilization, flows must be characterized by piecewise linear functions consisting of more than two segments. [61] has shown that moderately bursty traffic (e.g., some MPEG-encoded movies) achieve high link utilization (about 90%) when using envelopes with 3-4 segments. Moreover, highly bursty traffic (e.g., advertisements) need 10-15 segments in order to achieve the same result, while two-segment envelopes achieve only a third of the maximum link utilization. Multiple-segment envelopes are easy to specify: each segment is characterized by a (rate, burst size) pair. Multiple-segment envelopes are also easy to regulate and police (see e.g., [25]). A multiple-segment regulator is constructed with a set of leaky bucket regulators in series. It has also been shown in [107] that such a tandem of regulators does not introduce any additional worst case delay for the regulated flow.

The remainder of the chapter is organized as follows. In Section 3.2 we describe the requirements imposed by IP and ATM flow setup protocols on the local (link) admission control. In Section 3.3 we derive simple admission control algorithms for flows characterized by multiple-segment envelopes. In Section 3.4 we propose admission control procedures for non-preemptive EDF schedulers with non-negligible packet sizes. In Section 3.5 we

evaluate by simulation the performance of exact and discrete admission control algorithms. Section 3.6 concludes the chapter.

3.2 Flow Admission Control in Networks: EDF Schedulers

Flow setup protocols for flows with maximum end-to-end packet delay requirements, such as ATM signaling and RSVP with Guaranteed Services, impose certain requirements on flow link-level admission control algorithms. In this section, we examine these requirements; in Section 3.3 we present specific admission control algorithms that meet these requirements.

Consider a source that wishes to establish a flow f to a destination, using ATM Signaling. It sends a SETUP message to the destination, which includes information such as the flow's traffic characteristics (maximum cell rate, sustained cell rate, maximum burst size [7]), and the maximum allowable end-to-end delay, d_f. At each link l along the path from source to destination, the minimum delay that link l can guarantee to f, $\overline{d}_l$, is computed and added to d_c, the cumulative delay, included in the SETUP message. If, at some node, the cumulative delay exceeds the maximum allowable delay, the flow is not accepted and a RELEASE message is returned. Otherwise, at the end of the first pass (at the destination node), $d_f \geq d_c$ and the flow is accepted. A CONNECT message is returned on the same path to the source, assigning a delay $d_{f,l} \geq \overline{d}_l$ to flow f at link l on path P, such that $\sum_{l \in P} d_{f,l} \leq d_f$ according to some delay division policy (we have explored this in detail in Chapter 2).

Consider next the RSVP protocol [14] in conjunction with the Integrated Services "Guaranteed QoS" specification [94]. The source of a real-time flow sends periodic $Path$ messages to a unicast or multicast IP address. The source includes the flow's traffic characteristics in the $Path$ message. Each link l on the path to the receiver computes the minimum delay it can guarantee to f and adds it to d_c, the cumulative delay, which is sent in the D_{tot} term of the $Path$ message. A receiver that requires an end-to-end delay guaran-

tee d_f and receives a $Path$ message, compares d_f with the minimum end-to-end delay that can be guaranteed by the network, d_c. If $d_f < d_c$ then the receiver decides that its delay requirement cannot be guaranteed. If $d_f \geq d_c$ then the requirement can be satisfied, and the receiver sends a $Resv$ message back to the sender over the same route that $Path$ traversed. This includes d_f, its delay requirement, as part of the delay slack term S. On its return to the source, $Resv$ assigns a delay $d_{f,l} \geq \overline{d}_l$ to flow f at link l, such that $\sum_{l \in P} d_{f,l} \leq d_f$, according to some QoS division policy (studied in Chapter 2).

We see that each of the above flow setup protocols requires that a local admission control procedure be invoked at each link l with the following capabilities:

- given a flow f and its traffic characterization (e.g., maximum and average bandwidth requirement and maximum burst size or, more generally, any traffic envelope), provide the minimum delay that link l can guarantee to f, $\overline{d}_l$, based on the current state (set of reserved flows) at the local scheduler;

- given a flow f, its traffic characterization (as above), a requirement $d_{f,l} \geq \overline{d}_l$, and a set of currently accepted flows, update the current "state" of the local scheduler to reflect the fact that a maximum packet delay $d_{f,l}$ is additionally being guaranteed to flow f.

In the following we examine how these capabilities can be provided in the case where EDF scheduling is used to provide maximum end-to-end packet delay guarantees. Consider a flow f with the amount of arrivals (measured in bits/second) in the time interval $[\tau_1, \tau_2]$ denoted by $A_f[\tau_1, \tau_2]$. The flow is characterized by a traffic constraint function, or minimum envelope A_f^*, an upper bound on the flow's arrival pattern [26, 21]:

$$A_f[\tau, \tau + t] \leq A_f^*(t), \quad \forall \tau \geq 0, \forall t \geq 0$$

which also satisfies the relation $A_f^*(t) \leq \tilde{A}_f^*(t)$, $\forall t \geq 0$ for any envelope $\tilde{A}_f^*$ having the above property. It is easy to see that A_f^* is non-decreasing. We take $A_f^*(t) = 0, \forall t < 0$. In

this chapter we measure the traffic as a number of data units (bits) rather than transmission time (seconds), as in [66].

Let us consider a set of N flows, where flow i is characterized by its envelope A_i^*. The stability condition for a work-conserving scheduler (including the EDF scheduler) is ([66], eq. (5)):

$$\lim_{t\to\infty} \frac{\sum_{i=1}^{N} A_i^*(t)}{ct} < 1 \tag{3.1}$$

where c is the constant rate of the server (bits/second).

We consider now and in Section 3.3 preemptive EDF schedulers, and in Section 3.4, non-preemptive EDF schedulers. In the case of a *preemptive* EDF scheduler we state the following variant of the schedulability condition proposed in [66] for a set of N flows:

Theorem 3.1 (Liebeherr, Wrege, Ferrari) *Consider a set of N flows, that satisfy (3.1), where flow i is characterized by its envelope A_i^* and has a maximum packet delay requirement at a given link of d_i. The set of flows is EDF-schedulable at that link if and only if:*

$$ct \geq \sum_{1\leq i\leq N} A_i^*(t \Leftrightarrow d_i), \quad \forall t \geq 0 \tag{3.2}$$

We say that the set $\{A_i^*, d_i\}_{1\leq i\leq N}$ is schedulable if (3.1) and (3.2) are satisfied.

Note that equation (3.2) provides only a schedulability condition; it does not provide an algorithm to test this condition. In this chapter we present efficient algorithms for testing this condition. The following two properties of equations (3.1) and (3.2) ensure that EDF schedulers are capable of supporting the flow setup protocols described earlier.

Proposition 3.1 *If a set of flows $(A_i^*, d_i)_{1\leq i\leq N}$ is schedulable, then it remains schedulable if the maximum tolerable delay requirement for any flow is increased from d_k to $d_k + \delta$, $\delta > 0$, for any $1 \leq k \leq N$.*

The intuition behind this result is that, by relaxing the delay requirement for a flow in a schedulable set, the set remains schedulable.

Proof. Obviously the stability condition (3.1) is not affected by the increase of d_k. It is easy to see that the schedulability inequalities in (3.2) remain true when d_k increases. Taking $\delta > 0$ and knowing that A_k^* is non-decreasing, we have:

$$\begin{aligned} ct &\geq \sum_{i=1}^{N} A_i^*(t \Leftrightarrow d_i) \\ &= \sum_{i=1, i\neq k}^{N} A_i^*(t \Leftrightarrow d_i) + A_k^*(t \Leftrightarrow d_k) \\ &\geq \sum_{i=1, i\neq k}^{N} A_i^*(t \Leftrightarrow d_i) + A_k^*(t \Leftrightarrow d_k \Leftrightarrow \delta), \quad \forall t \geq 0 \end{aligned}$$

Q.E.D.

Corollary 3.1 *Given an EDF scheduler with a set of N admitted flows, for any new flow A_f^* there is a unique delay $\overline{d}(A_f^*)$ such that (A_f^*, d) can be admitted iff $d \geq \overline{d}(A_f^*)$.*

The delay $\overline{d}$ defined in Corollary 3.1, is the minimum (best) delay that can be guaranteed to flow f by the given EDF scheduler having the given load of N flows. The existence and uniqueness of the minimum delay $\overline{d}$ ensures that EDF schedulers are capable of supporting the flow setup protocols described earlier.

3.3 Admission Control Algorithms for Flows Characterized by Multiple-Segment Envelopes

In this section we address the problem of computing the admissibility of flows characterized by multiple-segment envelopes at an EDF scheduler, when each flow has a maximum packet delay requirement. We begin by introducing the definitions and notations related to multiple-segment envelopes.

Definition 3.1 *The multiple-segment envelope A_i^* of flow i is a function $A_i^* : \mathbb{R} \rightarrow \mathbb{R}$ with the following properties:*

1. *A_i^* is a piecewise linear function with a finite number of segments n_i:*

$$A_i^*(t) = \sigma_{i,j} + \rho_{i,j}t \quad a_{i,j} \leq t < a_{i,j+1}, \quad 0 \leq j \leq n_i \tag{3.3}$$

where $\sigma_{i,0} = 0$, $\sigma_{i,1} = 0$, $\rho_{i,0} = 0$, and

$$\Leftrightarrow\infty = a_{i,0} < 0 = a_{i,1} < a_{i,2} < .. < a_{i,n_i} < a_{i,n_i+1} = \infty$$

2. *A_i^* is a continuous function:*

$$\sigma_{i,j-1} + a_{i,j}\rho_{i,j-1} = \sigma_{i,j} + a_{i,j}\rho_{i,j}, \quad 2 \leq j \leq n_i$$

3. *A_i^* is strictly increasing in $[0, \infty)$:*

$$\rho_{i,j} > 0, \quad 1 \leq j \leq n_i$$

4. *Let $h_{i,j} = A_i^*(a_{i,j})$. Because A_i^* is strictly increasing we have:*

$$0 = h_{i,0} = h_{i,1} < h_{i,2} < .. < h_{i,n_i} < h_{i,n_i+1} = \infty \tag{3.4}$$

Observe that the above definition of A_i^* covers not only concave envelope functions (for example multiple leaky buckets, [61]), but also more general envelope functions (for example D-BIND, [63]). Figure 3.1 shows an example of a multiple-segment envelope.

In the analysis that follows we will use the notions of concave, convex and flexion points, that we define below.

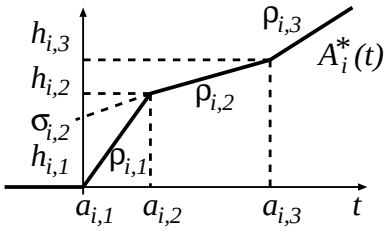


Figure 3.1. A multiple-segment envelope

Definition 3.2 *Given a continuous function $f : \mathbb{R} \to \mathbb{R}$ and $a \in \mathbb{R}$, $(a, f(a))$ is said to be a concave (convex) point of f if f is concave (convex) in a vicinity of a:*

$$\begin{aligned} &\exists \epsilon > 0 \quad \forall \alpha \in (0,1), \\ &f((1 \Leftrightarrow \alpha)(a \Leftrightarrow \epsilon) + \alpha(a + \epsilon)) > \\ &\qquad (1 \Leftrightarrow \alpha) f(a \Leftrightarrow \epsilon) + \alpha f(a + \epsilon) \end{aligned} \tag{3.5}$$

("$<$" respectively). A flexion point is a concave or a convex point.

Observe that a is an element of the domain of f, and thus a is the abscissa coordinate of the point $(a, f(a))$ on the graph of f. For example, in Figure 3.1, $a_{i,2}$ is a concave point and $a_{i,3}$ is a convex point of A_i^*.

Finally we introduce the following notation.

Notation 3.1 *Given a piecewise linear, continuous function f:*

1. *X_f^{cv} is the set of concave points of f and $Y_f^{cv} = f(X_f^{cv})$.*
2. *X_f^{cx} is the set of convex points of f and $Y_f^{cx} = f(X_f^{cx})$.*
3. *$X_f = X_f^{cv} \cup X_f^{cx}$ is the set of flexion points of f and $Y_f = f(X_f)$.*

Notation 3.2 *Given a function $f : \mathbb{R} \to \mathbb{R}$ and $S \subset \mathbb{R}$,*

1. *The inverse of the function f, f^{-1} is defined as*

$$f^{-1}(y) \stackrel{\Delta}{=} \{x | f(x) = y\}$$

If the resulting set has one element, we write $f^{-1}(y) = x$. The inverse of the function f in $y \in \mathbb{R}$, restricted to the interval (a, b) is

$$f^{-1}_{(a,b)}(y) \stackrel{\Delta}{=} \{x | x \in (a, b), f(x) = y\} \tag{3.6}$$

2. *The difference between a set $S \subset \mathbb{R}$ and $a \in \mathbb{R}$ is the set of differences*

$$S \Leftrightarrow a \stackrel{\Delta}{=} \{x \Leftrightarrow a | x \in S\}$$

3. *For any set S, $\widehat{S} = S \cup \{\Leftrightarrow\infty, \infty\}$.*

3.3.1 Exact Admission Control Algorithms for Multiple-Segment Envelopes

Let us consider N flows, where flow i is characterized by the multiple-segment envelope A^*_i and has a maximum packet delay requirement d_i. In order to compute the schedulability conditions (3.2), we introduce the following definition:

Definition 3.3 *The* (work) availability function $F : \mathbb{R} \Leftrightarrow\!\!\rightarrow \mathbb{R}$ *is defined by:*

$$F(t) = ct \Leftrightarrow \sum_{i=1}^{N} A^*_i(t \Leftrightarrow d_i) \tag{3.7}$$

Given this definition, the schedulability condition (3.2) for the set of N flows becomes $F(t) \geq 0 \quad \forall t \geq 0$. $F(t)$ gives the maximum amount of work (in bits) available over an interval of length t in the worst case at the EDF scheduler, while guaranteeing for each flow i (with envelope A^*_i) its maximum packet delay of d_i, for $1 \leq i \leq N$. This function plays a

central role in the development of admission control algorithms in the rest of this chapter. Given that A_i^* is a piecewise linear and continuous function, $i = 1, .., N$, it follows that F is also piecewise linear and continuous (Figure 3.2 gives an example of F).

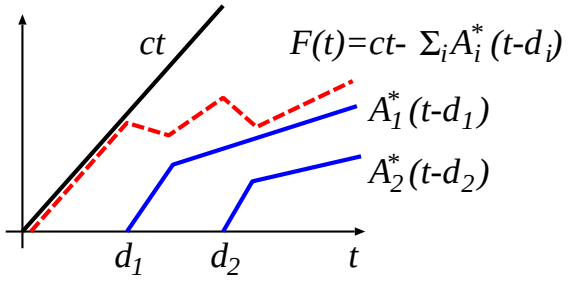


Figure 3.2. A work availability function F

We now consider the problem of admitting a new flow f, with envelope A_f^*, given that N flows (A_i^*, d_i) are scheduled at the EDF scheduler, and that the stability condition (3.1) is satisfied:

$$\sum_{i=1}^{N} \rho_{i,n_i} + \rho_{f,n_f} < c$$

In the following we adopt the convention that $\max \emptyset = 0$.

Theorem 3.2 *The minimum delay $\overline{d}$ guaranteeable to flow f is*

$$\overline{d} = \max\left(m_x, m_y\right)$$

where

$$m_x = \max_{u \in X_F} (u \Leftrightarrow A_f^{*-1}(F(u))) \tag{3.8}$$

$$m_y = \max \cup_{a \in X_{A_f^*}} (F^{-1}(A_f^*(a)) \Leftrightarrow a) \tag{3.9}$$

Observe that in (3.9) $F^{-1}(A_f^*(a))$ is a set that may contain more than one element since F is not necessarily a bijective function. Thus, m_y is the maximum element in a union of sets.

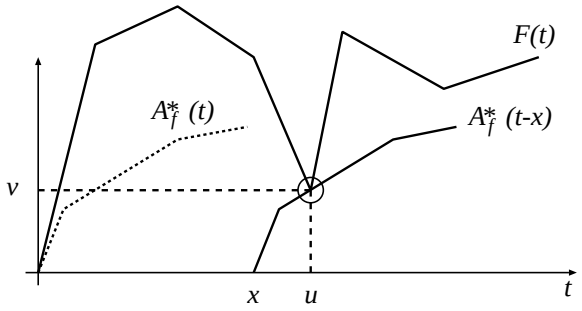


Figure 3.3. Constraint by flexion point of F

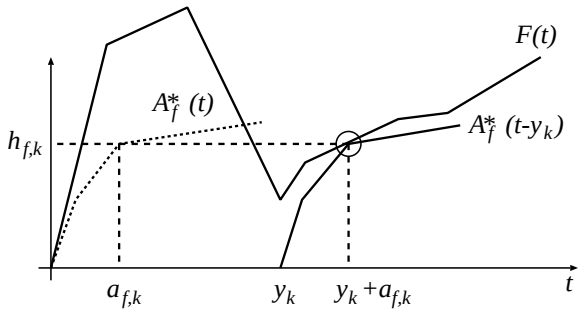


Figure 3.4. Constraint by flexion point of A_f^*

We provide an intuitive explanation of the theorem; see Appendix B.1 for its formal proof. First, note that given an availability function F, the schedulability condition $F(t) \geq A_f^*(t \Leftrightarrow d)$ is equivalent to being able to fit the A_f^* curve "below" $F(t)$. Figure 3.3 shows such an example, where the original envelope curve, $A_f^*(t)$, is translated x units to the right so that it just fits under $F(t)$. In this example, the minimum delay guaranteeable would be x. We note that this problem of fitting a multi-segment curve under another differs from polygon containment problems in computational geometry [83].

Our goal is to find the smallest value of d such that the original envelope curve, $A_f^*(t)$, translated d time units to the right, will lie completely below $F(t)$. Imagine for the moment (the algorithm we will present does not actually do this), taking the $A_f^*(t)$ curve and, starting from the right, translating the curve to the left (towards the origin) until it *first* intersects $F(t)$. $A_f^*(t)$ will either intersect $F(t)$ at a flexion point of $F(t)$ (as shown in Figure 3.3), or a line segment of $F(t)$ (as shown in Figure 3.4).

The equality for m_x in (3.8) considers the cases where flexion points of $F(t)$ just fit a translated $A_f^*(t)$. When $F(t)$ and a translated $A_f^*(t)$ are incident at such a flexion point, say at $t = u$ (as shown in Figure 3.3), then the height of the translated A_f^* at u is $F(u)$. The *untranslated* A_f^* has this height at ${A_f^*}^{-1}(F(u))$ and hence the amount of translation is $u \Leftrightarrow {A_f^*}^{-1}(F(u))$. One can show (see Appendix B.1) that if a translated A_f^* and $F(t)$ are incident at two flexion points of $F(t)$, with associated translations d_1 and d_2, where

$d_1 < d_2$, then the A_f^* that is translated d_1 lies above $F(t)$ for some values of t. Hence, in order to find the smallest delay that guarantees schedulability, we take a max in (3.8).

Similarly, suppose that $F(t)$ and a translated $A_f^*(t)$ intersect at a flexion point of $A_f^*(t)$ (see Figure 3.4). In this case, let a be such that the height of $F(t)$ at the incidence point is equal to the height of the untranslated $A_f^*(a)$. The amount of translation, and hence the guaranteeable delay would be $F^{-1}(A_f^*(a)) \Leftrightarrow a$.

In Figure 3.5 we give an algorithm to compute the minimum guaranteeable delay $\overline{d}$, which is a direct implementation of Theorem 3.2. Here we use the notation $h_{f,k} = A_f^*(a_{f,k})$, $v_l = F(u_l)$, $Y_{A_f^*} = A_f^*(X_{A_f^*})$, $Y_F = F(X_F)$ and $B = c \Leftrightarrow \sum_{i=1}^{N} \rho_{i,n_i}$.

MINIMUM_DELAY (Input: $X_{A_f^*}$, $Y_{A_f^*}$, ρ_{f,n_f}, X_F, Y_F, B;
Output: $\overline{d}_f$)

```
1  for each v_l ∈ Y_F
2      and each h_{f,k} ∈ Y_{A*_f}
3      if v_l ∈ [h_{f,k}, h_{f,k+1})
4          then x_l ← u_l ⇔ A*^{-1}_{f,[h_{f,k},h_{f,k+1})}(v_l)
5  m_x ← max x_l
6  for each h_{f,k} ∈ Y_{A*_f}
7      and each v_l ∈ Y_F
8      if h_{f,k} ∈ [v_l, v_{l+1})
9          then y_{l,k} ← F^{-1}(h_{f,k}) ⇔ a_{f,k}
10 m_y ← max y_{l,k}
11 d̄ ← max(m_x, m_y)
```

Figure 3.5. An $O(K^2N)$ algorithm for computing the minimum delay for a multi-segment envelope

To evaluate the computational complexity of this algorithm we first observe that, for any piecewise linear and continuous function g, if a_1 and a_2 are consecutive points in X_g, and $v \in [g(a_1), g(a_2))$, then computing $g^{-1}(v)$ requires $O(1)$ time, since g^{-1} is linear in $[g(a_1), g(a_2))$. (More precisely, $g^{-1}(v) = a_1 + (v \Leftrightarrow g(a_1))/g'(a_1^+)$, where $g'(a_1^+) = (g(a_2) \Leftrightarrow g(a_1))/(a_2 \Leftrightarrow a_1)$ for $a_2 < \infty$, and for $a_2 = \infty$, $g'(a_1^+) = \lim_{x\to\infty} g'(x)$, which is ρ_{f,n_f} for $g = A_f^*$ and B for $g = F$.) We assume that, at the time of the algorithm's invocation, there are N flows, and that any flow envelope has $|X_{A_f^*}| = |Y_{A_f^*}| = O(K)$

flexion points. It follows that $|X_F| = |Y_F| = O(KN)$. To compute m_x (steps 1-5), a lookup in $Y_{A_f^*}$ is done for each element in Y_F. Thus, the complexity of computing m_x is $O(|Y_F||Y_{A_f^*}|) = O(K^2N)$. To compute m_y (steps 6-10), a lookup Y_F is done for each element in $Y_{A_f^*}$, giving a complexity of $O(K^2N)$. The complexity of the entire algorithm is thus $O(K^2N)$.

We can speed up the computation using two independent methods which can be combined. We begin by observing that not all points in $X_{A_f^*}$ and X_F are relevant for computing the minimum delay. It is easy to show that only the concave points of F and convex points of A_f^* impose constraints on the position of A_f^* (see Figure 3.3 and 3.4). Thus,

$$\overline{d} = \max(m_x, m_y) \tag{3.10}$$

where

$$m_x = \max_{u \in X_F^{cv}} (u \Leftrightarrow {A_f^*}^{-1}(F(u))) \tag{3.11}$$

$$m_y = \max \cup_{a \in X_{A_f^*}^{cx}} (F^{-1}(A_f^*(a)) \Leftrightarrow a) \tag{3.12}$$

A second method that reduces the worst case complexity of computing the minimum guaranteeable delay, reduces the number of points for which F^{-1} is computed. We can accomplish this if, in (3.12), for each $a \in X_{A_f^*}^{cx}$, we compute only $F^{-1}(A_f^*(a)) \Leftrightarrow a$ within the concave interval of F where $m_x + a$ is situated (see Figure 3.6). In Appendix B.2 we show that this computation is sufficient for computing $\overline{d}$.

For a formal statement, given m_x defined in (3.11), we introduce $u_{a,1}, u_{a,2} \in \widehat{X^{cv}}$ (where $\widehat{X^{cv}} = X^{cv} \cup \{\Leftrightarrow\infty, \infty\}$) such that

$$u_{a,1} = \max\{x \in \widehat{X^{cv}} | x \leq m_x + a\} \tag{3.13}$$

$$u_{a,2} = \min\{x \in \widehat{X^{cv}} | x > m_x + a\} \tag{3.14}$$

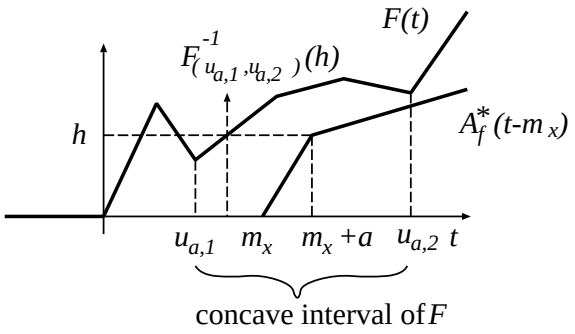


Figure 3.6. Reducing the number of F^{-1} computations

Theorem 3.3 *The minimum delay $\overline{d}$ guaranteeable to flow f is*

$$\overline{d} = \max(m_x, m_z) \tag{3.15}$$

where

$$m_x = \max_{u \in X_F^{cv}} (u \Leftrightarrow A_f^{*-1}(F(u))) \tag{3.16}$$

$$m_z = \max_{a \in X_{A_f^*}^{cx}} (F^{-1}_{(u_{a,1},u_{a,2})}(A_f^*(a)) \Leftrightarrow a)$$

where $F^{-1}_{()}()$ is defined in (3.6).

Observe that the set $F^{-1}_{(u_{a,1},u_{a,2})}(A_f^*(a)) \Leftrightarrow a)$ has at most one element since F is concave in $(u_{a,1}, u_{a,2})$.

The proof of the theorem can be found in Appendix B.2. In Appendix B.3 we present an algorithm for computing $\overline{d}$ based on the above theorem, having a computational complexity of $O(KN)$. Observe that this complexity is the best (lowest) possible since all of the segments of F must be evaluated for computing d, and there are $O(KN)$ such segments.

We observe here that, although a complexity of $O(KN)$ for admission control may be acceptable when the number of flows N is small, this amount of computation can be problematic when the number of flows reserved at a link is large (e.g., thousands of flows on an OC12 link). In the next section we explore a technique for further reducing the computation time for flow admission.

3.3.2 Discrete Admission Control for Multiple-Segment Envelopes

In this section, we show how the computational complexity of the admission control algorithm can be significantly reduced by ensuring that F consists of at most L linear segments. Let $\mathcal{D} = \{u_i | u_i \in \mathbb{R}^+, 1 \leq i \leq L\}$. We say that F is $\mathcal{D}$-*discrete* if the set of F's flexion points form a subset of $\mathcal{D}$, $X_F \subset \mathcal{D}$. Figure 3.7 shows an example of a $\mathcal{D}$-discrete work availability function F where $\mathcal{D} = \{u_1, u_2, u_3, u_4\}$. Consider a $\mathcal{D}$-discrete work availability function F and a new flow f with envelope A_f^* and maximum packet delay requirement d. Consider the problem that arises when we wish to admit f having envelope A_f^* and maximum packet delay requirement d. We define f's *minimum work requirement* function as

$$A_{f,d}(t) = A_f^*(t \Leftrightarrow d) \tag{3.17}$$

It is important that the admission result in a new work availability function F' that is also $\mathcal{D}$-discrete and that satisfies $F'(t) \leq F(t) \Leftrightarrow A_{f,d}(t)$. We construct F' as $F'(t) = F(t) \Leftrightarrow A_{f,d}^c(t)$, where $A_{f,d}^c(t) \geq A_{f,d}(t)$ and $A_{f,d}^c(t)$ is $\mathcal{D}$-discrete. Thus, F' has the required properties of being $\mathcal{D}$-discrete and $F'(t) \leq F(t) \Leftrightarrow A_{f,d}(t)$. $A_{f,d}^c$ is called a $\mathcal{D}$-discrete **cover** for the minimum work function $A_{f,d}$. An example of a $\mathcal{D}$-discrete cover for a work function $A_{f,d}$ is shown is Figure 3.8.

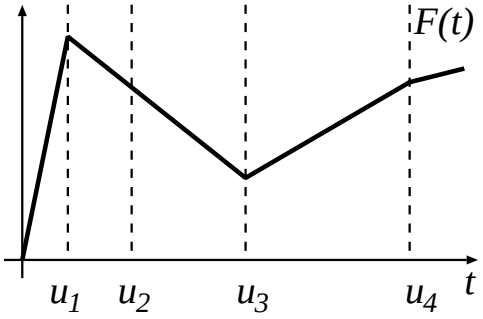


Figure 3.7. A discrete work availability function

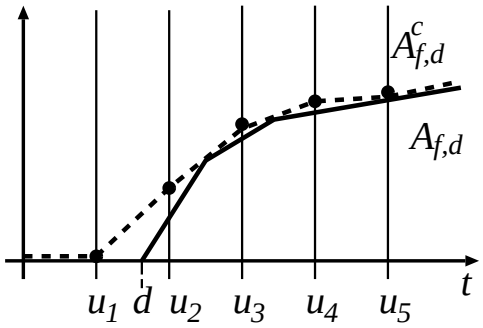


Figure 3.8. A discrete cover

We make two observations. First, it is clear that reserving a cover that is larger than the required work function implies that more resources (work) will be reserved than are

needed to accommodate the request. This will lead to lower resource utilization at the link scheduler and potentially fewer flows being admitted at the link, compared to what is possible with the exact admission control. We investigate this tradeoff in Section 3.5. The second observation is that, in general, there are many choices of discrete covers for a given work function. Among these choices, a cover that is "closer" to the original request is preferred because the amount of over-reservation is smaller. Unfortunately, it is not possible to establish a total order among the covers of a given envelope, and thus, there is no one cover that would minimize the over-reservation. We have developed and analyzed several heuristics to choose covers,

A class of covers that are interesting for resource reservation is defined below:

Definition 3.4 *Given a $\mathcal{D}$-discrete work availability function F and an envelope request A_{f,d_f}, a cover A^c_{f,d_f} is said to be F-feasible if:*

$$A^c_{f,d_f}(u) \leq F(u),\ \forall u \in X^{cv}$$
$$\lim_{t\to\infty} \frac{dA^c_{f,d_f}(t)}{dt} < \lim_{t\to\infty} \frac{dF(t)}{dt}$$

It follows that any $\mathcal{D}$-discrete, F-feasible cover A^c_{f,d_f} is schedulable with F:

$$A^c_{f,d_f}(t) \leq F(t),\ \forall t \geq 0$$
$$\lim_{t\to\infty} \frac{d}{dt}(F(t) \Leftrightarrow A^c_{f,d_f}(t)) > 0$$

We can now define a classification of cover choosing policies. We denote by $\mathbb{E}$ the set of envelope and cover functions

$$\mathbb{E} = \{f | f : \mathbb{R} \to \mathbb{R}^+\}$$

Definition 3.5 *Let $\mathcal{C}$ be a cover choosing policy (shortly, cover policy):*

$$\mathcal{C} : \mathbb{E} \to \mathbb{E}, \quad \mathcal{C}(A_{f,d_f}) = A^c_{f,d_f}$$

We say that $\mathcal{C}$ is F-respecting if, for any envelope A_{f,d_f} that is F-feasible, the cover $\mathcal{C}(A_{f,d_f})$ is also F-feasible.

In other words, an F-respecting policy admits any envelope that would be admitted by the exact admission control. The over-reservation due to discretization impacts only the subsequent flow admission requests. A non-F-respecting policy can reject some envelopes that would otherwise be admitted by the exact admission control. In the following sections we describe several F-respecting and non-F-respecting cover policies that we believe to provide efficient resource reservation. We compare their performance in Section 3.5.

3.3.2.1 Non-F-respecting Cover Policies

One way to construct a discrete cover for a work function is to shift each concave point to the largest discretization point smaller than itself. The remaining points of the cover are taken "on" the $A_{f,d}$ (i.e., in $A_{f,d}(\mathcal{D})$). The **horizontal translation** policy (Figure 3.9) constructs the cover $A^h_{f,d}$ by translating the concave points of the work function $A^*_{f,d}$ horizontally to the left. The **slope translation** policy (Figure 3.10) constructs the cover $A^s_{f,d}$ by translating each concave point of the work function $A^*_{f,d}$ to its left, following the slope of the envelope's segment that begins at that concave point of $A_{f,d}$. It is easy to show that $A^s_{f,d}(t) \leq A^h_{f,d}(t)$, i.e., the slope translation results in a smaller over-reservation than the horizontal translation. Henceforth we only consider the slope translation policy.

Observe that both policies may produce a cover that intersects F, even if the original envelope $A_{f,d}$ did not. Thus, the minimum delay guaranteeable by the discrete admission control using the above policies is generally larger than the delay guaranteeable by exact admission control.

