Service Disciplines For Packet-Switching Integrated-Services Networks

by

Hui Zhang

Service Disciplines For Packet-Switching Integrated-Services Networks

by

Hui Zhang

B.S. (Beijing University) 1988
M.S. (Rensselaer Polytechnic Institute) 1989

A dissertation submitted in partial satisfaction of the
requirements for the degree of
Doctor of Philosophy

in

Computer Science

in the

GRADUATE DIVISION
of the
UNIVERSITY of CALIFORNIA at BERKELEY

1993

The dissertation of Hui Zhang is approved:

Chair	Date
	Date
	Date

University of California at Berkeley

1993

To My Parents, My Sister, and Jackie

Abstract

Service Disciplines For Packet-Switching Integrated-Services Networks

Hui Zhang

Doctor of Philosophy in Computer Science

University of California at Berkeley

Professor Domenico Ferrari, Chair

Integrated-services networks will provide the communication infrastructure in the future. Unlike traditional communication networks, which are designed to offer a single type of service, integrated-services networks will offer multiple services, including data, voice, video, and others. Supporting applications with diverse traffic characteristics and performance objectives requires the offered services to be both flexible and guaranteed. To be flexible, integrated-services networks will use packet-switching technology. To provide performance guarantees in terms of throughput, delay, delay jitter, and loss rate, a proactive network control approach is needed. One of the most important components in this architecture is the service discipline at the switching nodes.

We first present a taxonomy and framework for studying and comparing service disciplines in integrated-services networks. Given the framework, we show the limitations of several existing solutions, and propose a new class of service policies called *rate-controlled service disciplines.* This class of service disciplines may be *non-work-conserving*, i.e., a server may be idle even when there are packets to be transmitted. Although non-work-conserving disciplines were seldom studied in the past, our research shows that non-work-conserving rate-controlled service disciplines have several advantages that make them suitable for supporting guaranteed performance communication in a high speed networking environment. In particular, rate-controlled service disciplines can provide end-to-end per-connection deterministic and statistical performance guarantees in very general networking environments. Unlike existing solutions, which only apply to simple network environments, rate-controlled service disciplines also apply to internetworking environments. Moreover, unlike existing solutions, which only apply to feed-forward networks and a restricted class of feedback networks, rate-controlled service disciplines can provide guarantees in arbitrary feed-forward and feedback networks.

The key feature of a rate-controlled service discipline is the separation of the server into two components: a rate-controller and a scheduler. This separation has several distinct advantages: it decouples the allocation of bandwidths and delay bounds, uniformly distributes the allocation of buffer space inside the network to prevent packet loss, and allows arbitrary combinations of rate-control policies and packet scheduling policies. Rate-controlled service disciplines provide a general framework under which most of the existing non-work-conserving disciplines can be naturally expressed. One discipline in this class, called Rate-Controlled Static Priority (RCSP), is particularly suitable for providing performance guarantees in high speed networks. It achieves simplicity of implementation as well as flexibility in the allocation of bandwidths and delay bounds to different connections.

To increase the average utilization of the network by real-time traffic, we present new admission control conditions for deterministic service, and new stochastic traffic models for statistical service. Compared to previous admission control algorithms for deterministic service, our solution ensures that deterministic services can be guaranteed even when the sum of the peak data rates of all connections exceeds the link speed. When the traffic is bursty, the new algorithm results in a multifold increase in the number of accepted connections. Further, to better characterize bursty traffic, we propose a traffic model that captures the interval-dependent behavior of traffic sources. With this traffic model and rate-controlled service disciplines, end-to-end per-connection statistical performance guarantees can be efficiently provided in a general networking environment.

To test our algorithms in real environments, we have designed and implemented the Real-Time Internet Protocol, or RTIP. RTIP is the network layer data delivery protocol within the Tenet real-time protocol suite, and is the first protocol of its kind that provides host-to-host bounded-delay and bounded-delay-jitter services in an internetworking environment. We implemented RTIP in Ultrix on DECstation 5000 workstations, in HP/UX on HP9000/7000 workstations, and in SunOS on SPARCstations. The service disciplines used in the implementation are rate-controlled service disciplines. Results from a measurement study show that the protocol is effective in supporting guaranteed performance services in an internetworking environment.

Committee Chair: ______________________________

Professor Domenico Ferrari

Contents

List of Figures

Acknowledgments

I would like to express my earnest gratitude to my advisor, Professor Domenico Ferrari, for his guidance and support throughout my research. In my many years of interacting with Professor Ferrari, I am often reminded of an ancient Chinese proverb: "A great mentor not only conveys his knowledge, but also nurtures his students." Professor Ferrari is indeed such a mentor.

I thank the members of my thesis committee, Professors Randy Katz and Richard Barlow for reading my thesis and making suggestions for improvement. I am also thankful to Professor Jean Walrand for serving on my quals committee.

It has been a great experience for me to work in the Tenet group. Collaborations with other group members have been a vital component in my research throughout my Ph.D years. My first project in the group was with Dinesh Verma on delay-jitter control. The next one was with Srinivasan Keshav on a comparison study in service disciplines. These two projects introduced me to the networking research, and finally led me to the current thesis topic. I would like to thank both of them for the invaluable experiences. I also enjoyed working with Ed Knightly on deriving new stochastic traffic models, with Colin Parris on Dynamic Channel Management algorithms, and with Colin Parris and Giorgio Ventre on graceful adaptation schemes. I will always cherish the memory of the protocol design meetings and pizza lunches with Anindo Banerjea, Bruce Mah, Amit Gupta, Mark Moran, Dinesh Verma and other members in the group. I thank Mark for always carving the pizza perfectly with the most primitive knife. I also thank Tom Fisher for helping me with the porting of the RMTP/RTIP protocols to the Ultrix environment, and Fred Templin for answering all the questions I had about Ultrix. Thanks also go to Riccardo Gusella, Ramon Caceres, Vern Paxon, and Steve McCanne for sharing with me their expertise.

I thank John Limb of Hewlett Packard Co. for making it possible for me to implement the first prototype of RMTP/RTIP on a HP9000/700 workstation. I thank Chuck Kalmannek of AT&T Bell Laboratory for being genuinely interested in my research and giving me useful advice. Thanks also go to Ed Knightly and Riccardo Bettati for reading my thesis and making several grammatic and technical suggestions.

Although my parents and sister are on the other side of the Pacific, they are a constant source of love and encouragement. I would not have finished this thesis without their support.

Last but not least, I would like to thank my dear Jackie Xu for her love and understanding throughout my graduate years.

Chapter 1

Introduction

This dissertation is about communication networks. Three additional key phrases in the title of the dissertation: *service disciplines*, *packet switching* and *integrated-services*, define the scope of the research. In this chapter, we will describe these three key words in turn.

In section 1, we motivate the need for integrated-services networks, and describe the most important difference between these networks and the other existing networks, and how this difference will influence the design of future networks. We then briefly review the packet-switching technology, which has been widely accepted as the switching technology of choice for future integrated-services networks, and describe the difficulties in supporting integrated-services using the current packet-switching technology. We argue that a connection-oriented and reservation-based network architecture is needed to support guaranteed service in a packet-switching network. We identify as one of the most important components in such a architecture the service discipline at the switches of the network. In section 2, we define more precisely the network model used in this study, and the properties of the services that are to be offered in the network. In section 3, we give some background on service disciplines and specify the requirements of service disciplines for integrated-services networks. In section 4, we present an overview of the dissertation.

1.1 Background

1.1.1 Integrated-Services Networks

Communication systems have been revolutionized by technological advances in the last decade. The speed and capacity of various components in a communication system, such as transmission media, switches, memory, processors, have all followed technological curves that have grown either linearly or exponentially over the last ten years [34]. At the periphery of the network, driven by the same underlying technology — microelectronics, the capability of computers has been drastically increased while the cost has been significantly reduced. To take advantage of the technological trends, and to satisfy the growing need for people and computers to communicate, new integrated-services networks are being designed. Unlike the current communication networks which are designed to offer a special type of service, the integrated-services network will offer multiple services that include data, voice, video, and others. The integration of multiple services into one network will offer a number of advantages, including vast economies of scale, ubiquity of access, and improved statistical multiplexing. However, this integration also presents new challenges for network designers.

Traditional communication networks offer a single type of service that supports one type of applications. For example, since the telephone network has been designed to support interactive voice, which requires a low delay, low rate, fixed bandwidth, two-way, jitter-free service. Alternately, the cable TV network has been designed to support broadcasting of analog video, which requires a high rate, fixed bandwidth, one-way, jitter-free service. Finally, the data network has been designed to support communication between computers. Although quality of service has been considered, the current data network only offers a best effort service – there are no guarantees on performance parameters such as delay or throughput. The specialization of each of these networks have allowed the network design to be optimized for that specific type of service.

Integrated-services networks will have to support applications with diverse traffic characteristics and performance objectives. There have been many applications proposed, and there is no doubt that many more will emerge once the networks are in place. These applications can be subdivided into at least two classes: those that have stringent performance requirements in terms of bandwidth, delay, delay jitter and loss rate, and those that do not. Some examples from the first class are: multi-party multi-media interactive applications (video conferencing, distributed seminars); scientific visualizations (nuclear reactions, molecular modeling); medical imaging and remote diagnostics (CAT scans, magnetic resonance imagery). Some examples from the second class are electronic mail and distributed computation. Corresponding to these two classes of applications, at least two classes of services are to be offered – *real-time service* and *non-real-time service* [24].

The non-real-time service corresponds to the best-effort service provided by the current data network, but the real-time services provided by the integrated-services networks are far from the simple merging of services provided by the current voice and video networks. The phone network supports only two-way communication with homogeneous sources, and cable networks support only broadcasting of one type of video, while the traffic characteristics and performance requirements for the integrated-services networks will be diverse. Scientific visualization and medical imaging will have very different characteristics from video. Even for video, conferencing applications, movie applications and HDTV require different qualities of service. Other factors, such as different coding algorithms and different resolutions, also contribute to the diversity of video requirements.

Because of the vast diversity of traffic characteristics and performance requirements of existing applications, as well as the uncertainty about future applications, a network should be flexible in its ability to support applications with different requirements. Moreover, the network also has to offer performance guarantees. Although the packet-switching data networks are highly flexible and can support applications that span a wide range of rates, no performance guarantees can be provided. Circuit-switched voice networks can provide excellent guarantees; however, they are extremely inflexible and are very limited in the variety of services they can offer. The design of integrated-services networks is challenging because these networks have to provide both *flexible* and *guaranteed* services.

1.1.2 Packet-Switching Techniques

Packet switching was proposed as an alternative to circuit switching to support data communication. The most important advantages of packet switching are its flexibility and ability to exploit statistical multiplexing. In a packet-switching network, any single source can potentially utilize all the network resources, and potentially there are unlimited numbers of users that can use the network at any one time. This is achieved by allowing sources to statistically share the network resources. In a networking environment where traffic sources are bursty, it is not likely that all sources will transmit information at their peak rates at the same time. Allowing users to statistically share the network resources will potentially achieve a higher network utilization than the sum of the peak rates of all the sources.

It has been widely accepted that future integrated-services networks will use packet switching because of its flexibility and ability to do statistical multiplexing. However, this flexibility is not free. Statistical multiplexing also introduces the problem of congestion – there is the possibility that the aggregate rate of the input traffic to the network temporarily exceeds the capacity of part of the network, in which cases, packets may experience long delays or may be dropped by the network.

Although networks are expected to become even faster, the problem of congestion is not likely to go away [43]. Various congestion control algorithms have been proposed in the literature. These solutions can be classified into two classes: reactive and proactive.

Reactive approaches have been widely studied in the context of data networks. In such an approach, the control is usually at the end system; sources constantly sample the network state and try to detect symptoms of network congestion. Various signals, either from the receiver or from the network, can be used by a source to detect network congestion [42, 78, 48]. When a congestion is detected, sources usually lower the transmission rates to alleviate the congestion. Such reactive or feedback-based controls operate in a time scale of the magnitude of one *round-trip time* (RTT) since the length of the interval between the time when the congestion is detected and the time when the congestion signal is passed back to the source is on the order of one round-trip time. A related important control parameter is the delay-bandwidth product, which is the product of the RTT and the link bandwidth. This is the amount of data that is in transit within the network along a given path. Potentially, this amount of data may get dropped due to congestion, and re-transmission of the data may be needed. A high-speed network has high-speed links, but the RTT is bounded from below by the propagation delay of signals, which is a function of distance and the speed of the light, but not of the link speed. Thus, as link speeds become higher, the delay-bandwidth product will also increase. Consider a cross-country gigabit network: the roundtrip propagation delay is around 50 ms, so the delay-bandwidth product is at least $1Gbps \times 50ms$, or 50Mbits. In such a network, potentially, 50Mbits of data may get dropped due to congestion and have to be re-transmitted.

Large delay-bandwidth products have made the reactive congestion-control approach less effective in the context of high-speed networks. Besides the large delay-bandwidth product, the control time scale also makes the reactive approach not suitable for delay-sensitive applications. With the reactive approach, data is re-transmitted when lost; however, data from re-transmission can arrive at the destination only after at least a couple of round-trip times, and may not be useful for delay-sensitive applications.

For applications that require performance guarantees, a proactive congestion management approach is needed. Such a scheme needs resource management within the network, and usually requires a connection-oriented approach. An example of proactive congestion management is the Tenet Real-Time Channel Scheme [31]. In the Tenet scheme, before communication starts, the client specifies its traffic characteristics and performance requirements to the network; the client's traffic and performance parameters are translated into local parameters, and a set of connection admission control conditions are tested at each switch; the new connection is accepted only if its admission would not cause the performance guarantees made to other connections to be violated; during data transfers, each switch will service packets from different connections according to a service discipline; by ensuring that the local performance requirements are met at each switch, the end-to-end performance requirements can be satisfied. Notice that there are two levels of control in this paradigm: connection admission control at the connection level, and service discipline at the packet level. Usually, different service disciplines need different admission control algorithms. A complete solution needs to specify both the service discipline and the associated connection admission control conditions.

1.2 Problem Specification

1.2.1 Network Model

There are two architectures proposed for packet-switching integrated-services networks. One is based on fixed-size packets, the other is based on variable-sized packets. CCITT has adopted Asynchronous Transfer Mode (ATM) [11] as the transfer mode for the Broadband Integrated Services Network (B-ISDN). ATM uses small, fixed-size packets called *cells* as the transmission and multiplexing unit. Other researchers, largely from the computer communication community, are advocating a packet-switching architecture based on variable-size packets [17, 15]. It seems certain that the two architectures will co-exist at least for the near future. The final outcome, whether one architecture will be dominating or both will co-exist for a longer time, depends not only on technical, but also on political, economical and administrative issues. One way they can co-exist is through internetworking. For example, in Xunet [36], a wide-area network testbed, gateways or routers are used to link an ATM network with FDDI networks into an internetwork. Variable-size packets are used at the internetworking level, while the ATM network is used as a subnetwork. The Xunet testbed is shown in Figure 1.1

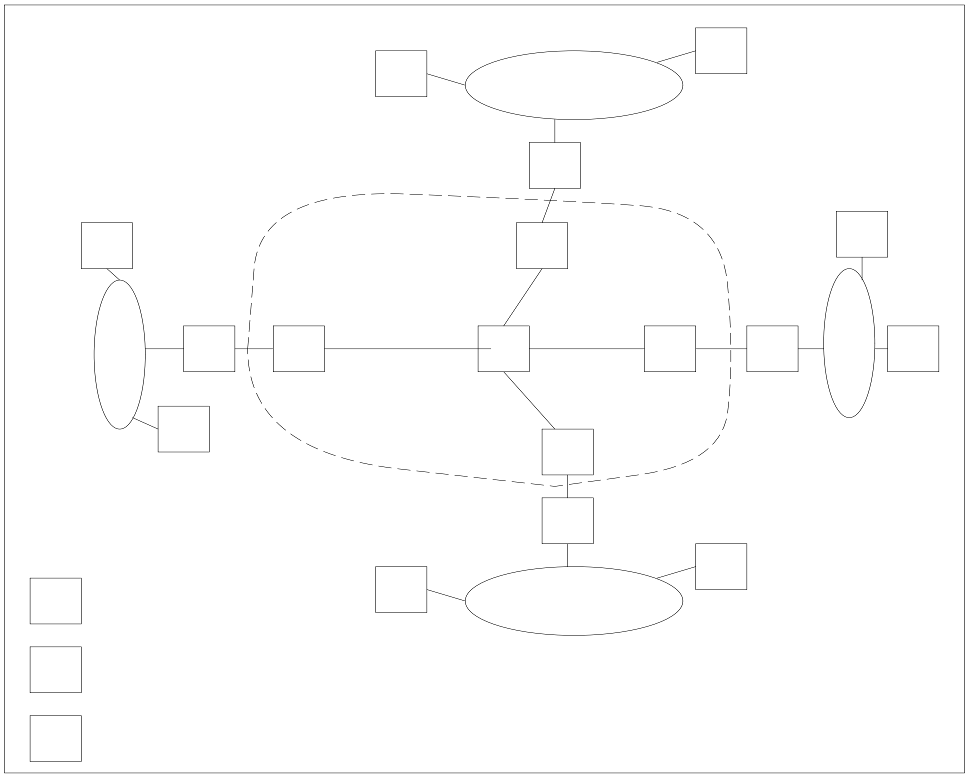


Figure 1.1: Xunet 2 wide-area network testbed

In this research, we adopt a general model that is applicable to both cases. We consider a network with arbitrary topology of links and switches[1]. The network switches variable-size packets.