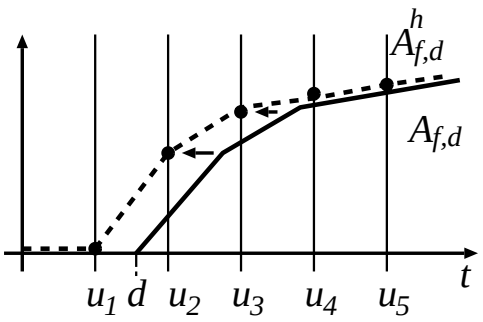


Figure 3.9. A horizontal translation cover

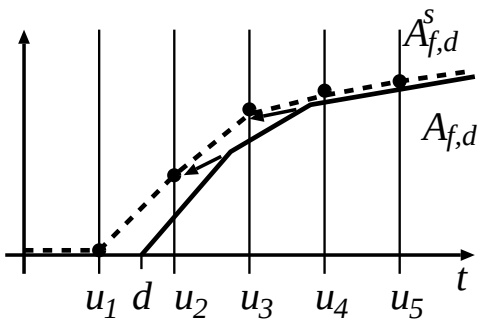


Figure 3.10. A slope translation cover

We begin by deriving a method for computing the minimum delay $\overline{d}^s$ guaranteeable by the slope translation policy, given that the stability condition (3.1) is satisfied. We introduce the following notation.

$$A^*_{f,a_{f,j}}(t) = \sigma_{f,j} + \rho_{f,j}t, \quad t \in \mathbb{R}, 1 \leq j \leq n_f \tag{3.18}$$

thus,

$$A^*_f(t) = A^*_{f,a_{f,j}}(t), \quad \text{for} \quad t \in [a_{f,j}, a_{f,j+1}) \tag{3.19}$$

Given an envelope A^*_f and a maximum delay request d, the slope translation cover envelope $A^s_{f,d}$ is a piecewise linear, continuous function defined by its flexion points:

$$A^s_{f,d}(u) = \max(A_{f,d}(u), A^+_{f,d}(u)), \quad u \in \mathcal{D} \tag{3.20}$$

where

$$\begin{aligned} &A^+_{f,d}(u_i) = \max\{0\} \cup \\ &\{A^*_{f,a}(u_i \Leftrightarrow d) | a \in X^{cx}_{A^*_f}, u_i \leq a + d < u_{i+1}\}, \quad u_i \in \mathcal{D} \end{aligned} \tag{3.21}$$

In Figure 3.11 we illustrate the construction of such a cover. Observe that the second set in (3.21) is empty when there is no $a \in X^{cx}_{A^*_f}$ such that $u_i \leq a + d < u_{i+1}$; in this case

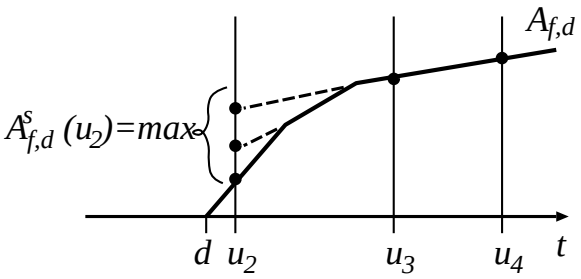


Figure 3.11. Construction of slope translated cover

$A^+_{f,d}(u_i) = 0$ and thus $A^s_{f,d}(u_i) = A^*_f(u_i \Leftrightarrow d)$. This corresponds to $A_{f,d}$ having no flexion points in the interval $[u_i, u_{i+1})$. This case is exemplified in Figure 3.11 for u_3.

Theorem 3.4 *The minimum delay $\overline{d}^s$ guaranteeable to flow f using the discrete admission control algorithm coupled with the slope translation cover policy is*

$$\overline{d}^s = \max(m_x, m_z), \tag{3.22}$$

where

$$m_x = \max_{u \in \mathcal{D}} \{u \Leftrightarrow {A^*_f}^{-1}(F(u))\} \tag{3.23}$$

$$m_z = \max_{a \in X^{cx}_{A^*_f}} \{\min(u_{i_a} \Leftrightarrow {A^*_{f,a}}^{-1}(F(u_{i_a})), u_{i_a+1} \Leftrightarrow a)\} \tag{3.24}$$

$$u_{i_a} \leq m_x + a < u_{i_a+1} \quad u_{i_a}, u_{i_a+1} \in \mathcal{D}^\infty \tag{3.25}$$

We provide some intuition behind the above result; the formal proof can be found in Appendix B.4. Given a $\mathcal{D}$-discrete availability function F, the schedulability condition $F(t) \geq A^s_{f,d}(t)$ for all $t \in \mathbb{R}^+$ reduces to $F(u) \geq A^s_{f,d}(u)$ for all $u \in \mathcal{D}$, since the slope translation cover $A^s_{f,d}$ is by construction $\mathcal{D}$-discrete. The minimum delay $\overline{d}^s$ guaranteeable

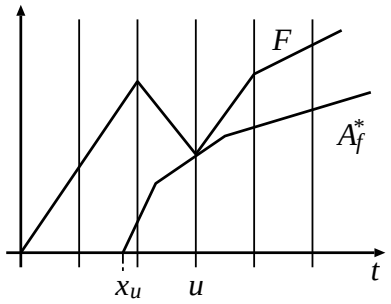


Figure 3.12. F constrains points "on" original envelope

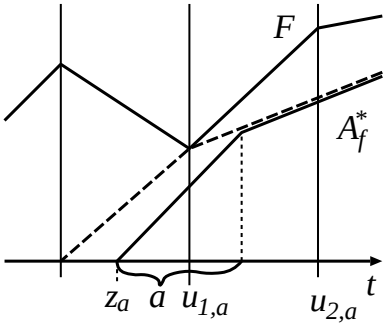


Figure 3.13. F constrains slope translating points

to f is the leftmost position of $A_{f,d}$, while its cover $A^s_{f,d}$ is below F. $A^s_{f,d}$ is defined as the maximum of two components, $A_{f,d}$ and $A^+_{f,d}$ which are both increasing functions. m_x, the leftmost position of $A_{f,d}$, accounts for all points "on" the original envelope A^*_f being constrained by F, as in Figure 3.12. m_z, the leftmost position of $A^+_{f,d}$, accounts for all slope translated points of A^*_f being constrained by F, as in Figure 3.13. $\overline{d}^s$ is the smallest delay (leftmost position) admissible for both components, which is the largest (rightmost) of the two individual positions, $\overline{d}^s = \max(m_x, m_z)$.

MINIMUM_DELAY_SLOPE_TR (Input: $X^{cx}_{A^*_f}$, $Y_{A^*_f}$, ρ_{f,n_f}, $\mathcal{D}$, $F(\mathcal{D})$, B; Output: $\overline{d}_f$)

1 **for** each $v \in F(\mathcal{D})$
2 find $h_{f,i} \in Y_{A^*_f}$, $h_{f,i} \le v < h_{f,i+1}$, $h_{f,i} = A^*_f(a_{f,i})$
3 $x_u \leftarrow u \Leftrightarrow {A^*_{f,a_{f,i}}}^{-1}(v)$
4 $m_x \leftarrow \max_u x_u$
5 **for** each $a \in X^{cx}_{A^*_f}$
6 find $u_j \in \mathcal{D}$, $u_j \le m_x + a < u_{j+1}$
7 $z_a \leftarrow \min\left(u_j \Leftrightarrow {A^*_{f,a}}^{-1}(F(u_j)), u_{j+1} \Leftrightarrow a\right)$
8 $m_z \leftarrow \max_a z_a$
9 $\overline{d}^s \leftarrow \max(m_x, m_z)$

Figure 3.14. An $O(L + K)$ algorithm for computing the minimum delay for the slope-translating policy

In Figure 3.14 we give an algorithm to compute the minimum guaranteeable delay $\overline{d}^s$, which is a direct implementation of Theorem 3.4. To evaluate the computational complexity of this algorithm we first observe that, the computation of m_x in lines 1-3 requires a lookup in $Y_{A_f^*}$ and $X_{A_f^*}$, which can be done in tandem if both sets are sorted. Thus, the complexity of computing m_x is $O(K+L)$ since $|Y_F| = |\mathcal{D}| = L$ and $|Y_{A_f^*}| = |X_{A_f^*}| = K$. A similar analysis applies to the computation of m_z in lines 4-7, where the sets $X^{cx}_{A_f^*}$ and $\mathcal{D}$ can be looked up in tandem if sorted, giving a complexity of $O(K_2+L)$ since $|X^{cx}_{A_f^*}| = K_2$. Thus, the total complexity of computing $\overline{d}_s$ is $O(K+L)$. This is an important improvement over the $O(KN)$ complexity of exact admission control algorithm when the number of flows N is large (thousands) and the number L of discretization points is small (tens). We compare the performace of the two algorithms through simulation in Section 3.5.

3.3.2.2 Admission Control Using F-respecting Cover Policies

The minimum delay guaranteeable by F-respecting cover policies is given by the following proposition.

Proposition 3.2 *For a given work availability function F and a given envelope A_f^*, the minimum delay guaranteeable to A_f^* by a discrete admission control using any F-respecting cover policy is equal to the minimum delay guaranteeable by the exact admission control.*

Proof. 1. Given F, A_f^*, and the minimum delay $\overline{d}_f$ guaranteeable by the exact admission control (as defined in Theorem 3.3), we show that $\overline{d}_f$ can be guaranteed to A_f^* for some $\mathcal{D}$-discrete cover reservation.

It is sufficient to show that there is a $\mathcal{D}$-discrete, F-feasible function A_f^c that is a cover for $A_{f,\overline{d}_f}$. Take the cover $A_f^c(u) = F(u)$, $\forall u \in \mathcal{D}$ and $\lim_{t\to\infty} \frac{dA_f^c(t)}{dt} = \lim_{t\to\infty} \frac{dF(t)}{dt} \Leftrightarrow \epsilon$, where ϵ is any value in $0 < \epsilon < \lim_{t\to\infty} \frac{dF(t)}{dt}$. Then A_f^c is $\mathcal{D}$-discrete and F-feasible.

2. Given F, A_f^*, and $\overline{d}_f$, it is clear that $\overline{d}_f$ is the minimum guaranteeable delay to A_f^* when using a $\mathcal{D}$-discrete cover reservation, since $\overline{d}_f$ is the minimum guaranteeable delay to A_f^* when using no cover. *Q.E.D.*

The consequence of Proposition 3.2 is that, given a work availability function F, the minimum guaranteeable delay for a new flow is not affected by discretization. We observe here that, although choosing the cover $A_f^c = F$ does not impact the minimum guaranteeable delay $\overline{d}_f$, using A_f^c will result in an over-reservation for flow f, that may make it impossible to admit any subsequent flows. In general, the choice of a cover envelope impacts the availability of resources (work) for subsequently arriving flow reservation requests. The problem of choosing a $\mathcal{D}$-discrete, F-feasible cover that features "little" over-reservation is the object of the next section.

The algorithms for computing minimum delay, resource reservation and release in the discrete framework are similar to the exact algorithms described in Section 3.3.1.

Specifically, MINIMUM_DELAY_F-RESPECTING (Figure 3.15) takes as input $\mathcal{D} = (u_l)_{1 \leq l \leq L}$, the availability function F given by its set of values $Y^{cx} = (w_l)_{1 \leq l \leq L}$ where $w_l = F(u_l)$ and B, and the envelope characterization $(\sigma_{f,k}, \rho_{f,k})_{1 \leq k \leq n_f}$ of the new flow f. It outputs the minimum delay guaranteeable to f. First (lines 1-2) it checks whether there is sufficient capacity to accept the new flow. If there is, it computes (lines 3-6) a lower bound m_x on $\overline{d}_f$ based on the constraints imposed by the local minima of F on the each segment of A_f^*. It then determines (line 7-8), the discretization interval (u_{l_k}, u_{l_k+1}) of F where A_f^*'s concave point $(m_x + a_{f,k}, h_{f,k})$ is situated. If the condition in line 9 is met in this interval, another lower bound on $\overline{d}_f$, z_k is computed (lines 10-11) based on the constraint imposed by the concave point of A_f^*. The final value of $\overline{d}_f$ is the maximum of all the computed lower bounds (lines 13-14). The $O(K + L)$ computational complexity of MINIMUM_DELAY_F-RESPECTING is a consequence of line 4, where the ordered lists Y^{cx} and $(h_{f,k})_{1 \leq k \leq n_f}$ are each looked up once.

The RESERVE algorithm first chooses a $\mathcal{D}$-discrete, F-feasible cover envelope A_f^c by choosing the set of values $\Delta w_{f,l} = A_f^c(u_l)$, $1 \leq l \leq L$ and $\Delta B_f = \lim_{t \to \infty} \frac{dA_f^c(t)}{dt}$. Next, these values are subtracted from the corresponding values $(w_l)_{1 \leq l \leq L}$ and B of F. RELEASE reclaims the reservation of A_f^c from F by performing the complement of operations in

RESERVE. RESERVE has complexity $O(K + L \log L)$ since the set Y^{cx} is sorted and we will see in Section 3.3.2.3 that CHOOSE_COVER has complexity $O(K + L)$. RELEASE has complexity $O(L \log L)$ as it updates and sorts the set Y^{cx} that has $O(L)$ elements.

As we observed earlier, a cover envelope is not uniquely determined. We examine some F-respecting policies in Section 3.3.2.3.

MINIMUM_DELAY_F-RESPECTING(Input: $\mathcal{D}$, Y^{cx}, B, $(\sigma_{f,k}, \rho_{f,k})_{1 \le k \le n_f}$;
Output: $\overline{d}_f$)

1 **if** $B \le \rho_{f,n_f}$
2 **then** $\overline{d}_f \leftarrow \infty$, **exit**
3 **for** $w_l \in Y^{cx}$ in increasing order **do**
4 find k, $1 \le k \le n_f$ such that $h_{f,k} \le w_l < h_{f,k+1}$
5 $x_l \leftarrow u_l \Leftrightarrow \frac{w_l - \sigma_{f,k}}{\rho_{f,k}}$
6 $m_x \leftarrow \max_{1 \le l \le L} (x_l)^+$
7 **for** $k = 2$ to n_f **do**
8 find l_k, $0 \le l_k \le L$ such that $u_{l_k} \le m_x + a_{f,k} < u_{l_k+1}$
9 **if** $w_{l_k} < h_{f,k} < w_{l_k+1}$
10 **then** $\alpha \leftarrow \begin{cases} B & \text{if } l_k = L \\ \frac{w_{l_k+1} - w_{l_k}}{u_{l_k+1} - u_{l_k}} & \text{if } l_k < L \end{cases}$
11 $z_k \leftarrow \Leftrightarrow a_{f,k} + u_{l_k} + (h_{f,k} \Leftrightarrow w_{l_k})/\alpha$
12 **else** $z_k \leftarrow 0$
13 $m_z \leftarrow \max_{2 \le k \le n_f} (z_k)^+$
14 $\overline{d}_f \leftarrow \max(m_x, m_z)$

Figure 3.15. An $O(L + K)$ algorithm for computing the minimum delay for a multiple-segment envelope

RESERVE(Input: $\mathcal{D}$, Y^{cx}, B, $((\sigma_{f,k}, \rho_{f,k})_{1 \le k \le n_f}, d_f)$;
Output: Y^{cx}, B)

1 CHOOSE_COVER($\mathcal{D}$, Y^{cx}, B, $((\sigma_{f,k}, \rho_{f,k})_{1 \le k \le n_f}, d_f)$;$(\Delta w_{f,l})_{1 \le l \le L}$, ΔB_f)
2 **for** $l = 1$ to L **do**
3 $w'_l \leftarrow w_l \Leftrightarrow \Delta w_{f,l}$
4 $B \leftarrow B \Leftrightarrow \Delta B_f$
5 $Y^{cx} \leftarrow$ ordered set $(w'_l)_{1 \le l \le L}$

Figure 3.16. An $O(K + L \log L)$ algorithm for reserving resources for a multiple-segment envelope

RELEASE(Input: $\mathcal{D}$, Y^{cx}, B, $(\Delta w_{f,l})_{1 \le l \le L}$, ΔB_f;
Output: Y^{cx}, B)

1 **for** $l = 1$ to L **do**
2 $\quad w'_l \leftarrow w_l + \Delta w_{f,l}$
3 $Y^{cx} \leftarrow$ ordered set $(w'_l)_{1 \le l \le L}$
4 $B \leftarrow B + \Delta B_f$

Figure 3.17. An $O(L \log L)$ algorithm for releasing resources reserved for a cover envelope

INIT_STATE(Input: c, $\mathcal{D}$;
Output: Y^{cx}, B)

1 $B \leftarrow c$
2 **for** $l = 1$ to L **do**
3 $\quad w_l \leftarrow cu_l$

Figure 3.18. State initialization at an empty EDF scheduler

3.3.2.3 F-respecting Cover Policies

We consider the problem of choosing a $\mathcal{D}$-discrete, F-feasible cover A_f^c for a given envelope A_f^*. We approach this problem by defining

$$\Delta w_{f,l} = A_f^c(u_l), \quad \text{for} \quad l = 1, .., L, \quad \Delta B_f = \lim_{t \to \infty} dA_f^c(t)/dt$$

and attempting to minimize $\Delta w_{f,l}$ for $l = 1, .., L$ and ΔB_f. We present several heuristics that attempt this. We begin by assuming that the concave points of A_f^* are situated in non-adjacent discretization intervals, an assumption that will be removed later. By examining the example in Figure 3.19, we see that all points of A_f^c can be taken "on" A_f^*:

$$A_f^c(u_l) = A_f^*(u_l \Leftrightarrow d_f)$$

except for the segments that fall in the same discretization interval with a concave point of A_f^*. For example in Figure 3.19 the cover coincides with the envelope in u_2 and u_5, but not in u_3 and u_4.

as is the case with segment $D \Leftrightarrow E$ in Figure 3.19.

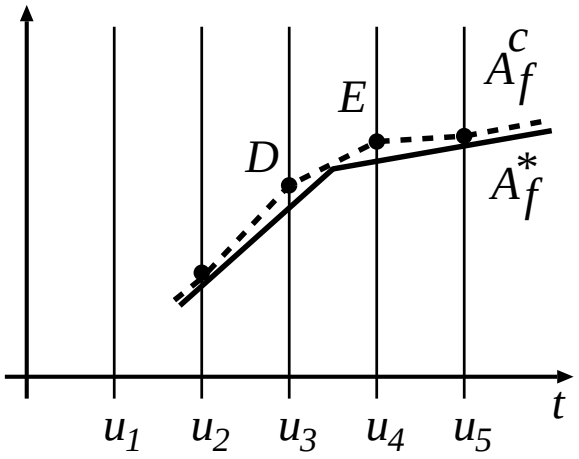


Figure 3.19. A cover A_f^c in the vicinity of a concave point of A_f^*

Observe here that this "concave point" segment is not uniquely determined. To be close to A_f^*, the segment should include A_f^*'s concave point $(d_f + a_{f,k}, h_{f,k})$. Let l_k, $1 \leq l_k \leq L$ such that $u_{l_k} \leq d_f + a_{f,k} \leq u_{l_k+1}$. The segment's slope must be in the range $[\rho_{f,k-1}, \rho_{f,k}]$ as in Figure 3.20, by w_{l_k} and w_{l_k+1} as in Figure 3.21, by w_L as in Figure 3.22, or by 0 as in Figure 3.23.

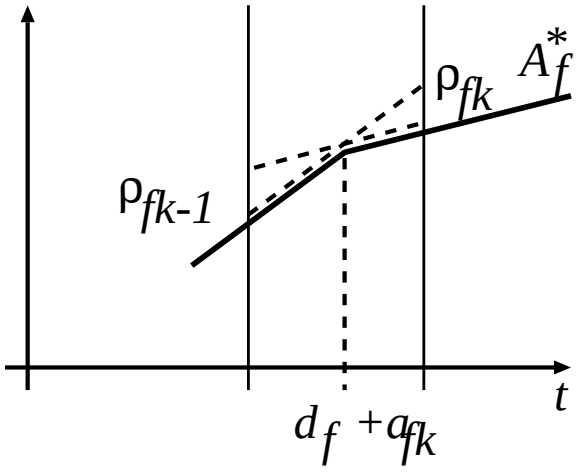


Figure 3.20. Slope of cover segment limited by $\rho_{f,k-1}$ and $\rho_{f,k}$

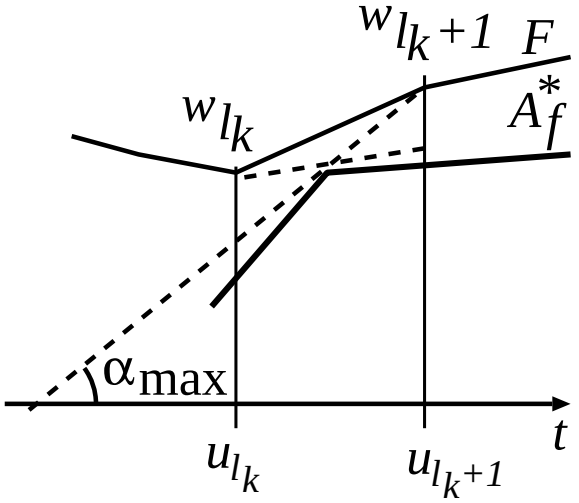


Figure 3.21. Slope of cover segment limited by w_{l_k} and w_{l_k+1}

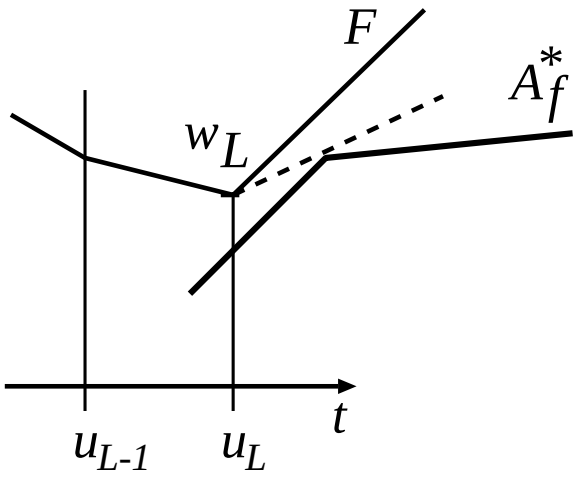


Figure 3.22. Slope of cover segment limited by w_L

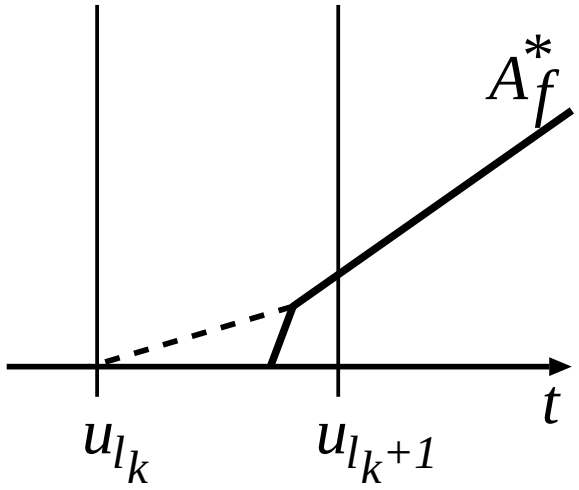


Figure 3.23. Slope of cover segment limited by 0

The first policy that we propose, named *proportional*, chooses a segment that is as close to the following "objective" segment, as the above constraints permit. The objective segment contains the concave point of A_f^* and the work reservations are proportional to the available work, i.e., Δw_{l_k} and Δw_{l_k+1} satisfy

$$\frac{\Delta w_{l_k}}{w_{l_k}} = \frac{\Delta w_{l_k+1}}{w_{l_k+1}}$$

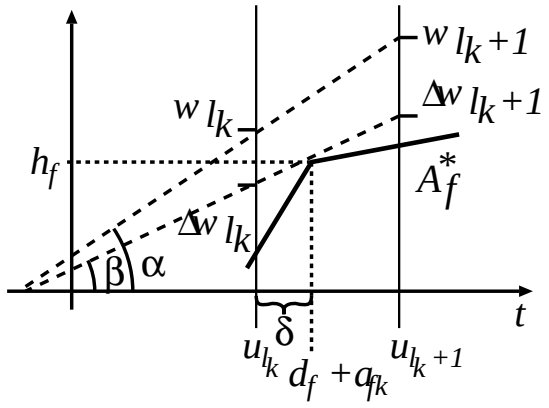


Figure 3.24. "Proportional" position of a cover segment

as in Figure 3.24. Observe that the segment satisfying this relation does not violate the bounds w_{l_k}, w_{l_k+1} for any $d_f \geq \overline{d}_f$.

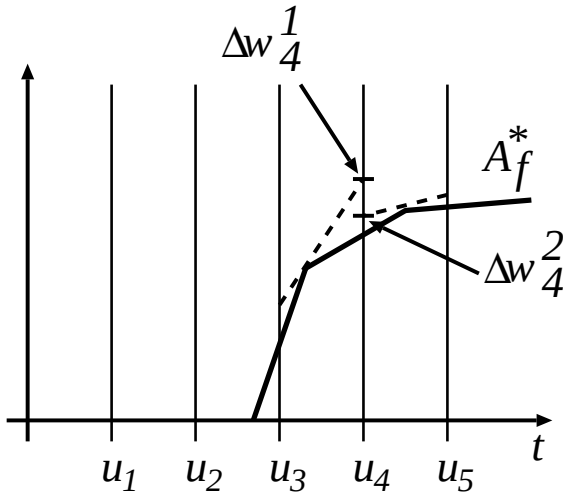


Figure 3.25. Non-contiguous cover

In the case where two concave points of A_f^* fall into adjacent discretization intervals, the cover segments chosen as above generally yield a non-contiguous cover envelope, as in Figure 3.25. In that case we have to determine what value $\Delta w_{f,i}$, $1 \leq i \leq L$ to reserve. We take the approach of reserving the maximum of the individual requirements, $\Delta w_{f,i} = \max_k \Delta w_i^k$. Since the proportional policy chooses the Δw_i^k values such that $\Delta w_i^k \leq w_i$

for any $d_f \geq \overline{d}_f$, it follows that $\Delta w_{f,i} \leq w_i$, i.e., CHOOSE_COVER never violates the w_i constraints (i.e., yields an F-feasible cover) for any $d_f \geq \overline{d}_f$.

The algorithm CHOOSE_COVER_PROP in Figure 3.27 chooses an F-feasible cover $((\Delta w_{f,l})_{1 \leq l \leq L}, \Delta B_f)$ for the envelope A_{f,d_f}. In lines 1-4, the cover envelope is initialized to be identical to $A^*_f(t \Leftrightarrow d_f)$ for all points in $\mathcal{D}$. Then, the algorithm computes for each concave point of A^*_f (lines 5-15) a cover segment, and the cover is constructed (lines 15-20) as the maximum of all these cover segments and the initial, on-envelope cover points.

To compute the cover segment for the concave point k, the algorithm first determines the value of reservation at the left of the discretization interval, ΔW^{left}, i.e., in u_{l_k}. First, it determines (lines 7-9) the slope of F's segment, α, and the maximum slope of the cover segment, α_{max}, given the limitation $\Delta w_{l_k+1} \leq w_{l_k+1}$ or ΔB. The slope β of the objective segment is computed in line 11. Line 12 computes the value of work reservation ΔW^{free} in u_{l_k} of the objective (unrestricted segment). The reservation ΔW^{left} in u_{l_k} of the cover segment (line 15) is given by the unrestricted value ΔW^{free} upper limited by Δw_{l_k} and the slope $\rho_{f,k}$, and lower limited by the maximum slope α_{max} and the slope $\rho_{f,k-1}$ (line 13). Lines 17-20 compute the reservation Δw_{l_k+1} in u_{l_k+1} or ΔB_f based on the reservation $\Delta w_{f,l_k}$ in u_{l_k} and with the condition that the segment includes the envelope's concave point k.

We can easily see that the algorithm CHOOSE_COVER_PROP has a computational complexity that is $O(L + K)$ since, in the lines 3 and 6 the sets $\widehat{X^{cv}}$ and $(d_f + a_{f,k})_{0 \leq k \leq n_f}$ are each looked up once.

A variation of the proportional policy is obtained by biasing the slope of the objective segment with the S parameter in line 11. As extreme cases, "min. slope" and "max. slope" have values 0 and ∞ for the S parameter respectively, thus forcing the slope of the objective segment to be as small or as high as possible, while remaining in the described limits and while also including the concave point.

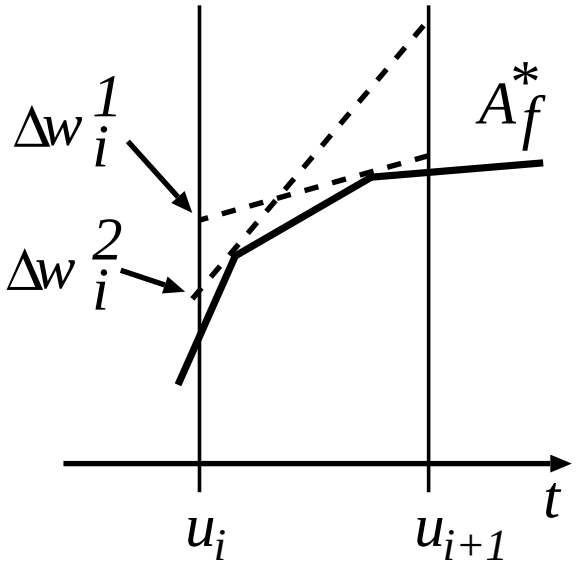


Figure 3.26. Two concave points in one discretization interval

Last, it is possible that two or more concave points of A_f^* be situated in the same discretization interval (see Figure 3.26). We observe that a segment covering one concave point, covers A_f^* on the entire discretization interval. In our heuristic implemented in the algorithm CHOOSE_COVER_PROP, the cover segment computed for the rightmost concave point in the interval is chosen to cover the interval because it has the lowest rate (slope) to reserve on this interval. This is achieved by running the loop starting at line 5 from the rightmost to the leftmost concave point, in conjunction with taking the maximum of $\Delta w_{f,l_k}$ in line 16.

CHOOSE_COVER_PROP(Input: S, $\mathcal{D}$, Y^{cx}, B, $((\sigma_{f,k}, \rho_{f,k})_{1 \le k \le n_f}, d_f)$;
Output: $(\Delta w_{f,l})_{1 \le l \le L}$, ΔB_f)

$\Delta B_f \leftarrow \rho_{f,n_f}$
for $l = 1$ to L **do**
    find k, $0 \le k \le n_f$, such that $d_f + a_{f,k} \le u_l < d_f + a_{f,k+1}$
    $\Delta w_{f,l} \leftarrow \sigma_{f,k} + \rho_{f,k}(u_l \Leftrightarrow d_f)$
for $k = n_f$ downto 2 **do**
    find l_k, $0 \le l_k \le L$ such that $u_{l_k} \le d_f + a_{f,k} < u_{l_k+1}$
    if $l_k = L$
      then $\alpha \leftarrow B$, $\alpha_{max} \leftarrow B$
      else $\alpha \leftarrow \frac{w_{l_k+1} - w_{l_k}}{u_{l_k+1} - u_{l_k}}$, $\alpha_{max} \leftarrow \frac{w_{l_k+1} - h_{f,k}}{u_{l_k+1} - (d_f + a_{f,k})}$
    $\delta \leftarrow d_f + a_{f,k} \Leftrightarrow u_{l_k}$
    $\beta \leftarrow S \frac{h_{f,k}}{\delta + w_{l_k}/\alpha}$
    $\Delta W^{free} \leftarrow h_{f,k} \Leftrightarrow \delta\beta$
    $\Delta W^{min} \leftarrow \max(h_{f,k} \Leftrightarrow \delta\alpha_{max}, \sigma_{f,k-1} + \rho_{f,k-1}(u_{l_k} \Leftrightarrow d_f))$
    $\Delta W^{max} \leftarrow \min(w_{l_k}, (\sigma_{f,k} + \rho_{f,k}(u_{l_k} \Leftrightarrow d_f))^+)$
    $\Delta W^{left} \leftarrow \min(\Delta W^{max}, \max(\Delta W^{min}, \Delta W^{free}))$
    $\Delta w_{f,l_k} \leftarrow \max(\Delta w_{f,l_k}, \Delta W^{left})$
    $\gamma \leftarrow \frac{h_{f,k} - \Delta w_{f,l_k}}{\delta}$
    if $(l_k = L)$
      then $\Delta B_f \leftarrow \max(\Delta B_f, \gamma)$
      else $\Delta w_{f,l_k+1} \leftarrow \max(\Delta w_{f,l_k+1}, \Delta w_{f,l_k} + (u_{l_k+1} \Leftrightarrow u_{l_k})\gamma)$

Figure 3.27. An $O(L + K)$ algorithm for choosing a feasible cover for multiple-segment envelopes

3.4 Admission Control at a Non-preemptive EDF Scheduler

So far we have studied admission control procedures for EDF schedulers assuming that the scheduling discipline is preemptive or that the packet transmission time is negligible. While preemptive schedulers are usually not of practical applicability, the above assumption is valid for scheduling ATM cells, since the cell transmission time (on the order of microseconds) is much smaller than practical delay requirements (on the order of miliseconds or greater).

In this section we remove this assumption and develop admission control algorithms for non-preemptive EDF (NPEDF) schedulers. This is important in the case that packet transmission times are non-negligible, as might be the case for IP packets. We present a simple correction to the results proposed in previous sections for preemptive EDF (PEDF) that accounts for non-zero packet transmission times.