For simplicity of discussion, we assume that switches are 'non-blocking', i.e., when packets arrive at an input link, they can be routed directly to the appropriate output links without switching conflicts. Packets destined for different output links do not interfere with

[1]In the literature, the term 'switch' is used in the context of ATM networks, while 'gateway' or 'router' is used more often in an internetworking environment. In this research, we will call all switching elements as 'switches'.

each other, and queueing occurs only at the output ports of the switch. With this assumption, a connection in such a network can be modeled as traversing a number of queueing servers, with each server modeling the output link of an ATM switch. The assumption is largely valid in existing ATM networks, where switches contain high-speed fabrics or buses [5, 46]. In an internetworking environment, computers are usually used as gateways to switch packets. The processor of the gateway can potentially become a bottleneck. In such a case, both the processor of the gateway and the output link can be modeled as queueing servers so that a connection traverses two servers in each switch.

A link in such a network is assumed to have bounded, but not necessarily constant delay. This will allow us to model an internetwork in a hierarchical way. A subnetwork, as long as it can bound its end-to-end delay, can be modeled as a logical link at the immediately higher internetworking level [26]. Various algorithms can be used to bound delay in subnetworks. For example, in [3, 4], algorithms are designed to bound packet delays in an FDDI network.

It should be noted that most of the current delay bounding solutions assume a *simple network* model where link delays between switches are constant [20, 21, 51, 66, 68, 38, 6]. Constant-delay links have the desirable property that the traffic pattern at the receiving end of the link is the same as that at the transmitting end of the link. This property is important for these solutions because they use the characterization of the output traffic from a scheduler as the characterization of the input traffic to the next-hop scheduler. However, in an internetworking environment, links connecting switches may be subnetworks such as ATM or FDDI networks. Though it is possible to bound delay over these subnetworks, the delays for different packets will be *variable*. Thus, these solutions do not apply to an internetworking environment. By relaxing such an assumption and having a more general model that assumes only bounded-delay links, the solutions proposed in this dissertation apply to both simple network and internetwork environments.

In this network model, the end-to-end delay of a packet consists of four components: (1) the link delay, which includes the propagation delay and other delays incurred in the lower level subnetwork if some of links are subnetworks, (2) the switching delay, which depends on the implementation of the switches, (3) the transmission delay, which is a function of the packet length and the link speed, and (4) the queueing delay at each switch. By assumption, link delay is bounded. Switching delay and transmission delay are fixed. Queueing delay is the component that can be affected by controlling the load of each switch or using an appropriate service discipline, and thus is the major concern for this research.

Packets submitted to the network may get lost for two reasons: transmission errors and buffer overflow at switches. As the technology advances, the transmission error rate drops drastically. For example, with current fiber technology, the bit error rate is below 10^{-14} [35]. Thus, buffer overflows due to congestion is the major cause of packet losses.

1.2.2 Services

We assume that there are two types of services offered by an integrated services network: real-time service and non-real-time service.

In a real-time service, before the communication starts, the client needs to specify its traffic characteristics and desired performance requirements; when the network accepts the client's request, it guarantees that the specified performance requirements will be met provided that the client obeys its traffic specification.

We believe that the real-time service provided by the integrated-services network should have the following properties [27]:

(1) The service interface provided by the network should be general. Instead of supporting only a fixed class of applications (e.g., conference quality video, high definition television, CD-quality sound, animated images with voice track, and so on; or low-delay high-throughput communication, medium-delay low-throughput communication, high-delay low-jitter communication, and so on), the network should provide a parameterized interface abstraction that allows the client to specify a continuum of values and an unrestricted combination of values for its specification of both traffic characteristics and performance requirements.

One example of such a general service interface [24], which we will use in this dissertation, consists of the following traffic and performance parameters:

- Minimum packet inter-arrival time $Xmin$
- Average packet inter-arrival time $Xave$
- Averaging interval I
- Maximum packet size $Smax$

- Delay bound D
- Delay violation probability bound Z
- Buffer overflow probability bound W
- Delay jitter bound J

(2) *The solution should be applicable to a wide variety of internetworking environments.* Most end-to-end communication will go through several networks. Guaranteed-performance communication services that could not be easily implemented in a wide spectrum of internetworks would have limited value.

(3) The guarantees should be quantitative, and mathematically provable. It is our belief that the clients of a guaranteed performance service require predictable performance. Computer networks are strongly *coupled non-linear* systems with complex interactions between different entities. Although simulation can give insights to the performance of a network, results from simulation of smaller networks are not easily extended to larger networks due to the non-linear nature of the network. In particular, it has been observed that adding a single extra node may make a stable system unstable [33]. These considerations make the problem more challenging and favor solutions where the results are guaranteed in general environments. As long as the guarantees are *a priori*, they can either be deterministic or statistical [24]. To achieve this, we believe that the network should manage resources explicitly; this entails (a) channel admission control with resource reservation; and (b) connection-oriented communication with pre-computed routes.

The non-real-time service may be further divided into multiple classes. One class corresponds to the best-effort service, where the client can send packets at any time, but there is no guarantee that a packet submitted to the network will be delivered, and there is no upper bound on the delay of a packet even when it gets delivered. Recently, a new type of service called *predicted service* has been proposed [17], in which the performance delivered by the network in terms of delay bounds is computed by measuring the current load, rather than by using precomputed, worst-case data. This service has been proposed for a particular class of multimedia applications, called *adaptive playback applications*. These applications use an estimated *post facto* delay bound, and can adapt the *playback point* when the post facto bound changes due to network load fluctuations. Since in the predicted service, fluctuations in the network load can produce variations in the provided quality of service, the network cannot commit to offering a well determined performance. In addition, applications are required to endure a high packet loss rate and even service disruption due to the adaptation of the playback point to the variations of the end-to-end communication delay. For the purposes of this research, we classify the predicted service among the non-real-time services.

1.3 Service Disciplines

In a packet-switching network, packets from different connections will interact with each other at each switch. Without proper control, these interactions may adversely affect the network performance experienced by clients. The service disciplines of the switching nodes, which control the order in which packets are serviced, determine how packets from different connections interact with each other.

The service discipline at the output port of a switch manages three nearly independent resources: bandwidth (*which* packets get *transmitted*), promptness (*when* do those packets get *transmitted*) and buffer space (*which* packets are *discarded*) [22], which, in turn, affects three performance parameters: throughput, delay, and loss rate.

A service discipline can be classified as either *work-conserving* or *non-work-conserving*. With a work-conserving discipline, a server is never idle when there is a packet to send. With a non-work-conserving discipline, the server may be idle even when there are packets waiting to be sent. Non-work-conserving disciplines were seldom studied in the past mainly due to two reasons. First, in most of previous performance analyses, the major performance indices are the *average* low delay of all packets and the *average* high throughput of the server. With a non-work-conserving discipline, a packet may be held in the server even when the server is idle. This may increase the average delay of packets and decrease the average throughput of the server. Secondly, most previous study assumed a single server environment. The potential advantages of non-work-conserving disciplines in a networking environment were therefore not realized. In integrated-services networks, end-to-end delay *bounds* are to be guaranteed on a per-connection basis in a *networking* environments. Since guaranteeing end-to-end delay bounds requires *worst case* analysis in a networking environment, the above reasons for not using non-work-conserving disciplines do not hold any more. In this dissertation, we will show that non-work-conserving disciplines have some distinct advantages that make them suitable for providing performance guarantees in packet switching integrated-services networks.

Although it is possible to build an integrated-services network on top of a vast class of service disciplines [26], we would like a service discipline to be efficient, protective, flexible, and simple.

1.3.1 Efficiency

To guarantee certain performance requirements, we need a connection admission control policy to limit the real-time traffic load in the network, or limit the number of real-time connections that can be accepted. A service discipline is more efficient than another one if it can meet the same *end-to-end* performance guarantees under a heavier load. An efficient service discipline will result in a higher utilization of the network.

1.3.2 Protection

Guaranteed service requires that the network protects well-behaving real-time clients from three sources of variability: ill-behaving users, network load fluctuation and best-effort traffic. It has been observed in operational networks that ill-behaving users and malfunctioning equipments may send packets to a switch at a higher rate than declared. Also, network load fluctuations may cause a higher instantaneous arrival rate from a connection at some switch, even though the connection satisfies the traffic constraint at the entrance to the network. Another variability is the best-effort traffic. Although the real-time traffic load is limited by connection admission control, best-effort packets are not constrained. It is essential that the service discipline should meet the performance requirements of packets from well-behaving real-time clients even in the presence of ill-behaving users, network load fluctuation and unconstrained best-effort traffic.

1.3.3 Flexibility

The integrated service network needs to support applications with diverse traffic characteristics and performance requirements. A general parameterized service interface instead of a fixed type service interface should be offered. Mapping the service interface into the properties of the network mechanisms, the service discipline should be flexible to allocate different delay, bandwidth and loss rate quantities to different real-time connections.

1.3.4 Simplicity

We would like to have a service discipline that is both conceptually simple so that it can be studied with simple analysis techniques, and easy to implement in very high speed switches.

1.4 Overview of the Thesis

In this research, we study the issues and tradeoffs that are to be faced when designing service disciplines for integrated-services packet-switching networks. We adopt the following research methodology: a comparison study is first conducted to understand the strengthes and weaknesses of previous work; a new algorithm, which overcomes the limitations of existing solutions, is then proposed and analyzed by using both deterministic and probabilistic techniques; finally, the new algorithm is implemented and evaluated in real-time environments.

Chapter 2 reviews previous work. A novel taxonomy is presented to form a framework for classifying and comparing existing solutions. Within the context of a framework, we discuss various tradeoffs in designing service disciplines in integrated-services networks, and identify strengths and weaknesses of each of the approaches.

Chapter 3 presents a new class of non-work-conserving service disciplines called *rate-controlled service disciplines.* that have several distinct advantages that make them suitable for supporting guaranteed performance communication in a high speed networking environment. In particular, rate-controlled service disciplines can provide end-to-end per-connection deterministic performance guarantees in very general networking environments. Unlike existing solutions, which only apply to simple network environments, rate-controlled service disciplines also apply to internetworking environments. Moreover, unlike existing solutions, which only apply to feed-forward networks and a restricted class of feedback networks, rate-controlled service disciplines can provide guarantees in arbitrary feed-forward and feedback networks.

The key feature of a rate-controlled service discipline is the separation of the server into two components: a rate-controller and a scheduler. This separation has a number of advantages: it decouples the allocation of bandwidths and delay bounds, uniformly distributes the allocation of buffer space inside the network to prevent packet loss, and allows arbitrary combinations of rate-control policies and packet scheduling policies. Rate-controlled service disciplines provide a general framework under which most of the existing non-work-conserving disciplines can be naturally expressed. One discipline in this class, called Rate-Controlled Static Priority (RCSP), is particularly suitable for providing performance guarantees in high speed networks. It achieves simplicity of implementation as well as flexibility in the allocation of bandwidths and delay bounds to different connections.

It has been shown that with *non-work-conserving* rate-controlled service disciplines, although the *average* delay of packets on a connection may be increased, the end-to-end delay bound is not increased.

Chapter 4 analyzes rate-controlled service disciplines using statistical techniques. End-to-end statistical performance bounds are derived in general networking environments. The fact that rate-controlled service disciplines are non-work-conserving greatly simplies the analysis and improves the network utilization. Compared to the approach of using work-conserving disciplines, rate-controlled service discipline allows *more* connections to be admitted in *more general* networking environments.

Chapter 5 presents the design and implementation of the Real-Time Internet Protocol, which is the first protocol of its kind that provides host-to-host bounded-delay and bounded-delay-jitter services in an internetworking environment. RTIP has been implemented on several hardware and software platforms in a number of networking environments. The service disciplines used in the implementation are rate-controlled service disciplines. Results from a measurement study show that the protocol is effective in supporting guaranteed performance services in an internetworking environment.

Chapter 6 summarizes the dissertation by discussing the contributions and the weakness of our approach. We also outline directions in which this work can be extended.

Chapter 2

Taxonomy and Previous Work

Service disciplines and associated performance problems have been widely studied in the contexts of hard real-time systems and queueing systems. However, results from these studies are not directly applicable in the context of integrated-services networks. Analyses of hard real-time systems usually assume a *single* server environment, *periodic* jobs, and the job delay bounded by its *period* [80]. However, the network traffic is *bursty*, and the delay constraint for each individual connection is *independent* of its bandwidth requirement. In addition, bounds on *end-to-end* performance need to be guaranteed in a *networking* environment, where traffic dynamics are far more complex than in a single server environment. Queueing analysis is often intractable for realistic traffic models. Also, the classical queueing analyses usually study *average* performance for *aggregate* traffic [49, 87], while in integrated-services networks *performance bounds* need to be derived on a *per-connection* basis [24, 52]. In addition to the challenge of providing end-to-end per-connection performance guarantees to heterogeneous, bursty traffic, service disciplines must be *simple* enough that they can be implemented at very high speeds.

Recently, several new service disciplines, which aim to provide different qualities of service to different connections, have been proposed. Among them are Delay Earliest-Due-Date (Delay-EDD) [31], Virtual Clock [97], Fair Queueing [22] and its weighted version, also known as General Processor Sharing [67], Jitter Earliest-Due-Date (Jitter-EDD) [86], Stop-and-Go [38], and Hierarchical Round Robin (HRR) [44]. They are closely related, but also have some important differences [93].

In this chapter, we present a novel taxonomy that classifies the existing solutions along two dimensions:

1. how the service discipline allocates different delay bounds and bandwidths to different connections in a single switch;
2. how the service discipline handles traffic distortions in a network of switches.

We show that existing solutions adopt one of two approaches along each dimension. We describe the approaches and discuss the tradeoffs between them. In section 2.1, we present the two approaches to allocating delay/bandwidth to different connections in a single switch: the sorted priority queue mechanism and the framing strategy. In section 2.2, we present the two approaches to handling traffic distortions in a network of switches, i.e., accommodating the controlling the distortions. In section 2.3, we summarize the discussion and place each of the existing solutions within the framework of the taxonomy. In section 2.4, we discuss other related work described in the literature.

2.1 Allocating Delay/Bandwidth in a Single Switch

The objective of the allocation of delay/bandwidth is that, with a certain scheduling discipline, a connection can be guaranteed to receive certain throughput, and each packet on that connection can be guaranteed to have a bounded delay.

2.1.1 Sorted Priority Queue Mechanism

Virtual Clock, Fair Queueing and Earliest-Due-Date schemes use the sorted priority queue mechanism to allocate different delay bounds and bandwidths to different connections.

Figure 2.1 gives a simple example to illustrate how Virtual Clock works. In the figure, there are three connections sharing the same output link. All three connections specify their traffic characteristics and reserve resources accordingly. Connections 1 and 2 have an average packet inter-arrival time of 5 time units, connection 3 has an average packet inter-arrival time of 2 time units. For simplicity, assume packets from all the connections have the same size, and the transmission of one packet takes one time unit. Hence, each of connections 1 and 2 reserve 20% of the link bandwidth, while connection 3 reserves 50% of the link bandwidth.

In Figure 2.1, the first three timelines show the the actual arrival pattern of the three connections. Connections 1 and 2 send packets at higher rates than reserved, while

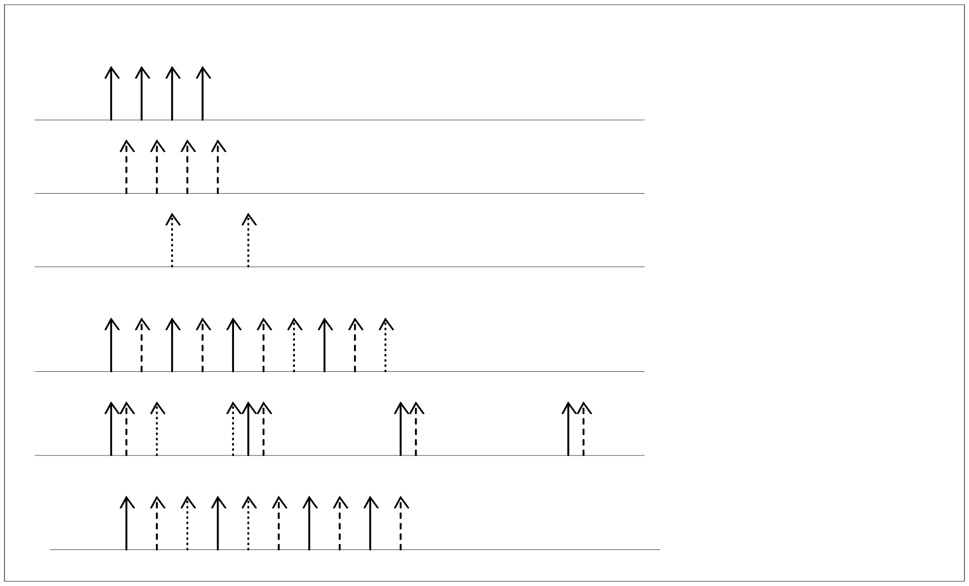


Figure 2.1: Comparison of Virtual Clock and FCFS

connection 3 sends packet according to the specified traffic pattern. The fourth timeline shows the service order of packets if the service discipline is FCFS. In this case, even though connection 3 reserves more resources, the misbehaviors of connections 1 and 2 affect its performance.

The Virtual Clock algorithm assigns each packet a virtual transmission time based on the arrival pattern and the reservation of the connection to which the packet belongs. The fifth timeline shows the virtual transmission time assignment. The transmissions of packets are then ordered by the virtual transmission times. The service order of packets under the Virtual Clock discipline is shown in the sixth timeline. Notice that although connections 1 and 2 are sending packets at higher rates, the Virtual Clock algorithm ensures that all well-behaving connections, in this case connection 3, get good performance.

Delay and Jitter Earliest-Due-Date, and Fair Queueing use a similar sorted priority queue mechanism. In such a mechanism, there is a state variable associated with each channel to monitor and enforce the rate for each connection. Upon arrival of each packet from that connection, the variable is updated according to (a) the reservation made for the connection during the connection establishment time and, (b) the traffic arrival history of the connection during the data transfer. The packet is then stamped with the value of the state variable for the connection to which it belongs. The stamped value is used as a

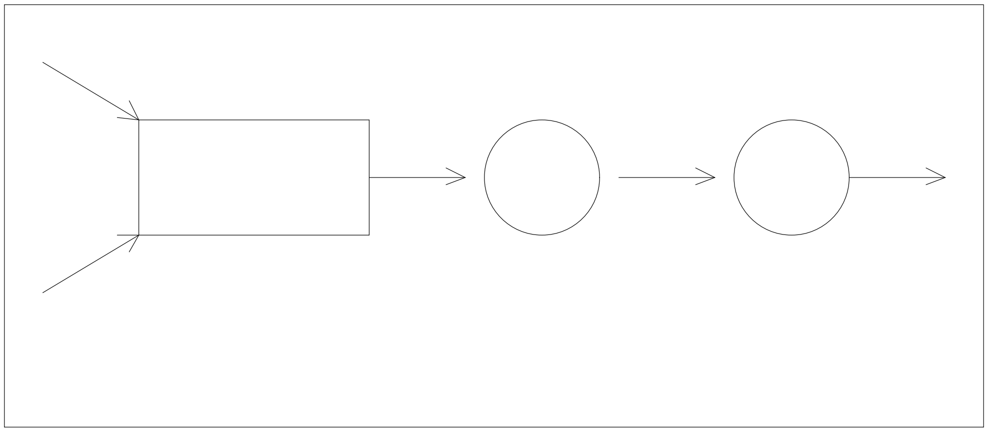


Figure 2.2: Sorted priority mechanism

priority index. Packets are served in the order of increasing priority index values. This is shown in Figure 2.2.

In Virtual Clock, the state variable is called auxiliary Virtual Clock [1] ($auxVC$); in Fair Queueing, it is called the Finish Number (F); in Delay-EDD, it is called Expected Deadline (ExD). In all three cases, $auxVC$, F and ExD are used as priority indices of packets; the computations of $auxVC$, F and ExD are based on the formula shown in Table 2.1. In the table, the subscripts i and j denotes switch number and connection number, respectively. In Delay-EDD, $Xmin_j$ is the minimum packet inter-arrival time for connection j, $\overline{d}_{i,j}$ is the local delay bound assigned to connection j at switch k at connection establishment time. In Virtual Clock, $Vtick_j$ is the average packet inter-arrival time for connection j. In Virtual Clock and Delay-EDD, AT is the packet arrival time. In Fair Queueing, R^i is the number of rounds that have been completed for a hypothetical bit-by-bit round robin server at switch i, n_j is a weighting factor [2] and P_j is the packet length measured in number of bits.

The state updating formulas upon arrival of each packet for Virtual Clock and Fair Queueing are completely equivalent. This is clear from the following correspondences:

$$AT \Longleftrightarrow R^i$$

[1]There is another variable $VirtualClock$ associated with each connection that is periodically checked to measure the difference between the actual rate and the allocated rate. $VirtualClock$ is irrelevant in this discussion.

[2]In [66], where the weighted version of Fair Queueing or the General Processor Sharing discipline is discussed, ϕ is used as the symbol of the weighting factor.

Virtual Clock	**Fair Queueing**	**Delay-EDD**
$auxVC_{i,j} \leftarrow max(AT, auxVC_{i,j})$	$F_{i,j} \leftarrow max(R^i, F_{i,j})$	$ExD_{i,j} \leftarrow ExD_{i,j} + Xmin_j$
$auxVC_{i,j} \leftarrow auxVC_{i,j} + Vtick_j$	$F_{i,j} \leftarrow F_{i,j} + \frac{P_j}{n_j}$	$ExD_{i,j} \leftarrow max(AT + \overline{d}_{i,j}, ExD_{i,j})$
Stamp packet with $auxVC_{i,j}$	Stamp packet with $F_{i,j}$	Stamp packet with $ExD_{i,j}$

Table 2.1: Comparison of Virtual Clock, Fair Queueing and Delay-EDD

$$auxVC_{i,j} \Longleftrightarrow F_{i,j}$$

$$Vtick_j \Longleftrightarrow \frac{P_j}{n_j}$$

If we combine the two state equations for both Virtual Clock and Delay-EDD, we have

- In Virtual Clock, $auxVC_{i,j} \leftarrow max(AT + Vtick_j, auxVC_{i,j} + Vtick_j)$
- In Delay-EDD, $ExD \leftarrow max(AT + \overline{d}_{i,j}, ExD_{i,j} + Xmin_j)$

We have the following mapping:

$$auxVC_{i,j} \Longleftrightarrow ExD$$

$$Vtick_j \Longleftrightarrow Xmin_j \ or \ \overline{d}_{i,j}$$

Recall that, in Delay-EDD, $Xmin_j$ is the minimum packet inter-arrival time and $\overline{d}_{i,j}$ is the local delay bound; in Virtual Clock, $Vtick_j$ is the average packet inter-arrival time. As can be seen, there are two differences here:

1. Delay-EDD imposes the restriction of minimum spacing between packets, while Virtual Clock does not.

2. Delay-EDD decouples the delay and bandwidth requirements by using both $Xmin_j$ and $\overline{d}_{i,j}$, while Virtual Clock just has one counterpart for bath, namely, $Vtick_j$.

The two differences are the reason why Delay-EDD, in conjunction with the establishment and admission control scheme described in [30], can provide both delay and bandwidth guarantees, while Virtual Clock can provide only bandwidth guarantees.

2.1.2 Framing Strategy

Both Stop-and-Go and HRR use a multi-level framing strategy. For simplicity, we just describe one-level framing. In a framing strategy, the time axis is divided into periods of some constant length T, each called a frame[3]. Bandwidth is allocated to each connection as a certain fraction of the frame time.

Stop-and-Go defines *departing* and *arriving* frames for each link. At each switch, the arriving frame of each incoming link is mapped to the departing frame of the output link by introducing a constant delay θ, where $0 \leq \theta < T$. All the packets from one arriving frame of an incoming link and going to output link l are delayed by θ and put into the corresponding departing frame of l. According to the Stop-and-Go discipline, the transmission of a packet that has arrived on any link l during a frame f should always be postponed until the beginning of the next frame. Since packets arriving during a frame f of the output link are not eligible for transmission until the next frame, Stop-and-Go is a non-work-conserving service discipline. Within each frame, the service order of packets is arbitrary[4].

One-level HRR is equivalent to a non-work-conserving round robin service discipline. Each channel is assigned a fraction of the total available bandwidth, and receives that bandwidth in each frame, if it has sufficient packets available for service. The server ensures that the assigned bandwidth is also the maximum service rate for that channel in each frame. This means that, in a frame, after providing a channel's bandwidth allocation, even if the server is available and more packets from that channel are queued for transmission, the packets will not be served until the next frame. Since these extra packets are not eligible for transmission until the next frame, HRR is a non-work-conserving service discipline. Within each frame, the service order of packets is arbitrary.

2.1.3 Tradeoffs Discussion

The two approaches of allocating delay/bandwidth to different connections have both advantages and disadvantages.

The sorted priority queue mechanism is flexible in allocating different combination of delay/bandwidth resources to different connections. When a sorted priority queue mech-

[3]The notation FT is used in HRR, while T is used in Stop-and-Go; we adopt T for both in this discussion.

[4]FCFS is used as the service discipline within each frame in [37, 38]; however, as pointed out by the author, this is just for convenience and is not part of Stop-and-Go.

anism is used, if the calculation of the state variable takes into account both the delay and bandwidth requirements of a connection as done in Delay-EDD, arbitrary combinations of delay and bandwidth requirements can be allocated to a connection. However, there are two drawbacks with the approach. The insertion operation for a sorted priority queue has an intrinsic complexity of $O(\log N)$ [50], where N is the number of packets in the queue. In a network that is designed to support many connections with bursty traffic, the switch usually has buffer space for a large number of packets. For example, the queue module of each link of the Xunet switch contains memory to store 512K ATM cells [45]. Potentially, the queue length can be long. It may not be feasible to implement an operation that has an $O(\log N)$ complexity at very high speed. Recently, it was reported that a high-speed sequencer chip has been implemented to support service disciplines such as Virtual Clock; however, the chip can only sort a maximum of 256 packets [12]. It has yet to be seen whether a high-speed implementation can be built to support a sorted priority queue mechanism in an environment with large numbers of packets.

The second drawback with a sorted priority queue mechanism has to do with the admission control tests. In order to decouple the delay and bandwidth allocations as in Delay-EDD, a schedulability test has to be performed in addition to the bandwidth and buffer tests [30]. This is due to the fact that scheduling saturation can occur even if bandwidth is not overbooked. For example, two packets with delay bounds of 4 time units and service times of 3 time units each may arrive at a node at the same time. In this situation, it is impossible to meet the delay bounds for both packets. The schedulability test as outlined in [23] is rather complicated. Although connection establishment is a much less frequent event than packet forwarding, and there is usually no real-time requirement, it is important to have efficient admission control tests for fast establishment schemes like [82].

Compared to the sorted priority queue mechanism, the framing strategy is easier to implement. The Hierarchical Round Robin discipline has been implemented in custom VLSI at gigabit speed [36]. Also, there is no separate schedulability test at connection establishment time.

However, the framing strategy has an important drawback: it couples the allocations of delay bound and bandwidth. In a framing strategy, the delay of a packet is bounded by the frame size multiplied by a constant factor (in Stop-and-Go, the factor is between 2 and 3, in HRR the factor is 2). To provide a smaller delay bound, a smaller frame size is needed. However, the frame size is also linked to the bandwidth allocation granularity. If the frame size has N slots, then bandwidth can only be allocated in units of $\frac{1}{N}$ of the total link bandwidth. A finer bandwidth allocation granularity requires a larger frame size. The relationships between frame size, local delay bound and bandwidth allocation granularity are shown in the following formula, where C is a constant:

$$bandwidth\ allocation\ granularity = \frac{packet\ size}{T}$$

$$delay\ bound = C \times T$$

It is clear that there is a coupling between the bandwidth allocation granularity and the delay bound. Low delay bound and fine granularity of bandwidth allocation cannot be achieved simultaneously. To get around this coupling, both HRR and Stop-and-Go propose the use of multiple frames with different frame times. In this case, it is possible to provide low delay bounds to some channels by putting them in frames with a smaller frame time, and to allocate bandwidth with finer granularity to other channels by putting them in levels with a larger frame time. However, the coupling between delay and bandwidth allocation granularity still remains within each frame.

2.2 Handling Traffic Distortions

A switch can provide local performance guarantees to a connection only when the traffic on that connection satisfies certain traffic specifications. However, network load fluctuations may distort the traffic pattern of a connection and cause an instantaneously higher rate at some switch even when the connection satisfies the client-specified traffic constraints at the entrance to the network.

Consider the example shown in Figure 2.3. Four packets from one connection are sent with a certain inter-packet spacing from the source into a simple network where links have constant delay. At the first switch, the first packet is delayed by a certain amount of time (less than the local delay bound) due to instantaneous cross traffic load, but the other three packets pass through instantly. Because the first packet was delayed longer than the second packet, the spacing between first and second packets becomes smaller when they arrive at the second switch. At the second switch, let the first and the second packet be delayed some more time, but packets 3 and 4 pass through instantly. At the third switch,

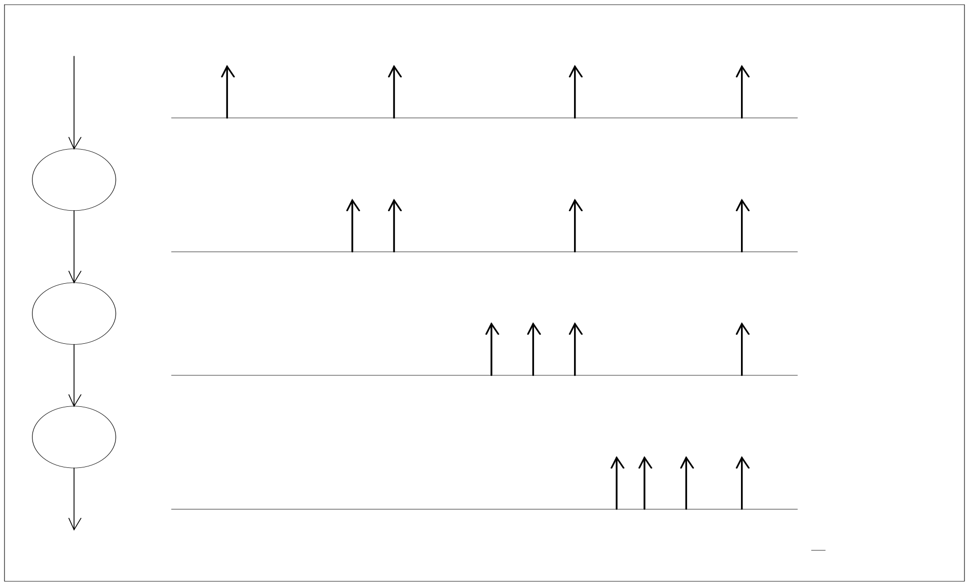


Figure 2.3: Traffic pattern distortions due to load fluctuations

the first three packets are delayed but packet 4 passes through instantly. The figure shows traffic patterns at the entrance of each of the switches. Two things can be observed: (a) the traffic pattern of a connection can be distorted due to network load fluctuations; the distortion may make the traffic burstier and cause instantaneously higher rates; (b) in the worst case, the distortion can be accumulated, and downstream switches potentially face burstier traffic than upper stream switches.

In a simple network, where links have constant delay, only load fluctuations at switches cause distortion to the traffic pattern. In an internetworking environment, where the logical link between two switches may be a subnetwork, load fluctuations within subnetworks may also introduce traffic pattern distortions. For example, if the subnetwork is an ATM network, the delay experienced by a packet in the subnetwork consists of a number of components: segmentation and reassembly delay, propagation delay, queueing delay at ATM switches, switching delay, and transmission delay. The queueing delay at ATM switches varies with the load in the ATM network. Therefore, the end-to-end delay in the ATM network, which is the logical link delay at the internetworking level, varies with the load in the ATM subnetwork. Most of the existing solutions that provide end-to-end performance bounds consider only traffic distortion due to load fluctuation at the switches. Therefore, these solutions apply only to simple network environments, but not to internetworking environment. In this chapter, we assume a simple network environment. Solutions that apply to internetworking environments are discussed in Chapter 3.

While a switch can provide local performance guarantees to a connection only when the traffic on that connection satisfies certain traffic specifications, distortions to connection's traffic pattern invalidate the client specified traffic characteristics inside the network. There are two solutions to this problem:

1. accommodating the distortion by allocating more buffer space;
2. controlling the traffic distortion within the network.

2.2.1 Accommodating Traffic Distortions

If the maximum distortion to traffic patterns can be bounded, buffer space can be allocated at each switch to accommodate the distortion. However, in a feedback network, where the union of the routes of several connections may result in cycles inside the network, the bounding of traffic pattern distortions can be very complicated. Feedback effects may drive the system towards instability. In [20], an example is given to show this phenomenon for an arbitrary service discipline. In [68, 66], the problem of stability in a feedback network is discussed in the context of the General Processor Sharing discipline. Although stability is proven to hold under certain conditions, the general stability problem in a GPS network is still unsolved.

In Delay-EDD, the maximum distortion of traffic patterns can be bounded because of the way the deadline of a packet is calculated. The deadline of a packet is the sum of the local delay bound (d) and the expected arrival time (ExA) of the packet. The service discipline and the admission control policy ensure that the packet is guaranteed to leave before the deadline, or at most d time units after the expected arrival time of the packet. If a connection satisfies the traffic specification at the entrance to the network, and the previous switches are scheduled by the Delay-EDD discipline, the difference between the expected arrival time of a packet and the actual arrival time of a packet can be bounded by the sum of the local delay bounds of previous switches. Formally, the following properties hold for Delay-EDD:

1. if the k^{th} packet on connection j enters the network at time $EnterTime$, the earliest time it can arrive at switch i is $(EnterTime + \sum_{h=0}^{i-1} t_{h,j}^{k} + \sum_{h=1}^{i} \pi_h)$, where $t_{h,j}^{k}$ is

the time to process the packet at switch h and π_h is the propagation delay between switch $h-1$ and switch h; the earliest time and the latest time it can leave the switch are $(EnterTime + \sum_{h=0}^{i} t_{h,j}^{k} + \sum_{h=1}^{i} \pi_h)$ and $(EnterTime + \sum_{h=0}^{i} \overline{d}_{h,j} + \sum_{h=1}^{i} \pi_h)$, respectively.

2. the maximum residence time of the k^{th} packet on connection j at switch i is $(\sum_{h=0}^{i} \overline{d}_{h,j} - \sum_{h=0}^{i-1} t_{h,j}^{k})$.

3. the maximum number of packets from connection j at switch i at any given time is $\lceil \frac{\sum_{h=0}^{i} \overline{d}_{h,j} - \sum_{h=0}^{i-1} t_{h,j}^{k}}{xmin} \rceil$.

For Delay-EDD, the maximum number of packets from a connection at a switch increases linearly along the path. If a per-connection buffer allocation scheme is used, the buffer space needed to prevent packet loss due to buffer overflow also increases linearly along the path. For other work-conserving disciplines, there are similar buffer requirements.

2.2.2 Controlling Traffic Distortion

In Jitter-EDD, Stop-and-Go, and HRR, traffic distortions are controlled at each switch so as to offset the effects of network load fluctuation and interaction between switches.

A comparison between Delay-EDD and Jitter-EDD is shown in Table 2.2.