There are two conditions required for a set of envelopes $(A_i^*, d_i)_{1 \leq i \leq N}$ to be schedulable at an NPEDF scheduler with capacity c, [66]. The stability condition

$$\lim_{t \to \infty} \sum_{i=1}^{N} A_i^*(t)/ct < 1$$

is identical to the stability condition of the corresponding PEDF scheduler. The schedulability condition includes a term accounting for the packet sizes:

$$ct \geq \sum_{i=1}^{N} A_i^*(t \Leftrightarrow d_i) + (\max_{\substack{1 \leq i \leq N \\ d_i > t}} p_i)\mathbf{1}(d_1 \leq t < d_N), \quad \forall t \geq 0 \tag{3.26}$$

where $d_1 \leq d_2 \leq ... \leq d_N$ are the delay requirements, p_i is the maximum packet size of flow i and $\mathbf{1}(P)$ is defined as:

$$\mathbf{1}(P) = \begin{cases} 1 & \text{if predicate } P \text{ is true} \\ 0 & \text{otherwise} \end{cases}$$

In Appendix B.5 we show that Proposition 3.1 no longer holds in the context of the schedulability condition (3.26), i.e., it is possible to guarantee a delay d but not d', even if $d < d'$. Consequently (3.26) cannot be used for admission control in the context of the flow setup protocols described in Section 3.2. In the following we propose a sufficient scheduling condition for NPEDF that eliminates the above-mentioned problem.

Let us assume that the NPEDF scheduler has an upper bound on the maximum packet size, p_{max}, such that $p_i \leq p_{max}$ for all i. This assumption is consistent with all current IP router architectures. For ATM switches, all flows have the same packet size, $p_i = p_{max} = 1$ cell. Also, this assumption has been used in other studies (see e.g., the study of Packetized Generalized Processor Sharing in [80]).

Our proposed schedulability condition is:

$$ct \geq \sum_{i=1}^{N} A_i^*(t \Leftrightarrow d_i) + p_{max}, \quad \forall t \geq \min_{1 \leq i \leq N}(d_i) \tag{3.27}$$

It is easy to see that (3.27)⇒(3.26). We further transform (3.27):

$$c(t \Leftrightarrow \frac{p_{max}}{c}) \geq \sum_{i=1}^{N} A_i^*(t \Leftrightarrow d_i), \quad \forall t \geq \min_{1 \leq i \leq N}(d_i)$$

$$ct \geq \sum_{i=1}^{N} A_i^*(t \Leftrightarrow d_i \Leftrightarrow \frac{p_{max}}{c}), \quad \forall t \geq 0$$

The last inequality is in a form that permits us to extend the admission control algorithms developed for PEDF schedulers to NPEDF schedulers.

In order to compute $\overline{d}_f^{NP}$, the minimum delay for A_f^* to be schedulable at a NPEDF, we first use the algorithm developed in the previous sections to determine a minimum delay $\overline{d}_f$, and then $\overline{d}_f^{NP} = \overline{d}_f + p_{max}/c$. Conversely, to reserve resources for (A_f^*, d_f^{NP}) at an NPEDF scheduler, we use the algorithms for reserving resources for $(A_f^*, d_f^{NP} \Leftrightarrow p_{max}/c)$ as if the EDF scheduler were preemptive.

3.5 Evaluation of Admission Control Algorithms Through Simulations

We are interested in answering two questions regarding the efficacy of the discrete admission control algorithms developed in Section 3.3.2. First, we are interested in empirically assessing the computational gains obtained by the discrete admission control over the exact algorithm. This is done by comparing the running times of an implementation of the exact and discrete admission control. Second, we wish to determine the performance degradation of the discrete admission control. This is done by comparing the link blocking probability yielded by the implementations of the exact and discrete algorithms.

We consider a link that forwards ATM traffic according to the EDF scheduling policy. The traffic characteristics of the flows to be serviced at this link are chosen randomly from the set of flow characterizations displayed in Table 3.1. Each row represents a four-segment characterization (σ_i, ρ_i) of a movie trace, where ρ_1 is the peak rate and ρ_4 is the mean rate. These characterizations have been derived as four-segment covers of the empirical envelopes of traces of MPEG-1 coded movies from [79, 78]. To account for possible variations in bandwidths associated with different encodings (MPEG-1, H.261, H.263, RealVideo, and Vxtreme), the (σ, ρ) characterizations from Table 3.1 are scaled with a random parameter 10^{θ}, where θ is uniformly distributed in $[\Leftrightarrow 2, 0]$.

Table 3.1. Four-segment characterization for six MPEG-coded movie traces

Movie	σ_1	ρ_1	σ_2	ρ_2	σ_3	ρ_3	σ_4	ρ_4
Advertisements	0	1600.0	800.0	800.0	1333.0	600.0	1600.0	533.0
Jurassic	0	4000.0	133.3	1054.0	400.0	853.3	1066.0	761.9
Mtv	0	6000.0	266.6	2356.5	933.3	1973.3	1866.6	1866.6
Silence	0	4000.0	266.6	666.5	533.0	600.0	1133.0	500.0
Soccer	0	5000.0	266.6	2500.0	1000.0	1238.0	2133.3	1066.6
Terminator	0	3400.0	133.3	787.8	266.6	586.6	800.0	366.6

Flow arrivals are generated according to a Poisson process with parameter α and their durations are exponentially distributed with mean $1/\beta$. The ratio α/β characterizes the

load offered to the link, i.e., the average number of flows that would exist at any time at a link with no capacity limitation. Each flow has a delay requirement d which is uniformly distributed in $[50ms, 3s]$. After a flow is generated with the above parameters, its EDF schedulability is verified by our admission control algorithms. We generate 100,000 flows in each simulation run, and we are interested in the link blocking probability, i.e., the ratio between the number of rejected flows and the total number of generated flows. We take the link blocking probability for an admission control algorithm as an indication of its performance. In our simulations, we use the method of independent replications to generate 90% confidence intervals for the link blocking probability.

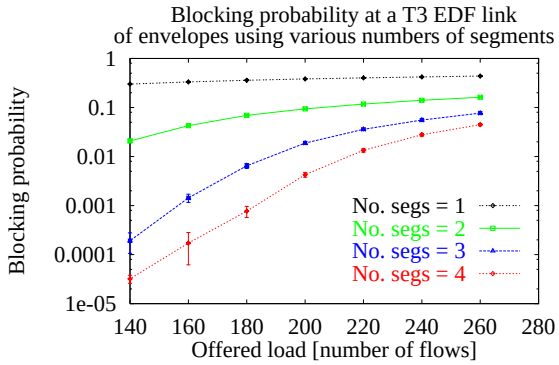


Figure 3.28. Impact of number of envelope segments on admission control performance

In the first experiment (Figure 3.28), we evaluate the efficiency of using multiple-segment approximation of data flows. The link capacity is $45Mb/s$ (a T3 link) and the same sequence of flows are offered to an EDF exact admission control algorithm, where the flow characterizations are take from Table 3.1, with 1, 2, 3, and all four (σ, ρ) pairs respectively. From the results in Figure 3.28 we see that multiple-segment flow characterization gives improved link blocking probability with respect to one- or two-segment characterizations, by orders of magnitude. This confirms the motivation of our present work that it is important to provide admission control algorithms for flows characterized by multiple-segment envelopes. In the remainder of our simulation experiments the flows are characterized by all four segments of Table 3.1.

Table 3.2. Comparison of computation times (in μs) for exact and discrete admission control algorithms

		Exact	Discrete
MINIMUM_DELAY	T3	77.63	13.39
	OC3	283.71	14.99
	OC12	1267.60	14.79
RESERVE	T3	129.85	7.03
	OC3	490.21	6.71
	OC12	2357.54	6.96
RELEASE	T3	84.06	7.30
	OC3	335.63	7.27
	OC12	1727.95	7.71

In the next experiment we compare the computational performance of discrete admission control algorithms (having 15 discretization points) with the exact algorithm when both operate in the same environment. Both algorithms input the same series of flows under three scenarios: link capacity $45Mb/s$ (a T3 link) and offered load 120 flows; link capacity $155.52Mb/s$ (an OC3 link) and offered load 414 flows; link capacity $622.08Mb/s$ (an OC12 link) and offered load 1658 flows. The offered loads have been chosen to incur the same blocking probability (0.05) in all three scenarios. Given this low rejection probability, the average number of flows N reserved at the link at any time is approximately equal to the offered load. We measure the computation time of the following algorithms: MINIMUM_DELAY (Figure B.1), RESERVE (Figure B.2) and RELEASE for exact admission control, and MINIMUM_DELAY_SLOPE_TR (Figure 3.14), RESERVE and RELEASE for discrete admission control using slope translation. The average computation time has been measured with the GNU code profiler $gprof$ on a 266MHz DECAlpha system.

First, observe that the average run times of the exact algorithms in Table 3.2 increase as a linear function of the average number of reserved flows N, which is consistent with the $O(KN)$ complexity found in Section 3.3.1. Second, observe that the run times of the discrete algorithms are independent of the number of reserved flows N, which was predicted by the $O(K + L)$ complexity found in Section 3.3.2. Also, observe that the run times on

all “OC12” lines in the table show a gain of about two orders of magnitude in computation time for the discrete admission control. Most important, the discrete algorithms have run times around $10\mu s$/call, which makes them a practical solution for flow admission control.

For the rest of our simulations we consider a T3 link ($45Mb/s$).

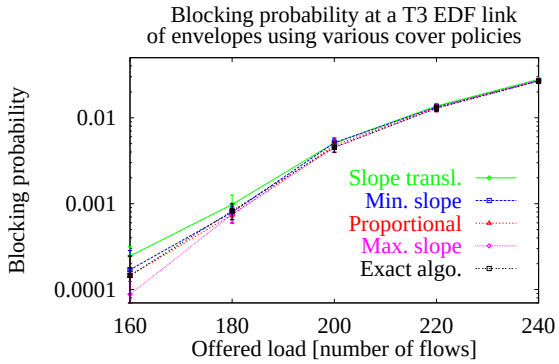


Figure 3.29. Impact of cover policies on admission control performance

We next compare the various cover policies proposed in Sections 3.3.2.3 and 3.3.2.1. The same sequence of flows is presented to exact and discrete admission control algorithms using slope translation, min slope, proportional, and max slope cover policies. From Figure 3.29 it follows that all cover policies perform almost identically and very close to the performance of the exact algorithm. Due to its simplicity, in the rest of our simulations we use the slope translation policy.

In the following we evaluate the penalty in link performance when using discrete admission control coupled with the slope translation policy. Recall that the discrete algorithms in Section 3.3.2 take their discretization point values from a finite set $\mathcal{D} = \{u_i | 1 \leq i \leq L\}$. A large spacing between discretization points implies a significant over-reservation for a flow, that would translate in fewer flows being admitted (higher blocking probability). A small spacing between discretization points, on the other hand, results in a large number of points and consequently a higher overhead for the admission control algorithms. In the following we address two questions. First, for a fixed number of discretization points, what is a good policy for choosing the spacing between points? Second, given that we have found a good

spacing policy, what number of points is sufficient for good link performance, yet small enough for low computational overhead?

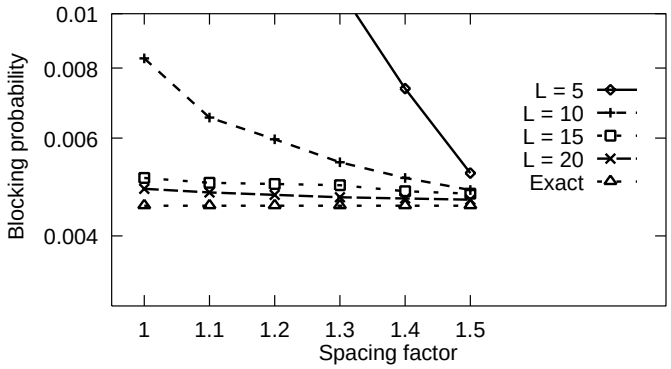


Figure 3.30. Impact of spacing of discretization points on admission control performance

One possibility for spacing of discretization points is equal (linear) spacing:

$$u_2 \Leftrightarrow u_1 = u_3 \Leftrightarrow u_2 = .. = u_L \Leftrightarrow u_{L-1}$$

Another possibility is to have the points geometrically spaced:

$$\frac{u_3 \Leftrightarrow u_2}{u_2 \Leftrightarrow u_1} = \frac{u_4 \Leftrightarrow u_3}{u_3 \Leftrightarrow u_2} = .. = \frac{u_L \Leftrightarrow u_{L-1}}{u_{L-1} \Leftrightarrow u_{L-2}} = \mathcal{S}$$

where $\mathcal{S}$ is a spacing factor. This latter spacing policy is expected to result in a smaller over-reservation for a small distance between discretization points compared to the linear policy, due to a smaller space the request falls in. In Figure 3.30 we plot the results of our simulations for spacing factors $\mathcal{S} = 1, 1.1, .., 1.5$, where a value of 1 corresponds to linear spacing. All of the half-width of the 90% confidence intervals are within 5% of the point value. The graph "Exact" corresponds to the exact admission control algorithm, which forms the base case for our comparison. First, we note that with less than 10 points, the blocking probability is unacceptably high compared to the base case. If 15 or more

discretization points are used, then the linear spacing policy provides a link performance close to that given by the geometric spacing policy, regardless of spacing factor. For this scenario, linear spacing is the solution of choice due to its simplicity and near optimal performance.

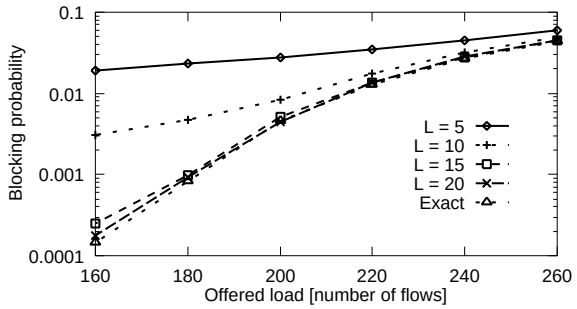


Figure 3.31. Impact of number of discretization points on admission control performance

In Figure 3.31 we plot the results of simulation experiments with algorithms using linear spacing and different number of discretization points as a function of offered load. All of the half-width of the 90% confidence intervals are within 5% of the point value. We observe that the blocking probability achieved by the discrete algorithm with 15 points is indeed quite close to the one achieved by the exact algorithm.

3.6 Conclusion

In this chapter we have proposed practical solutions to the problem of admission control for real-time flows with delay guarantees at an EDF scheduler, as a part of end-to-end flow admission control in IP and ATM networks. We applied the admission control conditions put forward by [66] to flows characterized by multiple-segment envelopes. We developed a first set of algorithms having a computation complexity of $O(KN \log (KN))$, where N is the number of flows admitted in the EDF scheduler at the time of algorithm invocation and K is the number of segments per envelope. A second set of algorithms places the horizontal position of concave points of flow envelopes into a predefined set of values (discretization

points), thus reducing the computational complexity of admission control to $O(K + L)$, where L is the number of predefined discretization points. A set of simulation experiments showed that the improvement in execution time achieved by the discrete admission control is indeed significant (two orders of magnitude faster for the examples we consider) and that the algorithm's execution time is independent of the number of flows admitted. The computation time of our admission control algorithm has been around $10\mu s$ per flow on a 266MHz DECAlpha system. Moreover, we have seen that the link performance degradation of the discrete admission control relative to the exact admission control is small, while using a small number of discretization points (15). Taken together, these results suggest that the algorithms we have studied in this chapter form the basis for a practical and highly efficient solution to the problem of admission control of real-time flows at EDF schedulers.

CHAPTER 4

SUPPORT FOR RSVP-BASED SERVICES OVER ATM NETWORKS

4.1 Introduction

Unicast and multicast applications with Quality of Service (QoS) guarantees are expected to become increasingly important and to require an increasingly larger fraction of network resources. This trend is reflected in the introduction of a number of protocols for flow establishment and allocation of network resources. Some of the major ones are ST-II [98, 29] and RSVP [108, 14] for IP networks, and UNI signaling [7, 91, 92] for ATM networks. While these varied proposals reflect the high level of activity in this area, they also introduce additional issues beyond what we have covered in the preceding chapters. Specifically, it is now necessary to enable these different protocols to interoperate, i.e. to provide a homogeneous service in a heterogeneous environment. In this chapter we study the problem of resource reservation for IP flows in IP over ATM networks and provide parctical solutions to the problem of how to permit the RSVP protocol to interoperate with ATM signaling.

4.1.1 Problem Statement

We consider a heterogeneous environment in which IP networks are operated over ATM networks. We also consider network applications (data flows) that require Quality of Service (QoS) guarantees. The applications reside in end systems (computers connected to a network) using the Internet (IP) protocol stack. To provide QoS guarantees, the IP network performs resource reservation using RSVP [14] as the reservation setup protocol. In the underlying ATM network, the resource reservation is performed by ATM signaling. In this

chapter we consider the problem of resource reservation for IP flows in IP over ATM networks and provide parctical solutions to the problem of interoperability of RSVP protocol with ATM signaling.

Our starting point is the classical IP over ATM model [64] in which an ATM Address Resolution Protocol (ATMARP) server is used for address resolution within a Logical IP Subnetwork (LIS), while the inter-LIS traffic is routed through IP routers. For an application with QoS requirements, the classical IP over ATM architecture does allow for QoS support over the VCs between the routers. Alternatively, resource reservations can be performed at the ATM level, by establishing a direct ATM connection ("shortcut") through an ATM network. Compared with the classical IP over ATM model [64], this shortcut architecture benefits from a better usage of network resources, assuming that the ATM network is topologically richer than the embedded IP network. This benefit is further increased when layer 2 forwarding and QoS enforcement are done more efficiently in ATM than in IP. The first of these benefits is likely to be significant, whereas the second depends on the capabilities of routers and switches in the network. In Chapter 5 we evaluate the benefits of ATM shortcuts and their dependence on the IP/ATM network topology.

In this chapter we describe a method to establish such shortcuts over ATM networks, first for unicast and then for multicast flows. In order to enable end-to-end performance guarantees we study the interplay between establishing RSVP flows and setting up ATM Variable Bit Rate (VBR) Virtual Circuits (VC). We provide a mapping between RSVP flow characteristics and ATM call parameters for the Guaranteed and Controlled-Load Service specifications [94, 104], of the Internet Integrated Services framework.

The rest of this chapter is structured as follows. In Section 4.1.2 we present related work on protocols for IP over ATM. In Section 4.2 we present the main functions of some protocols that we will refer to in the rest of this chapter, such as: address resolution in IP/ATM networks and the operation of resource reservation protocols such as RSVP and ATM signaling. In Sections 4.3 and 4.4 we describe mechanisms that relate flow establishment in

IP and call establishment in ATM, for the unicast and multicast cases, respectively. In Section 4.5 we address the issue of translating flow/call specifications (traffic parameters and QoS requirements) and the service requirements of the RSVP protocol into corresponding ATM signalling and QoS negotiations. In particular, we propose simple mappings between the IP Integrated-Services service specifications [94] and ATM QoS parameters [92]. Section 4.6 identifies both the necessary modifications to the current RSVP protocol and the extensions and changes to the ATM signalling that are needed to better support RSVP. Section 4.7 summarizes the proposal put forth in this chapter, and points to a number of open issues.

4.1.2 Related Work

As mentioned before, the emergence of ATM as a key networking technology has triggered a large number of studies addressing the interactions between IP and ATM networks. Some comprehensive surveys of such works can be found in [22, 2].

The mapping of IP packets to ATM cells at the data link layer is relatively well understood, and a stable set of solutions has been established. For example, the specification of multi-protocol encapsulation over ATM Adaptation Layer 5 can be found in [50], although special encapsulation provisions for IP multicast routing over ATM might still be needed [43]. The issue of packet fragmentation is addressed in [6], where a large IP MTU size (9180 bytes) is proposed for efficient IP packet handling.

On the other hand, when it comes to mapping IP flows/routes to ATM connections at the network layer, a large number of alternatives are still being proposed and considered, and this is where the focus of much of the recent work on IP over ATM has occured. Many of the problems that have been addressed so far are related to route establishment and address resolution for both unicast and multicast communications. For example, in the simple case of “local” ATM networks (where all IP switches have IP addresses coinciding on their network part), a now well-established solution to the problem of IP routing and address

resolution is described in [64, 81] (classical IP/ATM). It defines the rules for establishing direct ATM VC connections between local IP nodes, and for communicating with remote IP nodes via intermediate routers, that are themselves connected via a chain of VCs. Here, the notion of "local vs remote" is based on the source and destination IP addresses and the subnetwork mask specification. Specifically, within each LIS there is a single ATMARP server which resolves ATM addresses for all nodes in that subnet.

This model, typically refered to as the classical model, was extended in [87], to allow the establishment of direct ATM VCs based on QoS requirements, rather than on the IP destination addresses. Another extension of the Classical model, labeled the Conventional model [77], attempts to eliminate the overhead of hop-by-hop IP processing during communication to a remote destination by enabling the routers to "splice" the VCs that are associated with the same IP flow directly at the ATM level, thus forming a "bypass pipe."

The general issue of a scalable address resolution mechanism for non-broadcast multi-access (NBMA) networks, e.g. ATM, has also been addressed in [70, 19] which describes the Next Hop Resolution Protocol (NHRP). The NHRP describes how queries for the ATM address associated with a given IP resource are to be propagated in an NBMA network, and how and when replies carrying a destination ATM address are to be returned. Similarly, support for IP/ATM multicast and broadcast routing is proposed in [5, 95], where a set of multicast group membership servers (MARS) provides multicast address resolution within the ATM network. Finally, the problem of IP inter-domain routing over ATM is considered in [88].

All of the works mentioned above rely on the model of an "overlay" IP network on top of the ATM network, with some exchange of control information between the two. A different model is developed in [18, 17, 82, 16], which assume an environment where IP routers and ATM switches interact on a peer-to-peer basis (Peer or Integrated model). In this integrated model, interactions between ATM and IP are greatly simplified. However, this imposes constraints that may not always be easily satisfied. For example, the Integrated

model requires that IP and ATM routing be consistent and, therefore, be based on similar metrics. Similarly, the Peer model requires that both IP routers and ATM switches use a common (interchangeable) address format. This clearly represents a desirable goal, but it is not clear how soon this can be realized. As a result, in this chapter we do not require that the assumptions of this integrated model be satisfied.

With respect to the establishment of real-time flows with QoS requirements, the primary focus of this chapter is to deal with the heterogeneity of the flow establishment protocols (ST-II and RSVP for IP, and ATM signaling for ATM). In early works [45, 49], mechanisms were proposed for interactions between ST-II and ATM signalling. [57] outlines a scheme by which the construction of an ATM "bypass-pipe" can be triggered by RSVP messages, while [56, 44] dealt with the issue of how to advertise IP flow identifiers across an ATM network and how to map them onto ATM call identifiers. More recently, [12] identified a set of problems related to the establishment of real-time IP flows across ATM networks. A work relevant to our work, is [72, 73], which puts forth a proposal on how to support best-effort and real-time unicast and multicast IP flows over an ATM network. There the best effort traffic is handled through regular IP forwarding, i.e. a router overlay network, whereas the real-time flows are established through a combination of RSVP messages and ATM signalling using ATM VCs across the ATM network. A different method for establishing shortcuts through an ATM network for multicast sessions, one which is based on changes to the routing protocol, is described in [86].

Recently, a significant effort has been under way in the Internet Engineering Task Force (IETF) to unify various emerging approaches to IP reservations in ATM networks. This work in progress is summarized in the following four IETF Drafts. [23] outlines the issues and framework related to providing IP Integrated Services with RSVP over ATM. [13] discusses a proposed mapping of the Integrated Services models to ATM. [10, 9] discuss implementation requirements and guidelines for RSVP interoperation with ATM signaling. The above set of works take the approach of proposing a solution for RSVP-ATM inter-

operation based on the current RSVP and ATM specifications. Due to the different and in some aspects conflicting design principles of RSVP and ATM signaling, this solution is not able to exploit network resources optimally. By contrast, in this chapter we propose minimal extensions to RSVP and ATM signaling, thus enabling a well-structured and more efficient interoperation of RSVP and ATM.

4.2 Resource Reservation in IP and ATM Networks

In this chapter we provide a brief overview on the exisiting protocols for IP/ATM communication and resource reservation in IP and ATM networks. We will refer to these protocols many times in the rest of this thesis.

4.2.1 Reservations in IP: RSVP

The "Resource Reservation Protocol" (RSVP) [14] is used to reserve resources in IP networks for the purpose of providing Quality of Service guarantees to unicast and multicast IP flows. The protocol conveys information necessary for computing and performing reservations, and triggers the establishment of reservations at network nodes (routers).

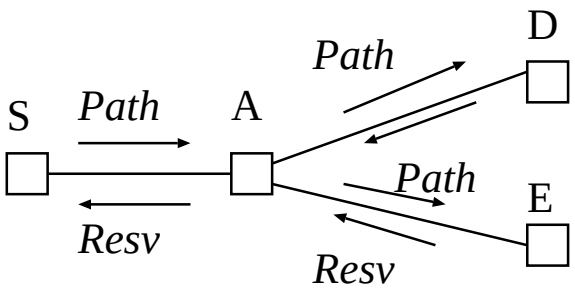


Figure 4.1. An example of RSVP operation

Consider the example in Figure 4.1. Source S sends periodic $Path$ messages to a multicast address. A $Path$ message contains, among other information, a data object named "Traffic Specification" or TSpec, which is a characterization of the QoS flow (for example

its average rate, maximum burst size). The $Path$ message follows the multicast route and a copy of it reaches each receiver (D and E). $Path$ is processed by the RSVP module at each node, where it accummulates information on link state (e.g., the minimum packet delay that the link can guarantee). Each receiver compares its QoS objective to the accummulated minimum end-to-end guarantee (for example compares its end-to-end delay requirement to the minimum end-to-end path delay). If the QoS objective is less stringent than the minimum guaranteeable QoS, the receiver sends a reservation message ($Resv$) to S. A $Resv$ message includes a data object named "Reservation Specification" or RSpec, which gives the amount of resources to be reserved (for example a rate). $Resv$ follows the $Path$'s route in the opposite direction and triggers resource reservation at each node it traverses. $Resv$ is also sent periodically by each receiver, and multiple $Resv$ messages can be merged into one $Resv$ message at each multicast branching point. The reservations are maintained in "soft" state at the nodes: if the periodic $Path$ and $Resv$ messages (also know as "refresh" messages) are no longer received, the information about the flow and its reservations are deleted after some period of time.

Flows with the same destination can share a reservation (this provides an efficient resource utilization for example in the case of teleconferences with one speaker at a time). The type of reservation sharing, along with the set of senders sharing the reservation, is specified as a "filter" or reservation style in the $Resv$ message.

4.2.2 Reservations in ATM: ATM Signaling

ATM signaling establishes reservations in the form of ATM Virtual Circuits (VCs). For establishing a unicast VC, the source S (see Figure 4.2) sends a Setup message to D including the flow's characteristics. At each node, resources are reserved and the receiver D returns a Connect message to S informing S that the flow was accepted. If resources are not available at any node, VC setup is terminated by a Release message returned to S.

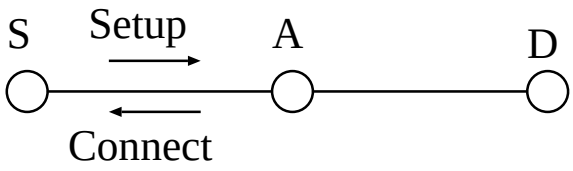


Figure 4.2. An example of unicast VC setup

To establish a multicast VC in ATM UNI 3.1 signaling, the source adds receivers to a multicast VC one by one. For example, in Figure 4.3 suppose that the communication $S \Leftrightarrow D$ is in place (established like a unicast VC) and S then requests the addition of a connection to E. S sends an "Add Party" message to E that is transformed into a Setup message at the branching point A. A reservation is done at A and E responds with a Connect message. This is transformed into an "Add Party Ack" message at A. Observe that each receiver is added separately by the source S.

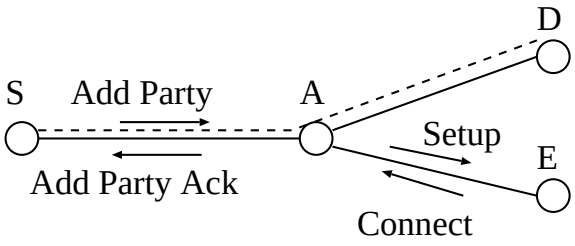


Figure 4.3. An example of multicast VC setup with ATM UNI 3.1 signaling

To avoid this potential overhead at S, in ATM UNI 4.0 signaling the Leaf Initiated Join (LIJ) was introduced to enable receivers to join a multicast VC without the source's participation. In this case (see Figure 4.4), the receiver E requests a join with "Leaf Setup Request" message. The branching point responds with a Setup message that reserves resources on this branch. E confirms the reservation by responding with a "Connect" message. The new VC branch can start transmitting after A responds with a "Connect Ack" message.

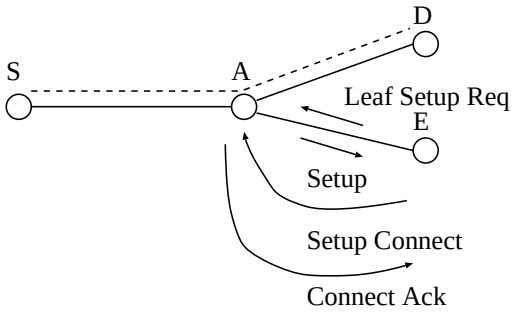


Figure 4.4. An example of multicast VC setup with ATM UNI 4.0 signaling

4.2.3 Protocols for IP/ATM Address Resolution

An IP/ATM network consists of an IP network spanning an ATM network (see Figure 4.5). The ATM network contains ATM switches connected by ATM links. The IP network contains IP routers connected by IP links. The IP routers are also ATM switches in the ATM domain. The ATM network is considered a "Non-Broadcast Medium Access" (NBMA) network from the point of view of IP network. When an IP flow has data to transfer between two IP nodes (routers) and the flow's route crosses the ATM network, a VC connection needs to be established. In Figure 4.5, a flow from S to D has route $S \Leftrightarrow A \Leftrightarrow B \Leftrightarrow D$ and, thus, a VC needs to be established between A and B. The VC establishment is initiated by A using ATM signaling (see Section 4.2.2), provided that A has the ATM address of B. The "NBMA Next Hop Resoultion Protocol" (NHRP) provides the service of ATM address resolution. Given the IP and ATM address of the requestor (A) and the IP address of the final destination (D), NHRP provides A with the IP and ATM addresses of the IP router/switch at the egress point of the ATM network (in our example B).

ATM address resolution is somewhat different when the communication is multicast (see Figure 4.6). Here, an IP multicast flow needs communication between a source S and a set of receivers (D and E). For communication over an ATM network, a multicast VC needs to be established. For this purpose, the "Multicast Address Resolution Server"

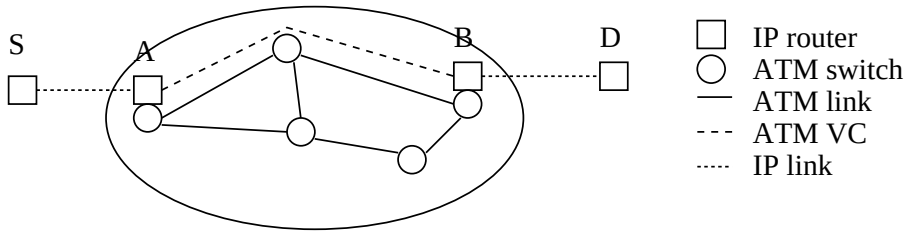


Figure 4.5. An example of NHRP operation

(MARS) protocol provides a mapping between an IP multicast address and a set of ATM addresses. Specifically, the IP router/switch A sends a multicast address resolution request (MARS_REQUEST) to a MARS server giving an IP multicast address. In its response, the MARS server provides a list of IP and ATM addresses (D, E) of the routers that are egress points at the ATM network for this multicast flow. These addresses are used by A to initiate a multicast ATM VC setup using ATM signaling (see Section 4.2.2).

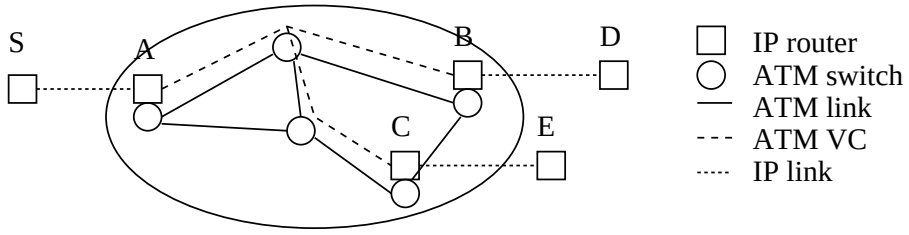


Figure 4.6. An example of MARS operation

4.3 Reservation Setup for Unicast Flows

This section focuses on RSVP-based reservation setup for *unicast* IP flows spanning ATM networks. It is assumed that the data flow traverses an ATM network, and that the source and the destination of the flow could be located on or off this network.

The schemes under consideration aim at setting up an ATM VBR VC (with QoS guarantees) through the ATM network for a given RSVP flow. The parameters necessary for

setting up these VCs are obtained via RSVP as described in Section 4.2. We describe four models of RSVP support over ATM networks.

- The "classical" RSVP support preserves IP routing but adds QoS connections (VCs) between the routers, and between the routers and the hosts (source and destination).
- The "sender-initiated ATM shortcut" and the "NHRP-based ATM shortcut" extend the classical RSVP support by enabling ATM shortcuts using the ingress router as the controlling entity for establishing shortcuts. The main difference between these shortcut methods is that the latter attempts to leverage and extend existing NHRP mechanisms to determine how ATM shortcuts should be established.
- The "receiver-initiated ATM shortcut" method exploits the duplex nature of ATM point-to-point VCs to allow the egress router to become the controlling entity for establishing shortcuts. The main benefit of this method is its synergy with the handling of multicast flows when support for Leaf Initiated Join (LIJ) becomes available from the ATM signaling, i.e. in UNI 4.0 (see Section 4.4.4 for details).