Delay-EDD	**Jitter-EDD**
$ExA_{i,j} \leftarrow max(ExA_{i,j} + Xmin_j, AT)$	$ExA_{i,j} \leftarrow max(ExA_{i,j} + Xmin_j, AT)$
$Ahead \leftarrow ExA_{i,j} - AT$	$Ahead \leftarrow max(ExA_{i,j} - AT, PreAhead)$
	Hold the packet for *Ahead* time
$Deadline \leftarrow AT + Ahead + \overline{d}_{i,j}$	$Deadline \leftarrow AT + Ahead + \overline{d}_{i,j}$
Stamp packet with *Deadline*	Stamp packet with *Deadline*

Table 2.2: Comparison between Delay-EDD and Jitter-EDD

Notice that the state equations for Delay-EDD are written in a form different from that in Table 2.1. This is only for the sake of exposition. It can be shown that these two representations of the state equations are equivalent. The variable $ExA_{i,j}$ is the expected arrival time of the next packet. It is related to $ExD_{i,j}$ of Table 1 as follows:

$$ExD_{i,j} \Longleftrightarrow ExA_{i,j} + Xmin_j$$

The variable *Ahead* is the amount of time the packet arrives ahead of schedule at the current switch.

As can be seen in Table 2.2, there are two differences between Delay-EDD and Jitter-EDD:

- There is one more term in the calculation of *Ahead* in Jitter-EDD: *PreAhead*. *Prehead* is a value carried in from the previous switch; it tells the switch *how much the packet is ahead of schedule with respect to the previous switch*, i.e., the difference between the deadline and the actual finishing time of the packet in the previous switch.
- Jitter-EDD holds the packet for *Ahead* units of time, and then calculates its deadline and hands it to the scheduler, while Delay-EDD extends the deadline by *Ahead* time units and hands the packet immediately to the scheduler.

It is this holding time that makes Jitter-EDD a non-work-conserving discipline. It has been shown in [86] that Jitter-EDD has the following properties:

1. The traffic pattern of a channel is preserved at each switch in spite of network load fluctuations;
2. Let switch 0 be the source, $\overline{d}_{h,j}$ be the local delay bound assigned to channel j at switch h, and $t^k_{h,j}$ be the service time of the packet at switch h. If the k^{th} packet on connection j enters the network at time $EnterTime$, the earliest time it can arrive at switch i is $(EnterTime + \sum_{h=0}^{i-2} \overline{d}_{h,j} + t^k_{i-1,j} + \sum_{h=1}^{i} \pi_h)$;

   the earliest time and the latest time the packet can leave the switch are $(EnterTime + \sum_{h=0}^{k-1} \overline{d}_{h,j} + t^k_{i,j} + \sum_{h=1}^{i} \pi_h)$ and $(EnterTime + \sum_{h=0}^{i} \overline{d}_{h,j} + \sum_{h=1}^{i} \pi_h)$, respectively.
3. From property 2, it immediately follows that the maximum residence time of a packet on connection j at switch i is $\overline{d}_{i-1,j} + \overline{d}_{i,j} - t^k_{i-1,j}$.
4. From property 3, the maximum number of packets from connection j at switch i is $\lceil \frac{\overline{d}_{i,j} + \overline{d}_{i,j} - t^k_{i-1,j}}{Xmin} \rceil$.

Property 2 provides jitter bounds for channels [86]; property 4 gives buffer space bounds to prevent packet loss.

Jitter-EDD was proposed in the context of delay-jitter control inside the network. Controlling jitter is strictly related to controlling traffic pattern distortions. In the literature, there are different definitions of *jitter*. In [44], the term is used to capture the *burstiness* of the traffic, and is defined to be the maximum number of packets in a *jitter averaging* *interval.* In [24, 86], the term is used to capture the magnitude of the distortion to the traffic pattern caused by the network, and is defined to be the maximum difference between the delays experienced by any two packets on the same connection. In this dissertation, we call these two quantities *rate jitter* and *delay jitter*, respectively. Correspondingly, there are two ways of controlling traffic distortions, rate-jitter control and delay-jitter control.

A zero delay jitter connection is equivalent to a constant delay line. The traffic patterns of a connection at the entrance and the exit of a zero delay jitter connection are exactly the same. Eliminating delay jitter at each switch along the path will *completely* reconstruct the traffic pattern at that switch; thus, if the traffic obeys the traffic specification at the entrance to the network, it will obey the specification throughout the network. This is what is done by Jitter-EDD. Stop-and-Go performs a similar function, except that instead of eliminating delay jitter totally, delay jitter is bounded to two times the frame size at each switch.

Controlling the rate jitter at each switch along the path will only *partially* reconstruct the traffic pattern; the partially reconstructed traffic pattern obeys the same traffic specification as the input traffic, but some inter-packet spacing information is lost [84]. Notice that, if the traffic at the entrance to the network has a small rate jitter, controlling delay jitter inside the network will also keep the rate jitter small; however, controlling rate jitter would not automatically eliminate or reduce delay jitter.

The following example illustrates the difference between rate-jitter control and delay-jitter control. Figure 2.4 is an example of Stop-and-Go, which uses delay-jitter control. Figure 2.5 is an example of Hierarchical Round Robin, which uses rate-jitter control. The time lines in the figures show the traffic pattern of a connection at each of the switches along the path of the connection. In the examples, it is assumed that 3 slots are allocated to the connection in each frame. In Stop-and-Go, as shown in Figure 2.4, packets that are transmitted in the same frame at the entrance to the network stay in the same frame throughout the network. The difference between delays experienced by any two packets from the source to any switch is bounded by $2 * T$, where T is the frame size. In Hierarchical-Round-Robin, as shown in Figure 2.5, packets that are transmitted in the same frame at the entrance to the network do not necessarily stay in the same frame inside the network; however, the property that *no more than three packets are transmitted during one frame time* holds throughout the network.

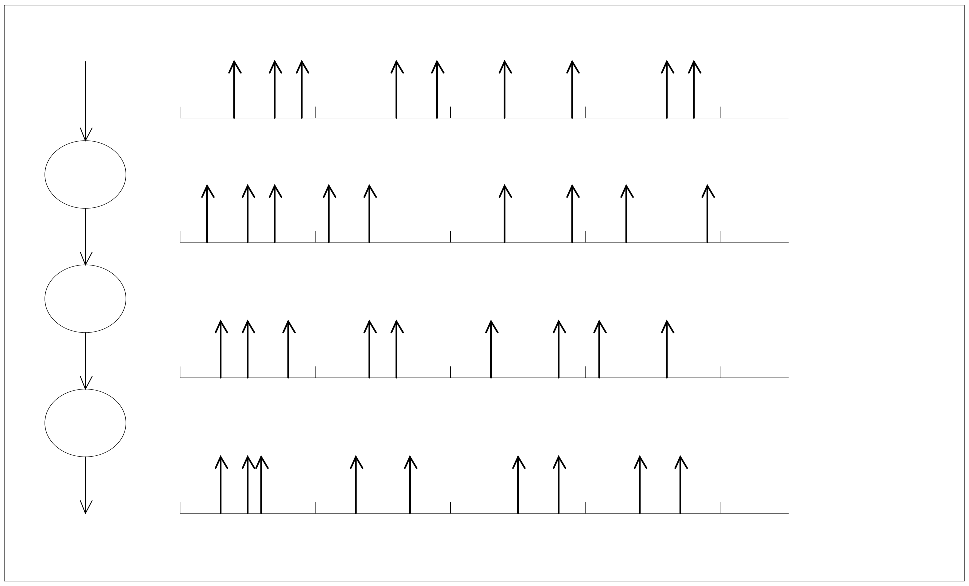


Figure 2.4: Delay-jitter control in Stop-and-Go

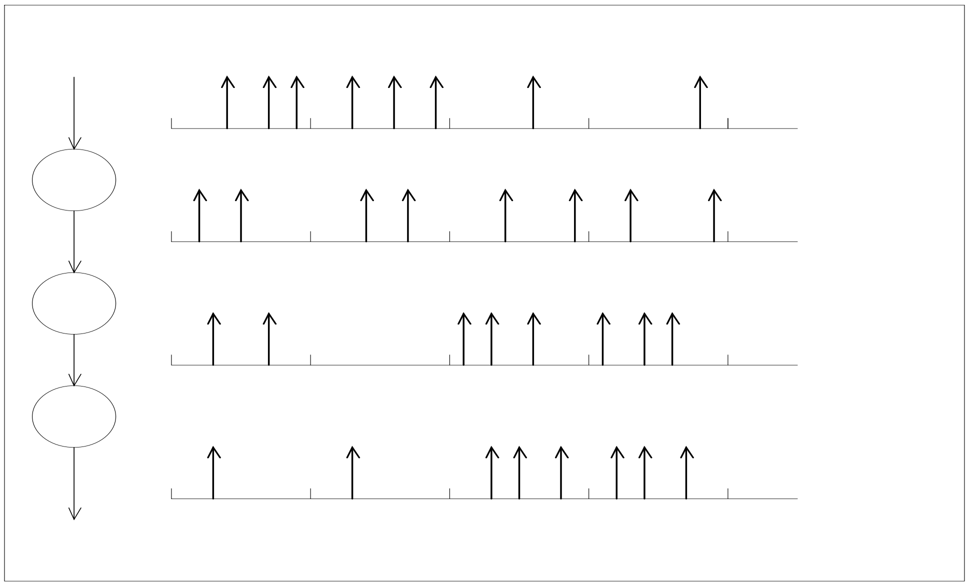


Figure 2.5: Rate-jitter control in HRR

2.2.3 Tradeoffs Discussion

Controlling traffic distortions within the network has several advantages: 1) it needs uniformly distributed buffer space within the network to prevent packet loss; 2) the traffic at the exit of the network satisfies certain desirable properties, for example, bounded rate jitter or delay jitter; 3) it simplifies the analysis of stability.

Table 2.3 shows the buffer space requirements for connection j at the i^{th} switch on the path for the Delay-EDD, Jitter-EDD, Stop-and-Go and HRR disciplines. For Delay-

Delay-EDD	$\frac{\sum_{h=0}^{i} \overline{d}_{h,j}}{Xmin_j}$
Jitter-EDD	$\frac{\overline{d}_{i-1,j}+\overline{d}_{i,j}}{Xmin_j}$
Stop-and-Go	$(2T_i + \theta_k) \times AR_i$
HRR	$2T_i \times AR_i$

Table 2.3: Buffer Space Requirements

EDD and Jitter-EDD, if local delay is assumed to be the same at each switch for connection j, the expressions for the buffer space requirements can be simplified to $\frac{(k+1)\overline{d}_j}{Xmin_j}$ and $\frac{2\overline{d}_j}{Xmin_j}$, respectively. It can be seen that the buffer space requirements for the three non-work-conserving disciplines are almost constant for each node traversed by the channel, while the buffer space requirement for the work-conserving Delay-EDD increases linearly for each node along the path.

Controlling traffic distortions within the network also maintains certain traffic properties at the exit of the network. A delay-jitter controlled network provides a tight upper bound on delay jitter, which is an important performance index for certain applications such as digital video and audio. Although the delay bound for a connection is also the delay-jitter bound, it is usually too loose to be useful. In [69], it is argued that the network needs only to provide a bounded-delay service, as an bounded-delay-jitter service can be built on top of the bounded-delay service by buffering at the destination. The amount of buffer space needed to build a service with delay jitter bounded by J on top of a service that can only bound delay-jitter to J', is $max(J - J', 0) \times Rate$. For a given delay-jitter bound requirement for an application, the looser the delay-jitter bound provided by the network, the more buffer space is needed at the destination.

Controlling traffic distortion also simplifies the stability analysis in arbitrary-topology networks. In a network of switches, the input of a switch is either from a traffic source, or from the output of another switch. Assuming traffic sources satisfy certain traffic characteristics, the key to the analysis of a network is to characterize the output of a switch and iterate the characterization throughout the network [20, 68]. The analysis is usually complicated, also general solutions are usually not available.

Controlling traffic distortion requires non-work-conserving disciplines at switches. Non-work-conserving disciplines may leave the link idle even when there are packets in the queue.

2.3 Taxonomy

As discussed in previous sections, existing solutions can be classified along two dimensions:

1. how the service discipline allocates delay/bandwidth to different connections in a single switch;

2. how the service discipline handles traffic distortion in a network of switches.

There are two approaches to allocate delay/bandwidth to a single switch, 1) a sorted priority mechanism or 2) a multi-level framing strategy. There are also two approaches to handle traffic pattern: 1) accommodating distortion by buffering or 2) controlling distortion by either rate-jitter control or delay-jitter control. Figure 2.6 shows how existing solutions fit in this taxonomy.

As discussed earlier, although sorted priority mechanism has the flexibility to allocate different delay bounds and bandwidths to different connections, it is difficult to be implemented at high speeds. Multi-level framing is easier to implement, but it suffers from the coupling between bandwidth and delay bound allocation.

Non-work-conserving disciplines have the property of maintaining traffic characteristics throughout the network. This results in 1) uniformly distributed buffer space within the network to prevent packet loss; 2) the traffic at the exit of the network satisfies certain desirable properties, for example, bounded rate or delay jitter; 3) and simplified stability analysis.

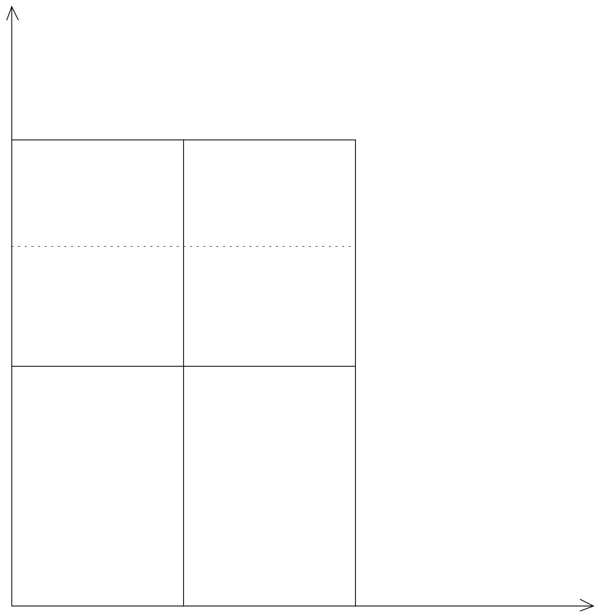


Figure 2.6: Taxonomy of service disciplines

2.4 Other Related Work

In [40, 41], a scheduling policy called MARS is proposed. MARS differs from the previous discussed disciplines in that it only supports three classes of services: video, voice and data. As discussed in Chapter 1, the integrated-services networks will offer services that are far from the simple merging of services provided by the current video, voice and data networks, and we believe that the network should be designed to support a parameterized service interface instead of a fixed type service interface, and this should be reflected in the design of the service discipline.

In [73, 72], a cost-based scheduling algorithm is proposed to support integrated-services networks. The algorithm is also based on the sorted priority queue mechanism. It differs from the other algorithms in that the calculation of the priority index is based on a more general concept of cost function. The algorithm is considered to be the multiplexing policy at the entrance to the network. It is unclear how to extend it to a network of switches.

Chapter 3

Rate-Controlled Service Disciplines: Deterministic Analysis

As discussed in the previous chapters, a service discipline for integrated-services packet-switching networks should address two issues: (1) how to allocate different delay bounds and bandwidths to different connections, and (2) how to deal with traffic distortions within the network.

In this chapter, we present a new class of service disciplines, called *rate-controlled service disciplines*, which address the problem by separating the server into two components: a rate controller and a scheduler. The rate controller monitors the traffic on each connection and forces the traffic to obey the desired traffic pattern. The scheduler orders transmissions of packets on different connections. Thus, the rate controller allocates bandwidth and controls traffic distortion, whereas the scheduler allocates service priorities to packets and controls the delay bounds of connections.

Rate-controlled service disciplines have several advantages that make them suitable for providing performance guarantees in high speed integrated-services networks:

1. End-to-end performance bounds can be obtained in a network of arbitrary topology. Unlike most work-conserving disciplines, which can provide bounds only in feed-forward networks and some restricted classes of feed-back networks, rate-controlled disciplines provide bounds in arbitrary feed-back and feed-forward networks.

2. Unlike most existing solutions, which assume *constant* link delays between switches, we only assume *bounded* link delays when rate-controlled disciplines are used. This

is particularly important in an internetworking environment, where switches are connected by subnetworks. The delays of packets traversing subnetworks may be *bounded*, but are usually *variable*.

3. Given this separation of functionality into rate control and packet scheduling, we can have arbitrary combination of rate-control policies and packet scheduling disciplines.

4. By having a server with two components, we can extend results previously obtained for a single scheduler to a networking environment. Any scheduler that can provide delay bounds to connections at a single switch can be used in conjunction with a rate controller to form a rate-controlled server, which can then be used to provide end-to-end performance bounds in a networking environment.

5. Separation of rate-control and delay-control functions in the design of a server allows decoupling of bandwidth and delay bound allocation to different connections. Most existing solutions have the drawback of coupling the bandwidth/delay bound allocation — allocating of a lower delay bound to a connection automatically allocates a higher bandwidth to the connection. Such solutions cannot efficiently support low delay/low bandwidth connections.

6. Unlike work-conserving disciplines, which require the reservation of more buffer space at the downstream switches traversed by a connection, rate-controlled disciplines also have the advantage of requiring evenly distributed buffer space at each switch to prevent packet loss.

One discipline in this class, called Rate-Controlled Static Priority (RCSP), is particularly suitable for providing performance guarantees in high speed networks. It achieves simplicity of implementation as well as flexibility in the allocation of bandwidth and delay bound to different connections.

The rest of the chapter is organized as follows: in Section 3.1, we present the rate-controlled service discipline by describing the two components of a rate-controlled server: the rate controller and the scheduler; in Section 3.2, we study the end-to-end delay characteristics of connections in a network with rate-controlled servers, assuming that local delay bounds can be guaranteed in the scheduler of each of the rate-controlled servers; in Section 3.3, we give the conditions for providing local delay bounds in two types of schedulers, the

j denotes the connection number
i denotes the switch number
k denotes the packet number
a bar is used to denote an upper bound
d is denotes the local delay
D is denotes the end-to-end delay

$d^k_{i,j}$	delay of the k^{th} packet on connection j at the i^{th} switch along its path
$\overline{d}_{i,j}$	local delay bound for connection j at the i^{th} switch along its path
$\overline{D}_j$:	end-to-end delay bound for connection j
$\pi^k_{i,j}$:	the link delay between the $(i-1)^{th}$ and the i^{th} switch for the k^{th} packet on connection j
$\overline{\pi}_{i,j}$:	the maximum link delay between the $(i-1)^{th}$ and the i^{th} switch for all packets on connection j

Table 3.1: Notation used in this chapter

First-Come-First-Served (FCFS) scheduler and the Static Priority (SP) scheduler; in Section 3.4, we discuss the tradeoffs between two types of regulators, the rate-jitter controlling regulator and the delay-jitter controlling regulator; finally, in Section 3.5, we propose an implementation of RCSP that we believe is simple enough to be able to run at very high speed.

3.1 Rate-Controlled Service Disciplines

Table 3.1 shows the notation used in this chapter. In discussions when there is no ambiguity, some subscripts or superscripts are omitted for simplicity.

A rate-controlled server has two components: a rate controller and a scheduler. The rate controller shapes the input traffic on each connection into the desired traffic pattern by assigning an eligibility time to each packet; the scheduler orders the transmission of eligible real-time packets from all the connections. For the following discussion, we assume that real-time packets have non-preemptive priority over non-real-time packets. The scheduling policies discussed below only apply to the real-time packets. The service order of non-real-time packets is not specified, and is irrelevant in this discussion.

Conceptually, a rate controller consists of a set of regulators corresponding to each of the connections traversing the switch; each regulator is responsible for shaping the traffic of the corresponding connection into the desired traffic pattern. Regulators control the interactions between switches and eliminate jitter. Many types of regulators are possible;

they will be discussed in Section 3.4. In this section, we present two types of regulators:

1. *rate-jitter controlling regulator*, or RJ regulator, which controls rate jitter by partially reconstructing the traffic pattern;

2. *delay-jitter controlling regulator*, or DJ regulator, which controls delay jitter by fully reconstructing the traffic pattern.

The regulator achieves this control by assigning each packet an eligibility time upon its arrival and holding the packet till that time before handing it to the scheduler. Different ways of calculating the eligibility time of a packet will result in different types of regulators.

For a $(Xmin, Xave, I)$ RJ regulator, where $Xmin \leq Xave < I$ holds, the eligibility time of the k^{th} packet on a connection at the i^{th} switch along its path, ET_i^k, is defined with reference to the eligibility times of packets arriving earlier at the switch on the same connection:

$$ET_i^k = -I, \quad k < 0 \tag{3.1}$$

$$ET_i^1 = AT_i^1 \tag{3.2}$$

$$ET_i^k = max(ET_i^{k-1} + Xmin,\ ET_i^{k-\lfloor \frac{I}{Xave} \rfloor + 1} + I,\ AT_i^k), \quad k > 1 \tag{3.3}$$

where AT_i^k is the time the k^{th} packet on the connection arrived at switch i. (3.1) is defined for convenience so that (3.3) holds for any $k > 1$.

From this definition, we can see that

$$ET_i^k \geq AT_i^k \tag{3.4}$$

always holds, i.e., a packet is never eligible before its arrival. Also, if we consider the sequence of packet eligibility times at switch i, $\{ET_i^k\}_{k=1,2,\ldots}$, it always satisfies the $(Xmin, Xave, I)$ traffic characterization.

Proposition 3.1 *Assume a connection traverses a $(Xmin, Xave, I)$ RJ regulator, and the maximum packet size of the connection is bounded by $Smax$. The output traffic of the regulator satisfies the $(Xmin, Xave, I, Smax)$ specification.*

The eligibility time of a packet for a DJ regulator is defined with reference to the eligibility time of the same packet at the immediately upstream switch. The definition assumes that the queueing delays of packets on the connection, and the link delay from the upstream switch to the current switch, are bounded. Let $\overline{d}_{i-1}$ be the local delay bound for the connection in the scheduler at switch $i-1$, and $\overline{\pi}_i$ be the maximum link delay from switch $i-1$ to switch i. For a delay-jitter controlling regulator, ET_i^k, the eligibility time of the k^{th} packet on a connection that traverses switch i is defined as:

$$ET_0^k = AT_0^k \tag{3.5}$$

$$ET_i^k = ET_{i-1}^k + \overline{d}_{i-1} + \overline{\pi}_i, \quad i > 0 \tag{3.6}$$

where switch 0 is the source of the connection.

Thus, for a DJ regulator:

- inequality (3.4) holds, i.e., a packet is never eligible before its arrival;
- also we have the following:

$$ET_i^{k+1} - ET_i^k = AT_0^{k+1} - AT_0^k \ \ \forall k, i \geq 0, \tag{3.7}$$

  i.e., the traffic pattern on a connection at the output of the regulator of every switch traversed by the connection is exactly the same as the traffic pattern of the connection at the *entrance* to the network.

From (3.7), the following immediately follows:

Proposition 3.2 *Assume a connection traverses a set of switches with rate-controlled servers having DJ regulators. If its traffic obeys the specification* Θ *at the entrance to the network, the output traffic from the connection's regulator, or the input traffic into the connection's scheduler at each switch, will also obey* Θ.

This proposition is more general than Proposition 3.1. It applies to any traffic specification Θ, rather than just $(Xmin, Xave, I, Smax)$. For example, Θ can be the (σ, ρ) model as proposed in [19], or even stochastic models like Markov-Modulated Poisson Process (MMPP) models [59]. As will be discussed in Chapter 4, this property of completely reconstructing the traffic pattern allows us to extend local statistical performance bounds to end-to-end statistical performance bounds.

From Propositions 3.1 and 3.2, we can see that both types of regulators enforce the traffic specification requirement for each connection, so that the traffic going into the scheduler will always satisfy the traffic specification.

Any scheduler can be used in a rate-controlled server as long as it can provide local delay bounds for connections under certain admission control conditions. Many schedulers, including the most common First Come First Served (FCFS) discipline, can be used. We will give the admission control conditions for FCFS and Static Priority (SP) schedulers in Section 3.3.

3.2 End-To-End Delay Characteristics

The end-to-end delay of a packet consists of the link delays the packet experienced and the residence times of the packet in each of the switches along the path. The residence time of a packet in a switch with rate-controlled servers has two components: the *holding* time in the regulator and the *waiting time* in the scheduler. In this section, we will show that the end-to-end delays of all the packets on a connection can be bounded, as long as the delays on the links and the delays at each of the schedulers can be bounded. Holding in the rate controllers will *not* increase the *end-to-end delay bound* of the connection. Formally, we have the following theorem (End-to-End Theorem):

Theorem 3.1 *Consider a connection passing through n switches connected in cascade, with $\overline{\pi}_i$ and $\hat{\pi}_i$ being the upper and lower bounds on the delay of the link between the $i-1^{th}$ and the i^{th} switch. Assume that the scheduler of switch i can guarantee that the delays of all the packets on the connection be bounded by $\overline{d_i}$ as long as the connection's input traffic to the scheduler satisfies the given $(Xmin, Xave, I, Smax)$ specification. If the traffic on the connection obeys the $(Xmin, Xave, I, Smax)$ specification at the entrance to the first switch,*

1. *the end-to-end delay for any packet on the connection is bounded by $\sum_{i=1}^{n} \overline{d}_i + \sum_{i=2}^{n} \overline{\pi}_i$ if rate-jitter controlling regulators are used;*

2. *the end-to-end delay and the delay jitter for any packet are bounded by $\sum_{i=1}^{n} \overline{d}_i + \sum_{i=2}^{n} \overline{\pi}_i$ and $\overline{d}_{i_n}$, respectively, if delay-jitter controlling regulators are used;*

3. *reservation of the following amount of buffer space for the connection at switch i will prevent packet loss: $(\lceil \frac{\overline{d}_{i-1}+\overline{\pi}_i-\hat{\pi}_i}{Xmin} \rceil + \lceil \frac{\overline{d}_i}{Xmin} \rceil) Smax, \quad i = 1, \ldots, n; \ \overline{d}_0 = 0$*

Before we give the proof, we first briefly discuss the implications of the result. At first glance, the result seems trivial, because it states that an end-to-end delay bound can be guaranteed if local delay bounds can be guaranteed at each scheduler and link. However, several things should be noticed:

- If a switch's server does not include rate controller, but has only the scheduler, the result will not hold. Under the assumptions of the theorem, a local delay bound can be guaranteed for a connection only if the connection's *input traffic to the scheduler* satisfies the $(Xmin, Xave, I, Smax)$ specification. Even though the traffic satisfies that specification at the entrance to the network, without rate controllers the traffic pattern will be distorted inside the network, and the $(Xmin, Xave, I, Smax)$ specification may *not* be satisfied in subsequent schedulers.

- The theorem just assumes that there is an upper bound on delays in the scheduler; the holding times in the rate controllers have not be accounted for. The proof of the theorem needs to establish that holding in the rate controllers will not increase the end-to-end delay bound.

- The theorem assumes links with bounded, but possibly *variable*, delay. Most of the existing solutions proposed to bound end-to-end delay in a networking environment assume links with *constant* delay [20, 21, 51, 66, 68, 38, 6]. Constant delay links have the nice property that the traffic pattern at the receiving end of the link is the same as that at the transmitting end of the link. This property is important for existing bounding solutions because central to their analysis is the technique of characterizing output traffic from a scheduler and using it as the input traffic to the next-hop scheduler. However, in an internetworking environment, links connecting switches [1] may be subnetworks such as ATM or FDDI networks. It is possible to bound delay over these subnetworks; however, the delays for different packets will be *variable*. Rate-controlled service disciplines can provide end-to-end performance bounds not only in simple networks, where links have constant delays, but also in an internetworking environment, where subnetworks have bounded but variable delays.

- The theorem holds for *any* schedulers that can guarantee local delay bounds for connections satisfying certain traffic specifications. This has two implications: 1) any scheduler that can guarantee delay bounds to connections at a single switch can be used in conjunction with a rate controller to form a rate-controlled server, which can

[1]Switching elements in an internetworking environment are usually called routers or gateways.

then be used to provide end-to-end performance bounds in a networking environment; 2) rate-controlled servers with different schedulers can be used in a network, and end-to-end performance bounds can still be guaranteed. This is particularly important in an internetworking environment, where heterogeneity is unavoidable.

3.2.1 Two-Node Lemma

In the following lemma, we establish the delay characteristics of two switches connected in cascade, when each type of regulator is used.

Lemma 3.1 *Consider a connection with traffic specification $(Xmin, Xave, I, Smax)$ traversing two switches using rate-controlling service disciplines, and labeled by $i-1$ and i, respectively. For the k^{th} packet on the connection, let d_{i-1}^k be its delay in the scheduler of switch $i-1$, π_i^k its link delay between from the $(i-1)^{th}$ to the i^{th} switch, and h_i^k its holding time in the regulator of switch i. If $d_{i-1}^k \leq \overline{d}_{i-1}$ and $\pi_i^k \leq \overline{\pi}_i$ for all k's, we have that*

1. *if a delay-jitter controlling regulator is used at switch i,*

$$d_{i-1}^k + h_i^k + \pi_i^k = \overline{d}_{i-1} + \overline{\pi}_i \tag{3.8}$$

2. *if a rate-jitter controlling regulator is used at switch i,*

$$d_{i-1}^k + h_i^k + \pi_i^k \leq \overline{d}_{i-1} + \overline{\pi}_i \tag{3.9}$$

Proof.

Let ET_{i-1}^k and ET_i^k be the eligibility times for the k^{th} packet at switch $i-1$ and i, respectively. DT_{i-1}^k is the departure time of the k^{th} packet from switch $i-1$, and AT_i^k is the arrival time of the k^{th} packet at switch i. We have

$$d_{i-1}^k = DT_{i-1}^k - ET_{i-1}^k \tag{3.10}$$

$$h_i^k = ET_i^k - AT_i^k \tag{3.11}$$

$$\pi_i^k = AT_i^k - DT_{i-1}^k \tag{3.12}$$

Combining (3.10), (3.11), and (3.12), we have

$$d_{i-1}^k + h_i^k + \pi_i^k = ET_i^k - ET_{i-1}^k \tag{3.13}$$

1. For the case of a delay-jitter controlling regulator, from (3.13) and $ET_i^k - ET_{i-1}^k = \overline{d}_{i-1} + \overline{\pi}_i$, which is (3.6), we immediately have

$$d_{i-1}^k + h_i^k + \pi_i^k = \overline{d}_{i-1} + \overline{\pi}_i$$

2. For the case of a rate-jitter-controlling regulator, we will prove the lemma by induction with respect to k.

Step 1. When k=1, from (3.2), we have $ET_i^1 = AT_i^1$. It follows that

$$h_i^1 = ET_i^1 - AT_i^1 = 0 \tag{3.14}$$

Also we have

$$d_{i-1}^1 \leq \overline{d}_{i-1} \tag{3.15}$$

$$\pi_i^1 \leq \overline{\pi}_i \tag{3.16}$$

Summing (3.14), (3.15) and (3.16), we have

$$d_{i-1}^1 + h_i^1 + \pi_i^1 \leq \overline{d}_{i-1} + \overline{\pi}_i \tag{3.17}$$

So, (3.9) holds for k=1.

Step 2. Assume that (3.9) holds for the first k packets; we now consider the $(k+1)^{st}$ packet. From (3.3) we have,

$$ET_i^{k+1} = max(ET_i^k + Xmin, ET_i^{k-\lfloor \frac{I}{Xave} \rfloor + 2} + I, AT_i^{k+1}), \quad k > 1 \tag{3.18}$$

Since ET_i^{k+1} is the maximum of the three quantities at the right hand side of (3.18), we consider each of the following three cases in turn:

(a) $ET_i^{k+1} = AT_i^{k+1}$

(b) $ET_i^{k+1} = ET_i^k + Xmin$,

(c) $ET_i^{k+1} = ET_i^{k-\lfloor \frac{I}{Xave} \rfloor + 2} + I$

Case **(a):** If $ET_i^{k+1} = AT_i^{k+1}$, we have

$$h_i^{k+1} = ET_i^{k+1} - AT_i^{k+1} = 0 \tag{3.19}$$

Also,

$$d_{i-1}^{k+1} \leq \overline{d}_{i-1} \tag{3.20}$$

$$\pi_i^{k+1} \leq \overline{\pi}_i \tag{3.21}$$

It immediately follows that

$$w_{i-1}^{k+1} + h_i^{k+1} + \pi_i^{k+1} \leq \overline{d}_{i-1} + \overline{\pi}_i \tag{3.22}$$

Case **(b)**: If $ET_i^{k+1} = ET_i^k + Xmin$, we have

$$\begin{aligned}
d_{i-1}^{k+1} + h_i^{k+1} + \pi_i^{k+1} &= ET_i^{k+1} - ET_{i-1}^{k+1} && (3.23)\\
&= (ET_i^k + Xmin) - ET_{i-1}^{k+1} && (3.24)\\
&\leq (ET_i^k + Xmin) - (ET_{i-1}^k + Xmin) && (3.25)\\
&= ET_i^k - ET_{i-1}^k && (3.26)\\
&\leq \overline{d}_{i-1} + \overline{\pi}_i && (3.27)
\end{aligned}$$

(3.25) holds due to $ET_{i-1}^{k+1} \geq ET_{i-1}^k + Xmin$. (3.27) is derived from the assumption that (3.9) holds for the first k packets.

Case **(c)**: If $ET_i^{k+1} = ET_i^{k-\lfloor\frac{I}{Xave}\rfloor+2} + I$, we have,

$$\begin{aligned}
d_{i-1}^{k+1} + h_i^{k+1} + \pi_i^{k+1} &= ET_i^{k+1} - ET_{i-1}^{k+1} && (3.28)\\
&= (ET_i^{k-\lfloor\frac{I}{Xave}\rfloor+2} + I) - ET_{i-1}^{k+1} && (3.29)\\
&\leq (ET_i^{k-\lfloor\frac{I}{Xave}\rfloor+2} + I) - (ET_{i-1}^{k-\lfloor\frac{I}{Xave}\rfloor+2} + I) && (3.30)\\
&= ET_i^{k-\lfloor\frac{I}{Xave}\rfloor+1} - ET_{i-1}^{k-\lfloor\frac{I}{Xave}\rfloor+1} && (3.31)\\
&\leq \overline{d}_{i-1} + \overline{\pi}_i && (3.32)
\end{aligned}$$

(3.30) holds due to the fact that the traffic pattern into the scheduler of switch $i-1$ also satisfies the $(Xmin, Xave, I, Smax)$ specification, hence $ET_{i-1}^{k+2} \geq ET_{i-1}^{k-\lfloor\frac{I}{Xave}\rfloor+2} + I$. (3.32) is derived from the assumption that (3.9) holds for the previous k packets.

From (a), (b) and (c), we have proven that (3.9) holds for the $k+1^{st}$ packet if it holds for the first k packets. Thus, (3.9) holds for any packet on the connection. **Q.E.D.**

The Two-Node Lemma is an important step in establishing Theorem 3.1. Intuitively, a packet is held in a regulator only when the packet was transmitted ahead of schedule by the previous switch, or, when the packet experienced less delay over the link than the maximum link delay. The amount of holding time in the regulator is never greater than the amount of time the packet is ahead of schedule plus the difference between the maximum link delay and the actual link delay. Thus, holding does not increase the *accumulative delay bound*. The result is obvious for delay-jitter controlling regulators, since the eligibility time of a packet at a switch is defined with respect to its eligibility time in the previous switch. In a sense, Equations (3.5) and (3.6), which are the definition of the eligibility time of a packet for delay-jitter controlling regulators, have already had the result built in. However, for a rate-jitter controlling regulator, the result is non-trivial, since the definition of the eligibility time of a packet is based on the packet spacing requirement in the same switch.