4.3.1 "Classical" RSVP Support

Figure 4.7 shows an ATM network consisting of four LISs. A is the ingress IP router to the ATM network, B is the egress router. RSVP messages follow the IP route $AEFGB$. Thus, a $Path$ message will travel downstream from A to B, while the corresponding $Resv$ message will travel upstream from B to A. When the $Resv$ message arrives at G the router sets up a VC from G to B (see Section 4.5 for details). Similarly, VCs will be set up from F to G, from E to F, and from A to E.

The above description concerns initial $Path$ and $Resv$ messages which trigger the reservation setup. $Path$ refresh messages are also forwarded along the IP route in order to track route changes. Observe that $Resv$ refreshes are not needed, since the RSVP 'soft' state has been replaced in the ATM environment by a 'hard' state. Therefore, non-refresh $Resv$ messages will be sent only if the QoS parameters of the flow change.

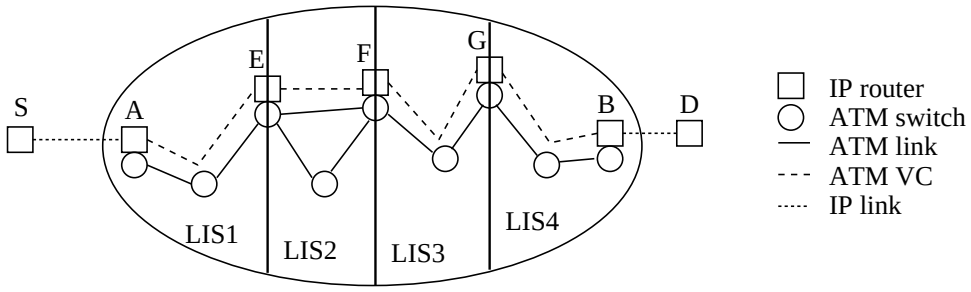


Figure 4.7. Reservation setup using "classical" RSVP support

If the ATM network consists of a single LIS then the route from A to B consists of one hop, although there could be multiple hops at the ATM level. This would also be the case if all hosts were served by a single Route Server in the Multiprotocol over ATM (MPOA) model [16].

For the multi-hop case, while RSVP messages travel over best-effort VCs, data packets flow over QoS VCs and enjoy QoS support in the routers. Traversing the routers, however, entails IP-level processing and thus is less desirable than a shortcut VC from A to B. In the rest of this section we discuss several schemes to avoid this overhead by using ATM shortcuts.

4.3.2 Sender-initiated ATM Shortcuts

In this scheme we modify RSVP operation in order to identify the appropriate egress router for the purpose of establishing a shortcut route through the ATM network (Figure 4.8). When the first $Path$ message for a session arrives at A, the node determines that the message will be forwarded over an ATM link and thus node A is the ingress node into the ATM network. The $Path$ message is routed along the overlay IP route, and is modified to carry both the ATM address and the IP address of A (the IP address of A is the 'previous hop' or PHOP). Each node along the route performs an ATM connectivity check to determine whether it is the egress point from the ATM network. This decision would be based on the ATM connectivity between the current router, the upstream router, and the down-

stream router. If the current router is not an egress router, it forwards the $Path$ message to the downstream router *without updating the PHOP address field*. This router does not create any $Path$ state for the session. If the current router is an egress router (e.g., B) it processes the $Path$ message in the default manner, creates $Path$ state for the session and stores, among other things, the IP address and the ATM address of A.

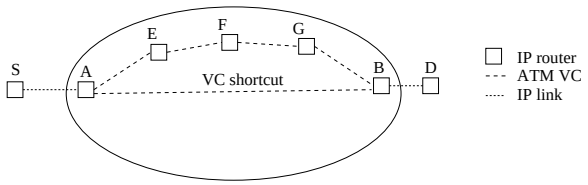


Figure 4.8. Reservation setup using ATM shortcuts

When a *new*[1] $Resv$ message arrives at B, B inserts its own ATM address as an object into this message, and forwards the message along the IP-routed path to A. Intermediate routers recognize the $Resv$ message but do not create any session or reservation and simply forward the message upstream. When this $Resv$ message arrives at A it carries in addition to the regular RSVP information, both the ATM address of the egress router B and QoS information necessary to determine the type of ATM VC that needs to be setup (see Section 4.5.3 for details). A will then initiate the VC setup.

After the shortcut VC from A to B is set up, the egress router B can suppress the transmission of *Resv* refreshes towards router A, unless they carry a modified service specification. This suppression of *Resv* refreshes reduces the overhead in traffic and processing of *Resv* messages. To achieve this, B needs to be able to associate the newly created VC with the RSVP flow. In order to accomplish this, the flow identifier consisting of the tuple `(source address, destination address, transport layer)` is carried in the SETUP message in the Broadband High Layer Information (B-HLI) ele-

[1]By new we mean both reservation requests for new flows and requests to modify the reservation of existing flows.

ment[2]. The source and destination addresses themselves further consist of pairs of the form `(IP address, port number)`. Note also that the receipt of the SETUP message provides an implicit acknowledgment that the *Resv* message was received at router A. This means that router A has received all of the information necessary to forward *Resv* messages upstream, i.e., the RSVP filter and service specifications that are not directly available from the ATM connection characteristics.

Figure 4.8 shows a shortcut VC from A to B which bypasses nodes E, F and G. The shortcut VC is used for the RSVP data traffic, but $Path$ messages continue to flow along the default routed path. Note that this scheme for creating shortcut routes is independent of the underlying routing mechanism and is oblivious to any IP routing domain boundaries. Moreover, RSVP state is required only at the edge routers A and B.

4.3.3 NHRP-based ATM Shortcuts

An alternate but functionally equivalent method to set up an ATM shortcut route is to instead, rely on an extension of NHRP. In this method the ingress router A creates an NHRP Query and stores the contents of the RSVP $Path$ message in the QoS object of the query. The NHRP query travels to the appropriate egress router B which then recreates the $Path$ message. B also creates an NHRP response and returns it to A. When a new $Resv$ message arrives at B it is forwarded to A, which then sets up the ATM VC to B.

The main advantage of this approach is that it avoids modifications to the handling of *Path* and *Resv* messages, by instead relying on the availability of NHRP mechanisms. Furthermore, as discussed in Section 4.3.5 it also makes possible the suppression of *Resv* and *Path* refresh messages once the ATM shortcut has been established. However, because the approach is not readily extendable to the multicast case (NHRP protocol is specified

[2]The length of this field would have to be extended from its current size of 8 bytes. The source and destination IP addresses cannot be inferred from the ATM addresses in the router–router case.

only for unicast addresses), the sender-initiated ATM shortcut (Section 4.3.2) is probably the preferred one of the two.

4.3.4 Receiver-initiated ATM Shortcuts

This last shortcut method is identical to the "sender-initiated" approach (presented in Section 4.3.2) in its handling of *Path* messages, but different in that the responsibility of establishing the ATM shortcut VC is shifted from ingress router A to egress router B (see Figure 4.8). This is possible because ATM unicast calls are always duplex, and resources can be reserved in both directions. Specifically, when a $Resv$ message arrives at the egress router B, B can generate a SETUP message towards A and specify the resources required in both directions. The SETUP message will specify QoS requirements in the direction A to B to accommodate the service specifications carried in the $Resv$ message. Conversely, it will not request any QoS or bandwidth guarantees from B to A since there is no data flow in this direction. While the VC setup is now handled by the egress router, it is still necessary to forward the *Resv* message to the ingress router, so that it can propagate that information upstream (it cannot be accurately inferred from the traffic and QoS parameters carried in the SETUP message). In order to do so, *Resv* messages including refreshes for reliability purposes, will continue to be forwarded on the IP route. However, as with the "sender-initiated" method of Section 4.3.2, they are not acted upon at intermediate routers. Another alternative is to include the *Resv* message as higher layer information in the SETUP message.

The main advantage of this scheme is that it is consistent with the preferred solution for multicast flows when the LIJ capability of UNI 4.0 becomes available (see Section 4.4.4 for details). As a result, we recommend that it be the solution of choice in the UNI 4.0 environment, while either the sender-initiated (Section 4.3.2) or the NHRP-based (Section 4.3.3) should be used with UNI 3.1.

4.3.5 Failure Handling and Route Changes

Failure handling in both the ATM and IP domains is important, as is the ability to react to changes in IP routes. It is particularly important to avoid the formation of persistent routing loops that may be caused by interactions between the ATM and IP level paths. As a general rule, this is achieved by giving precedence to the IP level mechanisms to decide when to tear-down a connection or establish a new one.

To detect connection failures in the ATM domain, we rely on the ATM mechanisms based on Operation And Management (OAM) flows [7] and hard connection states that the ATM network maintains. Briefly stated, OAM cells are periodically sent between the source and receiver ATM switches of any ATM VC. When the VC fails for any reason the OAM flow is interrupted and the ATM switches are informed of the failure.

Any IP route change should eventually be reflected on the underlying ATM connection. In the context of the solutions of Sections 4.3.2 and 4.3.4, this is achieved by having $Path$ messages continue to flow along the IP route. As a result, whenever the IP route changes, the *Path* messages will automatically follow the new route. If the route change does not affect the ingress and egress routers, the RSVP session remains undisturbed. However, if the $Path$ message reaches a new egress router, the RSVP session must be modified. This modification will be triggered when the ingress router A receives a $Resv$ message from a new egress router C. In response, router A sets up a new VC to C, and initiates the tear-down of the VC to the previous egress router B.

On the other hand, if the ingress router A changes to A', then $Path$ messages begin flowing through A' and stop flowing through A. This will trigger a new reservation in ATM starting in A' and ending in the egress router C. After some time, since $Path$ or *Resv* no longer visit A, the reservation (ATM VC) $A \Leftrightarrow C$ is terminated.

Note that because this scheme relies on refresh $Path$ messages to detect route changes, it is susceptible to transient loops. However, the duration of the transient loop is limited by the refresh period and the amount of data/packet loss (and the network impact) can be

bounded by adjusting the refresh period based on the flow characteristics. This can be accomplished quite easily by indexing the refresh period to the source traffic specification contained in the RSVP messages.

In the case of the "NHRP-based" solution (Section 4.3.3), robustness to changes in IP routes relies on NHRP. However, while the current NHRP mechanism works well when A and B belong to the same IP routing domain, its application to a more general environment is complicated. It has been shown that, under certain circumstances, the use of NHRP for the general router-to-router case can lead to the creation of persistent routing loops. Rekhter and Cole [85] have proposed three approaches to extend NHRP for this general environment:

1. terminate NHRP at IP routing boundaries;

2. maintain NHRP state at routing boundaries for queries that pass through;

3. detect routing changes using refresh messages between the ingress and egress routers.

Clearly, the first approach is the simplest and requires little state information or refreshes and, hence, their recommendation is to use this approach for the general environment. However, this solution requires additional IP hops through the ATM network, which is what we are trying to avoid for RSVP flows with QoS requirements. As a result, we advocate that this simple approach be used only for NHRP queries without specific QoS requirements. On the other hand, queries which contain QoS requirements, as is the case with the "NHRP-based" approach (Section 4.3.3), should be processed using the third approach to obtain an ATM shortcut spanning the whole ATM network. In this case, NHRP refresh queries essentially replace the refresh $Path$ messages used in the "sender-initiated" and "receiver-initiated" approaches.

4.4 Reservation Setup for Multicast Flows

This section focuses on RSVP-based reservation setup for *multicast* IP flows over ATM networks. We consider the general case in which the source of the data flow may reside outside the ATM network, and where the data flow traverses an ATM network in order to reach the destinations. These could be located on or off the ATM network.

The IP multicast model is a receiver-initiated model and permits many-to-many communication within a multicast group. Receivers wishing to subscribe to a multicast group use the IGMP protocol to inform their local router. Routers use multicast routing protocols such as DVMRP, MOSPF, or PIM to disseminate membership information. A sender wishing to send data to a multicast group simply sends IP packets to the IP address of the multicast group.

In this section we consider three models of RSVP support over ATM networks, and we focus on the case of multicast flows with fixed filter reservations. The first approach, "classical" RSVP support, preserves IP routing for the data flow but adds QoS support through the use of ATM VCs between the multicast routers, and between the routers and the participating hosts. The "sender-initiated ATM shortcut" model, and the "receiver-initiated ATM shortcut" model, extend the "classical" RSVP support by enabling ATM shortcuts. The main motivation for presenting both the sender- and the receiver-initiated approaches is that the former is better suited to the UNI 3.1 environment, while the latter is the preferred model in the context of UNI 4.0 with LIJ capability. Next, we consider the interplay between RSVP 'soft' state and ATM 'hard' state in the context of failure handling and route changes. Finally, we discuss the handling of RSVP filters in the context of multiple-sender multicast transmissions.

4.4.1 Multicast Address Resolution

In order to implement IP multicasting over an ATM network, a mechanism is needed to resolve IP group addresses to the corresponding ATM addresses. For example, referring to

Figure 4.9, forwarding the first $Path$ message received at A to the next IP hop E requires that the IP multicast group address be mapped to the ATM address of E. At times routers are statically configured with PVC connections, and then the routing of RSVP messages can proceed without this address resolution step. In the general case, however, an SVC would have to be set up between two routers or between a router and a host attached to the ATM network, in which case the address resolution step is required. We describe below two methods for resolving IP group address to ATM addresses. The term 'host' is taken here to mean either an end system or a router.

In the first method, one starts by mapping the IP group address to the set of IP addresses of member hosts on the LIS. Milliken [72] accounts for the fact that the ATM subnetwork is not a broadcast medium by modifying the IGMP mechanism. In his scheme the router elicits IGMP reports from hosts on the LIS. The IGMP reports sent by hosts are not re-transmitted to other hosts and thus hosts are not aware of other hosts' presence and are thus forced to send their own IGMP report. After collecting the IP addresses of the hosts in the multicast group, the router can map these addresses to ATM addresses by using the ATMARP server on the LIS.

An alternate method is to use the MARS address resolution mechanism [5], which generalizes the ATMARP server to multicasting, as described in Section 4.2. Our proposals on integrating IP and ATM resource reservation for multicast application in independent of which of these two (or any new protocol) is used.

The rest of this section does not depend on the scheme used for multicast address resolution. Either of these two schemes, or possibly some other scheme, can be used.

4.4.2 "Classical" RSVP Support

We extend the "classical" RSVP support discussed in Section 4.3.1 to single-sender multicast flows. The $Path$ messages traveling downstream are routed by the multicast-capable routers towards the members of the multicast group. The example in Figure 4.9

shows the route followed by these messages through an ATM network. A is the ingress router to the ATM network, while B and C are the egress routers. Consider now the $Resv$ messages from the receivers of the multicast which follow the reverse path upstream. When the first $Resv$ message from B arrives at F, the router at F sets up a VC from F to B. $Resv$ messages from F and C travel independently towards E. The arrival of these messages at E will eventually result in the setup of a point-to-multipoint VC rooted at E and with leaves F and C. Another VC will be set up later from A to E.

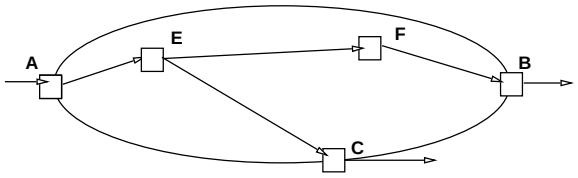


Figure 4.9. Reservation setup using "classical" RSVP support

The above description concerns the initial $Path$ and $Resv$ messages which trigger the reservation setup. $Path$ refresh messages are also forwarded along the IP route, in order to track route changes. Observe that $Resv$ refreshes are not needed, since the RSVP 'soft' state has been replaced in the ATM environment by a 'hard' state. Therefore, non-refresh $Resv$ messages will be sent only if the QoS parameters of the flow change.

4.4.3 Sender-initiated ATM Shortcuts

We extend the "sender-initiated" method to establish unicast ATM shortcuts (Section 4.3.2) to single-sender multicast flows, as illustrated in Figure 4.10. In this method, the establishment of multicast VC is initiated at the ingress router A, the root of the VC. This sender-controlled approach is suited to a UNI 3.1 environment. The determination of the ATM shortcut follows the same steps as in Section 4.3.2. When a $Path$ message for a session arrives at node A, the node determines[3] that the message will be forwarded over an

[3]This step only needs to be performed upon receipt of the first *Path* message.

ATM link and thus node A is the ingress node into the ATM network. The ATM address of A is inserted as an object into the $Path$ message, which is routed over the IP route. At each node along the route an ATM connectivity check is performed to determine whether the current node is an egress point from the logical ATM network. If the current node, such as F in Figure 4.10, is not an egress point then the $Path$ message is forwarded to the downstream nodes without updating the PHOP (previous hop) address field. As in the unicast case, F does not create and maintain a $Path$ state for this flow. Note that this means that we are in fact using automatic tunnelling back to the ingress router A. However, this also implies that the *Resv* messages will be handled as default IP traffic and not as control messages. This lack of preferential treatment for *Resv* messages is the price paid for avoiding state at intermediate routers.

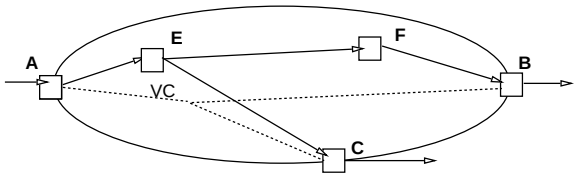


Figure 4.10. Reservation setup with maximum shortcut

When the first $Resv$ message arrives at an egress point, say B, it forwards the message along the reverse path to A. The ATM address of B is carried as an object in the $Resv$ message. Intermediate routers, F and E in this case, simply forward the message upstream towards A. Specifically, they do not merge *Resv* messages and do not perform any reservation. When the first $Resv$ message arrives at A, say from B, A has all the information necessary to create a shortcut point-to-multipoint VC with root A and leaf B. In order for B to associate the newly created VC with the RSVP flow, the flow identifier consisting of the pair `(source IP address, destination IP address)` is carried in the Broadband High Layer Information (B-HLI) field (specified in the UNI standard [7]) of the SETUP message. Later, when the $Resv$ message from C arrives at A, A adds C to

the point-to-multipoint VC with an ADD PARTY signaling message. The ADD PARTY message will also carry the flow identifier in the B-HLI element.

In order to track route changes and changes in group membership, $Path$ refresh messages continue to flow normally over the IP route. However, *Resv* refreshes from each router are suppressed as soon as the egress router receives the ATM setup message (ADD PARTY or SETUP for the first leaf). This is because the setup message indicates that the initial *Resv* message has been received by the ingress router, and that the reservation through the ATM network has been successfully performed. This suppression prevents the implosion of refresh *Resv* messages at the ingress router. However, the ingress router is still required to perform as many ATM connection SETUPs as there are leaves in the ATM network for the multicast address. This is because, the scheme always results in the use of a "maximum" ATM shortcut that directly connects the ingress and egress routers. A more promising and systematic approach to eliminate the possibility of signaling overload at the ingress router, is the use Leaf-Initiated Join (LIJ) in UNI 4.0, which is discussed next.

4.4.4 Receiver-initiated ATM Shortcuts

Consider the ATM network in Figure 4.10 and assume that the flow of $Path$ messages is as described in the previous section. That is, $Path$ messages continue to use the default IP route. As before, the $Path$ messages are not processed at intermediate routers and the PHOP is not modified. $Path$ messages are also extended at the ingress router A to carry the ATM address of A. In addition, A also chooses an ATM "global connection identifier" (GCID), and inserts it into the $Path$ message. This global connection identifier consists of a call identifier uniquely chosen by the root, which is paired with the root's ATM address for LIJ setup. For a given RSVP session, there may be multiple flows transiting through A and, for each flow, A would choose a distinct global connection identifier. This connection identifier will be used by egress routers when generating an ATM LIJ request to join the point-to-multipoint connection associated with the IP multicast address.

When the first $Resv$ message reaches an egress router, say B, the router has all the information needed to generate a Leaf Initiated Join (LIJ) request to the connection identified by the GCID received. The ATM point-to-multipoint connection is then created at this time, with the ingress router A as its root and B as the first leaf. As other egress routers, such as C in Figure 4.10, also receive their first *Resv* message, they signal their intention to join the connection in exactly the same manner, i.e., through a LIJ request to the specified GCID. They are then added as new leaves to the existing point-to-multipoint connection, and the ingress router A is not notified of this new join, which eliminates the potential processing overload at router A.

However, note that as a result of not notifying the ingress router of new leaves joining, the information carried in the *Resv* messages arriving at the associated egress routers is not forwarded to the ingress router during the ATM setup process. This information is, however, necessary for the ingress router to further propagate *Resv* messages upstream, i.e., it needs information elements such as the RSVP service and filter specifications, which as mentioned before cannot always be directly inferred from the ATM traffic and QoS parameters. In order to achieve this, *Resv* messages, including refreshes, will continue to be propagated and merged on the IP path, but no reservation will be triggered at intermediate routers. The merging on the IP path ensures that the ingress router is not overwhelmed by the volume of refresh *Resv* messages it receives, while providing it with all the information it needs to forward *Resv* messages to its upstream neighbor. Note that $Resv$ refreshes are not suppressed in order to ensure reliable delivery of *Resv* messages to the ingress router.

4.4.5 Failure Handling and Route Changes

Consider an RSVP-based multicast data flow which traverses an ATM network, and for which an ATM point-to-multipoint connection has been established. As with unicast connections, the issue is to ensure that the ATM and IP failure and recovery mechanisms interact consistently, so that permanent loops are avoided and robustness against failures is

provided whenever feasible. Failures can take place in both the ATM and in the IP domains, and when this happens there will be failure detection and recovery activity at both levels. As before, precedence is given to IP-level mechanisms when dealing with error recovery procedures.

At the IP level, the soft-state mechanism of $Path$ refreshes is used to recover from route changes and failures. When an egress router stops receiving $Path$ refreshes, it can conclude that it is no longer on the path to the destination and remove itself, directly (with LIJ) or indirectly, from the ATM multicast connection. On the other hand, if the ATM connection is broken while the RSVP session is still in place the egress router will attempt to re-establish the ATM connection by sending a $Resv$ message to the ingress router and re-initiating the leaf initiated join.

4.4.6 Handling of RSVP Filters

We consider here multiple-sender multicasts and examine the impact of RSVP filters on the reservation setup models. Figure 4.11 shows a multicast flow with three sources, S1, S2 and S3, traversing an ATM network. The egress router B receives a $Resv$ message from R, the receiver of the multicast.

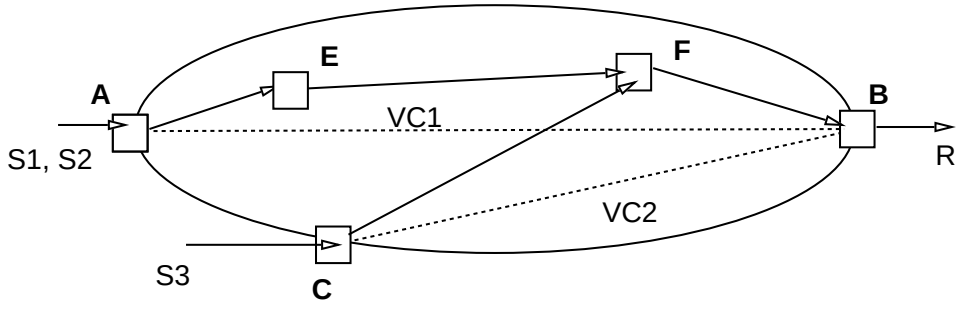


Figure 4.11. ATM shortcuts for an RSVP filter spec

Assume first that the $Resv$ message at B specifies a Fixed Filter (FF) involving the three sources. The egress router B maps this FF request into two separate FF requests, one

for $S1, S2$ and the other for $S3$. These FF requests are then forwarded upstream towards the ingress routers. Separate VCs would be setup for the three flows either by the ingress routers or by the egress router (in LIJ).

Assume now that the $Resv$ message received at B specifies a wildcard filter (WF) involving the three sources. Wildcard filters allow the sharing of resources across multiple flows as specified in the *Resv* message. This sharing is, however, only possible if the different flows are indeed routed on a common link. This is a valid assumption in an IP-only environment where *Resv* (and *Path*) messages travel on the same route as the data flow since routing decision are made independently of the flow resources and services requirements. This then implies that flows from different senders but destined to the same address (unicast or multicast), are likely to be routed on paths with significant overlap (of IP links).

The situation is, however, quite different when flows are to be routed across ATM networks. This is because the path assigned to a given flow is selected as a function of the flow traffic parameters and service requirements. Furthermore, since paths for different flows are likely to be requested and generated independently of each other, i.e., often triggered by requests from different receivers, the potential for overlap and sharing of resources in the ATM network is, therefore, much more limited. In addition, ATM currently does not provide the ability, in terms of signalling, to specify the sharing of resources between multiple ATM connections that is implied by wildcard filters. Furthermore, even if this ability was provided, it would also require that ATM switches on the path of such connections be capable of enforcing this sharing, i.e., controlling the number of cells sent. It is not clear if and when such capability might become ubiquitous in ATM switches as it requires tight per-connection queueing and scheduling.

As a result, requests that specify wildcard filters can currently only be handled in one of two ways. The wildcard filter can be translated into a set of fixed filters, one for each individual flow, and the creation or joining of an ATM connection is then performed for each of them. This amounts to ignoring the possibility of resources sharing within the

ATM network, which may be acceptable when, for example, most of the flows specified in the wildcard filter have already been established across the ATM network as a result of (fixed filter) requests from other receivers. In such configurations, the potential for sharing within the ATM network is relatively small. Note that this also applies to an IP-only environment. The second possible approach for handling wildcard filters, is to simply defer such requests to the IP overlay network. This clearly preserves all the sharing potential, but suffers from the obvious drawbacks of requiring IP level processing at each hop within the ATM network, and of forcing flows to be routed on links where resources might be constrained. Such an approach may, however, be the preferred one for sessions with large numbers of senders and receivers, where typically only one sender is active at the time, e.g., large audio conferences.

4.5 Issues Related to Flow/Call Characteristics

The previous sections have dealt with many of the issues related to the mapping between RSVP and ATM control flows. In this section, we focus on similar problems but at the level of the data flows. Specifically, we consider issues related to the mapping of traffic parameters and QoS guarantees as well as function placement. Some of these mappings consist of relating ATM cell-based measures to the corresponding packet/byte level quantities used in RSVP. Others are caused by differences in service specifications and capabilities, or are needed simply to identify where and how each step in the establishment of a connection is to be performed.

4.5.1 Mapping Traffic Parameters

Traffic is characterized in both IP and ATM using leaky bucket models:

- peak rate p, mean rate r and maximum burst size b in IP;
- Peak Cell Rate (PCR), Sustained Cell Rate (SCR) and Maximum Burst Size (MBS) in ATM.

If we assume a perfect fluid model for both the IP flow and the ATM call and ignore the potential impact of the granularity of the ATM cell size and of the segmentation overhead, the following relations can be established [94]:

$$\text{SCR} = \frac{r}{48} \tag{4.1}$$

$$\text{MBS} = \frac{b}{48} \tag{4.2}$$

$$\text{PCR} = \frac{\mathcal{P}}{48} \tag{4.3}$$

where 48 is the number of user data bytes per ATM cell, and $\mathcal{P}$ corresponds to the minimum between the speed of the incoming link and the peak rate p of the flow. Note that for a Controlled Load flow, $\mathcal{P}$ is always set to the speed of the incoming link since the Controlled Load TSpec does not include a peak rate term. It should be pointed out that the above expressions need to be adjusted to reflect the impact of the ATM segmentation in fixed size cells.

As mentioned before, the above expressions need to be adjusted to properly reflect the impact of the ATM segmentation in fixed size cells. There are two types of required adjustments. The first is to account for the fact that packet sizes need not be integer multiple of cell payloads, i.e., the last cell is typically not full. The second is to include any overhead introduced by the segmentation layer, which depends on the AAL type. If AAL5 is used, the overhead should be minimal (8 bytes in the last cell of a message).

Determining the exact amount by which the above expressions need to be adjusted is, however, not a simple matter as it depends on the entire packet size *distribution*. For example, if all packets are 41 bytes, then assuming AAL5, two cells are needed to transmit each of them (the 8-byte trailer and the 41-byte payload do not fit in a single cell). In this case, equations (4.1-4.3) are off by more than a factor 2. On the other hand, if all packets are 280 bytes long, then six cells (288 bytes) suffice to transmit them and equations (4.1-

4.3) are accurate. There are two possible approaches to resolve this problem. The first is to be conservative, while the second is to rely on some approximation.

A conservative estimate can be obtained based on the minimum packet size information provided in the RSVP TSpec. The basic idea is that while the error in equations (4.1-4.3) is not a continuous function of packet size (one cell for 40-byte packets and two cells for 41-byte packets), it is nevertheless a "regularly" decreasing function of packet size. Specifically, it has a saw-tooth behavior with jumps for every packet size of the form $n \times 48 \Leftrightarrow 7$ bytes, i.e., the number of cells needed goes from n to $n+1$ at these values. Based on this, conservative estimates for SCR, MBS, and PCR can be obtained by assuming that all packets are of minimum size and always require the maximum possible number of cells. The corresponding relations are then of the form:

$$\text{SCR} = \alpha r \tag{4.4}$$

$$\text{MBS} = \alpha b \tag{4.5}$$

$$\text{PCR} = \alpha \mathcal{P} \tag{4.6}$$

where $\alpha = (1 + \lceil P/48 \rceil)/P$ represents the worse case overhead due to the ATM segmentation with AAL5 and a minimum packet size of P (in bytes). Note that if some traffic shaping is performed at the IP/ATM boundary, as it probably should, the value of PCR could actually be decreased (more on this in Section 4.5.4).

4.5.2 Mapping of Controlled Load Service Specifications

The Controlled Load Service can be mapped to the Available Bit Rate service of ATM [92] in the case of unicast flows. In this case, the token bucket rate r of the Controlled Load TSpec is mapped to a corresponding value of the ABR Minimum Cell Rate (MCR). This value guarantees a floor rate to the flow, which is in keeping with the spirit of the Controlled Load specifications. The ABR service also allows transmission at a higher rate when resources are available. This is also in keeping with the spirit of the Controlled Load

specifications, that allows a Controlled Load flow to exceed its TSpec but without any real guarantee for that traffic. In that respect, the ABR service will actually provide a better level of service through the ATM network since it determines, through feedback messages, the maximum rate at which the flow can transmit without risking excessive losses. However, note that this ability of the ABR service to identify the "bottleneck" link for the flow comes at a price, i.e., the overhead of generating and processing RM cells.

While ABR provides a suitable service mapping for unicast Controlled Load flows, it cannot be used for multicast flows. This is because the ABR specifications have currently not been defined to cover the case of point-to-multipoint connections. Instead, Controlled Load multicast flows will have to be mapped onto Variable Bit Rate Non Real Time (VBR-NRT) ATM connections. Unfortunately, this mapping does not readily allow Controlled Load flows to exceed their specified rate, unless the rate parameter of the corresponding VBR connection has been set to a higher value than that corresponding to the token bucket rate r of the Controlled Load flow. This could be wasteful of resources in the ATM network. Another, possibly preferable, alternative is to rely on the marking feature of ATM networks, where excess traffic from the Controlled Load flow would be sent as CLP=1 cells through the ATM network. Note that such a mapping could also be used for unicast flows.

4.5.3 Mapping of Guaranteed Service Specifications

In the IP Guaranteed Service model [94], a maximum packte delay $\tilde{d}$ is guaranteed to a flow by reserving a minimum buffer clearing rate R such that

$$\tilde{d} = \frac{M + C_{tot}}{R} + D_{tot} + \begin{cases} \frac{b-M}{R}\frac{p-R}{p-r} & r \leq R < p \\ 0 & p \leq R \end{cases} \tag{4.7}$$

where

- M is the maximum packet size,
- p, r, b are the peak rate, mean rate and burst size respectively,

- $\mathcal{C}_{tot} = \sum_{i \in path} \mathcal{C}_i$ and $\mathcal{D}_{tot} = \sum_{i \in path} \mathcal{D}_i$, where,
- $\mathcal{C}_i$ is the rate-dependent delay term of link i,
- $\mathcal{D}_i$ is the rate-independent delay term of link i (see [94])

$\mathcal{C}_i$ and $\mathcal{D}_i$ are values accumulated by $Path$ on its way from source to receiver. R is computed at the flow's receiver given its delay objctive $\tilde{d}$. If the receiver can tolerate a maximum packet delay $d > \tilde{d}$, then a delay slack $\mathcal{S} = d \Leftrightarrow \tilde{d}$ is also computed. R and $\mathcal{S}$ are included in the RSpec section of the $Resv$ message that is sent toward source S.