The idea of holding a packet by the amount of time that it is ahead of schedule in the previous switch was first proposed in [86]. The relationship between the regulators defined in [86] and the delay-jitter controlling regulators is discussed in Section 3.4.

3.2.2 Proof of the End-To-End Theorem

With Lemma 3.1 proven, we are now ready to prove the main result of the section, Theorem 3.1.

Proof of Theorem 3.1.

For the first two parts of the theorem, consider the end-to-end delay of the k^{th} packet on the connection, D^k, which can be expressed as:

$$D^k = \sum_{i=1}^{n}(h_i^k + w_i^k) + \sum_{i=0}^{n-1}\overline{\pi}_i \tag{3.33}$$

where h_i^k and w_i^k are the holding and the waiting times at switch i, respectively. If we rearrange the terms, (3.33) becomes:

$$D^k = h_1^k + \sum_{i=2}^{n}(w_{i-1}^k + h_i^k + \overline{\pi}_i) + w_n^k \tag{3.34}$$

If the traffic obeys the $(Xmin, Xave, I, Smax)$ characterization at the entrance to the first switch, there is no holding time in the first regulator, or $h_1^k = 0$. The D_k can be further simplified to be:

$$D^k = \sum_{i=2}^{n-1}(w_{i-1}^k + h_i^k + \overline{\pi}_i) + w_n^k \tag{3.35}$$

(1) If rate-jitter controlling regulators are used, according to Proposition 3.1 the traffic obeys the $(Xmin, Xave, I, Smax)$ characterization at the entrance to each of the schedulers. From Lemma 3.1, we have

$$w_{i-1}^k + h_i^k + \pi_i^k \leq \overline{d}_{i-1} + \overline{\pi}_i \tag{3.36}$$

Combining (3.35) and (3.36), we have,

$$\begin{aligned} D^k &= \sum_{i=2}^{n}(w_{i-1}^k + h_i^k + \overline{\pi}_i) + w_n^k && (3.37)\\ &\leq \sum_{i=2}^{n}(\overline{d}_{i-1} + \overline{\pi}_i) + \overline{d}_n && (3.38)\\ &= \sum_{i=1}^{n} d_i + \sum_{i=2}^{n}\overline{\pi}_i && (3.39)\end{aligned}$$

(2) If delay-jitter controlling regulators are used, according to Proposition 3.2 the traffic obeys the $(Xmin, Xave, I, Smax)$ characterization at the entrance to each of the schedulers. From Lemma 3.1, we have

$$w_{i-1}^k + h_i^k + \pi_i^k = d_{i-1} + \overline{\pi}_i \qquad (3.40)$$

Combining (3.35) and (3.40), we have,

$$D^k = \sum_{i=2}^{n}(d_{i-1} + \overline{\pi}_i) + w_n^k \qquad (3.41)$$

$$0 < w_n^k \leq d_n \qquad (3.42)$$

and

$$\sum_{i=1}^{n-1} d_i + \sum_{i=2}^{n}\overline{\pi}_i < D^k \leq \sum_{i=1}^{n} d_i + \sum_{i=2}^{n}\overline{\pi}_i \qquad (3.43)$$

(3) To verify the third part of the theorem, notice that the longest times a packet can stay in the regulator and the scheduler of the i^{th} switch are $d_{i-1} + \overline{\pi}_i - \hat{\pi}_i$ and d_i, respectively; since the minimum packet inter-arrival time is $Xmin$, it follows that the maximum numbers of packets in the regulator and the scheduler are $\lceil \frac{d_{i-1}+\overline{\pi}_i-\hat{\pi}_i}{Xmin} \rceil$ and $\lceil \frac{d_i}{Xmin} \rceil$, respectively. **Q.E.D.**

3.3 Bounding Delay in Schedulers

In the previous section, we established the end-to-end delay characteristics of a connection traversing switches with rate-controlled service disciplines, assuming that local delay bounds can be guaranteed in the scheduler of each of the rate-controlled servers. In this section, we derive admission control conditions that guarantee deterministic delay bounds for several schedulers.

In [31], conditions are given for a variation of the Earliest-Due-Date discipline. One of the conditions is called *deterministic test* [2]:

$$\sum_{j=1}^{n} \frac{Smax_j}{Xmin_j} \leq l \tag{3.44}$$

where l is the link speed and n is the number of channels traversing the simplex link. Although, the tests guarantee that deterministic delay bounds can be provided, they are rather restrictive in the sense that the sum of the peak rates of all real-time connections is limited to be no greater than the link speed. This works well for smooth traffic. However, many clients requiring performance guarantees have bursty sources, for example, those producing compressed digital video. If the condition of the *deterministic test* has to be satisfied, new requests will be rejected when the sum of the peak rates of all the connections reaches the link's speed. In this case, if the peak-to-average-rate ratio is high, the average link utilization by real-time traffic will be low.

Therefore, it is important to derive solutions that can provide performance guarantees even when the sum of the peak rates of all the connections is greater than the link's speed, and to understand the relationship between server utilization and traffic burstiness. In this section, we present an analysis that provides delay bounds for the First-Come-First-Served (FCFS) discipline and the Static Priority (SP) discipline in even when the sum of the peak rates of all the connections is larger than 1.

3.3.1 Bounding Delays in a FCFS Scheduler

In this section, we derive the admission control conditions for providing delay bounds in a FCFS scheduler. We use the bounding techniques developed by Cruz [19]. In [19], a fluid traffic model (σ, ρ) is used. A connection satisfies traffic specification (σ, ρ) if, for any time interval of length u, the number of bits arriving during the interval is no more than $\sigma + \rho u$. The model we use is the discrete $(Xmin, Xave, I, Smax)$ model. A discrete model has the advantage that it better characterizes multimedia sources than fluid approximations, thus giving tighter performance bounds [56].

Figure 3.1 illustrates some of the concepts used in the analysis. The horizontal axis is the time. The upper curve in the figure is the sum of the bits that have arrived

[2]The deterministic test in [31] is $\sum_{j=1}^{n} \frac{t_j}{Xmin_j} \leq 1$. Here we consider the output link as the server, where $t_j = \frac{Smax_j}{l}$, thus (3.44) is equivalent to the deterministic test.

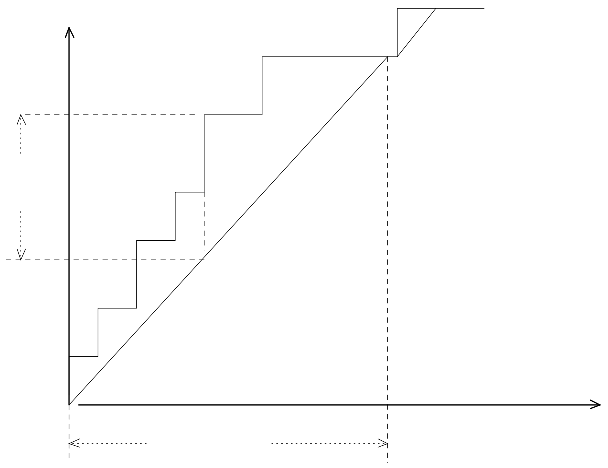


Figure 3.1: Concepts: delay, backlog and busy period

since the origin of time. Each arrival of a packet causes an upward jump in this curve. The lower curve is the number of bits that have been transmitted. The difference between the two curves is the *backlog* function. A work-conserving server always transmits packets at a constant rate when there is a backlog. The points where the arrival curve and the service curve meet, or where the backlog is zero, divide the time axis into *busy periods* and *idle periods*. Within such a framework, the following two propositions immediately follow.

Proposition 3.3 *For a work-conserving real-time scheduler, if for* any *realization of the input traffic that satisfies a given traffic constraint, the maximum length of a busy period is no greater than* $\overline{d}$*, then* $\overline{d} + \frac{\overline{Smax}}{l}$ *is an upper bound for the delays of all packets, where* $\overline{Smax}$ *is the maximum packet size that can be transmitted over the link, and* l *is the link speed.*

Proposition 3.4 *For a FCFS real-time scheduler, if for* any *realization of the input traffic that satisfies a given traffic specification, the maximum real-time traffic backlog divided by the link speed is no greater than* $\overline{d}$*, then* $\overline{d} + \frac{\overline{Smax}}{l}$ *is an upper bound for the delays of all packets, where* $\overline{Smax}$ *is the maximum packet size that can be transmitted over the link, and* l *is the link speed.*

To take into account the effect of non-real-time packets, which have lower priority than real-time packets, but cannot be preempted after the beginning of their transmission, $\frac{\overline{Smax}}{l}$ is included in the delay bounds.

Notice that, in both propositions, in order to derive the delay bound, certain upper bounds values (the length of busy period or the backlog) need to be calculated for *any* realization of the input traffic. It is impossible to calculate these upper bounds for every realization of the input traffic because there are infinite such realizations. If we can find one "worst-case" realization of the input traffic, and prove that this realization gives the maximum upper bound values, we can just analyze the scheduler for this realization and compute the delay bound.

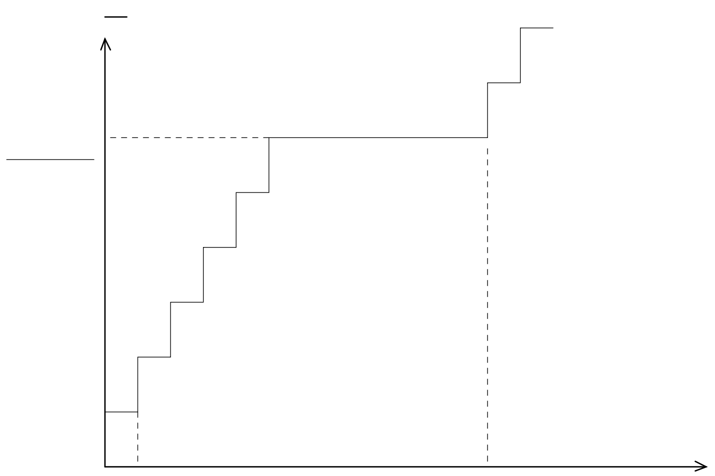


Figure 3.2: Traffic constraint function for (Xmin, Xave, I, Smax) specification

Central to the analysis is the concept of *traffic constraint function* $\overline{b}(.)$ [3]. $\overline{b}_j(u)$ is defined to be the maximum number of bits that can arrive on connection j during *any* interval of length u. For a traffic source that obeys the model $(Xmin_j, Xave_j, I_j, Smax_j)$, it is easy to show that

$$\overline{b}_j(u) = (min(\lceil \frac{u \ mod \ I_j}{Xmin_j} \rceil, \lceil \frac{I_j}{Xave_j} \rceil) + \lceil \frac{u}{I_j} \rceil \times \lceil \frac{I_j}{Xave_j} \rceil) \times Smax_j \tag{3.45}$$

Figure 3.2 illustrates the function $\overline{b}(.)$ for a channel with traffic specification $(Xmin, Xave, I, Smax)$.

The following theorem bounds the delay for a FCFS scheduler.

Theorem 3.2 *Let there be n channels multiplexed on a link with a FCFS scheduler and link speed l. If for $j = 1, \cdots, n$, the traffic on channel j is bounded by $\overline{b}_j(.)$, then the delays*

[3]We use a notation that differs slightly from that in [19], where $b(.)$ is used to denote the traffic constraint function. In this dissertation, $\overline{b}(.)$ is used to denote the traffic constraint function, and $b^{\xi}(s,t)$ denotes the number of bits that arrive between time instants s and t for an input traffic realization ξ.

of packets on all the channels are bounded by $\overline{d}$, where $\overline{d}$ is defined by

$$\overline{d} = \frac{1}{l} \max_{\forall u \geq 0} \{ \sum_{j=1}^{n} \overline{b}_j(u) - l \times u + \frac{\overline{Smax}}{l} \} \tag{3.46}$$

Proof.

Let ξ be a realization of the input traffic. If s is the starting time instant of a busy period, and t is a time instant within the busy period, define $b_j^{\xi}(s,t)$ to be the number of bits that arrive during the interval (s,t) on connection j. The maximum backlog in this realization ξ is:

$$\max_{\forall s,t} \{ \sum_{j=1}^{n} b^{\xi}(s,t) - l \times (t-s) \} \tag{3.47}$$

We have

$$\frac{1}{l} \max_{\forall s,t} \{ \sum_{j=1}^{n} b^{\xi}(s,t) - l \times (t-s)) \} \leq \frac{1}{l} \max_{\forall s,t} \{ \sum_{j=1}^{n} \overline{b}_j(t-s) - l \times (t-s)) \} \tag{3.48}$$

$$\leq \frac{1}{l} \max_{\forall u \geq 0} \{ \sum_{j=1}^{n} \overline{b}_j(u) - l \times u \} \tag{3.49}$$

Since ξ is an arbitrary realization of the input traffic, from Proposition 3.4 $\overline{d}$ as defined by Equation (3.46) is a delay bound for packets from all channels. **Q.E.D.**

Since $b_j(.)$ as defined in (3.45) is a piecewise linear function, evaluating (3.46) is equivalent to optimizing a piecewise linear function. Such an optimization can be done efficiently. In the following two corollaries, we give closed form solutions for two special cases. Corollary 3.1 considers the case when $\sum_{j=1}^{n} \frac{Smax_j}{Xmin_j} \leq l$, Corollary 3.2 considers the case of homogeneous sources when $\sum_{j=1}^{n} \frac{Smax_j}{Xmin_j} > l$.

Corollary 3.1 *Let there be n channels multiplexed on a link with a FCFS scheduler and link speed l. Assume channel j obeys the traffic specification $(Xmin_j, Xave_j, I_j, Smax_j)$, $(j = 1, \cdots, n)$. If $\sum_{j=1}^{n} \frac{Smax_j}{Xmin_j} \leq l$, then the delays of packets on all the channels are bounded by $\overline{d}$, where $\overline{d}$ is given by*

$$\overline{d} = \frac{1}{l} \sum_{j=1}^{n} Smax_j + \frac{\overline{Smax}}{l} \tag{3.50}$$

Proof.

$$\overline{d} = \frac{1}{l} \max_{u \geq 0} \{ \sum_{j=1}^{n} \overline{b}_j(u) - l \times u \} + \frac{\overline{Smax}}{l} \tag{3.51}$$

$$\leq \frac{1}{l}\max_{u\geq 0}\{\sum_{j=1}^{n}\lceil\frac{u}{Xmin_j}\rceil Smax_j - l\times u\} + \frac{\overline{Smax}}{l} \tag{3.52}$$

$$\leq \frac{1}{l}\max_{u\geq 0}(\sum_{j=1}^{n}(\frac{u}{Xmin_j}+1)Smax_j - l\times u) + \frac{\overline{Smax}}{l} \tag{3.53}$$

$$= \frac{1}{l}\max_{u\geq 0}\{\sum_{j=1}^{n}Smax_j + (\sum_{j=1}^{n}\frac{1}{Xmin_j}\times\frac{Smax_j}{l} - 1)\times l\times u\} + \frac{\overline{Smax}}{l} \tag{3.54}$$

$$\leq \frac{1}{l}\sum_{j=1}^{n}Smax_j + \frac{\overline{Smax}}{l} \tag{3.55}$$

Q.E.D.

In the proof, (3.52) holds due to the assumption that the minimum inter-packet spacing is $Xmin_j$. All three inequalities (3.52), (3.53) and (3.55) become equalities when $u = 0$, i.e., the worst case backlog is at the beginning of a busy period, when every connection sends out a maximum length packet.

Corollary 3.2 *Let there be n homogeneous channels with traffic specification (Xmin, Xave, I, Smax) multiplexed on a link with the FCFS service discipline and link speed l. If* $n\times\frac{Smax}{Xave}\leq l$ *and* $n\times\frac{Smax}{Xmin} > l$*, the delays of packets from all the channels are bounded by* $\overline{d}$*, where* $\overline{d}$ *is given by*

$$d = Xmin + (\mu_{ave} - \frac{1}{burstRatio})I + \frac{\overline{Smax}}{l} \tag{3.56}$$

where $\mu_{ave} = n\times\frac{Smax}{Xave}\times\frac{1}{l}$*, and* $burstRatio = \frac{Xave}{Xmin}$.

Proof.

Let $\hat{I}$ be the length of the longest interval that packets from one connection can arrive at the scheduler with $Xmin$ spacing, we have,

$$\hat{I} = \min\{u \mid \overline{b}(u) = \overline{b}(I)\} \tag{3.57}$$

$$= (\frac{I}{Xave} - 1)Xmin \tag{3.58}$$

It is easy to see that $\sum_{j=1}^{n}(\overline{b}_j(u) - l\times u)$ is maximized at $u = \hat{I}$.
From Theorem 3.2, we have,

$$\overline{d} = \frac{1}{l}\sum_{j=1}^{n}(b_j(\hat{I}) - l\times\hat{I}) + \frac{\overline{Smax}}{l} \tag{3.59}$$

$$= \frac{1}{l}(\sum_{j=1}^{n} Smax\frac{I}{Xave}) - \hat{I} + \frac{\overline{Smax}}{l} \quad (3.60)$$

$$= \mu_{ave} \times I - \hat{I} + \frac{\overline{Smax}}{l} \quad (3.61)$$

$$= \mu_{ave} \times I - (\frac{I}{Xave} - 1)Xmin + \frac{\overline{Smax}}{l} \quad (3.62)$$

$$= Xmin + (\mu_{ave} - \frac{1}{burstRatio})I + \frac{\overline{Smax}}{l} \quad (3.63)$$

Q.E.D.

In (3.56), μ_{ave} is the average utilization of the link, and $\frac{Xave}{Xmin}$ and I represent the burstiness of the connections. Intuitively, the more loaded the link, and the burstier the connection traffic, the larger the delay bound. This is exactly what is shown by (3.56). It is important to see that the averaging interval I affects the traffic burstiness. For the given values of $Xmin$ and $Xave$, I determines how long the source can continuously send packets at the peak rate in the worst case. This is illustrated in Equation (3.58).

We now show some numerical examples to illustrate the results presented in this section. Figure 3.3 shows the effects of the traffic burstiness and the average utilization on delay bounds. The horizontal axis is the peak-to-average-rate ratio of the channels, the vertical axis is the delay bound. The figures only show cases when the sum of the peak rates of all the channels is greater than 1. The first thing to be noticed from these figures is that, when the average utilization μ_{ave} and the averaging interval I are fixed, the delay bound that can be provided increases as the peak-to-average-rate ratio increases, i.e., burstier traffic results in a larger delay bound. A dual result is that, for the same delay bound, a higher average utilization can be achieved when the traffic is less bursty. This is shown in Figure 3.4. Figures 3.3(a) and 3.3(b) show the results with fixed averaging interval I but different average utilizations. It can be seen that, when the averaging interval and the peak-to-average-rate ratio are fixed, a higher average utilization of the link results in a larger delay bound, i.e., the more loaded the link, the larger the delay bound. Figures 3.3(c) and 3.3(d) show the results for fixed utilization but different I's. The values of I are chosen to be multiples of 33 ms, which is the time interval between two frames in a 30 frames/sec video stream. It can be seen that, when the average utilization and the peak-to-average rate ratio are fixed, a larger averaging interval results in a larger delay bound. The intuition is that, with the same peak-to-average-rate ratio, a larger averaging interval means a burstier traffic; under the same average utilization of the link, a burstier traffic results in a larger

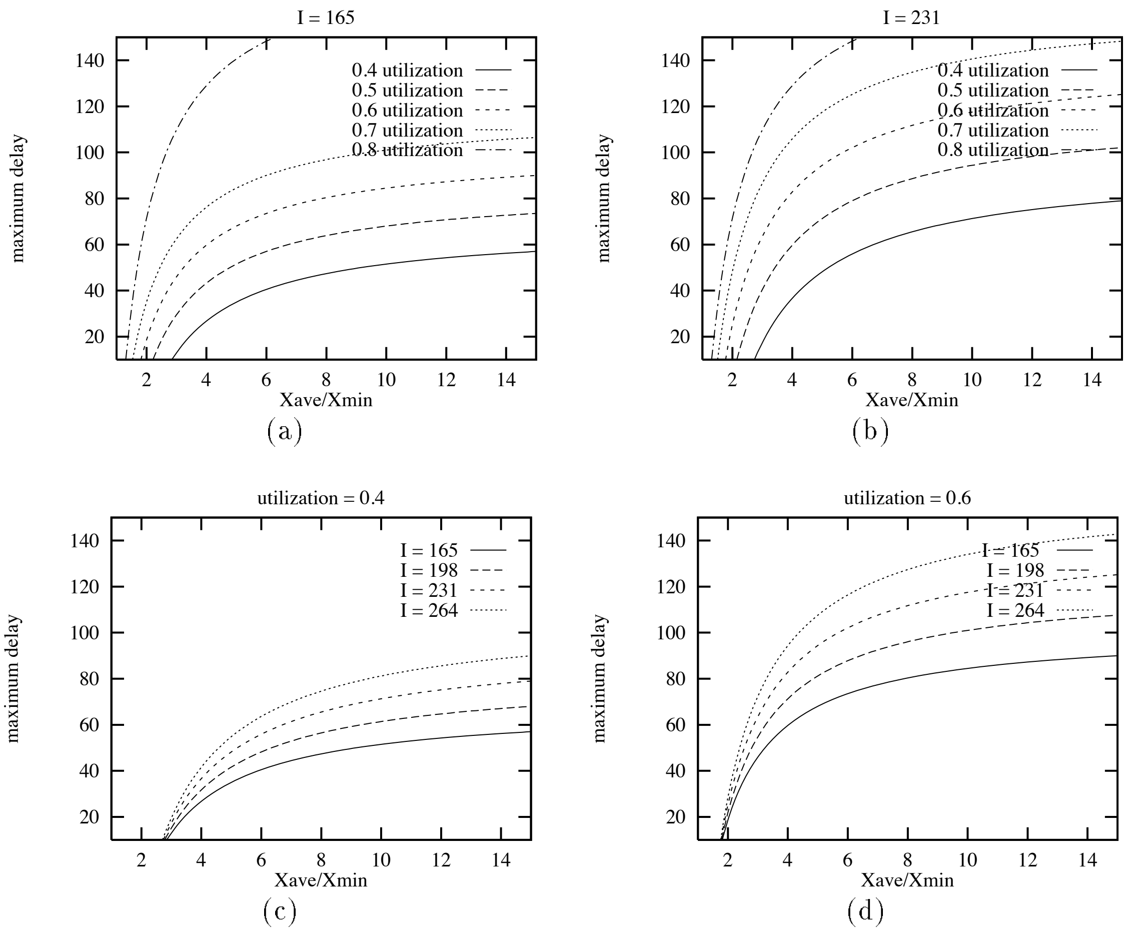


Figure 3.3: Effects of burstiness and utilization on delay bound

delay bound.

Figure 3.4 shows how much average or peak utilization can be achieved under certain delay bound constraints, where average and peak utilization are defined by $\sum_{j=1}^{n} \frac{Smax_j}{Xave_j \times l}$ and $\sum_{j=1}^{n} \frac{Smax_j}{Xmin_j \times l}$, respectively. Here *average utilization* means the maximum fraction of bandwidth that can be allocated to real-time traffic. The actual bandwidth used by real-time traffic is below the allocated bandwidth. The bandwidth unused by real-time traffic can be used by non-real-time traffic. Since the maximum peak utilization of the link is 1 under the old deterministic test defined in Equation (3.44), the peak utilization can be seen as an improvement factor, representing how many more connections can be accepted under the new admission control algorithm than under the old ones.

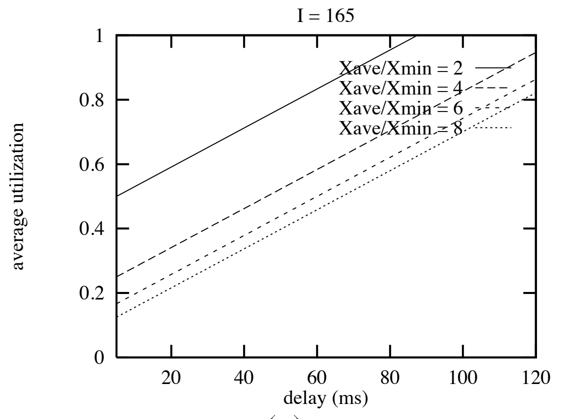
As can be seen, the peak utilization of the link by deterministically guaranteed performance traffic can be much greater than 1. Compared to the admission criteria where the sum of the peak rates of all channels have to be no greater than the link speed, using the admission control condition as stated in Theorem 3.2 may provide a multi-fold increase in the number of admitted connections when the traffic is bursty.

Figure 3.5 shows how average and peak utilization are affected by the burst ratio. As can be seen, on the one hand, the peak utilization, or the statistical multiplexing gain, is higher when the burst ratio is higher; on the other hand, the average utilization is lower when the burst ratio is higher. This matches our intuition that higher burst ratios provide more opportunity for statistical multiplexing but result in lower network utilization in order to meet a specific performance guarantee. Similar results can be obtained for statistical service, which will be discussed in detail in Chapter 4.

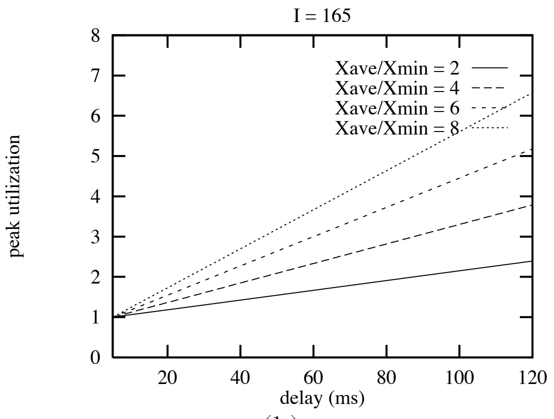
3.3.2 Bounding Delay for a Static Priority Scheduler

In the previous sections, we showed that even for simple service disciplines like FCFS, deterministic delay bounds can be obtained when the sum of the peak rates of all the connections is greater than the link bandwidth. The question naturally arises: why does one need more complex service disciplines than FCFS? One important reason is that an FCFS scheduler can only offer a single value of delay bound for all the connections, while the performance requirements for integrated-services networks will be diverse. It is important to support multiple Quality of Service classes.

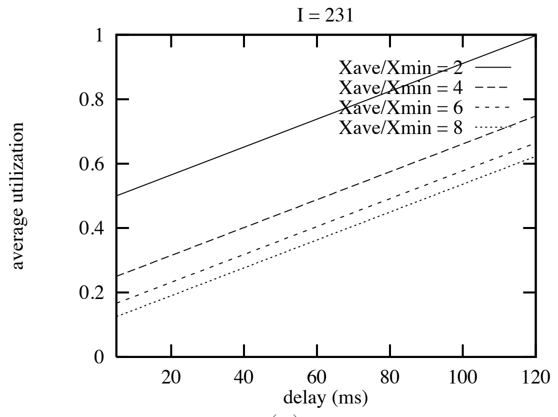
A Static Priority scheduler has the advantages of offering a number of priority



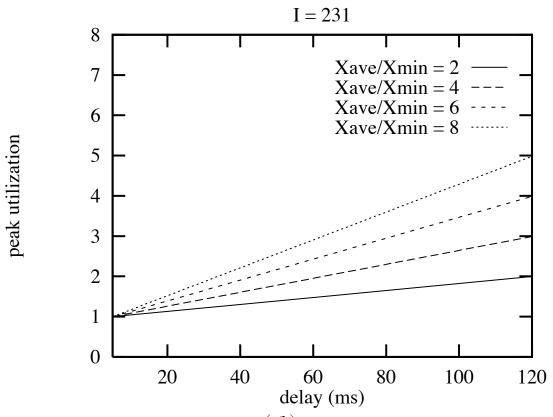
(a)



(b)

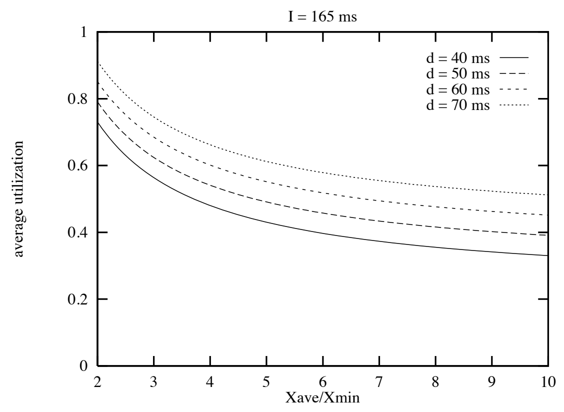


(c)

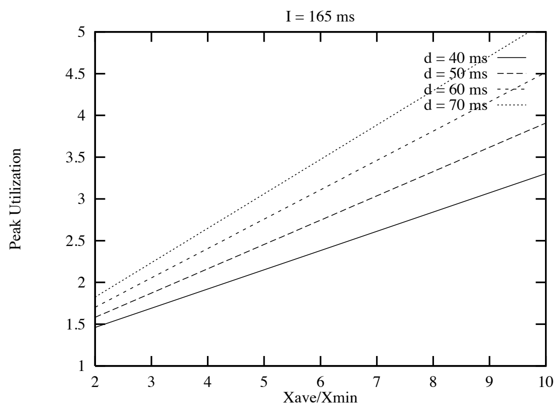


(d)

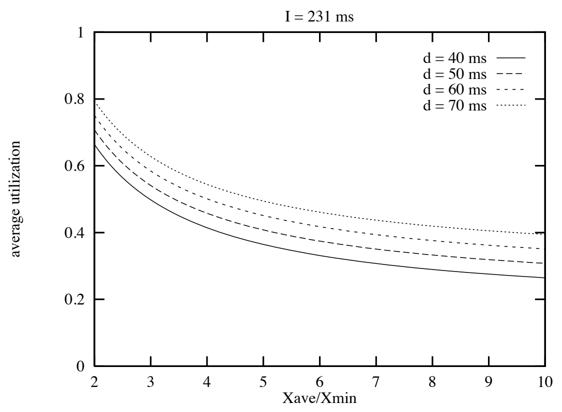
Figure 3.4: Average and peak utilization vs. delay bound



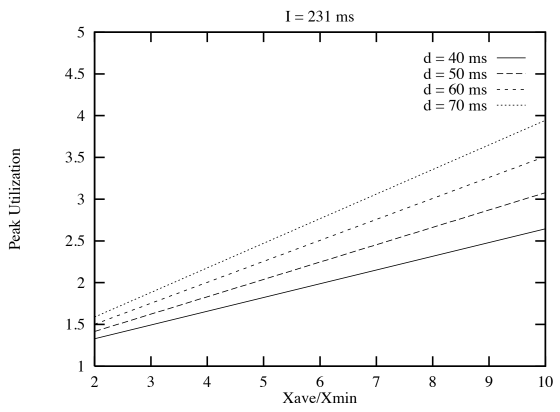
(a)



(b)



(c)



(d)

Figure 3.5: Average and peak utilization vs. burst ratio

levels, and also of being simple enough that it can be implemented at very high speeds.

An SP scheduler has a number of prioritized real-time packet queues and a non real-time packet queue. Packets at priority level 1 have the highest priority. A connection is assigned to a particular priority level at the connection's establishment time; all the packets from the connection will be inserted into the real-time packet queue at that priority level. Multiple connections can be assigned to the same priority level. The scheduler services packets using a non-preemptive static-priority discipline, which chooses packets in FCFS order from the highest-priority non-empty queue. Non-real-time packets are serviced only when there are no real-time packets; their order is not specified. The combination of a rate-controller and an SP scheduler is called a Rate-Controlled Static Priority (RCSP) server [91], which is shown in Figure 3.6.

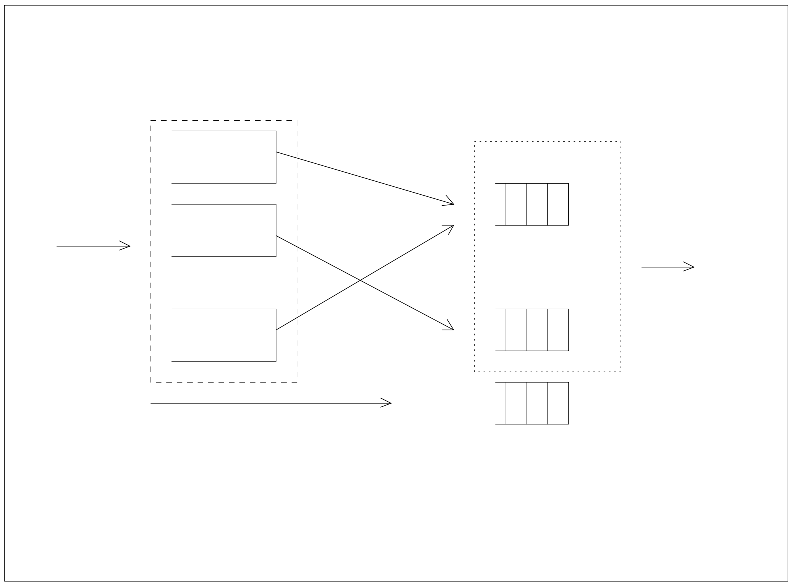


Figure 3.6: Rate-Controlled Static-Priority Queueing

By limiting the number of connections at each priority level via certain admission control conditions, the waiting time of each packet at a priority level can be bounded. The following theorem is used in [91] to give the admission control condition for an SP scheduler.

Theorem 3.3 *Let $\overline{d^1}, \overline{d^2}, \ldots, \overline{d^n}$ ($\overline{d^1} < \overline{d^2} < \cdots < \overline{d^n}$) be the delay bounds associated with each of the n priority levels, respectively, in a Static Priority scheduler. Assume that C_q is the set of connections at level q, and, that the j^{th} connection in C_q has the traffic specification ($Xmin_j^q, Xave_j^q, I_j^q, Smax_j^q$). Also assume that the link speed is l, and the size of the largest* *packet that can be transmitted onto the link is* $\overline{Smax}$. *If*

$$\sum_{q=1}^{m} \sum_{j \in C_q} \lceil \frac{\overline{d^m}}{Xmin_j^k} \rceil Smax_j^k + \overline{Smax} \leq \overline{d^m} l \tag{3.64}$$

the waiting time of a real-time packet at level m *is bounded by* $\overline{d^m}$.

Although condition (3.64) can guarantee delay bounds in an SP scheduler, it has the same limitation as the deterministic test that was used in [31]: the sum of the peak rates of all real-time connections can not exceed the link speed This will result a low average utilization of the connection when the peak-to-average-rate ratio is high.

In the following, we will apply the same analysis technique as used in Section 3.3.1 to derive tighter conditions for bounding delay in an SP scheduler. The following theorem has been stated and proven in [19].