A key aspect of the above approach, that complicates the interactions with ATM, is the decoupling between the advertising (accumulation of $\mathcal{C}_{tot}$ and $\mathcal{D}_{tot}$ as the *Path* message progresses) and the reservation phases (request for allocation of the clearing rate R). The main problem at the boundary of an ATM network is to determine which values to select for the terms $\mathcal{C}_{\text{ATM}}$ and $\mathcal{D}_{\text{ATM}}$ (that characterize the future ATM connection), while updating the $\mathcal{C}_{tot}$ and $\mathcal{D}_{tot}$ fields in the *Path* message. This is difficult for two reasons. First, the correct values are functions of the path through the ATM network, and this is not known at the time the *Path* message reaches the ingress (or egress) router of the ATM network (it will only be determined upon receipt of a *Resv* message at the egress router of the ATM network). Second, the form of the delay guarantees specified in [94], i.e., based on the specification of a clearing rate, will typically not be supported by ATM switches, and furthermore cannot be readily expressed through the ATM signaling. This means that the ATM network has to be modeled as a fixed delay component on the path. Hence there is a need to determine a value to advertise for $\mathcal{D}_{\text{ATM}}$, and further to comply with this advertised value when an ATM connection actually needs to be set up upon receipt of a *Resv* message. In the rest of this section, we review alternatives to address this problem.

4.5.3.1 Unicast Flows

The case of a unicast flow is illustrated in Figure 4.8. When $Path$ arrives at A, the fields $\mathcal{C}_{tot}$ and $\mathcal{D}_{tot}$ contain the values $\sum_S^A \mathcal{C}_i$ and $\sum_S^A \mathcal{D}_i$. *Path* now stops accumulating $\mathcal{C}_i$

and $\mathcal{D}_i$ for the duration of its journey through the ATM network, i.e., until it reaches router B. The issue is then to determine an estimate of the end-to-end delay guarantee $\mathcal{D}_{A,B}$, that can be provided between A and B by the ATM network, given the traffic parameters (p, r, b) provided in the TSpec of the *Path* message. We assume here that the mapping of the TSpec onto ATM traffic parameters is done following one of the methods of Section 4.5.1.

The first issue is to identify the router which is responsible for determining the value $\mathcal{D}_{A,B}$, and updating the *Path* message accordingly. There are two choices, the ingress or egress routers, i.e., router A or B. Both are equally capable of obtaining an estimate for $\mathcal{D}_{A,B}$, provided they know each other's ATM address. Access to this knowledge is dependent on the approach used to forward RSVP control information across the ATM network. From the discussion in Section 4.3, we know that A does not have the ATM address of B before the first $Resv$ arrival at A. It follows that the $\mathcal{C}_{tot}$ and $\mathcal{D}_{tot}$ fields contained in this first *Path* message cannot be updated by the ingress router A to advertise an estimate of the delay guarantee $\mathcal{D}_{A,B}$ across the ATM network. On the other hand, the egress router B can determine an appropriate value for $\mathcal{D}_{A,B}$ since the ATM address of the ingress router A is delivered to the egress router B together with the first *Path* message. In addition, as we shall see in the next section, this is also consistent with the approach that has to be used in the multicast case. However, note that this now requires that the selected value for $\mathcal{D}_{A,B}$ be communicated back to router A, so that it can specify the correct value in those cases where it is responsible for initiating the ATM call setup associated with the RSVP flow. This is done by including this information, together with the ATM address of router B, in the first *Resv* message that router B forwards to router A. Note that this problem does not arise if the receiver initiates the connection SETUP.

Once we have identified the router responsible for carrying out the determination of $\mathcal{D}_{A,B}$, it remains to specify how this is done. There are two generic approaches to obtaining an estimate of $\mathcal{D}_{A,B}$.

Local Determination of Delay Estimate. Router B generates an estimate for the delay $\mathcal{D}_{\text{ATM}}$ from router A to itself across the ATM network. This estimate can be a pre-configured value or could be inferred from information made available by the ATM network.

For example, if router B has a PNNI interface to the ATM network, it will then have access to the PNNI topology database [84]. This database contains link metrics from which it can obtain a reasonable delay estimate for a connection between A and itself, with traffic parameters as specified in the TSpec.

The advantage of this approach is its simplicity and the fact that it avoids the exchange of signaling messages with the ATM network. Its main disadvantage is its potential inaccuracy and lack of flexibility, especially in the case where a PNNI interface is not available and the advertised quantity is pre-configured.

Query ATM Network for Delay Estimate. In this solution, an ATM VC is established between A and B, and the ATM signaling for the Real-Time Variable Bit Rate (rtVBR) service model returns a value for maximum packet delay guarantee on this VC. (Note that B can initiate such a connection since it has the ATM address of A.) This interaction clearly involves additional overhead, but this overhead can be minimized by caching returned delay values. The main issue is, however, the selection of the delay requirement to be specified in the connection request.

Based on the ATM signalling specs for rtVBR [96], the user can specify in the SETUP message a Desired Maximum Cell Transfer Delay (DMCTD) for both the forward and backward directions of a connection. The VC establishment is initiated by B through a SETUP message sent to A. As the SETUP is routed through the ATM network and passes through different switches, each switch determines the maximum local delay it can guarantee in each direction and allocates resources accordingly. A Cumulative Maximum Cell Transfer Delay (CMCTD) field for each direction is carried in the SETUP message, and incremented by the local values at each switch. Once the SETUP message reaches its des-

tination, i.e., the ingress router A, the value carried in the CMCTD field for the backward direction identifies the delay guarantee that the network is actually able to provide to the connection. This value is then returned to the calling party, the egress router B, in the CONNECT message. This returned value can then be used by router B as its estimate for $\mathcal{D}_{A,B}$. Note that, since the value is also known to router A (it was included in the SETUP message), it may not be necessary to include it as well in the first *Resv* message sent by router B.

The main issue in the above procedure is for router B to determine which value of DMCTD to specify in the connection request. A default value could be used, but the benefit of querying the network would then be diminished. Preferably, the request should be for the "best" possible delay. Such a capability is not readily supported by the ATM signalling, but may be available in future extensions. For example, if it is possible to indicate that DMCTD is a "soft" value, i.e., the connection should not be rejected even if the specified value cannot be guaranteed. Router B could then safely choose an aggressive value, e.g., DMCTD $= 0$. The returned CMCTD value, typically larger than the required DMCTD, would then give the best feasible delay guarantees between A and B.

After router B has obtained a value for $\mathcal{D}_{A,B}$ and used it to update the $\mathcal{D}_{tot}$ field in the *Path* message, it remains to decide what to do with the ATM connection which was setup in order to obtain that information. The simplest solution is to disconnect it. This minimizes the amount of resources wasted (and paid for ...), but introduces a dependence on the state of the ATM network at the time the request was made. For example, if the network was unusually idle, the value returned for CMCTD would be much lower than a typical one, and the network may not be able to match this guarantee later when the *Resv* arrives and the actual connection needs to be established. This may, however, be acceptable since, even without crossing ATM networks, RSVP connections can be rejected for lack of available resources. Another solution is to keep the connection alive. This ensures that the advertised guaranteed delay can be provided when the *Resv* message eventually arrives.

This may result in a potentially significant wastage of resources. As a result, disconnecting the connection may be the preferred approach.

One feature common to the above solutions is that the advertised value is likely to be rather inaccurate, which can greatly increase the rejection rate of connections having to traverse ATM networks. It is, however, possible to greatly reduce the impact of this inaccuracy by allowing some flexibility in the delay guarantee that is eventually required from the ATM network, i.e., to provide a safety margin around the advertised value. Such a capability is included in [94] as the delay slack $\mathcal{S}$. The slack $\mathcal{S}$ corresponds to what remains of the end-to-end delay budget after the receiver has chosen a value for the reservation rate R. A receiver could purposely[4] select an R value so as to create some slack. The slack can then be used to provide some flexibility in the required delay guarantees through ATM networks. Specifically, each router on the path from S to D can take some of the slack if necessary, provided it properly updates the slack field to reflect the adjusted amount. This means that if router i consumes an amount $\mathcal{S}_i$ of the slack, it updates the slack field as follows: $\mathcal{S} \leftarrow \mathcal{S} \Leftrightarrow \mathcal{S}_i$, before forwarding the $Resv$ message to its upstream neighbor. In the context of a connection through an ATM network, the slack (if present) can be used to compensate for differences between the value currently feasible, and the quantity $\mathcal{D}_{A,B}$ that was initially advertised. This can improve the chances of success for the connection.

4.5.3.2 Multicast Flows

Multicast flows share the problems of unicast flows when mapping IntServ delay guarantees to corresponding ATM quantities. In the following we focus on aspects for which significant differences exist between the multicast and unicast cases.

The first major difference is the identity of the router making an estimate for $\mathcal{D}_{\text{ATM}}$ can be obtained. In the unicast case, this could be performed at either the ingress or the egress routers. In the multicast case, an estimate for $\mathcal{D}_{\text{ATM}}$ must be performed at the

[4]For example, if it knew it had to cross some ATM network.

egress routers for multicast flows because it is likely that different ATM addresses would yield different values for $\mathcal{D}_{\text{ATM}}$, and those could not be distinguished through a single *Path* message at the ingress router. Note that each egress router determines an estimate for $\mathcal{D}_{\text{ATM}}$ between itself and the ingress router whose ATM address was carried in the *Path* message. This corresponds to a direct ATM connection between the ingress and egress routers, which is unlikely to be the case as connectivity to (all) the egress routers will typically be provided by a single point-to-multipoint connection. This is yet another source of inaccuracy in the determination of $\mathcal{D}_{\text{ATM}}$.

A second difference between unicast and multicast flows is in the way the information is provided to and used by the router responsible for setting up the ATM connection. In the multicast case, we need to distinguish two cases depending on the type of signaling available to establish point-to-multipoint connections.

Sender-initiated point-to-multipoint ATM connection. This is the only approach available in UNI 3.1. The point-to-multipoint connection is root initiated, i.e., it relies on ADD-PARTY messages that all originate from the root. It is thus imperative that the root be provided with both the ATM address of the egress router and the value of $\mathcal{D}_{\text{ATM}}$ to be used in each ADD-PARTY. These must, therefore, be included in the *Resv* message generated from all the egress points. As discussed in Section 4.4, the *Resv* messages should not be merged as information about each individual "leaf" is needed at the root to set up the point-to-multipoint ATM connection.

Leaf initiated join (LIJ) to a point-to-multipoint ATM connection. In this case, the egress routers directly join the point-to-multipoint connection, while specifying the desired delay guarantee $\mathcal{D}_{\text{ATM}}$ they determined and advertised in the *Path* messages. Note that while UNI 4.0 defines the LIJ capability, it does not yet allow specification of different guarantees to different leaves. The unavailability of such a feature is clearly a major problem in supporting multicast RSVP flows using LIJ.

4.5.4 Handling Changes in Flow Specifications

In this section, we briefly address how flow reservation changes within RSVP can be handled across ATM networks. Specifically, RSVP allows receivers to change the service specification carried in the *Resv* messages for a given flow at any time. Request for such changes should essentially be transparent to the ATM network since it does not fully participate in the resource (clearing rate) allocation process triggered by a reservation request. Specifically, recall that the ATM network is handled in the *Path* advertising phase as a *fixed* delay component. Hence, it is unaffected by any change in the requested clearing rate R. This certainly simplifies the handling of RSVP reservation changes in ATM networks, but also points to a limitation in terms of the flexibility with which ATM networks can support RSVP services.

It is possible to remedy this problem to some extent by using an approach based on the method of [41]. The basic idea is that the peak rate of an RSVP flow can actually be limited at the entrance of an ATM network to the clearing rate R, without impacting the end-to-end delay guarantees. The ATM network could, therefore, reshape the flow to a peak of R as per the clearing rate value specified in the received *Resv* message. Since lower peak rate connections are easier to deal with and typically require the allocation of fewer resources for a given delay guarantee, the ATM network could take advantage of this. In particular, it could keep the guaranteed delay to a given connection constant, i.e. equal to the value advertised in the *Path* message, but adjust its allocated resources as a function of the specified RSVP clearing rate. This requires reshaping of the traffic to ensure that its "peak rate" indeed complies with the clearing rate, as well as the ability to signal such changes to the ATM network.

While the above approach allows ATM networks take advantage of changes in RSVP reservations, ATM signaling may not always provide the capability to do so. In particular, UNI 3.1 does not allow the characteristics of a VC to be changed once it is established. A possible alternative may then be to tear down the existing VC and set up a new VC in

the ATM segment of the flow, but this can be expensive if user data interruption is to be eliminated (buffering and/or two coexisting calls will be needed). UNI should allow for dynamic changes of call characteristics (it currently does not), and the above approach may then be readily supported.

Finally, note that using the requested clearing rate R as the peak rate for the connection can also help improve the likelihood that the ATM network can guarantee the advertised delay $\mathcal{D}_{\text{ATM}}$ when the connection request is eventually generated. This is because, when $\mathcal{D}_{\text{ATM}}$ was determined, the requested clearing rate was not yet known and, therefore, $\mathcal{D}_{\text{ATM}}$ was obtained assuming a higher peak rate, i.e., the access link speed or the peak rate value specified in the TSpec, if available. The final decision as to which peak rate to specify to the ATM network when the connection request is generated depends on a number of other factors, such as the reshaping and buffering capabilities present at the access point, i.e. the ingress router, and is likely to vary for each implementation.

4.6 Summary of RSVP and ATM Signaling Extensions

In this chapter we have described several aspects of a novel method for resource reservation in IP/ATM networks. The solutions that show the greatest potential require a minimum amount of extensions to RSVP and ATM signaling. In this section we summarize these extensions.

4.6.1 RSVP Modifications for UNI 3.1 Environment

In the context of ATM UNI 3.1 signaling, the general approach we take can be characterized as *root oriented.* This means that most of the interactions with the ATM signalling needed to extend RSVP flows across ATM networks, originate at the ingress router. Such extensions require a number of modifications to the processing of *Path* and *Resv* messages.

The first step at the ingress router is to identify that the flow is to cross an ATM network and should, therefore, be handled differently. Once this has been determined, subsequent

modifications to *Path* message handling vary somewhat as a function of the approach used. Typically, the *Path* message will be forwarded on the normal IP path, and extended to carry the ATM address of the ingress router. *Path* processing is also different at intermediate (non-egress) routers. They do not update the PHOP field, so that it still points to the ingress router, and they do not maintain state information. This helps lower the processing overhead for such messages. In addition, the $\mathcal{D}_{tot}$ field (and $\mathcal{C}_{tot}$) is not updated until the *Path* message reaches the egress router(s), where it is incremented by an estimate of the maximum delay the ATM network would contribute. *Path* messages continue flowing on the IP route even after an ATM VC shortcut has been established for the flow.

In the unicast case, we also outlined a possible alternative to the forwarding of *Path* messages that relies on NHRP mechanisms to forward the *Path* information to the appropriate egress router. This eliminates the need to modify *Path* processing at intermediate routers, and allows leveraging of mechanisms that may be available. However, this approach cannot be extended to multicast flows.

The processing of *Resv* messages is also affected when crossing ATM networks. They are used to trigger the establishment of an ATM shortcut when received at an egress router(s). The connection request originates from the ingress router (ADD-PARTY for multicast flows, or SETUP for unicast flows) upon receipt of a new *Resv* message from an egress router. This *Resv* message carries the standard RSVP information, i.e., filter and service specifications, that are needed by the ingress router to forward *Resv* messages to its upstream neighbor. The *Resv* message also contains the ATM address of the egress router as well as the delay guarantees needed for the connection across the ATM network. Note that the receipt of the SETUP (or ADD-PARTY for multicast flows) at an egress router provides an implicit acknowledgment that the ingress router has received the *Resv* message and that the ATM reservation has been successful. Finally, refresh *Resv* messages are suppressed, i.e. not forwarded on the IP path, and connection liveness is guaranteed by ATM mechanisms.

4.6.2 RSVP Modifications for UNI 4.0 Environment

The major enhancement in UNI 4.0, from the point-of-view of providing support for RSVP, is the LIJ ability in point-to-multipoint connections. This allows us to use a *receiver-initiated* approach to support RSVP flows (both unicast and multicast) which ensures better scalability.

The handling of *Path* messages remain essentially the same as in the case of UNI 3.1, in that they are forwarded on the normal IP path but not processed at intermediate routers, i.e., the PHOP field and OPWA objects are not modified and no state is created. In addition to carrying the ATM address of the ingress router, the *Path* message also carries a global ATM call identifier (GCID) in the case of multicast flows. This GCID is then specified in the LIJ message generated by egress routers upon receipt of a new *Resv* message, when they want to join an existing point-to-multipoint connection associated with a given multicast flow. In the case of a unicast flow, the egress router simply initiates a SETUP to the ATM address of the ingress router.

Because in the receiver-initiated approach the egress routers are responsible for the establishment of ATM connections, it is not necessary to forward *Resv* messages to the ingress router for that purpose. However, it is still necessary to transmit the RSVP information contained in the *Resv* message (filter and service specifications) to the ingress router, so that it can propagate this information upstream. This is achieved by forwarding all *Resv* messages (including refreshes for reliability) on the IP route to the ingress router. Note that although *Resv* messages are processed at intermediate routers they are not acted upon, i.e., merging of *Resv* messages will take place when required but no reservations will be triggered and no state will be maintained.

4.6.3 ATM Extensions and Modifications

As stated above, it is clear that many of the extensions to be included in UNI 4.0 are crucial to providing efficient support of RSVP flows across ATM networks. Foremost among

them is the LIJ capability, which is critical for handling large multicast connections. This capability should, however, be such that different leaves are allowed to specify different service requirements. Another desirable extension is the ability to renegotiate the characteristics of an established connection. However, there other desirable extensions which may not be provided in UNI 4.0. For example, it would be helpful for an RSVP router to query the ATM network to find the *best* delay that can be guaranteed to a given destination. This can be achieved either by allowing "soft" requests, or by supporting both "desired" and "acceptable" QoS parameters. Similarly, the ability to let the root of a point-to-multipoint call assign a GCID even before any leaf has requested to join, could simplify some of the processing when establishing such calls.

4.7 Conclusion

4.7.1 Discussion

There are a number of issues for discussion regarding our approach to resource reservation in IP/ATM networks that we have not addressed. One issue is that the establishment of shortcuts via RSVP, and the use of these shortcuts for multicast traffic, may result in receivers receiving duplicate packets. Another issue is that our approach requires changes to RSVP and ATM signaling. Yet another issue is that performing ATM shortcuts via RSVP can be viewed as RSVP performing a routing function. This would be contrary to a stated RSVP design principle, which requires that the routing function be separate from RSVP, and interoperable with it.

The problem of duplicate packets at some receivers arises sometimes because we want to ensure that we reach all receivers, including those that do not make any reservation. In order to achieve this, data packets should be sent on both the default VC, and the shortcut VC. Depending on how packets are delivered to those receivers which did not make reservations, some receivers with active reservations may receive packets on both the shortcut path, and the IP path.

This duplication of packets does not seem to be a serious problem. In normal cases, a simple filter can eliminate the problem altogether. For example, a receiver connected to a shortcut VC stops listening for packets of that flow on the default path. This type of filtering, whose overhead is typically negligible for the end-station, is common in multicast routing.

In contrast, the fact that packets are duplicated by the sender (for sending to both the default and shortcut VC), and then sent on a single ATM interface, is potentially more troublesome. Both problems, however, could be avoided altogether if, and when, 'variegated' ATM VCs become available, i.e., point-to-multipoint VCs which provide different QoS to different receivers, as needed. Until this type of connection is available, one could avoid packet duplication at the receivers by extending the shortcut VC to include all receivers, including those without a reservation. This solution, which gives receivers without reservation a "free ride", is our preferred solution, and it is similar to the case of a broadcast medium in which some receivers obtain enhanced service based on other receivers' reservations. Our conclusion, therefore, is that packet duplication is not a major problem, and can be handled in a number of ways. If variegated VCs become a reality, it will be eliminated altogether without any sacrifice in efficiency.

Another issue is whether RSVP is performing a routing function when the ATM shortcut is established. As noted above, this would be against one of the RSVP's design principles.

Current routing protocols do not take QoS into account. Likewise, there is no RSVP mechanism to influence routing based on QoS attributes. While RSVP designers stated a worthy goal, the goal of separating the routing function from the resource reservation function, it is still an open question how QoS is to be attained with such a separation. Meanwhile, it is likely that RSVP, and routing protocols, will change if the promise of QoS is to become reality.

There are additional considerations, concerning the actual changes to RSVP which are needed in order to implement our approach. In our proposal, RSVP messages serve as a vehicle for the shortcut information, but this artifact does not convert RSVP into a routing protocol. While RSVP messages always follow the IP route, once the shortcut is in place the data flows over it, and not on the IP route. Having the data traffic follow a different path from the path of the signaling messages is required in order to exploit the capabilities of ATM. While QoS information carried in the RSVP messages is used by the shortcut managing entity, RSVP has no say in establishing the shortcut.

One could decouple RSVP from the shortcut mechanism by using NHRP instead. If so, NHRP state must be maintained at the source and destination endpoints for each QoS session, which is essentially a duplication of part of RSVP state. Moreover, a temporary connection, along the same path as the shortcut is required to carry RSVP messages prior to creation of the QoS connection. This alternate solution suffers from duplication of functionality, and thus is not efficient.

The approach described here has, without doubt, shortcomings. Some of these can be traced to the differences between RSVP and ATM signaling and their sometimes conflicting design principles. Our approach is offered as a possible first step in supporting QoS flows in a heterogeneous environment with ATM networks. If adopted and carried through to implementation, the experience thus gathered may be beneficial in the design of a better next scheme.

4.7.2 Summary

In this chapter we focused on the establishment of QoS connections in a heterogeneous environment which includes ATM networks. For sessions whose path extends across ATM networks we investigated the interplay between RSVP flows at the network layer and ATM calls at the subnetwork layer.

We have used an approach, classical RSVP over ATM with shortcuts, which is intended to leverage the strengths of the ATM technology in support of applications with QoS requirements. Establishing shortcuts through an ATM network avoids the performance penalty associated with layer 3 processing in a classical IP over ATM approach.

We provided solutions applicable to both unicast and multicast flows. While attempting to provide solutions which apply equally to signaling in the UNI 3.1 environment, as well as the UNI 4.0 environment, we found that different solutions are required in order to account for significant differences between the two signaling protocols.

In this chapter we have touched on a number of topics for which much work remains to be done. First, a method is needed to better account for an ATM network's contribution during the advertising phase carried out through RSVP *Path* messages. This can mean better estimates for the delay guarantees that an ATM network can provide, or extensions to ATM signaling and service specifications to better emulate the Integrated Services model [94]. Second, while the LIJ of the UNI 4.0 is intended to improve scalability by removing the bottleneck associated with the ingress router, it does so by shifting the processing burden from the ingress router to the ATM network. It is, therefore, important to ensure that the the server infrastructure which handles ATM signaling is designed in a scalable way, and is able to support large multicast groups.

CHAPTER 5

PERFORMANCE EVALUATION OF ATM SHORTCUT CONNECTIONS IN OVERLAID IP/ATM NETWORKS

5.1 Introduction

As we have seen in Chapter 4, the recent development and deployment of network layer ATM protocols such as ATM signaling [91] and PNNI [84] makes it possible for IP to interact with ATM over the same physical network. Recently, several proposals have been put forward to improve the efficiency of IP/ATM networks through an interaction between IP and ATM network layers. One proposal [75] is to *cut through* the IP processing of IP packets and transmit the data of an IP flow through a separate ATM virtual circuit (VC). The VC connects two IP routers at the edge of the ATM network, and the route of the ATM VC follows the IP route. Another set of proposals goes one step further and permits the ATM VC, called *ATM shortcut*, connecting two IP routers, to be routed by the ATM network. ATM shortcuts are proposed by the Next Hop Resolution Protocol (NHRP) [70] for unicast IP flows and by Multicast Address Resolution Protocol (MARS) [5] for multicast IP flows. Last, in Chapter 4 we proposed a combination of RSVP and ATM signaling for resource reservation of IP flows with Quality of Service (QoS) requirements in IP/ATM networks.

IP cut-through has proved to be beneficial in experimental systems and commercial products by increasing the average throughput of IP routers embedded in ATM networks [67, 34]. Additional performance improvements are possible in the ATM shortcut operation mode since routing IP flows in the ATM network provides the opportunity for more efficient network resource utilization than the IP cut through mode. However, there has not yet been any theoretical or practical confirmation of the benefit of ATM shortcuts. In this chapter we address the following question:

Given an overlaid IP/ATM network, what is the performance benefit of ATM shortcuts?

This question is important in IP/ATM network design, where the benefits of ATM shortcuts can be weighed against the simplicity of a separate model where IP and ATM are maintained at separate layers. The purpose of this chapter is to develop a methodology for evaluating ATM shortcuts, given an IP/ATM network, and to assess the benefits of ATM shortcuts in a variety of scenarios.

To answer the above question, we first model an IP/ATM network, with and without ATM shortcuts, as two distinct network graphs. We then propose a metric to compare the performance of the two networks: the Network Load Ratio. For a given IP/ATM configuration, the Network Load Ratio represents the additional load the IP/ATM network can accept on average when using ATM shortcuts (compared with the "IP cut through" case), for the same network blocking probability. We develop a method to compute the Network Load Ratio for a given network blocking probability based on the Fixed Point Method [89]. We also develop the notion of Asymptotic Load Ratio which is an approximation to the Network Load Ratio when the networks operate in underload conditions. We find experimentally through simulation that this provides a good approximation (the 90-percentile of the relative error is less than 0.12) to the Network Load Ratio for a range of network blocking probabilities (less than 0.02), thus giving a general indication of the relative performance of the two networks. This second method is very attractive due to its low computational complexity. We use this method in simulation experiments to identify network topological conditions where ATM shortcuts bring high benefits. We find that in many cases the utilization of an IP/ATM network is improved significantly when the average route length decreases upon using ATM shortcuts. We also find that there is almost no correlation between the increase in network utilization (when using ATM shortcuts) and the IP to ATM node ratio.

The chapter is organized as follows. In Section 5.2 we provide a short background on overlaid IP/ATM networks and motivate the problem of evaluating the benefits of ATM shortcuts. In Section 5.3 we propose a metric for network performance comparison, the Network Load Ratio and its limiting value for underloaded conditions, the Asymptotic Load Ratio, and two methods for their approximate computation. In Section 5.4 we evaluate the accuracy with which the Asymptotic Load Ratio estimates the Network Load Ratio. In Section 5.5 we use the Asymptotic Load Ratio in simulation experiments to determine network topological conditions where ATM shortcuts bring high benefits. Section 5.6 concludes the chapter .

5.2 Background and Motivation

5.2.1 IP over ATM

IP and ATM networks are increasingly coexisting on the same medium as overlaid networks. It is unlikely that either IP or ATM will disappear in the foreseeable future; rather both architectures are expanding and currently provide different types of service. IP is the established protocol for data communication networks, such as the Internet. ATM is progressively deployed in telecommunication networks that mainly carry voice traffic (for example AT&T and MCI have had ATM backbones for years). Data networks using the IP protocol are currently carried over the telecom networks, but the two networks (IP and ATM) are virtually separated (such as is the case with NSF's very High Speed Backbone Network Service, vBNS [54]). It is possible, however, for IP to become aware of the underlying ATM infrastructure and to take advantage of the capabilities of the underlying ATM network through ATM shortcutting, as we have investigated in Chapter 4.

5.2.2 Overview of ATM Shortcut Performance Benefits

As presented in Chapter 4, so far, a significant amount of work has been done to specify the protocol details for establishing ATM shortcuts, but the benefit of ATM shortcuts has

not been quantitatively evaluated. No practical confirmation of the benefit exists either, as the protocols are still in the development phase. The main benefit of ATM shortcuts comes from the better utilization of network resources when IP flows are routed in ATM, as can be seen in the following example.

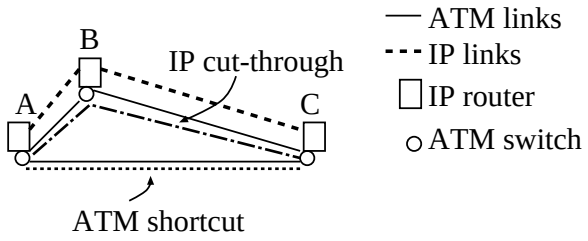


Figure 5.1. ATM shortcut versus IP cut through

Consider a simple network as in Figure 5.1. A flow with source A and destination C has route $A \Leftrightarrow B \Leftrightarrow C$ in IP and $A \Leftrightarrow C$ in ATM. If the flow requires one unit of bandwidth, then the network bandwidth consumption by the given flow is two units when using IP routing and one unit when using ATM routing. It follows that routing IP flows in the ATM network may yield a lower network bandwidth consumption. This in turn can result in a lower network blocking probability, or equivalently, more flows can be admitted for a given network blocking probability.

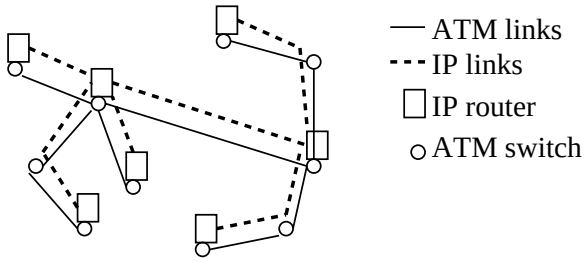


Figure 5.2. IP/ATM network with no ATM shortcut benefit

The benefit of ATM shortcuts depends on the topology of IP and ATM networks. Consider, for example, the network in Figure 5.2, where both IP and ATM networks have a tree

structure. In this case, an IP flow between any pair of IP nodes has the same route in both the IP and ATM networks. Thus, for a given IP flow, the IP cut-through VC is identical to the corresponding ATM shortcut and, thus, an ATM shortcut brings no additional benefit beyond that of IP cut-through in this network.

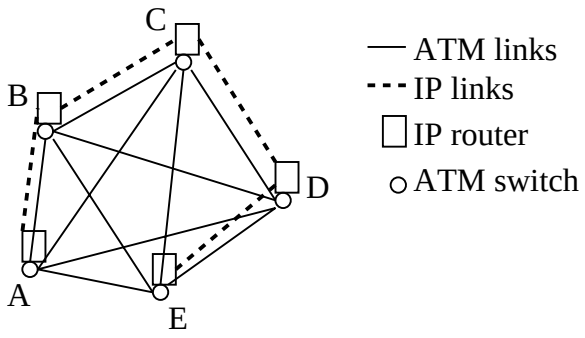


Figure 5.3. IP/ATM network with significant ATM shortcut benefit

At the other end of the spectrum, consider the example in Figure 5.3, where the IP network is a tandem network (the IP nodes are arranged in series), and the ATM network is fully connected (forms a complete graph). Given an IP flow between A and H, its IP route crosses four ATM links, whereas its ATM route has only one link. In this case we expect a significant benefit when using ATM shortcuts over IP cut-through.

Clearly, an evaluation of the benefit of ATM shortcuts is needed, based on the characteristics of the IP and ATM networks. Specifically, a systematic method is needed to identify the cases where ATM shortcuts are beneficial and to quantify such a benefit. We develop two such methods in the following section.

5.3 A Metric and Two Methods for Network Performance Comparison

Our approach in quantitatively evaluating the benefits of ATM shortcuts consists of comparing the flow loads offered to two networks having the same blocking probability.

We will compare a given IP/ATM network where IP flows are routed by IP routing, with the same IP/ATM network where IP flows are routed by ATM routing. In this section we propose a metric to compare the performance of two loss networks, and we develop two methods for estimating this metric. The technical details rely on the theory of loss networks; see [102, 58, 59, 89] and references therein.

We assume that both IP and ATM routing are fixed routing. Alternate routing, possible in ATM and also under study for IP (see for example [24]), may bring additional benefit in network utilization. Nevertheless, computing network loss probability is much more complex when using alternate routing (see for example [46]). Moreover, alternate routing in ATM, as well as QoS-based routing in IP, is work in progress, and therefore, not completely specified. Therefore, here we consider fixed routing as a first step in evaluating the performance of ATM shortcuts. An extension to our study that would include alternate routing constitutes an interesting future work.

We consider a network consisting of a set of nodes $\mathcal{V}$ and a set of links $\mathcal{L}$. A link $l \in \mathcal{L}$ has capacity C_l. The flows that arrive to the network have origin and destination in a set $\mathcal{A} \subseteq \mathcal{V}$, called *access nodes*. (In the next sections, $\mathcal{V}$ will be the set of ATM nodes, and $\mathcal{A}$, the set of IP nodes.) These flows are classified into a set of classes $\mathcal{K}$ according to their end-point nodes, i.e., there is a separate class $k \in \mathcal{K}$ for each pair of nodes (A, B), $A, B \in \mathcal{A}$.

Since we assumed that routing is fixed, it follows that all flows of class $k \in \mathcal{K}$ are routed on route r_k, where r_k is the sequence of links traversed by flows of class k. All flows have the same bandwidth requirement (henceforth taken as a unit of bandwidth), irrespective of class. A class k flow is admitted iff one unit of capacity can be reserved at each link $l \in r_k$. We define $\mathcal{R} = \{r_k | k \in \mathcal{K}\}$ to be the set of routes in the given network.

Flows arrive to the network according to a Poisson process with rate λ and have exponential holding times with mean $1/\mu$. The intensity of offered load is $\rho = \lambda/\mu$. With probability ν_k a flow becomes a class k flow. A flow is blocked at a link if the flow's band-

width requirement is larger than the link's available bandwidth. The probability that a flow of any class is blocked at link j is denoted by L_j. A flow is blocked (its establishment is denied) if it is blocked at any link on its path (route). The probability that a class k flow is blocked (not admitted) is denoted by B_k. The probability that any flow is blocked, called the network blocking probability, is denoted by P and is given by:

$$P = \sum_{k \in \mathcal{K}} \nu_k B_k = P(\rho) \tag{5.1}$$

where P depends on ρ through B_k. Let us define ϕ_k to be the intensity of offered load of class k:

$$\phi_k = \nu_k \rho \quad \forall k \in \mathcal{K} \tag{5.2}$$

We also introduce η_i, the probability that a flow traverses link i,

$$\eta_i = \sum_{k \in \mathcal{K}, i \in r_k} \nu_k \quad \forall i \in \mathcal{L} \tag{5.3}$$

and ψ_i, the intensity of offered load on link i,

$$\psi_i = \sum_{k \in \mathcal{K}, i \in r_k} \phi_k = \eta_i \rho \quad \forall i \in \mathcal{L} \tag{5.4}$$

Let us now consider two networks N_1 and N_2 with link sets $\mathcal{L}_1$ and $\mathcal{L}_2$ and route sets $\mathcal{R}_1$ and $\mathcal{R}_2$, that have the same set of access nodes, $\mathcal{A}_1 = \mathcal{A}_2 = \mathcal{A}$ (i.e., the same set of flow end points), the same set of flow classes, $\mathcal{K}_1 = \mathcal{K}_2 = \mathcal{K}$ and the same offered load probabilities $(\nu_k)_{k \in \mathcal{K}}$. We define the following metric for comparing the performance of the two networks.

Definition 5.1 *The* Network Load Ratio *of networks N_1 and N_2 at blocking probability p, $R(N_1, N_2, p)$, is defined to be the value of R such that*

$$P_1(R\rho) = P_2(\rho) = p \tag{5.5}$$

Network N_1 is said to perform R times better than N_2 at blocking probability p.

In other words, if N_1 and N_2 have a Network Load Ratio of R, N_1 can handle a load that is R times greater than N_2 for a network blocking probability p. Observe that the Network Load Ratio R depends on the network blocking probability p and on the characteristics of the two networks: $R = R(N_1, N_2, p)$. Henceforth, when no confusion is possible, we drop the parameters N_1 and N_2 from the expression of R.

The Network Load Ratio for a given network blocking probability p is given by:

$$R(N_1, N_2, p) = \frac{P_1^{-1}(p)}{P_2^{-1}(p)} \tag{5.6}$$

where $P_i^{-1}(.)$ is the inverse of function $P_i(.)$, $i = 1, 2$. In general, the network blocking probability cannot be computed exactly due to its computational complexity [69]. However, a good approximation is given by the Fixed Point method, also known as the Reduced Load approximation [59]. Following the Fixed Point method, an approximation of the blocking probability B_k of flows in class k, B_k^*, is given by:

$$B_k^* = 1 \Leftrightarrow \prod_{j \in r_k} (1 \Leftrightarrow L_j^*) \tag{5.7}$$

where L_j^*, an approximation of the blocking probability L_j at link j, is the solution of the following system of equations,

$$L_j^* = Er(\sum_{k \in \mathcal{K}, j \in r_k} \phi_k \prod_{i \in r_k - \{j\}} (1 \Leftrightarrow L_i^*), C_j) \quad j \in \mathcal{L} \tag{5.8}$$

Here $Er()$ is the Erlang loss formula,

$$Er(\rho, C) = \frac{\rho^C / C!}{\sum_{n=0}^{C} \rho^n / n!} \tag{5.9}$$

It follows that an approximation for the network blocking probability is,

$$P^* = \sum_{k \in \mathcal{K}} \nu_k B_k^* \tag{5.10}$$

The system of equations (5.8) can be solved by repeated substitutions, and is known to converge in most practical cases, giving a good approximation to B_k. The inverse of the network blocking probability function, $\rho = P^{-1}(p)$, can be computed using any numerical method for approximate root computation (e.g., the secant method) in conjunction with the Fixed Point method. The procedure based on equations (5.2), (5.6)-(5.10) allows us to compute the Network Load Ratio of two networks that operate at a given network blocking probability. It constitutes the first method that we propose for network performance comparison.

A problem with this method is its rather high computational complexity. The computation of Network Load Ratio based on equations (5.6-5.10) has complexity $O(L^2CKFS)$, where L is the number of links, C is the maximum of link capacities, K is the number of flow classes, F is the number of iterations in the Fixed Point method, and S is the number of iterations of the secant method. This complexity arises from the fact that the computation in (5.8) is done for all L links, with the product in (5.8) being $O(L)$, the sum in (5.8) being $O(K)$, and the computation of the Erlang function being performed in $O(C)$. From our experiments in Section 5.4, for an approximation error of 0.01, F typically falls between 3 and 100, and S typically falls between 10 and 50.

Another problem with this network comparison method is that it yields the Network Load Ratio $R(p)$ of two networks for a given blocking probability p. The point value $R(p)$ may not provide much insight into the relative behavior of the two networks for a range of network blocking probability values p.

After a series of empirical experiments with various networks of different topologies and loads, we have come to the conclusion that the Network Load Ratio $R(p)$ generally exhibits little variation in a range of values for p that are of interest to us. For example

in Figure 5.5 we plot the network blocking probability of four networks with 17 access nodes and four different topologies: the complete graph, tandem, star and a model (that we labeled "N17") of the NSF Backbone (see Figure 5.4) as it was in 1995. In each network, for each pair of nodes (and thus for each flow class k), we take $\nu_k = \nu$, $\forall k \in \mathcal{K}$. We observe that the Network Load Ratio is almost constant for blocking probabilities in the range $p \in [10^{-5}, 0.01]$. In the following we provide a formal motivation for this empirical observation and derive a second method for comparing two networks.

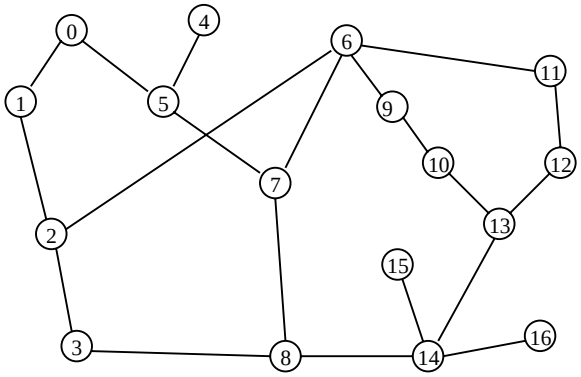


Figure 5.4. Topology of N17 (a model of NSF Backbone)

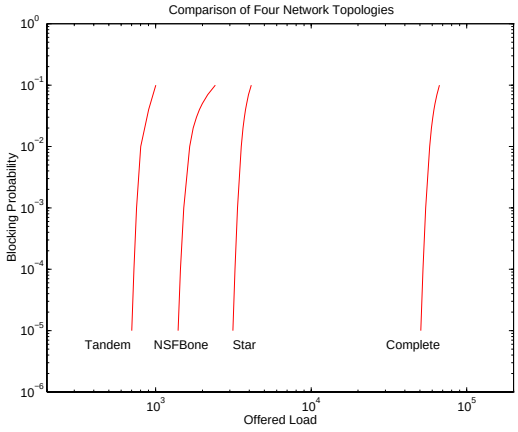


Figure 5.5. Comparison of four network topologies

We observe (as illustrated in Figure 5.5) that a "small" value of network blocking probability (i.e., $p \in [10^{-5}, 10^{-1}]$) does not necessarily correspond to a small value of ρ (in Figure 5.5, $\rho \in [10^2, 10^5]$). Thus, we consider the limit of the network load ratio as $p \to 0$, and not as $\rho \to 0$ (ρ approaches 0 more slowly).

Let us consider two networks N_1 and N_2 having all parameters defined earlier.

Definition 5.2 *The limit of the Network Load Ratio $R(N_1, N_2, p)$ as $p \to 0$,*

$$A_0(N_1, N_2) = \lim_{p \to 0} R(N_1, N_2, p) = \lim_{p \to 0} \frac{P_1^{-1}(p)}{P_2^{-1}(p)}$$

is the Low-Blocking Load Ratio of networks N_1 and N_2.

Observe that A_0 is a function of link capacities C_i. In Appendix C.1 we establish the following result.

Proposition 5.1 *If all link capacities are the same in both networks, $C_i = C$, $\forall i \in \mathcal{L}_q$, $q = 1, 2$, then the limit to the Low-Blocking Load Ratio as $C \to \infty$, named Asymptotic Load Ratio, is:*

$$A(N_1, N_2) = \lim_{C \to \infty} A_0(N_1, N_2) = M_2/M_1 \tag{5.11}$$

where

$$M_q = \max_{i \in \mathcal{L}_q} \eta_i \quad q = 1, 2 \tag{5.12}$$

Intuitively, Proposition 5.1 says that the relative performance of two networks in underload conditions is given by the ratio of loads on their bottleneck links. This result confirms the heuristic of "minimizing the maximum congestion" used in several alternate routing algorithms in circuit switching networks (see [89], Section 7.1 and references therein).

Proposition 5.1 yields a simple approximation of the Asymptotic Load Ratio when link capacities are large ($C > 100$, as is the case with most current networks).

For the general case, where the values of C_i are not restricted, we make the following conjecture:

Conjecture 5.1 *The Low-Blocking Load Ratio is well approximated by the Asymptotic Load Ratio:*

$$A(N_1, N_2) = M_2/M_1 \tag{5.13}$$

where

$$M_q = \max_{i \in \mathcal{L}_q} \eta_i/C_i \quad q = 1, 2 \tag{5.14}$$

In Section 5.4 we present a set of experiments where we find the Asymptotic Load Ratio to be a good approximation for the Network Load Ratio.

The computation of Asymptotic Load Ratio given in equations (5.11),(5.12), (5.13) and (5.14), constitutes our second proposed method for network comparison. Note that the complexity of the computation is $O(LK)$ where L is the number of links in the two networks and K is the number of flow classes. This makes the computation of Asymptotic Load Ratio much simpler than that of Network Load Ratio.

5.4 Accuracy of Asymptotic Load Ratio

In this section we evaluate the accuracy of Asymptotic Load Ratio. First, we verify that the Asymptotic Load Ratio, computed as in (5.11), (5.12) and (5.13), (5.14), is the limit of Network Load Ratio as the network blocking probability approaches zero. Second, we seek to determine the range of network blocking probabilities for which the Asymptotic Load Ratio is a good approximation to the Network Load Ratio. Third, we measure and compare the computation times incurred while computing the Network Load Ratio and Asymptotic Load Ratio.

We perform the evaluation through a set of simulation experiments, where the two methods are applied to pairs of randomly generated network topologies. We use the net-

works generated at Georgia Tech using the methods proposed in [100], here named RAND, and in [106], named Transit-Stub (TS). [1] The RAND networks have unstructured topologies, whereas the TS networks exhibit hierarchical structures and smaller diameters, which are claimed [106] to more accurately resemble real networks. All networks contain 100 nodes. All links in the RAND networks have OC3 (155 Mb/s) capacity, whereas only the backbone links in the TS networks have OC3 capacity and the rest, T3 (45 Mb/s) capacity. The bandwidth of a flow is 100 Kb/s. The shortest path routing policy is used to generate the routes.

In the first experiment, 10 networks of each network type were considered, and compared to each other using the two comparison methods. Specifically, the Network Load Ratio $R(N_i, N_j, p)$ is computed for each pair of networks N_i, N_j and for the following values of network blocking probability:

$$p \in \{10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}, 0.02, 0.03, 0.04, 0.05, 0.07, 0.1\}$$

For each network pair N_i, N_j, the Asymptotic Load Ratio $A(N_i, N_j)$ is also computed. We then compute the relative error of Asymptotic Load Ratio compared to Network Load Ratio for each network blocking probability value and each pair of networks:

$$e_{N_i,N_j}(p) = \frac{A(N_i, N_j) \Leftrightarrow R(N_i, N_j, p)}{R(N_i, N_j, p)}$$

In Figures 5.6 and 5.7 we plot, for each value of network blocking probability p, the average of the relative error values $e_{N_i,N_j}(p)$, over the set of all pairs (N_i, N_j) of networks. We also plot a vertical bar indicating the 5-percentile and the 95-percentile of the set $(e_{N_i,N_j}(p))_{i,j}$. From the two graphs we observe that the relative error converges to zero

[1] The networks and the generating code can be found at http://www.cc.gatech.edu/fac/Ellen.Zegura/-graphs.html

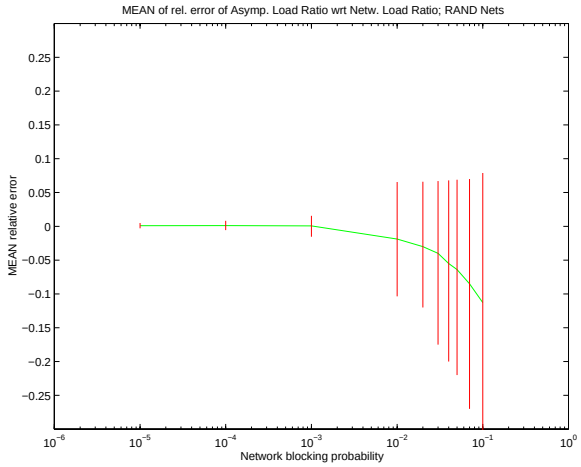


Figure 5.6. Mean of relative error of Asymptotic Load Ratio with respect to Network Load Ratio; RAND networks

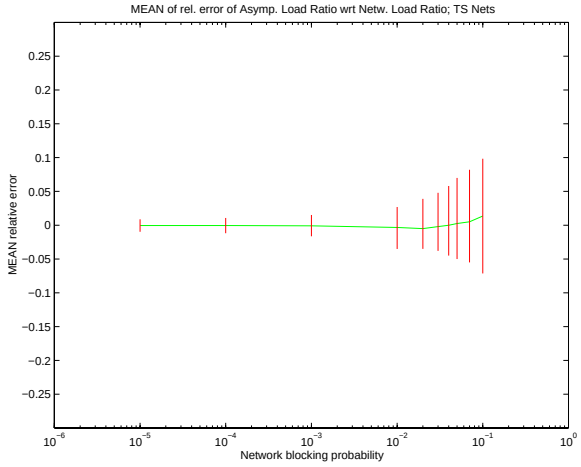


Figure 5.7. Mean of relative error of Asymptotic Load Ratio with respect to Network Load Ratio; TS networks

as $p \to 0$, which confirms Proposition 5.1 and Conjecture 5.1 that the Asymptotic Load Ratio is the limit of Network Load Ratio.

Second, we note that the 90-percentile of the relative error is less than 0.12 for a network blocking probability less than 0.02, indicating that the Asymptotic Load Ratio is a good estimate for the Network Load Ratio in this range.

Last, we remark that the approximation is better for TS networks than for RAND, which suggests that the Asymptotic Load Ratio has greater potential for “real” networks.

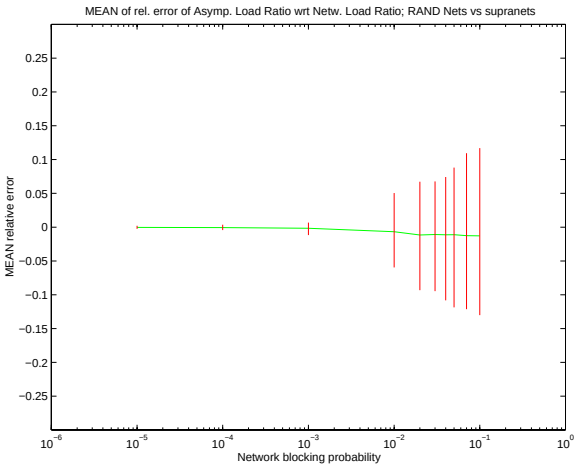


Figure 5.8. Mean of relative error of Asymptotic Load Ratio with respect to Network Load Ratio; RAND networks versus supranets

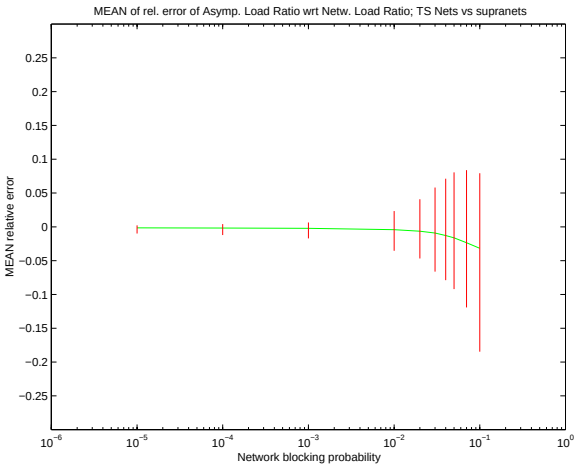


Figure 5.9. Mean of relative error of Asymptotic Load Ratio with respect to Network Load Ratio; TS networks versus supranets

In the second experiment we treat each of the networks considered in the first experiment as a base network. Starting from each base network we create 10 “supranets” by adding a random number of links (between 1 and 10). We then compute the Network Load

Ratio and Asymptotic Load Ratio for each base network and supranet combination. The relative error statistics are processed as in the first experiment and displayed in Figures 5.8 and 5.9. We obtain similar results as before: the relative error converges to zero, and the 90-percentile of the Asymptotic Load Ratio is within 10% of Network Load Ratio for $p < 0.02$.

Given the above experiments, we conclude that the Asymptotic Load Ratio is a good approximation of the Network Load Ratio for $p < 0.02$, and thus is a good metric for comparing networks.

For the above experiments we have also recorded the time required to compute the Network Load Ratio and Asymptotic Load Ratio for each pair of networks compared. The average computation time has been measured with the Unix code profiler *prof* on an 100MHz SGI workstation. Table 5.1 displays the average computation times for RAND and TS networks respectively. The difference in computation time in favor of Asymptotic Load Ratio confirms the simplicity predicted at the end of Section 5.3.

Table 5.1. Comparison of computation times for Network Load Ratio and Asymptotic Load Ratio

	RAND	TS
Network Load Ratio	103.927 s	129.676 s
Asymptotic Load Ratio	0.008 s	0.012 s

In conclusion, we consider the Asymptotic Load Ratio to be preferable to the Network Load Ratio for comparing network performance for two reasons:

- The Asymptotic Load Ratio is empirically found to give a good approximation (the 90-percentile of the relative error is less than 0.12) for the relative performance of two networks (the Network Load Ratio) for a range of network blocking probability values ($p \in (0, 0.02)$).
- The Asymptotic Load Ratio is very simple computationally (four orders of magnitude faster) compared to the Network Load Ratio method.

5.5 Evaluating the Benefit of ATM Shortcuts on Random Networks

Informally, we have seen in Section 5.2.2 that the relative performance of two networks depends on their topologies. Thus, it is not possible to provide a general, comprehensive answer to the question of whether and when ATM shortcuts are beneficial. Nevertheless, we can study the sensitivity of the Asymptotic Loss Ratio (ALR) with respect to several topological characteristics of the IP and ATM networks such as average route (path) length or average node degree. For this purpose we have performed simulation experiments where we generate IP and ATM network topologies using the Georgia Tech network generator [106]. We computed the networks' topological characteristics and ALR, and derived statistical dependencies of ALR on the networks' characteristics.

The objective of these experiments is to generate pairs of (IP and ATM) networks with realistic topologies. The current overlayed IP/ATM networks exist mostly in the Internet backbones (such as vBNS [54]). ATM backbone topologies are expected to carry both telephone and data traffic. Hence, we expect ATM networks to be designed similar to telephone networks. Telephone networks backbones are "nearly fully connected" ([60], page 10): the average node degree is high and the average path length is small. By contrast, IP networks (including backbones) are usually "less connected". Also, backbones usually have a flat (non-hierarchical) topology. In our experiments we generate networks with these observations in mind.

We generate random networks where a link is generated between two nodes with probability α. Thus, α is a parameter that influences the "connectedness" of the network. We generate ATM networks with $N_A \in \{16, 32\}$ nodes and with connectedness parameter $\alpha_A \in \{0.2, 0.4, 0.6, 0.8\}$. We generate 10 ATM networks for each combination of (N_A, α_A) values. All links have capacity OC3 ($155Mb/s$). The IP networks are generated with $N_I = q * N_A$, where $q \in \{0.25, 0.5, 0.75, 1.0\}$, and using a connectedness parameter $\alpha_I \in \{0.25, 0.5, 0.75, 1\} * \alpha_A$. This produces 16 sets of (N_I, α_I) values for each (N_A, α_A) pair, and we generate 10 IP networks for each set of (N_I, α_I) values.

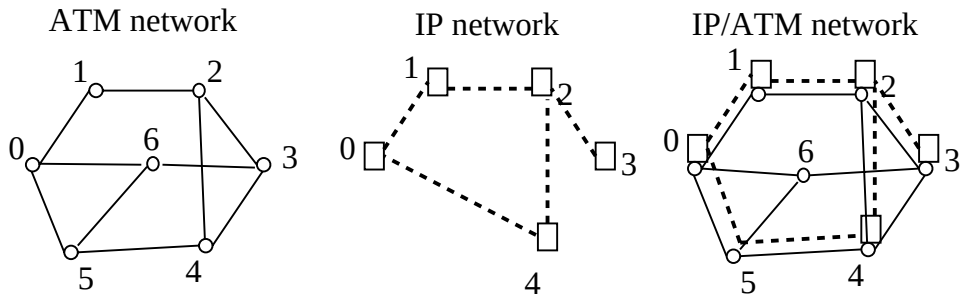


Figure 5.10. Overlaying an IP network over an ATM network

For each pair of ATM and IP networks, we overlay IP over ATM by mapping IP nodes to ATM nodes and the IP links to the shortest paths (hop count) in the ATM network (see example in Figure 5.10). For each overlayed IP/ATM network, the load consists of IP flows requesting reservations of $100Kb/s$ between any pair of IP nodes with the same intensity $\phi_k = \phi, \forall k \in \mathcal{K}$ (ϕ_k as defined in (5.2)). The IP flows can be routed in two ways: one is over the shortest path in the IP network following the path of each IP link in the ATM network; the other is over the shortest path in ATM network (i.e., using ATM shortcuts). For example, in Figure 5.10 an IP flow between nodes 0 and 3 has route $0 \Leftrightarrow 1 \Leftrightarrow 2 \Leftrightarrow 3$ if routed in IP, and route $0 \Leftrightarrow 6 \Leftrightarrow 3$ if routed in ATM (i.e., $0 \Leftrightarrow 6 \Leftrightarrow 3$ is an ATM shortcut).

We evaluate the benefit of ATM shortcuts for each IP/ATM overlayed network by computing the ALR for each of the two ways of routing of IP flows. Also, for each IP/ATM overlayed network, we compute the following topological characteristics for each of the two ways of routing of IP flows:

- $N_A =$ the number of ATM nodes, $N_I =$ the number of IP nodes (these are also ATM nodes);
- B_A, B_I, the average degree = average number of links connected to each node;
- C_A, C_I, the normalized average degree, $C_A = B_A/N_A$, $C_I = B_I/N_I$;
- D_A, D_I, the diameter = length of longest route;

- E_A, E_I, the average depth = average route length;
- F_A, F_I, the number of biconnected components. Here a biconnected component is a maximal set of edges such that any two edges in the set are on a common simple cycle.

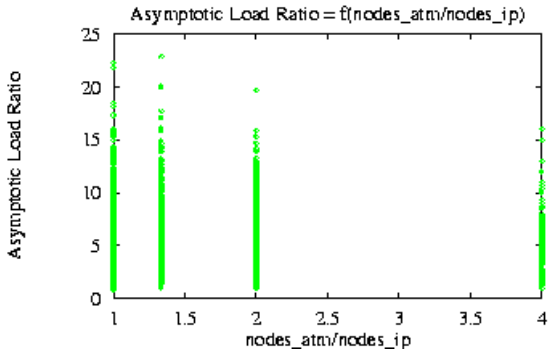


Figure 5.11. Asymptotic Load Ratio versus Node Ratio

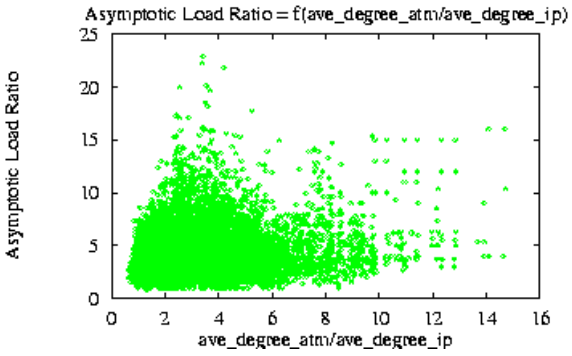


Figure 5.12. Asymptotic Load Ratio versus Average Degree Ratio

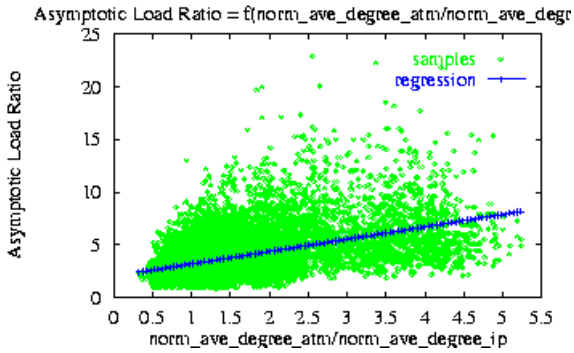


Figure 5.13. Asymptotic Load Ratio versus Normalized Average Degree Ratio

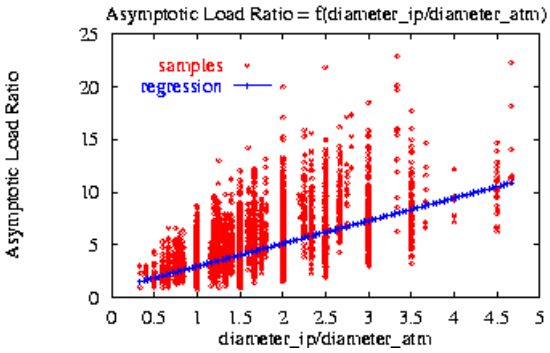


Figure 5.14. Asymptotic Load Ratio versus Diameter Ratio

In Figures 5.11-5.16, we display scatter plots of ALR as a function of ratios of the topological characteristics enumerated above, where each point represents an IP/ATM network pair (we have 12800 points in each figure). We seek to establish if ALR depends on any of these topological ratios. First, we observe that the node ratio N_A/N_I has little influence on ALR (see Figure 5.11), whereas ALR increases as a function of the ratio of average

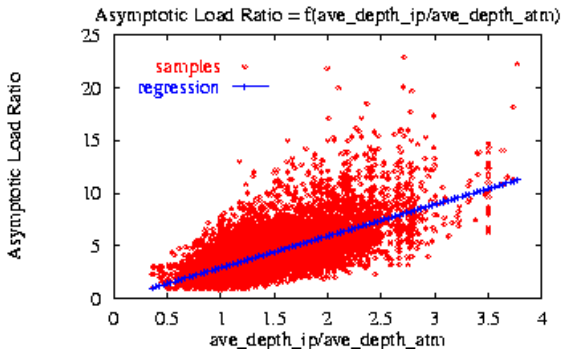


Figure 5.15. Asymptotic Load Ratio versus Average Depth Ratio

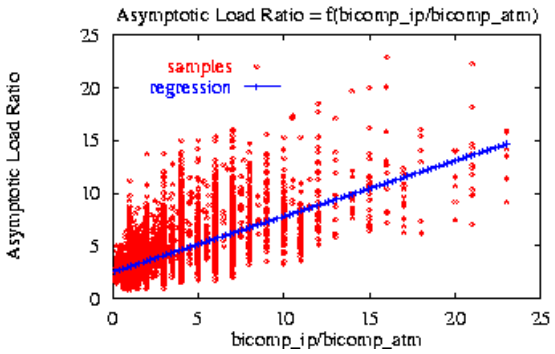


Figure 5.16. Asymptotic Load Ratio versus Number of Bicomponent Ratio

Table 5.2. Coefficient of correlation between Asymptotic Load Ratio and various topological ratios

	Coefficient of correlation w.r.t. ALR
No. Nodes ratio (N_A/N_I)	0.232
Ave Degree ratio (B_A/B_I)	0.164
Normalized Ave Degree ratio (C_A/C_I)	0.469
Diameter ratio (D_I/D_A)	0.610
Ave Depth ratio (E_I/E_A)	0.672
No. Bicomponents ratio (F_I/F_A)	0.622

Table 5.3. Coefficient of correlation between various topological ratios

	Norm ave degree ratio (C_A/C_I)	Diameter ratio (D_I/D_A)	Ave Depth ratio (E_I/E_A)	No. Bicomponents ratio (F_I/F_A)
Norm ave degree ratio (C_A/C_I)	1.000	0.791	0.746	0.364
Diameter ratio (D_I/D_A)	0.791	1.000	0.930	0.604
Ave Depth ratio (E_I/E_A)	0.746	0.930	1.000	0.686
No. Bicomponents ratio (F_I/F_A)	0.364	0.604	0.686	1.000

depth (see Figure 5.15). In order to quantify the correlation, we present, in Table 5.2, the coefficients of correlation between ALR and each of the topological ratios. These values indicate a weak correlation of ALR with respect to node ratio and average degree ratio, and a significant correlation with respect to diameter ratio, average depth ratio and number of

bicomponents ratio. The linear regression computed using the least square error method is depicted by the lines in Figures 5.13-5.16.

A further question to ask is what is the cumulative dependence of ALR on diameter, average depth, normalized average degree and number of bicomponents ratios. If the parameters exhibited independent effects on ALR, we could perform a multiple linear regression [53]. In Table 5.3 we present the coefficient of correlation between the four topological ratios of interest. We observe a strong correlation between the diameter ratio and the average depth ratio (it is somewhat natural for them to be related, since the diameter and average depth measure the maximum and average of the same set: the path lengths). Given this, we can eliminate diameter ratio from the multiple linear regression. Consequently, we have computed the following multiple linear regression:

$$\text{ALR} = 0.5 + 1.92 * (E_I/E_A) + 0.26 * (F_I/F_A) + 0.08 * (C_A/C_I)$$

We observe in the above regression that ALR has only a weak dependence on the Normalized Average Degree Ratio. By also observing in Table 5.3 a significant correlation between Normalized Average Degree Ratio and Average Depth Ratio, we conclude that we may drop it from the multiple linear regression, which then becomes:

$$\text{ALR} = 0.5 + 2.0 * (E_I/E_A) + 0.25 * (F_I/F_A)$$

In conclusion, we observe the following trends regarding the benefits of ATM shortcuts in IP/ATM networks:

- The network using ATM shortcuts is likely to accept more IP flows than the network not using them provided the network not using shortcuts has a high average depth or large number of bicomponents.
- The ATM shortcut benefits are generally not influenced by the number of nodes or the average degree of the IP/ATM network, with or without ATM shortcuts.

5.5.1 Discussion

Our simulation experiments above have provided us with statistically significant results, but did not provide an explanation for the correlation or independence of parameters. In this section we discuss these results and propose possible interpretation.

The fact that a significant decrease in average route length when using ATM shortcuts is generally an indication for significant ATM shortcut benefit is not surprising. In general, when route lengths are high, it is more likely that some links are heavily loaded (i.e., have very many routes crossing them). Since the Asymptotic Load Ratio is a load ratio of bottleneck links, it is then natural that two networks with high and low route lengths respectively to have a high ALR.

On the other hand, the fact that an IP/ATM network with a high ATM to IP node ratio does not necessarily have a high ATM shortcut benefit may appear in contradiction with the intuition that a "rich" ATM network has a high potential for ATM shortcut benefits.

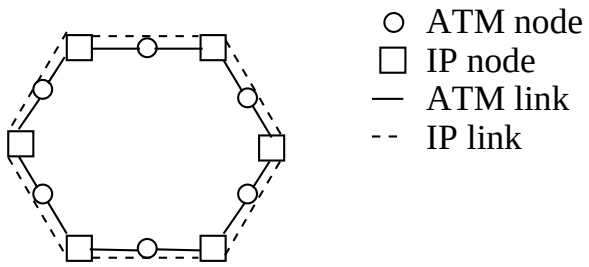


Figure 5.17. IP/ATM networks with ALR=1

Let us consider an example of IP/ATM network in Figure 5.17. This topology was designed such that between any two adjacent IP nodes there is an ATM node. If routing in both IP and ATM is shortest path, then the IP route between any two IP nodes is the same as the corresponding ATM route. Therefore, NLR and ALR are 1, i.e., there is no ATM shortcut benefit, although $N_A/N_I = 2$.

On the other hand, in the example in Figure 5.18, $N_A/N_I = 7/6$, but ALR is $A = 9/4$ (the maximum number of IP routes over an IP link is 9: on $C \Leftrightarrow D$, from A, B, C to

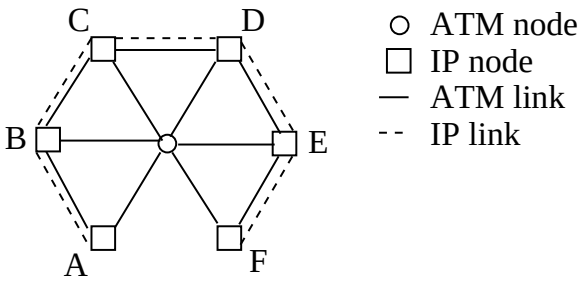


Figure 5.18. IP/ATM networks with ALR=1

D, E, F; the maximum number of ATM routes over an ATM link is 4: on $A \Leftrightarrow G$, from A to C, D, E, F).

From the above two examples we can see that high node ratio may bring no benefit, whereas there may be significant benefit for a small node ratio. A possible explanation is that the ATM shortcut benefits, as measured by NLR and ALR is essentially given by a difference in link loads, and not in number of nodes.

5.6 Conclusion

In this chapter we have considered the problem of evaluation of benefit of ATM shortcuts in IP/ATM networks, i.e., the benefit of ATM routing of IP flows in ATM networks. We proposed to measure the benefit of ATM shortcuts with the Network Load Ratio, that expresses the increase in the number of flows accepted by an IP/ATM network for the same network blocking probability. We developed a low complexity computation for Asymptotic Load Ratio, which estimates the Network Load Ratio relatively well (the 90-percentile of the relative error is less than 0.12) for underload network conditions (network blocking probability less than 0.02). An important result of this chapter is a methodology for comparing network performance, which can be used to evaluate the benefit and tradeoff of ATM shortcuts, and in the more general context of network design. We use this method in simulation experiments using random networks. These experiments indicate that in many cases the utilization of an IP/ATM network increases proportionally to the decrease in the

average path length when ATM shortcuts are used. We have also found that the decrease in the number of biconnected components of the IP/ATM network when using ATM shortcuts has also an influence on the network utilization, although not as strong as the average path length. Moreover, we have found that there is almost no correlation between the increase in network utilization (when using ATM shortcuts) and the IP to ATM node ratio.

This work can be extended in several directions. First, the conjecture that extends the Asymptotic Load Ratio to networks with heterogeneous capacities is an open problem that needs to be proved. Second, the results and methods may also be extended to networks with alternate routing, flows with heterogeneous bandwidth requirements, and multicast flows.

CHAPTER 6

SUMMARY AND FUTURE WORK

6.1 Summary of the Dissertation

In this section we summarize the research presented in this dissertation. In Chapter 1 we presented the general framework of resource reservation for multicast communication with Quality of Service guarantees. We discussed the need for supporting resource reservation for multicasting while accommodating network heterogeneity. We identified several problems and discussed the motivation for providing efficient solutions to each of them. The problems are: admission control and resource reservation algorithms for multicast sessions, admission control algorithms for piecewise linear envelopes at Earliest Deadline First schedulers, support for RSVP reservations in IP over ATM networks, and evaluation of ATM shortcuts in IP over ATM networks.

In Chapter 2 we have developed solutions to the problem of admission control and resource reservation for a multicast application once a route (multicast tree) has been chosen. We have developed centralized resource reservation algorithms, where the QoS computation is done at a single place (for example the multicast source node), and distributed algorithms, where the QoS computations and resource reservation are done at the nodes in the multicast tree. These algorithms accommodate heterogeneity of link resource availability and heterogeneity in receiver QoS requirements. We have evaluated these algorithms in two different network settings: one having packet loss probability as the QoS guarantee and the other, maximum packet delay. Our simulation results show that when our algorithms are used, the network can accept about 10% more sessions for the loss probability

model and about 50% more sessions for the delay model compared to the case where the reservation algorithms do not accommodate heterogeneity.

In Chapter 3 we have proposed practical solutions to the problem of admission control for real-time flows with delay guarantees at an EDF scheduler, as a part of end-to-end flow admission control. We applied the admission control conditions put forward by [66] to flows characterized by multiple-segment envelopes. We developed a first set of algorithms with a computation complexity of $O(KN)$, where N is the number of flows admitted in the EDF scheduler at the time of algorithm invocation and K is the number of segments per envelope. A second set of algorithms places the horizontal position of flexion points of flow envelopes into a predefined set of values (discretization points), thus reducing the computational complexity of admission control to $O(K + L)$, where L is the number of predefined discretization points and K is the number of segments per envelope. A set of simulation experiments showed that the improvement in execution time achieved by the discrete admission control is indeed very important (240 times faster for an OC12 link) and that the algorithm's execution time is independent of the number of flows admitted. Moreover, we have seen that the link performance degradation of the discrete admission control relative to the exact admission control is less that 5%, when using a small number of discretization points (15). Taken together, these results suggest that the algorithms we have developed in this chapter form the basis of a practical and highly efficient solution for the problem of admission control of real-time flows at EDF schedulers.

In Chapter 4 we focused on the establishment of reservations for flows with QoS requirements in the heterogeneous environment of IP over ATM networks. For sessions whose path extends across ATM networks, we investigated the interplay between RSVP flows at the network layer and ATM calls at the subnetwork layer. We have used an approach where RSVP interoperates with ATM signaling for establishing ATM shortcuts, which is intended to leverage the strengths of the ATM technology in support of IP applications with QoS requirements. Establishing shortcuts through an ATM network avoids

the performance penalty associated with layer 3 processing in a classical IP over ATM approach.

In Chapter 5 we have considered the problem of evaluation of benefit of ATM shortcuts in IP/ATM networks, i.e., the benefit of ATM routing of IP flows in ATM networks. We proposed to measure the benefit of ATM shortcuts with the Network Load Ratio, that expresses the increase in the number of flows accepted by an IP/ATM network for the same network blocking probability. We developed a low complexity computation for Asymptotic Load Ratio, which estimates the Network Load Ratio relatively well (maximum relative error less than 0.12) for underload network conditions (network blocking probability less than 0.01). An important result of this chapter is a methodology for comparing network performance, which can be used to evaluate the benefit and tradeoff of ATM shortcuts, and in the more general context of network design. We use this method in simulation experiments using random networks. These experiments indicate that in many cases the utilization of an IP/ATM network increases proportionally to the decrease in the average path length when ATM shortcuts are used. We have also found that the decrease in the number of biconnected components of the IP/ATM network when using ATM shortcuts has also an influence on the network utilization, although not as strong as the average path length.

6.2 Directions for Future Work

In this section we indicate several possible areas of future work which follow naturally from the work in this dissertation.

An interesting avenue for future work is to study how our multicast resource reservation algorithms apply to QoS guarantees other than packet loss probability and maximum packet delay. Also interesting is to study the behavior of various QoS division policies in the context of session correlations as is the case of filters proposed by RSVP.

In the area of RSVP-ATM interoperability, much work remains to be done. Here are two specific items. First, a method is needed to better account for an ATM network's

contribution during the advertising phase carried out through RSVP *Path* messages. This can mean better estimates for the delay guarantees that an ATM network can provide, or extensions to ATM signaling and service specifications to better emulate the Integrated Services model [94]. Second, while the Leaf Initiated Join of the ATM UNI 4.0 is intended to improve scalability by removing the bottleneck associated with the ingress router, it does so by shifting the processing burden from the ingress router to the ATM network. It is, therefore, important to ensure that the the server infrastructure which handles ATM signaling is designed in a scalable way, and is able to support large multicast groups.

A third avenue of future work is to extend the evaluation of ATM shortcuts in several directions. First, the conjecture that extends the Asymptotic Load Ratio to networks with heterogeneous capacities is an open problem that needs to be proved. Second, the results and methods may also be extended to networks with alternate routing, flows with heterogeneous bandwidth requirements, and multicast flows.

APPENDIX A

PROOF OF THEOREM IN CHAPTER 2

A.1 Proof of Theorem 2.1

Proof. Consider two paths $P^i = (S, D^i), i = 1, 2$ that share a common part. First we consider the case where there are no resource limitations. Then the QoS division policy defined in (2.1) is uniform because from (2.1) we have

$$\frac{Q_n^1}{C_n(A_n)} = \frac{Q_m^1}{C_m(A_m)} \quad \forall n, m \in \mathcal{L}(P^1) \quad \text{and} \quad \frac{Q_n^2}{C_n(A_n)} = \frac{Q_m^2}{C_m(A_m)} \quad \forall n, m \in \mathcal{L}(P^2) \quad .$$

so, if there exists $n \in \mathcal{L}(P^1 \cap P^2)$ such that $Q_n^1 < Q_n^2$, then

$$\frac{C_n(A_n)}{C_m(A_m)} Q_m^1 < \frac{C_n(A_n)}{C_m(A_m)} Q_m^2 \Longrightarrow Q_m^1 < Q_m^2 \quad \forall m \in \mathcal{L}(P^1 \cap P^2)$$

and, thus, the property in (2.3) holds. A similar argument applies for the case $Q_n^1 > Q_n^2$.

We consider now the case of QoS division with resource limitations. Let $P^1 = (S, D^1)$, $P^2 = (S, D^2)$ be two paths, $Q_{(S,D^1)}$ and $Q_{(S,D^2)}$ their end-to-end QoS requirements and $(Q_n^1)_{n \in \mathcal{L}(P^1)}$, $(Q_n^2)_{n \in \mathcal{L}(P^2)}$ the local QoS requirements obtained from the QoS division algorithm described in Figure 2.1.

If there exists $n \in \mathcal{L}(P^1 \cap P^2)$ such that $Q_n^1 \neq Q_n^2$, then let us assume without loss of generality that

$$Q_n^1 > Q_n^2 \tag{A.1}$$

We prove by contradiction that $Q_m^1 < Q_m^2 \; \forall m \in \mathcal{L}(P^1 \cap P^2)$. We assume that there exists $m \in \mathcal{L}(P^1 \cap P^2)$, $m \neq n$ such that

$$Q_m^1 < Q_m^2 \tag{A.2}$$

and proceed to find a contradiction. From (A.1) we have:

$$Q_n^1 > Q_n^2 \geq Q_n^{min}$$

Since $Q_n^1 \neq Q_n^{min}$, we have from (2.2) that

$$Q_n^1 = Q_{free}^{P^1} C_n(A_n) \tag{A.3}$$

From (2.2) we also have that $Q_m^1 \geq Q_{free}^{P^1} C_m(A_m)$, and in combination with (A.3),

$$Q_m^1 \geq \frac{C_m(A_m)}{C_n(A_n)} Q_n^1 \tag{A.4}$$

We consider two cases: $Q_m^1 = \frac{C_m(A_m)}{C_n(A_n)} Q_n^1$ and $Q_m^1 > \frac{C_m(A_m)}{C_n(A_n)} Q_n^1$.

Case 1 $Q_m^1 = \frac{C_m(A_m)}{C_n(A_n)} Q_n^1$. From (A.4)

$$Q_m^1 = \frac{C_m(A_m)}{C_n(A_n)} Q_n^1 = Q_{free}^{P^1} C_m(A_m) \tag{A.5}$$

We have

$$\begin{aligned} Q_m^2 &> Q_m^1 \quad \text{from (A.2)} \\ &= \frac{C_m(A_m)}{C_n(A_n)} Q_n^1 \quad \text{from (A.5)} \\ &> \frac{C_m(A_m)}{C_n(A_n)} Q_n^2 \quad \text{from (A.1)} \\ &\geq Q_{free}^{P^2} C_m(A_m) \quad \text{from (2.2)} \end{aligned}$$

This implies $Q_m^2 > Q_{free}^{P^2} C_m(A_m)$ and that

$$\begin{aligned} Q_m^2 &= Q_m^{min} \quad \text{from (2.2)} \\ &\leq Q_m^1 \quad \text{from (2.2)} \end{aligned}$$

which contradicts the statement $Q_m^2 > Q_m^1$ in (A.2).

Case 2 $Q_m^1 > \frac{C_m(A_m)}{C_n(A_n)} Q_n^1$. Since $Q_n^1 = Q_{free}^{P1} C_n(A_n)$ from (A.3), we have from (2.2) that

$$Q_m^1 = Q_m^{min} \tag{A.6}$$

and:

$$\begin{aligned} Q_m^2 &> Q_m^1 \quad \text{from (A.2)} \\ &> \frac{C_m(A_m)}{C_n(A_n)} Q_n^1 \quad \text{from the hypothesis of this case} \\ &> \frac{C_m(A_m)}{C_n(A_n)} Q_n^2 \quad \text{from (A.1)} \\ &\geq Q_{free}^{P2} C_m(A_m) \quad \text{from (2.2)} \end{aligned}$$

Then, from (2.2), we have $Q_m^2 = Q_m^{min}$ and from (A.6):

$$Q_m^1 = Q_m^2$$

which contradicts (A.2). Since both cases lead to contradictions, the assumption in (A.2) is refuted and Theorem 2.1 is proven. *Q.E.D.*

APPENDIX B

PROOF OF THEOREMS IN CHAPTER 3

B.1 Proof of Theorem 3.2

By Theorem 3.1, a delay $d \in \mathbb{R}$ can be guaranteed to A_f^* iff (3.2), i.e.:

$$G(t) \stackrel{\Delta}{=} F(t) \Leftrightarrow A_f^*(t \Leftrightarrow d) \geq 0, \quad \forall t \geq 0$$

where F is defined in (3.7). Given that F and A_f^* are piecewise linear and continuous, G has the same properties. Since a segment is above 0 if its ends are above 0, it follows that the schedulability condition (3.2) is equivalent to

$$G(u) \geq 0, \quad \forall u \in X_G \tag{B.1}$$

By observing that $X_G = X_F \cup (X_{A_f^*} + d)$, (3.2) is further equivalent to

$$G(u) \geq 0, \quad \forall u \in X_F \tag{B.2}$$

$$G(u) \geq 0, \quad \forall u \in X_{A_f^*} + d \tag{B.3}$$

To prove Theorem 3.2, it is sufficient to prove that (B.2) and (B.3) is equivalent to

$$d \geq u \Leftrightarrow {A_f^*}^{-1}(F(u)), \quad \forall u \in X_F \tag{B.4}$$

$$d \geq v \Leftrightarrow a, \quad \forall v \in F^{-1}(A_f^*(a)), \quad \forall a \in X_{A_f^*} \tag{B.5}$$

First we observe that (B.2) $\Leftrightarrow$ (B.4) since A_f^* is invertible on $[0, \infty)$ (because it is strictly increasing and thus bijective), and since $F(u) \in [0, \infty) \quad \forall u \in \mathbb{R}$ (we assumed the set of N flows to be schedulable).

We prove the rest of the equivalence in two parts:

(B.2),(B.3) $\Rightarrow$ (B.5) We know that (B.2),(B.3) $\Rightarrow G(t) \geq 0$, $\forall t \in \mathbb{R}$, so it is sufficient to show $G(t) \geq 0$, $\forall t \in \mathbb{R}$ implies (B.5). We prove this by contradiction. Assume that there is $a \in X_{A_f^*}$ and $v \in F^{-1}(A_f^*(a))$ such that $d < v \Leftrightarrow a$. Then $v \Leftrightarrow d > a$ and thus

$$A_f^*(v \Leftrightarrow d) > A_f^*(a) = F(v)$$

which contradicts the statement $G(t) \geq 0 \quad \forall t \in \mathbb{R}$.

(B.5) $\Rightarrow$ (B.3) We prove this by contradiction. Assume that there exists $u \in X_{A_f^*} + d$ such that $G(u) < 0$. Then there exists $a \in X_{A_f^*}$ such that $G(a + d) < 0$, which is $F(a + d) < A_f^*(a)$. But $\lim_{t \to \infty} F(t) = \infty$ (otherwise $\lim_{t \to \infty} F(t) = 0$, contradicting the stability condition (3.1)). Since F is continuous, there exists $v \in (a + d, \infty)$ such that $F(v) = A_f^*(a)$, or, equivalently,

$$\exists \quad v \in F^{-1}(A_f^*(a)) \text{ such that } a + d < v$$

which contradicts (B.5).

B.2 Proof of Theorem 3.3

We prove that $\overline{d}$ computed in Theorem 3.3 is equal to the one computed in (3.10), (3.11), (3.12). Since

$$\forall a, \quad F^{-1}_{(u_{a,1}, u_{a_2})}(A_f^*(a)) \subset F^{-1}(A_f^*(a))$$

we have that $m_y \geq m_z$. It suffices to show that, if $m_y > m_x$, then $m_y = m_z$, where m_y is defined in (3.12).

First, we show that the set $F^{-1}_{(u_{a,1},u_{a_2})}(A^*_f(a))$ is either empty or has one element. We have:

$$\begin{aligned} u_{a,2} &> m_x + a && \text{by (3.14)} \\ &\geq a + u_{a,2} \Leftrightarrow {A^*_f}^{-1}(F(u_{a,2})) && \text{by (3.16)} \end{aligned}$$

It follows that ${A^*_f}^{-1}(F(u_{a,2})) > a$, and thus $F(u_{a,2}) > A^*_f(a)$ since A^*_f is strictly increasing. Since F is continuous and concave on $(u_{a,1}, u_{a,2})$ (there is no convex point of F between $u_{a,1}$ and $u_{a,2}$ by definition of $u_{a,1}$ and $u_{a,2}$), we have that exactly one of the following holds:

$$\begin{aligned} \text{if } F(u_{a,1}) &> A^*_f(a) \text{ then } F^{-1}_{(u_{a,1},u_{a_2})}(A^*_f(a)) = \emptyset \\ \text{if } F(u_{a,1}) &\leq A^*_f(a) \text{ then } F^{-1}_{(u_{a,1},u_{a_2})}(A^*_f(a)) \text{ has one element} \end{aligned}$$

Let us define:

$$y_a = \max F^{-1}(A^*_f(a)) \Leftrightarrow a \tag{B.6}$$

$$z_a = F^{-1}_{(u_{a,1},u_{a_2})}(A^*_f(a)) \Leftrightarrow a \tag{B.7}$$

It is then sufficient to prove that, for any $a \in X^{cx}_{A^*_f}$, if $y_a > m_x$ then $y_a = z_a$. Let us assume $y_a > m_x$. By the definition of $u_{a,1}$ it follows that

$$y_a + a > u_{a,1} \tag{B.8}$$

We have

$$F(u) > A^*_f(a) \quad \forall u \in \widehat{X^{cv}}, u \geq u_{a,2} \tag{B.9}$$

because

$$\begin{aligned} u \Leftrightarrow a &\geq u_{a,2} \Leftrightarrow a \quad \text{by (B.9)} \\ &> m_x \quad \text{by (3.14)} \\ &\geq u \Leftrightarrow A_f^{*-1}(F(u)) \quad \text{by (3.16)} \end{aligned}$$

and thus $A_f^{*-1}(F(u)) > a$, or $F(u) > A_f^*(a)$ since A_f^* is strictly increasing. It follows that $F(t) > A_f^*(a) \quad \forall t \geq u_{a,2}$, since $[u_{a,2}, \infty)$ can be partitioned into intervals $[u_i, u_{i+1})$, $[u_{i+1}, u_{i+2})$, ..., for i such that $u_i = u_{a,2}$, and u_i, u_{i+1} consecutive in $\widehat{X^{cv}}$. Then, for any $t \in (u_i, u_{i+1})$, F is concave and continuous in (u_i, u_{i+1}), and thus $F(t) > \min(F(u_1), F(u_2)) > A_f^*(a)$. From $F(t) > A_f^*(a) \quad \forall t \geq u_{a,2}$, we have that $F^{-1}_{(u_{a,2},\infty)}(A_f^*(a))$ is an empty set, and thus:

$$y_a + a < u_{a,2} \tag{B.10}$$

From (B.8) and (B.10) it follows that $y_a \in F^{-1}(A_f^*(a)) \Leftrightarrow a$. Then $F^{-1}(A_f^*(a))$ has one element and, since $z_a \in F^{-1}(A_f^*(a)) \Leftrightarrow a$, it follows that $y_a = z_a$.

B.3 An Admission Control Algorithm Based on Theorem 3.3

In Figure B.1 we present an algorithm to compute the minimum guaranteeable delay, based on Theorem 3.3. We use the notation $h_{f,k} = A_f^*(a_{f,k})$, $W_l = F(u_l)$, for $u_l \in X_F^{cv}$, $V_q = F(v_q)$, for $v_q \in X_F$ and $B = c \Leftrightarrow \sum_{i=1}^{N} \rho_{i,n_i}$.

We assume that any flow envelope has $|X_{A_i^*}^{cv}| = |Y_{A_i^*}^{cv}| = O(K_1)$ convex points, $|X_{A_i^*}^{cx}| = |Y_{A_i^*}^{cx}| = O(K_2)$ concave points and $|X_{A_i^*}| = |Y_{A_i^*}| = O(K)$ total points. Since $X_F^{cx} = \cup_i X_{A_i^*}^{cv}$ and $X_F^{cv} = \cup_i X_{A_i^*}^{cx}$, we have $|X_F^{cx}| = |Y_F^{cx}| = O(K_1 N)$ and $|X_F^{cv}| = |Y_F^{cv}| = O(K_2 N)$. To compute m_x, a lookup in both Y_F^{cx} and $Y_{A_f^*}$ is needed. Observe that, if we assume $Y_{A_f^*}$ is sorted in increasing order, then

$$\forall v \in \mathbb{R}^+ \quad \text{there is a unique } h_{f,k} \in Y_{A_f^*}$$

MINIMUM_DELAY (Input: $X^{cx}_{A^*_f}$, $Y_{A^*_f}$, ρ_{f,n_f}, X_F, Y_F, X^{cv}_F,
Y^{cx}_F, B;
Output: $\overline{d}_f$)

1 **for** each $W_l \in Y^{cx}_F$
2 and each $h_{f,k} \in Y_{A^*_f}$
3 **if** $W_l \in [h_{f,k}, h_{f,k+1})$
4 **then** $x_l \leftarrow u_l \Leftrightarrow A^{*-1}_f(W_l)$
5 $m_x \leftarrow \max x_l$
6 **for** each $u_l \in X^{cv}_F$
7 and each $a_{f,k} \in X^{cx}_{A^*_f}$
8 **if** $u_l \leq m_x + a_{f,k} < u_{l+1}$ and $W_l < h_{f,k} < W_{l+1}$
9 **then** find $v_i \in X_F \cap [u_l, u_{l+1})$
such that $V_i < h_{f,k} < V_{i+1}$, where $V_i \in Y_F$
10 $z_k \leftarrow F^{-1}_{(v_i, v_{i+1})}(h_{f,k}) \Leftrightarrow a_{f,k}$
11 $m_z \leftarrow \max z_k$
12 $\overline{d} \leftarrow \max(m_x, m_z)$

Figure B.1. An $O(KN)$ algorithm for computing the minimum delay for a multi-segment envelope

such that $v \in [h_{f,k}, h_{f,k+1})$

If, in addition, Y^{cx}_F is also sorted, the lookups in steps 1 and 2 and the test in step 3 can be done in tandem, with two pointers that advance in Y^{cx}_F and $Y_{A^*_f}$, one at a time, without ever returning. It follows that the complexity for computing m_x is $O(|Y^{cx}_F| + |Y_{A^*_f}|) = O(K_1 N + K)$. To compute m_z, a search in X^{cv}_F and $X^{cx}_{A^*_f}$ is needed. Observe that, if X^{cv}_F sorted in increasing order,

$\forall a_{f,k} \in X^{cx}_{A^*_f}$ there exists a unique $u_l \in X^{cv}_F$
such that $a_{f,k} \in [u_l, u_{l+1})$

If, in addition, $X^{cx}_{A^*_f}$ is also sorted, the lookups in steps 6 and 7 and the test in step 8 can be done in tandem, with two pointers that advance in X^{cv}_F and $X^{cx}_{A^*_f}$, one at a time, without ever returning, giving a complexity of $O(K_2 N + K_2) = O(K_2 N)$. Line 9 requires a search in X_F between consecutive convex points u_l and u_{l+1}, i.e., a search among the concave points

of F in that interval. Since in the loop 6-10 there is one lookup over all convex points of F, it follows that there is a total of one lookup over all concave points of F in the same loop, giving an aggregate complexity of $O(K_1N)$. The total complexity of the algorithm is then $O(K_1N + K + K_2N + K_1N) = O(KN)$. We observe here that this is also a lower bound on the algorithm's complexity since all F's segments have to be considered in computing d, and the number of segments in F is $O(KN)$. We conclude that the computation of $\overline{d}$ cannot be further simplified.

To complete the admission control algorithm at EDF schedulers, we show how the sets X_F, Y_F, X_F^{cv} and Y_F^{cx} are updated when a flow is admitted (reservation), and a flow is terminated (release). The reservation algorithm in Figure B.2 updates the availability function, $F(t) \leftarrow F(t) \Leftrightarrow A_f^*(t \Leftrightarrow d)$, after a flow is admitted. Specifically, a new value of F is computed at each existing flexion point of F (lines 1-3). Next, all flexion points of A_f^* become new flexion points for F, and the value of F at these points is computed (lines 4-6). Finally (lines 7-10), the sets X_F, Y_F, X_F^{cv} and Y_F^{cx}, containing the new values, are sorted, as required by the MINIMUM_DELAY algorithm. To determine the complexity of this RESERVE algorithm, we observe that loops 1-3 and 4-6 can be performed in $O(KN)$ time if X_F and $X_{A_f^*}$ are sorted. Sorting X_F^{cv} and Y_F^{cx} requires $O(K_2N \log(K_2N))$, and sorting X_F and Y_F requires $O(KN \log(KN))$. It follows that the total complexity of RESERVE is $O(KN \log(KN))$. A complementary but similar algorithm updates the same sets upon a flow termination, by changing all values of F in its flexion points according to $F(t) \leftarrow F(t) + A_f^*(t \Leftrightarrow d)$, where f is the terminating flow.

B.4 Proof of Theorem 3.4

We need to prove

$$d \geq \overline{d}^s \quad \Leftrightarrow \quad A_{f,d}^s(t) \leq F(t), \quad \forall t \in \mathbb{R} \tag{B.11}$$

RESERVE (Input: d, $X_{A_f^*}$, ρ_{f,n_f}, X_F, Y_F, X_F^{cv}, Y_F^{cx}, B;
Output: X_F, Y_F, X_F^{cv}, Y_F^{cx})

for each $v_l \in X_F$
  find $a_{f,k} \in X_{A_f^*}$ such that $v_l \Leftrightarrow d \in [a_{f,k}, a_{f,k+1})$
  $W_l' \leftarrow W_l \Leftrightarrow A^*_{f_{[a_{f,k}, a_{f,k+1}]}}(v_l \Leftrightarrow d)$
for each $a_{f,k} \in X_{A_f^*}$
  find $v_l \in X_F$ such that $d + a_{f,k} \in [v_l, v_{l+1})$
  $W_{f,k} \leftarrow F_{[v_l, v_{l+1})}(d + a_{f,k}) \Leftrightarrow A_f^*(a_{f,k})$
$Y_F^{cx} \leftarrow \{W_l' | u_l \in X_F^{cv}\} \cup \{W_{f,k} | a_{f,k} \in X_{A_f^*}^{cx}\}$; sort Y_F^{cx}
$Y_F \leftarrow \{W_l'\} \cup \{W_{f,k}\}$; sort Y_F
$X_F^{cv} \leftarrow X_F^{cv} \cup (X_{A_f^*}^{cx} + d)$; sort X_F^{cv}
$X_F \leftarrow X_F \cup (X_{A_f^*} + d)$; sort X_F

Figure B.2. An $O(KN \log(KN))$ algorithm for resource reservation for flow f

It suffices to restrict ourselves to $t \in \mathcal{D}$ since, by definition, the cover envelope has all its flexion points in the fixed set $\mathcal{D}$. Given (3.22) and (3.20), the following two statements yield (B.11) for $t \in \mathcal{D}$:

$$d \geq m_x \quad \Leftrightarrow \quad A_{f,d}(u) \leq F(u), \quad \forall u \in \mathcal{D} \tag{B.12}$$

Given $d \geq m_x$,

$$d \geq m_z \quad \Leftrightarrow \quad A^+_{f,d}(u) \leq F(u), \quad \forall u \in \mathcal{D} \tag{B.13}$$

Proof of (B.12). From (3.23) and the definition of $A_{f,d}$, (3.17) it is sufficient to prove

$$d \geq u \Leftrightarrow {A_f^*}^{-1}(F(u)) \quad \Leftrightarrow \quad A_{f,d}(u) \leq F(u), \quad \forall u \in \mathcal{D}$$

This follows from

$${A_f^*}^{-1}(F(u)) \geq u \Leftrightarrow d \quad \Leftrightarrow \quad F(u) \geq A_f^*(u \Leftrightarrow d) = A_{f,d}(u)$$

since A_f^* is increasing and bijective and from (3.17).

Proof of (B.13). Given $d \geq m_x$, it is sufficient to prove

$$d \geq m_z \quad \Leftrightarrow \quad A^+_{f,d}(u) \leq F(u), \quad \forall u \in \mathcal{D} \text{ such that } A^+_{f,d}(u) > 0$$

Based on (3.24) and (3.21), it is sufficient to show that

$$d \quad \geq \quad \min(u_{i_a} \Leftrightarrow {A^*_{f,a}}^{-1}(F(u_{i_a})), u_{i_a+1} \Leftrightarrow a) \quad \Leftrightarrow$$

$$A^*_{f,a}(u_j \Leftrightarrow d) \leq F(u_j), \quad \forall a \in X^{cx}_{A^*_f}, \tag{B.14}$$

$$\text{where} \quad u_j \leq a + d < u_{j+1} \tag{B.15}$$

Let us consider an arbitrary $a \in X^{cx}_{A^*_f}$. We are given $d \geq m_x$, so $d + a \geq m_x + a$, and, from (3.25) and (B.15), it follows that $u_j \leq u_{i_a}$. It follows that we have two cases, $u_j = u_{i_a}$ and $u_j > u_{i_a}$, which are considered separately in the following.

Case $u_j = u_{i_a}$. We have to prove

$$d \geq \min(u_j \Leftrightarrow {A^*_{f,a}}^{-1}(F(u_j)), u_{j+1} \Leftrightarrow a) \quad \Leftrightarrow \tag{B.16}$$

$$A^*_{f,a}(u_j \Leftrightarrow d) \leq F(u_j) \tag{B.17}$$

From (B.15) we have $d < u_{j+1} \Leftrightarrow a$. It follows that (B.16) $\Leftrightarrow d \geq u_j \Leftrightarrow {A^*_{f,a}}^{-1}(F(u_j))$. But the latter is equivalent to (B.17) since $A^*_{f,a}$ is increasing.

Case $u_j > u_{i_a}$. We have to prove

$$d \geq \min(u_{i_a} \Leftrightarrow {A^*_{f,a}}^{-1}(F(u_{i_a})), u_{i_a+1} \Leftrightarrow a) \quad \Leftrightarrow \tag{B.18}$$

$$A^*_{f,a}(u_j \Leftrightarrow d) \leq F(u_j) \tag{B.19}$$

From the hypothesis of this case, $u_j \geq u_{i_a+1}$. Also, from (B.15), $u_j \leq a + d < u_{j+1}$, we have $u_{i_a+1} \leq a + d$, and thus, (B.18) is true. Thus, to prove (B.18) $\Leftrightarrow$ (B.19), it is sufficient

to prove that (B.19) is true for any d, $d \geq u_{i_a+1} \Leftrightarrow a$. Given that A^*_f is increasing, we have

$$\begin{aligned}
F(u_j) &\geq A^*_f(u_j \Leftrightarrow m_x) \quad \text{by } m_x \geq u_j \Leftrightarrow A^{*-1}_f(F(u_j)), \text{(3.23)} \\
&\geq A^*_f(u_{i_a+1} \Leftrightarrow m_x) \quad \text{by } u_j \geq u_{i_a+1} \\
&\geq A^*_f(a) \quad \text{by } m_x + a < u_{i_a+1}, \text{(3.25)} \\
&= A^*_{f,a}(a) \quad \text{by (3.19)} \\
&\geq A^*_{f,a}(u_j \Leftrightarrow d) \quad \text{by } u_j \leq a + d \quad \text{by (B.15)}
\end{aligned}$$

B.5 Analysis of Non-Preemptive EDF Admission Control

Let us examine the problem of determining the minimum delay of a flow f having envelope A^*_f, with the condition that f is schedulable along with a set of flows $(A^*_i, d_i)_{1 \leq i \leq N}$ at an NPEDF scheduler with capacity c. As in previous sections we define the work availability function

$$F(t) = ct \Leftrightarrow \sum_{i=1}^{N} A^*_i(t \Leftrightarrow d_i) \Leftrightarrow (\max_{\substack{1 \leq i \leq N \\ d_i > t}} p_i)\mathbf{1}(d_1 \leq t < d_N)$$

Let us denote the sets of flows $\mathcal{N} = \{1, ..N\}$ and $\mathcal{N}' = \{1, ..N\} \cup \{f\}$. Suppose that $d_1 \leq d_f \leq d_N$. It is easy to see that the condition of schedulability of flows in $\mathcal{N}'$ is:

$$ct \geq \sum_{i=1}^{N+1} A^*_i(t \Leftrightarrow d_i) + (\max_{\substack{1 \leq i \leq N+1 \\ d_i > t}} p_i)\mathbf{1}(d_1 \leq t < d_{N+1}) \tag{B.20}$$

We have:

$$\begin{aligned}
\max_{\substack{i \in \mathcal{N}' \\ t < d_i}}(p_i) &= \max(\max_{\substack{i \in \mathcal{N} \\ t < d_i}} p_i, p_f \mathbf{1}(d_1 \leq t < d_f)) \\
&= \max_{\substack{i \in \mathcal{N} \\ t < d_i}}(p_i) + (p_f \mathbf{1}(d_1 \leq t < d_f) \Leftrightarrow \max_{\substack{i \in \mathcal{N} \\ t < d_i}}(p_i))^+
\end{aligned}$$

$$= \max_{\substack{i\in\mathcal{N}\\ t<d_i}}(p_i) + (p_f \Leftrightarrow \max_{\substack{i\in\mathcal{N}\\ t<d_i}}(p_i))^+ \mathbf{1}(d_1 \le t < d_f)$$

and thus:

$$\max_{\substack{i\in\mathcal{N}'\\ t<d_i}}(p_i)\mathbf{1}(d_1 \le t < d_N) =$$
$$\max_{\substack{i\in\mathcal{N}\\ t<d_i}}(p_i)\mathbf{1}(d_1 \le t < d_N) + (p_f \Leftrightarrow \max_{\substack{i\in\mathcal{N}\\ t<d_i}}(p_i))^+ \mathbf{1}(d_1 \le t < d_f)$$

where $\mathbf{1}(d_1 \le t < d_N)\mathbf{1}(d_1 \le t < d_f) = \mathbf{1}(d_1 \le t < d_f)$ from our assumption $d_1 \le d_f \le d_N$. The condition (B.20) becomes:

$$F(t) \ge A_f^*(t \Leftrightarrow d_f) + (p_f \Leftrightarrow G(t))^+ \mathbf{1}(d_1 \le t < d_f) \tag{B.21}$$

where

$$G(t) = \max_{\substack{1\le i\le N\\ d_i>t}} p_i$$

Since $G(t)$ is a non-increasing, piecewise constant function on interval (d_1, d_N), it follows that $(p_f \Leftrightarrow G(t))^+ \mathbf{1}(d_1 \le t < d_f)$ is a piecewise constant function, non-decreasing on interval (d_1, d_f). Assuming all envelope functions $(A_i^*)_{1\le i\le N}$ and A_f^* to be piecewise linear (as is the case with all multiple-segment envelopes), it follows that the feasibility function F is also piecewise linear. The problem of determining the minimum delay $\overline{d}_f$ that satisfies (B.21) reduces to one of fitting the function

$$\hat{A}_f(t) \triangleq A_f^*(t \Leftrightarrow d_f) + (p_f \Leftrightarrow G(t))^+ \mathbf{1}(d_1 \le t < d_f)$$

below F, at its leftmost position, as in Figure B.3.

We can see a problem here. Since $\hat{A}_f$ is not necessarily an increasing function (it can actually decrease in the neighborhood of d_f), it is no longer the case (as in previous sections) that if $\hat{A}_f$ is schedulable for a d_f then $\hat{A}_f$ is schedulable for all $d > d_f$. In the example

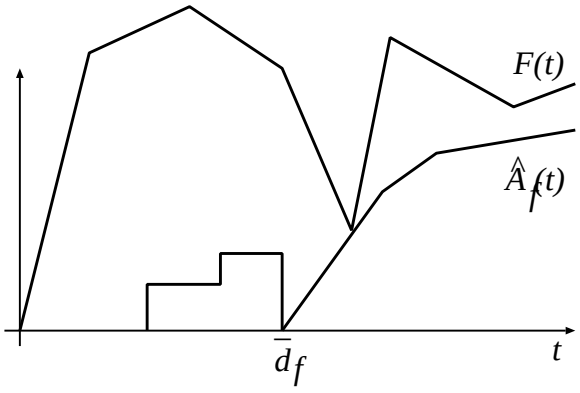


Figure B.3. Determining minimum delay for $\hat{A}_f(t)$

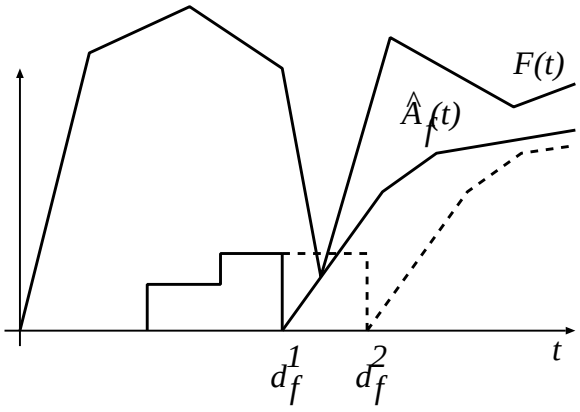


Figure B.4. Non-compactness of admissible delay range

in Figure B.4, $\hat{A}_f$ is schedulable for d_f^1, but not schedulable for $d_f^2 > d_f^1$. Consequently, instead of having a compact interval of schedulability for $\hat{A}_f$, $[\overline{d}_f, \infty)$, it is possible that the set of schedulable delay values consists of a union of separate intervals. This complicates the schedulability computations, and is not covered by any existing or emerging flow setup protocol standards such as RSVP/Guaranteed-Services or ATM Signaling.

APPENDIX C

PROOF OF THEOREM IN CHAPTER 5

C.1 Derivation of Asymptotic Load Ratio

We start by establishing the following:

Lemma C.1 *For any network,*

1.

$$\lim_{p\to 0} L_i^*/Er(\psi_i, C_i) = 1 \quad \forall i \in \mathcal{L} \tag{C.1}$$

2.

$$\lim_{p\to 0} B_k^*/\sum_{j\in r_k} L_j^* = 1 \quad \forall k \in \mathcal{K} \tag{C.2}$$

where $\psi_i = \sum_{k\in\mathcal{K}, i\in r_k} \phi_k \quad \forall i \in \mathcal{L}$, as defined in (5.4), and $\phi_k = \nu_k \rho$, as defined in (5.2).

We observe that a result similar to (C.1) was established by Whitt in [102], Corollary 2.3, point (ii):

Proposition C.1 *For any network,*

$$\lim_{\rho\to 0} L_i^*/Er(\psi_i, C_i) = 1 \quad \forall i \in \mathcal{L} \tag{C.3}$$

The result in (C.1) is stronger than (C.3) in the sense that ρ approaches 0 slower than p (e.g., $p \ll 1$ and $\rho > 1$).

Proof.

1. Let $p = P^*(\rho) = \sum_{k \in \mathcal{K}} \nu_k B_k^*$ as in (5.10). It follows that:

$$p > \nu_k B_k^* = \nu_k (1 \Leftrightarrow \prod_{j \in r_k} (1 \Leftrightarrow L_j^*)) \quad \text{by (5.7)}$$

which implies $\lim_{p \to 0} \prod_{j \in r_k} (1 \Leftrightarrow L_j^*) = 0$, and thus, $\lim_{p \to 0} L_j^* = 0$, $\forall j \in \mathcal{L}$ since $0 \leq L_j^* \leq 1$, as by (5.8), L_j^* is the result of an Erlang function, which has values in $[0, 1]$. It follows that

$\lim_{p \to 0} \prod_{i \in r_k - \{j\}} (1 \Leftrightarrow L_i^*) = 1$, and thus

$$\begin{aligned} 1 &= \lim_{p \to 0} L_j^* / Er(\sum_{k \in \mathcal{K}, j \in r_k} \phi_k, C_j) \quad \text{by (5.8)} \\ &= \lim_{p \to 0} L_j^* / Er(\psi_j, C_j) \quad \text{by (5.4)} \end{aligned}$$

2. We have:

$$\sum_{j \in r_k} L_j^* \geq 1 \Leftrightarrow \prod_{j \in r_k} (1 \Leftrightarrow L_j^*) \geq \sum_{j \in r_k} L_j^* \Leftrightarrow \sum_{i < j \in r_k} L_i^* L_j^*$$

since $L_j^* \geq 0 \; \forall j \in \mathcal{L}$. Thus,

$$1 \geq \frac{1 \Leftrightarrow \prod_{j \in r_k} (1 \Leftrightarrow L_j^*)}{\sum_{j \in r_k} L_j^*} \geq 1 \Leftrightarrow \frac{\sum_{i < j \in r_k} L_i^* L_j^*}{\sum_{j \in r_k} L_j^*} \tag{C.4}$$

We have

$$\lim_{p \to 0} \frac{\sum_{i < j \in r_k} L_i^* L_j^*}{\sum_{j \in r_k} L_j^*} = 0$$

because $0 \leq \frac{\sum_{i < j \in r_k} L_i^* L_j^*}{\sum_{j \in r_k} L_j^*} \leq \sum_{j \in r_k} L_j^*$, and from $\lim_{p \to 0} \sum_{j \in r_k} L_j^* = 0$. Thus, taking the limit $p \to 0$ in (C.4), we have

$$\lim_{p \to 0} \frac{1 \Leftrightarrow \prod_{j \in r_k} (1 \Leftrightarrow L_j^*)}{\sum_{j \in r_k} L_j^*} = 1$$

which gives $\lim_{p \to 0} B_k^* / \sum_{j \in r_k} L_j^* = 1$ from (5.7).

Q.E.D.

We can now prove the derivation of the Asymptotic Load Ratio:

Proposition 5.1 *If all link capacities are the same in both networks,* $C_i = C$*,* $\forall i \in \mathcal{L}_q$*,* $q = 1, 2$*, then the limit for the Low-Blocking Load Ratio as* $C \to \infty$*, named Asymptotic Load Ratio is:*

$$A(N_1, N_2) = \lim_{C \to \infty} A_0(N_1, N_2) = M_2/M_1$$

Proof. The limit of class k flow blocking probability as $p \to 0$ is:

$$\begin{aligned} 1 &= \lim_{p \to 0} B_k^* / (1 \Leftrightarrow \prod_{i \in r_k} (1 \Leftrightarrow L_i^*)) \quad \text{by (5.7)} && \text{(C.5)} \\ &= \lim_{p \to 0} B_k^* / \sum_{i \in r_k} L_i^* \quad \text{by (C.2)} && \text{(C.6)} \\ &= \lim_{p \to 0} B_k^* / \sum_{i \in r_k} Er(\psi_i, C_i) \quad \text{by (C.1)} && \text{(C.7)} \end{aligned}$$

The limit of network blocking probability becomes:

$$\begin{aligned} 1 &= \lim_{p \to 0} P^* / \sum_{k \in \mathcal{K}} \nu_k B_k^* \qquad \text{by (5.10)} && \text{(C.8)} \\ &= \lim_{p \to 0} P^* / \sum_{k \in \mathcal{K}} \nu_k \sum_{i \in r_k} Er(\psi_i, C_i) \qquad \text{by (C.7)} && \text{(C.9)} \\ &= \lim_{p \to 0} P^* / \sum_{i \in \mathcal{L}} Er(\psi_i, C_i) \sum_{k \in \mathcal{K}, i \in r_k} \nu_k \qquad \text{by rearranging terms} && \text{(C.10)} \\ &= \lim_{p \to 0} P^* / \sum_{i \in \mathcal{L}} \eta_i Er(\eta_i \rho, C_i) \qquad \text{by (5.3)} && \text{(C.11)} \end{aligned}$$

Let all link capacities be equal, $C_i = C$, and let M be the maximum link factor,

$$M = \max_{i \in \mathcal{L}} \eta_i$$

$\mathcal{L}$ can thus be partitioned in $\mathcal{L}'$, the set of links with maximum link factor, and $\mathcal{L}'' = \mathcal{L} \Leftrightarrow \mathcal{L}'$:

$$\mathcal{L}' = \{i | i \in \mathcal{L}, \eta_i = M\}, \quad \mathcal{L}'' = \{i | i \in \mathcal{L}, \eta_i < M\}$$

and let n and m be the number of links in the two sets,

$$n = |\mathcal{L}'|, \quad m = |\mathcal{L}''|$$

By factoring out the term with the maximum link factor in (C.11) we have:

$$1 = \lim_{p \to 0} \frac{P^*(\rho)}{MEr(M\rho, C)(n + \sum_{i \in \mathcal{L}''} \frac{\eta_i Er(\eta_i \rho, C)}{MEr(M\rho, C)})}$$

and we define $\overline{P^*}(\rho)$ as:

$$\overline{P^*}(\rho) = MEr(M\rho, C)(n + \sum_{i \in \mathcal{L}''} \frac{\eta_i Er(\eta_i \rho, C)}{MEr(M\rho, C)})$$

Consider the networks N_1 and N_2 having all link capacities $C_{iq} = C$, $\forall q = 1, 2$. By definition (5.5), the Network Load Ratio $R(p)$ is given by the equation $P_1^*(R(p)\rho) = P_2^*(\rho) = p$. Thus, the limit as $p \to 0$ of the Network Load Ratio, $A_0 = \lim_{p \to 0} R(p)$, named Low-Blocking Load Ratio, is given by the equation:

$$\overline{P_1^*}(A_0 \rho) = \overline{P_2^*}(\rho) = p$$

which is equivalent to:

$$\begin{aligned} M_1 Er(M_1 A_0 \rho, C)(n_1 + \sum_{i \in \mathcal{L}_1''} \frac{\eta_{i1} Er(\eta_{i1} \rho, C)}{M_1 Er(M_1 \rho, C)}) &= \\ M_2 Er(M_2 \rho, C)(n_2 + \sum_{i \in \mathcal{L}_2''} \frac{\eta_{i2} Er(\eta_{i2} \rho, C)}{M_2 Er(M_2 \rho, C)}) &= p \end{aligned} \tag{C.12}$$

We want to find the limit $A = \lim_{C \to \infty} A_0$, where A_0 is the solution of the above equation. Since the equation cannot be solved analytically, we proceed to find the limit of the equation first, and then solve it. Observe first that

$$0 < \frac{\eta_{iq} Er(\eta_{iq} \rho, C)}{M_q Er(M_q \rho, C)} < 1, \quad i \in \mathcal{L}_q'' \quad i = 1, 2$$

because $\eta_{iq} < M_q$ for $i \in \mathcal{L}_q''$. Then, (C.12) becomes:

$$aEr(bA_0\rho, C) = Er(\rho, C) = p \tag{C.13}$$

where $b = M_1/M_2$, and $M_1 n_1/M_2(m_2 + n_2) < a < M_1(n_1 + m_1)/M_2 n_2$. We have the following limits:

$$\lim_{C\to\infty} Er(\rho, C) = \begin{cases} 1 \Leftrightarrow C/\rho & \text{for } \rho > C \\ 0 & \text{for } \rho \leq C \end{cases} \quad \text{(for example, see [97])} \tag{C.14}$$

$$\lim_{C\to\infty} \frac{Er(\rho, C)}{(\rho^C/C!)e^{-\rho}} = \begin{cases} 1 & \text{for } \rho < C \\ 2 & \text{for } \rho = C \end{cases} \quad \text{(for example, see [1], 6.5.34)} \tag{C.15}$$

In (C.13) we take $p \to 0$, and thus $Er(\rho, C) \to 0$ and $Er(bA_0\rho, C) \to 0$, and by (C.14), both networks N_1 and N_2 are in underload conditions: $\rho \leq C$ and $bA_0\rho \leq C$. It follows that, when taking the limit $C \to \infty$ in (C.13), (C.15) can be applied to both sides of (C.13):

$$ad\frac{(bA\rho)^C}{C!}e^{-bA\rho} = \frac{\rho^C}{C!}e^{-\rho}$$

where $d = 1, 2$ or $1/2$. By taking the log of both sides:

$$\frac{\log ad}{C} + \log(bA) + \frac{\rho}{C}(1 \Leftrightarrow bA) = 0$$

The unique solution is $A = 1/b = M_2/M_1$ for $C \to \infty$. *Q.E.D.*

BIBLIOGRAPHY

[1] Abramowitz, M., and Segun, I. *Handbook of Mathematical Functions*. National Bureau of Standards, 1964.

[2] Alles, A. ATM Internetworking. Cisco Systems, Inc., May 1995.

[3] Anderson, David A. Metascheduling for Continuous Media. *ACM Transactions on Computer Systems 11*, 3 (August 1993).

[4] Anick, D., Mitra, D., and Sondhi, M. M. Stochastic theory of a data-handling system with multiple sources. *Bell System Technical Journal 61* (1982).

[5] Armitage, G. Support for Multicast over UNI 3.0/3.1 based ATM Networks. Internet RFC 2022, November 1996.

[6] Atkinson, R. Default IP MTU for use over ATM AAL5. Internet RFC1626, May 1994.

[7] ATM Forum. ATM User-Network Interface Specification. Version 3.1, September 1994.

[8] Ballardie, Tony, Francis, Paul, and Crowcroft, Jon. Core Based Trees. In *ACM SIGCOMM '93* (September 1993), pp. 85–95.

[9] Berger, L. RSVP over ATM Implementation Guidelines. Work in progress, draft-ietf-issll-atm-imp-guide-04.txt, Apr 1998.

[10] Berger, L. RSVP over ATM Implementation Requirements. Work in progress, draft-ietf-issll-atm-imp-req-03.txt, Apr 1998.

[11] Birman, Alex, Firoiu, Victor, Guérin, Roch, and Kandlur, Dilip. Support for RSVP-based Services over ATM Networks. In *IEEE Global Internet* (1996), pp. 10–15.

[12] Borden, M., Crawley, E., Davie, B., and Batsell, S. Integration of Real-time Services in an IP-ATM Network Architecture . Internet RFC1821, August 1995.

[13] Borden, M., and Garrett, M. Interoperation of Controlled-Load and Guaranteed Services with ATM. Work in progress, draft-ietf-issll-atm-mapping-06.txt, Apr 1998.

[14] Braden, R., Zhang, L., Berson, S., Herzog, S., and Jamin, S. Resource ReSerVation Protocol (RSVP) - Version 1, Functional Specification. Request For Comments (Proposed Standard) RFC 2205, Internet Engineering Task Force, September 1997.

[15] Brady, P. T. A statistical analysis of on-off patterns in 16 conversations. *Bell System Technical Journal 47* (January 1968), 73–91.

[16] Brown, C. Baseline Text for MPOA. ATM Forum 95-0824R1, July 1995.

[17] Callon, R. Integrated PNNI for Multi-Protocol Routing. ATM Forum 94-0789, September 1994.

[18] Callon, R., and Salkewicz, B. An Outline of Integrated PNNI for IP Routing. ATM Forum 95-0649, August 1995.

[19] Canserver, D. NHRP Protocol Applicability Statement. Internet draft-ietf-rolc-nhrp-appl-01.txt, March 1995.

[20] Casner, S. Major MBONE Routers and Links. ftp://ftp.isi.edu/mbone/mbone-topology.ps, May 1994.

[21] Chang, Cheng-Shang. Stability, Queue Length and Delay of Deterministic and Stochastic Queuing Networks. *IEEE Transactions on Automatic Control 39*, 5 (May 1994).

[22] Cole, R.G., Shur, D.H., and Villamizar, C. IP over ATM: A Framework Document. Internet draft-ietf-ipatm-framework-doc-04.ps, July 1995.

[23] Crawley, E., Berger, L., Berson, S., Baker, F., Borden, M., and Krawczyk, J. A Framework for Integrated Services and RSVP over ATM. Internet draft-ietf-issll-atm-framework-04.txt, May 1998.

[24] Crawley, E., Nair, R., Rajagopalan, B., and Sandick, H. A Framework for QoS-based Routing in the Internet. Internet draft, work in progress, draft-ietf-qosr-framework-05.txt, May 1998.

[25] Cruz, R. Quality of Service Guarantees in Virtual Circuit Switched Networks. *IEEE JSAC 13*, 6 (Aug 1995).

[26] Cruz, Rene L. A Calculus for Network Delay, Part I: Network Elements in Isolation. *IEEE Transactions on Information Theory 37*, 1 (January 1991).

[27] de Veciana, G., Courcoubetis, C., and Walrand, J. Decoupling Bandwidths for Networks. Tech. Rep. M93/50, UCB/ERL, June 1993.

[28] Deering, Stephen E., and Cheriton, David R. Multicast Routing in Datagram Internetworks and Extended LANs. *ACM Transactions on Computer Systems 8*, 2 (May 1990), 85–110.

[29] Delgrossi, L., and Berger, L. Internet STream Protocol Version 2+. Internet draft-ietf-st2-spec-02.txt, March 1995.

[30] Demers, A., Keshav, S., and Shenker, S. Analysis and Simulation of a Fair Queuing Algorithm. *Computer Communication Review (ACM SIGCOMM'89) 19*, 4 (1989).

[31] Doar, Matthew, and Leslie, Ian. How Bad is Naïve Multicast Routing ? In *IEEE INFOCOM '93* (1993), pp. 82–89.

[32] Effelsberg, Wolfgang, and Müller-Menrad, Eberhard. Dynamic Join and Leave for Real-Time Multicast. Tech. Rep. TR-93-056, Tenet Group, U.C. Berkeley and ICSI, October 1993.

[33] Elwalid, Anwar I., and Mitra, Debasis. Effective Bandwidth of General Markovian Traffic Sources and Admission Control of High Speed Networks. *Transactions on Networking 1*, 3 (June 1993).

[34] Feldmann, A., Rexford, J., and Caceres, R. Reducing Overhead in Flow-Switched Networks: An Empirical Study of Web Traffic. In *IEEE INFOCOM* (1998).

[35] Ferrari, Domenico, and Verma, Dinesh. A Scheme for Real-Time Channel Establishment in Wide-Area Networks. *IEEE Journal on Selected Areas in Communications 8*, 3 (April 1990).

[36] Firoiu, Victor, Kurose, Jim, and Towsley, Don. Efficient Admission Control of Piece-Wise Linear Traffic Envelopes at EDF Schedulers. To appear in *IEEE/ACM Transactions on Networking*. (A short version was presented at *IEEE INFOCOM'97*.).

[37] Firoiu, Victor, Kurose, Jim, and Towsley, Don. Performance Evaluation of ATM Shortcut Connections in Overlaid IP/ATM Networks. Submitted to IEEE INFOCOM.

[38] Firoiu, Victor, and Towsley, Don. Call Admission and Resource Reservation for Multicast Sessions. In *IEEE INFOCOM* (1996), pp. 94–101.

[39] Frederik, R. *nv*. Manual Pages, Xerox Palo Alto Research Center.

[40] Georgiadis, L., Guérin, R., and Parekh, A.K. Optimal multiplexing on a single link: Delay and buffer requirements. *IEEE Trans. Infor. Theory 43*, 5 (September 1997), 1518–1535. (A short version was presented at INFOCOM'94).

[41] Georgiadis, L., Guérin, R., Peris, V., Rajan, R., and Shenker, S. Issues in the Provision of Guaranteed Delay Service and the Use of Rate-Controlled Service Disciplines. Private Communication, July 1995.

[42] Georgiadis, L., Guérin, R., Peris, V., and Sivarajan, K. N. Efficient network QoS provisioning based on per node traffic shaping. *IEEE/ACM Trans. Networking 4*, 4 (August 1996), 482–501.

[43] Gleeson, B., and Armitage, G. Issues sorrounding a new encapsulation for IP over ATM. Internet draft-armitage-ipatm-encaps-02.txt, June 1995.

[44] Goto, Y. Session Identity Notification Protocol. Internet draft-goto-sinp-00.txt, July 1995.

[45] Graf, M., and Stüttgen, H.J. An Internetworking Architecture for Multimedia Communication over Heterogeneous Networks. In *Dagstuhl Workshop on Communication* (1993).

[46] Greenberg, A., and Srikant, R. Computational Techniques for Accurate Performance Evaluation of Multirate, Multihop Communication Networks. *IEEE/ACM Transactions on Networking 5*, 2 (April 1997), 266–290.

[47] Grossglauser, M., Keshav, S., and Tse, D. RCBR: A Simple and Efficient Service for Multiple Time-Scale Traffic. In *ACM SIGCOMM* (1995).

[48] Guérin, R., Ahmadi, H., and Naghshineh, M. Equivalent Capacity and Its Application to Bandwidth Allocation in High-Speed Networks. *IEEE Journal on Selected Areas in Communications 9* (September 1991), 968–981.

[49] Hagsand, O., and Pink, S. ATM as a Link in an ST-2 Internet. In *4th International Workshop, NOSSDAV* (1993).

[50] Heinanen, J. Multiprotocol Encapsulation over ATM Adaptation Layer 5. Internet RFC1483, July 1993.

[51] ITU-T. B-ISDN Functional Architecture. I.327, Mar 1993.

[52] Jacobson, V., and McCanne, S. *vat*. Manual Pages, Lawrence Berkeley Laboratory, Berkeley, CA.

[53] Jain, R. *The Art of Computer Systems Performance Analysis*. John Wiley & Sons, 1991.

[54] Jamison, J., and Wilder, R. vBNS: The Internet Fast Lane for Research and Education. *IEEE Communications Magazine 35*, 1 (Jan 1997).

[55] Kandlur, D., Shin, K., and Ferrari, D. Real-Time Communications in Multihop Networks. *IEEE Transactions on Parallel and Distributed Systems 5*, 10 (October 1994).

[56] Katsube, Y., and Nagami, K. Mapping of IP Flow to Datalink Layer Connection. Internet draft-katsube-flow-mapping-dl-conn-00.txt, May 1995.

[57] Katsube, Y., Nagami, K., and Esaki, H. Router Architecture Extensions for ATM: Overview. Internet draft-katsube-router-atm-overview-02.txt, March 1996.

[58] Kelly, F.P. Blocking Probabilities in Large Circuit-Switched Networks. *Adv. Appl. Prob. 18* (1986).

[59] Kelly, F.P. Loss Networks. *The Annals of Applied Probability 1*, 3 (1991).

[60] Keshav, S. *An Engineering Aprroach to Computer Networking*. Addison-Wesley, 1997.

[61] Knightly, E., Wrege, D., Liebeherr, J., and Zhang, H. Fundamental Limits and Tradeoffs of Providing Deterministic Guarantees to VBR Video Traffic. In *ACM SIGMETRICS'95* (1995).

[62] Knightly, E., and Zhang, H. Traffic Characterization and Switch Utilization using a Deterministic Bounding Interval Dependent Traffic Model. In *IEEE INFOCOM* (1995).

[63] Knightly, E., and Zhang, H. D-BIND: An Accurate Traffic Model for Providing QoS Guarantees to VBR Traffic. *IEEE/ACM Transactions on Networking 5*, 2 (April 1997).

[64] Laubach, M. Classical IP and ARP over ATM. Internet RFC1577, January 1994.

[65] Liebeherr, J., Wrege, D., and Ferrari, D. Exact Admission Control for Networks with Bounded Delay Services. Tech. Rep. CS-94-29, University of Virginia, 1994.

[66] Liebeherr, J., Wrege, D., and Ferrari, D. Exact Admission Control for Networks with Bounded Delay Services. *IEEE/ACM Transactions on Networking* (December 1996).

[67] Lin, S., and McKeown, N. A Simulation Study of IP Switching. In *ACM SIGCOMM* (1997).

[68] Liu, C., and Layland, J. Scheduling Algorithms for Multiprogramming in a Hard-Real-Time Environment. *Journal of ACM 20*, 1 (January 1973).

[69] Louth, G. *Stochastic Networks: Complexity, Dependence and Routing*. PhD thesis, University of Cambridge, 1990.

[70] Luciani, J., Katz, D., Piscitello, D., and Cole, B. NBMA Next Hop Resolution Protocol (NHRP). Internet draft-ietf-rolc-nhrp-11.txt, 1997.

[71] Lynch, N. *Distributed Algorithms*. Morgan Kaufmann, 1996.

[72] Milliken, W. Integrated Services IP Multicasting over ATM. Internet draft-milliken-ipatm-services-00.txt, July 1995.

[73] Milliken, W. IP Multicasting over ATM: System Architecture Issues. Internet draft-ietf-ipatm-arch-00.txt, July 1995.

[74] Moy, J. Multicast Extensions to OSPF. Internet RFC 1584, Mar 1994.

[75] Newman, P., Lyon, T., and Minshall, G. Flow Labelled IP: a Connectionless Approach to ATM. In *IEEE INFOCOM* (1996).

[76] Noronha Jr., Ciro A., and Tobagi, Fouad A. Optimum Routing of Multicast Streams in Communications Networks. Tech. Rep. CSL-TR-94-618, Department of Electrical Engineering and Computer Science, Stanford University, April 1994.

[77] Ohta, M., Esaki, H., and Nagami, K. Conventional IP over ATM. Internet draft-ohta-ip-over-atm-02.txt, March 1995.

[78] O.Rose. MPEG coded video traces. ftp://ftp-info3.informatik.uni-wuerzburg.de/-pub/MPEG/.

[79] O.Rose. Statistical Properties of MPEG Video Traffic and Their Impact on Traffic Modelling in ATM Systems. Tech. Rep. TR 101, University of Wuerzburg, Feb 1995.

[80] Parekh, A. K. *A Generalized Processor Sharing Approach to Flow Control in Integrated Services Networks*. PhD thesis, Department of Electrical Engineering and Computer Science, MIT, February 1992.

[81] Perez, M., Liaw, F., Mankin, A., Hoffman, E., Grossman, D., and Malis, A. ATM Signaling Support for IP over ATM. Internet RFC1755, February 1995.

[82] Perkins, D., and Liaw, F. Beyond Classical IP - Integrated IP and ATM Architecture Overview. ATM Forum 94-0935, September 1994.

[83] Preparata, F.P., and Shamos, M.I. *Computational Geometry*. Springer-Verlag, 1985.

[84] R. Cherukuri and D. Dykeman (eds.) and M. Goguen (chair). ATM PNNI Draft Specification. ATM Forum 94-0471, November 1995.

[85] Rekhter, Y., and Cole, B. Alternatives for router-to-router NHRP. Submission to rolc mailing list, July 1995.

[86] Rekhter, Y., and Farinacci, D. Support for Sparse Mode PIM over ATM. Internet draft-rekhter-pim-atm-00.txt, Feb 1996.

[87] Rekhter, Y., and Kandlur, D. IP Architecture Extensions for ATM. Internet draft-rekhter-ip-atm-architecture-01.txt, July 1995.

[88] Rekhter, Y., and Villamizar, C. Inter-Domain Routing over ATM Networks. Internet draft-rekhter- idr-over-atm-00.txt, February 1995.

[89] Ross, K.W. *Multiservice Loss Models for Broadband Telecommunication Networks*. Springer, 1995.

[90] Sahu, Sambit, Firoiu, Victor, Towsley, Don, and Kurose, Jim. Traffic Models and Admission Control for Variable Bit Rate Continuous Media Transmission with Deterministic Service. In *SPIE Symposium on Voice, Video and Data Communications* (1998).

[91] Samudra, Pradeep. Draft of UNI Signaling 4.0. ATM Forum 94-1018 R4, July 1995.

[92] Sathaye, S. Traffic Management Specification 4.0. ATM Forum 95-0013R7, July 1995.

[93] Schulzrinne, Henning. Voice Communication Accross the Internet: A Network Voice Terminal. Tech. Rep. TR-92-50, Department of Computer Science, University of Massachusetts, Amherst, July 1992.

[94] Shenker, S., Partridge, C., and Guérin, R. Specification of guaranteed quality of service. Request For Comments (Proposed Standard) RFC 2212, Internet Engineering Task Force, September 1997.

[95] Smith, T., and Armitage, G. IP Broadcast over ATM Networks. Internet draft-smith-ipatm-bcast- 01.txt, July 1995.

[96] Spiegel, E. M. Signalling of Individual QoS Parameters. ATM Forum 95-1009R1, October 1995.

[97] Syski, R. *introduction to Congestion Theory in Telephone Systems*. Oliver and Boyd, Edinburgh, 1960.

[98] Topolcic, C. *Experimental Internet Stream Protocol: Version 2 (ST-II)*. Internet RFC 1190, October 1990.

[99] Waitzman, D., Partridge, C., and Deering, S. Distance Vector Multicast Routing Protocol. Internet RFC 1075, Nov 1988.

[100] Waxman, Bernard M. Routing of Multipoint Connections. *IEEE Journal on Selected Areas in Communications 6*, 9 (December 1988), 1617–1622.

[101] Waxman, Bernard M. Performance Evaluation of Multipoint Routing Algorithms. In *IEEE INFOCOM '93* (1993), pp. 980–986.

[102] Whitt, W. Blocking when Service is Required from Several Facilities Simultaneously. *AT&T Technical Journal 64*, 8 (October 1985).

[103] Wrege, D., and Liebeherr, J. A Near-Optimal Packet Scheduler for QoS Networks. In *IEEE INFOCOM* (1997).

[104] Wroclawski, J. Specification of the controlled-load network element service. Request For Comments (Proposed Standard) RFC 2211, Internet Engineering Task Force, September 1997.

[105] Yaron, Opher, and Sidi, Moshe. Calculating Performance Bounds in Communication Networks. In *IEEE INFOCOM '93* (1993).

[106] Zegura, E., Calvert, K., and Bhattacharjee, S. How to Model an Internetwork. In *IEEE INFOCOM* (1996).

[107] Zhang, H., and Ferrari, D. Rate-Controlled Service Disciplines. *Journal of High Speed Networks 3*, 4 (1994).

[108] Zhang, Lixia, Deering, Stephen E., Estrin, Deborah, Shenker, Scott, and Zappala, Daniel. RSVP: A New Resource ReSerVation Protocol. *IEEE Network 7*, 5 (September 1993), 8–18.

[109] Zheng, Q., and Shin, K. On the Ability of Establishing Real-Time Channels in Point-to-Point Packet-Switched Networks. *IEEE Transactions on Communications 42*, 2/3/4 (1994).