Theorem 3.4 *Consider a multiplexer that is locally FCFS with respect to input stream 1. The maximum delay experienced in the multiplexer by any data bit from input stream 1 is bounded above by* $\overline{d^1}$, *where*

$$\overline{d^1} = max\{u : u \geq 0, b_1'(u) \geq l \times u\} \tag{3.65}$$

and $b'(\alpha)$ *is defined for all* α *by*

$$b_1'(\alpha) = \max_{\beta \geq 0}[l \times V + \overline{b}_1(\beta) + \overline{b}_2(\alpha + \beta) - l \times \alpha] \tag{3.66}$$

where V *is the bound on the vacation time the scheduler can take.*

To apply the result to an SP scheduler, we let all connections at level m to be stream 1, all the connections from level 1 to level $m-1$ to be stream 2, and the maximum vacation time be $\frac{\overline{Smax}}{l}$, and we get the following corollary.

Corollary 3.3 *Assume an SP scheduler has* n *priority levels. Let* C_q *be the set of the connections at level* q, *and the* j^{th} *connection in* C_q *satisfies the traffic constraint function* $\overline{b}_{q,j}(.)$. *Also assume that the link speed is* l, *and the size of the largest packet that can be transmitted onto the link is* $\overline{Smax}$. *The maximum delay of any packet at priority level* m *is bounded above by* $\overline{d^m}$, *where*

$$\overline{d^m} = max\{u : u \geq 0, b_m'(u) \geq l \times u\} \tag{3.67}$$

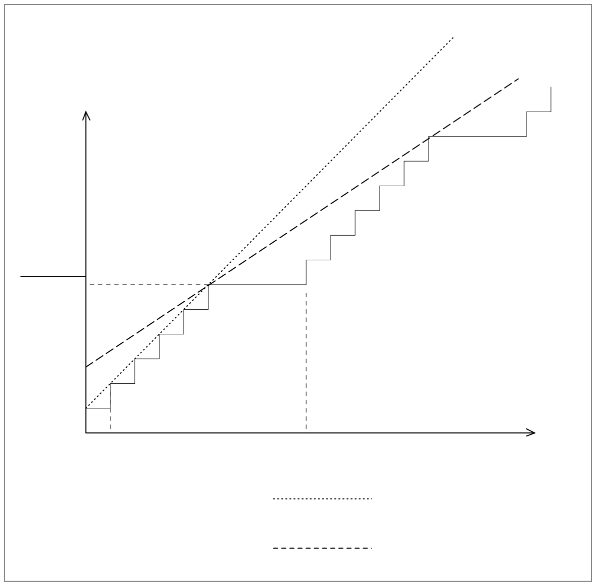


Figure 3.7: Bounding (Xmin, Xave, I, Smax) with (σ, ρ)

and $b'_m(\alpha)$ *is defined for all* α *by*

$$b'_m(\alpha) = \max_{\beta \geq 0}\{\overline{Smax} + \sum_{j \in C_m} \overline{b}_{m,j}(\beta) + \sum_{q=1}^{m-1} \sum_{j \in C_q} \overline{b}_{q,j}(\alpha + \beta) - l \times \beta\} \tag{3.68}$$

Solving 3.68 is equivalent to optimizing a piecewise linear function. Since the traffic constraint function for a connection with $(Xmin, Xave, I, Smax)$ specification is not convex, a single closed-form solution cannot be obtained. However, a closed-form solution exists for connections with (σ, ρ) traffic specification, whose traffic constraint function has the form $(\sigma + \rho u)$ [19]. A connection with an $(Xmin, Xave, I, Smax)$ traffic model can also be specified as obeying a (σ, ρ) model. In the following, we first relate the $(Xmin, Xave, I, Smax)$ model to the (σ, ρ) model, then give a closed-form solution for delay bounds in an SP scheduler for connections obeying a (σ, ρ) model, and finally we obtain a closed-form solution for delay bounds expressed in terms of $Xmin$, $Xave$, I and $Smax$.

Proposition 3.5 *If a connection satisfies the traffic specification* $(Xmin, Xave, I, Smax)$*, it also satisfies* (σ', ρ') *and* (σ'', ρ'')*, where* $\sigma' = Smax$*,* $\rho' = \frac{Smax}{Xmin}$*,* $\sigma'' = (I - \hat{I})\frac{Smax}{Xave}$*,* $\rho'' = \frac{Smax}{Xave}$*, and* $\hat{I}$ *is defined according to (3.57).*

Proof.

1) For any $u \geq 0$, we have

$$\begin{aligned}
\overline{b}(u) &\leq \lceil \frac{u}{Xmin} \rceil Smax && (3.69)\\
&\leq (1 + \frac{u}{Xmin}) Smax && (3.70)\\
&= Smax + \frac{Smax}{Xmin} u && (3.71)\\
&= \sigma' + \rho' u && (3.72)
\end{aligned}$$

Thus, the connection's traffic satisfies (σ', ρ').

2) We now show that the connection's traffic also satisfies (σ'', ρ'').

For any $u \geq 0$, let $u_1 = \lfloor \frac{u}{I} \rfloor I$ and $u_2 = u - u_1$. We have,

$$\overline{b}(u_1) = \frac{u_1}{Xave} Smax \tag{3.73}$$

$$\overline{b}(u_2) = \begin{cases} \lceil \frac{u_2}{Xmin} \rceil Smax & u_2 < \hat{I} \\ \frac{I}{Xave} Smax & u_2 \geq \hat{I} \end{cases} \tag{3.74}$$

$$\begin{aligned}
\overline{b}(u) &\leq \overline{b}(u_1) + \overline{b}(u_2) && (3.75)\\
&= \frac{u_1}{Xave} Smax + \overline{b}(u_2) && (3.76)\\
&= \frac{u - u_2}{Xave} Smax + \overline{b}(u_2) && (3.77)\\
&= \frac{u}{Xave} Smax + (\overline{b}(u_2) - \frac{u_2}{Xave} Smax) && (3.78)\\
&\leq \frac{u}{Xave} Smax + (\overline{b}(\hat{I}) - \frac{\hat{I}}{Xave} Smax) && (3.79)\\
&= (I - \hat{I}) \frac{Smax}{Xave} + \frac{Smax}{Xave} u && (3.80)
\end{aligned}$$

(3.75) holds because of the following property of a traffic constraint function:

$$\overline{b}(\alpha + \beta) \leq \overline{b}(\alpha) + \overline{b}(\beta) \quad \forall \alpha, \beta > 0 \tag{3.81}$$

(3.79) is due to the fact that the function $(\overline{b}(u_2) - \frac{u_2}{Xave} Smax)$ reaches its maximum at $\hat{I}$. (3.80) shows that the connection's traffic satisfies (σ'', ρ''). **Q.E.D.**

Proposition 3.5 shows that, for a connection with traffic specification $(Xmin, Xave, I, Smax)$, the traffic constraint function can be bounded by a convex function:

$$\overline{b}(u) \leq min(\sigma' + \rho' u, \sigma'' + \rho'' u) \tag{3.82}$$

The following theorem has been stated and proven in [19].

Theorem 3.5 *Assume an SP scheduler has n priority levels, and the input traffic to each priority level q satisfies a* (σ^q, ρ^q) *specification. Also assume that the maximum link speed is l, and the maximum size of a packet that can be transmitted over the link is* $\overline{Smax}$. *If* $\sum_{q=1}^{n} \rho^q < l$, *the maximum delay of any packet at priority level m is bounded above by* $\overline{d^m}$, *where*

$$\overline{d^m} = \frac{\sum_{q=1}^{m} \sigma^q + \overline{Smax}}{l - \sum_{q=1}^{m-1} \rho^q} \tag{3.83}$$

The theorem bounds the delay for each priority level when there is only one connection at each level. The following corollary extends the result to a Static Priority scheduler where more than one connection can be assigned to the same priority level.

Corollary 3.4 *Assume an SP scheduler has n priority levels. Let* C_q *be the set of the connections at level q, and the* j^{th} *connection in* C_q *satisfies the traffic specification function* (σ_j^q, ρ_j^q) . *Also assume that the maximum link speed is l, and the maximum size of a packet that can be transmitted over the link is* $\overline{Smax}$. *If* $\sum_{q=1}^{n} \sum_{j \in C_q} \rho_j^q < l$, *the maximum delay of any packet at priority level m is bounded above by* $\overline{d^m}$, *where*

$$\overline{d^m} = \frac{\sum_{q=1}^{m} \sum_{j \in C_q} \sigma_j^q + \overline{Smax}}{l - \sum_{q=1}^{m-1} \sum_{j \in C_q} \rho_j^q} \tag{3.84}$$

Proof.

It is easily seen from the following equation that, for any K connections, if the k^{th} connection satisfies a (σ_k, ρ_k) traffic specification, $k = 1, \cdots, K$, the aggregate traffic of the K connections satisfies $(\sum_{k=1}^{K} \sigma_k, \sum_{k=1}^{K} \rho_k)$:

$$\sum_{k=1}^{K} (\sigma_k + \rho_k u) = \sum_{k=1}^{K} \sigma_k + (\sum_{k=1}^{K} \rho_k) u \quad \forall\ u \tag{3.85}$$

If we consider the aggregate traffic of all the connections at level m as one stream, Corollary 3.4 follows immediately from Theorem 3.5. **Q.E.D.**

Corollary 3.5 *Assume an SP scheduler has n priority levels. Let* C_q *be the set of the connections at level q, and the* j^{th} *connection in* C_q *satisfies the traffic specification* $(Xmin_j^q, Xave_j^q, I_j^q, Smax_j^q)$. *Also assume that the maximum link speed is l, and the maximum size of a packet that can be transmitted over the link is* $\overline{Smax}$. *If*

$$\sum_{q=1}^{n} \mu_{ave}^q \leq 1 \tag{3.86}$$

where

$$\mu_{ave}^q = \sum_{j \in C_q} \frac{Smax_j^q}{Xave_j^q \times l} \tag{3.87}$$

the maximum delay of any packet at priority level m is bounded above by $\overline{d^m}$, where

$$\overline{d^m} = \begin{cases} min(\overline{d^{m'}}, \overline{d^{m''}}) & \sum_{q=1}^{m-1} \mu_{peak}^q < 1 \\ \overline{d^{m''}} & \sum_{q=1}^{m-1} \mu_{peak}^q \geq 1 \end{cases} \tag{3.88}$$

where

$$\mu_{peak}^q = \sum_{j \in C_q} \frac{Smax_j^q}{Xmin_j^q \times l} \tag{3.89}$$

$$\overline{d^{m'}} = \frac{\frac{\overline{Smax}}{l} + \sum_{q=1}^{m} \sum_{j \in C_q} \frac{Smax_j^q}{l}}{1 - \sum_{q=1}^{m-1} \mu_{peak}^q} \tag{3.90}$$

$$\overline{d^{m''}} = \frac{\frac{\overline{Smax}}{l} + \frac{1}{l} \sum_{q=1}^{m} \sum_{j \in C_q} \frac{Smax_j^q}{Xave_j^q}[I_q(1 - \frac{Xmin_j^q}{Xave_j^q}) + Xmin_j^q]}{1 - \sum_{q=1}^{m-1} \mu_{ave}^q} \tag{3.91}$$

Proof.

From Proposition 3.5, we know that, if a connection satisfies the traffic specification $(Xmin, Xave, I, Smax)$, it also satisfies (σ', ρ') and (σ'', ρ''), where $\sigma' = Smax$, $\rho' = \frac{Smax}{Xmin}$, $\sigma'' = [I - (\frac{I}{Xave} - 1)Xmin]\frac{Smax}{Xave}$, $\rho'' = \frac{Smax}{Xave}$.

Equations (3.90) and (3.91) follow directly from Corollary 3.4 with (σ', ρ') and (σ'', ρ''). Notice that Corollary 3.4 applies only when $\sum_{q=1}^{n} \sum_{j \in C_q} \rho_j^q < l$. In the case of (σ', ρ'), this means $\sum_{q=1}^{m-1} \mu_{peak}^q < 1$; in the case of (σ'', ρ'') this means $\sum_{q=1}^{m-1} \mu_{ave}^q < 1$. Thus, (σ', ρ') can be used to calculate the delay bound according to Corollary 3.4 only when $\sum_{q=1}^{m-1} \mu_{peak}^q < 1$; while (σ'', ρ'') can be used to calculate the delay bound in both cases of $\sum_{q=1}^{m-1} \mu_{peak}^q < 1$ and $\sum_{q=1}^{m-1} \mu_{peak}^q \geq 1$ as long as $\sum_{q=1}^{m-1} \mu_{ave}^q < 1$. **Q.E.D.**

3.4 Generalization and Comparison

In Section 3.1, we presented two types of regulators: the rate-jitter controlling, or *RJ*, regulator, and the delay-jitter controlling, or *DJ*, regulator. In Section 3.3, we gave conditions that guarantee delay bounds for FCFS and SP schedulers. There are other rate-controllers and schedulers that can be used in rate-controlled service disciplines. In this section, we show that the definition of rate-controlled service disciplines is quite general. By having different combinations of rate-controllers and schedulers, we can have a wide class of service disciplines. Most of the proposed non-work-conserving disciplines, including Jitter-EDD, Stop-and-Go and Hierarchical Round Robin, either belong to this class, or can be implemented by a rate-controlled service discipline with the appropriate choices of rate-controllers and schedulers. In the following, we first show the corresponding rate-controllers and schedulers for each of the above three disciplines, we then compare these disciplines with RCSP by showing the tradeoffs of using different rate-controllers and schedulers.

3.4.1 Jitter-EDD, Stop-and-Go, HRR

Jitter-EDD is a rate-controlled service discipline. The rate-controller in Jitter-EDD consists of regulators that control delay-jitter in a network with constant link delays. We will call this type of regulator the DJ_e regulator. In a DJ_e regulator, the eligibility time for packet k at the i^{th} switch along the path is defined as follows:

$$ET_i^k = AT_i^k + Ahead_{i-1}^k \tag{3.92}$$

where $Ahead_{i-1}^k$ is the amount of time the packet is ahead of schedule in switch $i-1$, the immediate upstream switch to switch i. The scheduler in Jitter-EDD is a variation of the Earliest-Due-Date scheduler [57].

It is easy to show that, in a DJ regulator, the eligibility time for packet k at the i^{th} switch along the path can also be represented as:

$$ET_i^k = AT_i^k + Ahead_{i-1}^k + (\overline{\pi}_i - \pi_i^k) \tag{3.93}$$

Comparing with (3.92), (3.93) has one more term: $\overline{\pi}_i - \pi_i^k$, which can be seen as the amount of time the packet is ahead of schedule in the link. In a network with constant delay links, the value of this term will always be zero. A DJ regulator differs from a DJ_e regulator in that the DJ regulator not only removes the traffic distortion introduced by the previous scheduler, but also removes the traffic distortion introduced by a variable-delay link.

A Stop-and-Go server with n frame sizes ($T_1 < T_2 < ... < T_n$) can be implemented by a rate-controlled service discipline with a variation of delay-jitter controlling regulators, which we call DJ_s regulators, and an n-level static priority scheduler. In a DJ_s regulator, the eligibility time for packet k at the i^{th} switch along the path is defined as follows:

$$ET_i^k = AT_i^k + Ahead_{i-1}^k + \theta \tag{3.94}$$

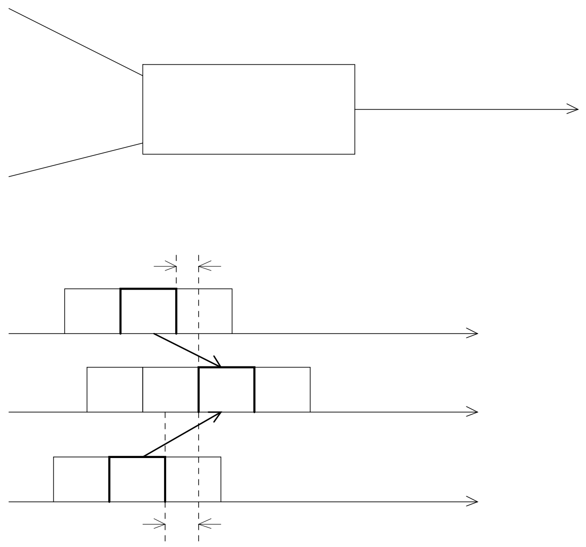


Figure 3.8: Synchronization between input and output links in Stop-and-Go

where $Ahead^k_{i-1}$ is the amount of time the packet is ahead of schedule in switch $i-1$, and θ is the synchronization time between the framing structures on the input and output links. Each pair of input and output links in a switch may have a different value of θ. Figure 3.8 illustrates this synchronization time. In ta static priority scheduler, the delay bound associated with level m is T_m, $1 \le m \le n$.

Although the above implementation of Stop-and-Go is very similar to RCSP, the allocation of delay bounds and bandwidth is more restrictive in Stop-and-Go than in RCSP. First, the traffic has to be specified with respect to the frame size that corresponds to the priority level the connection is assigned to. This introduces a coupling between the allocations of bandwidth and delay bounds. Secondly, there are dependencies among the local delay bounds at each priority level. $T_m = h_{m,m'} \times T_{m'}$ must hold for each pair of priority levels, with $1 \le m' \le m \le n$, and $h_{m,m'}$ being an integer. Thirdly, the delay bound allocations for each connection in different switches are coupled with one another. In [38], a connection has to have the same frame size in all the switches. In [93], a looser requirement is presented: the frame times of a connection along the path should be non-decreasing. None of these restrictions apply to RCSP.

A Hierarchical Round Robin server with n frame sizes ($T_1 < T_2 < ... < T_n$) can be implemented by a rate-controlled service discipline with a variation of the rate-jitter controlling regulator, which we call the RJ_h regulator, and an n-level static priority scheduler. In a level-m RJ_h regulator, the eligibility time for packet k at the i^{th} switch along the path is defined as follows:

$$ET_i^k = max(AT_i^k + \tau, ET_i^{k-a_i} + T_m) \tag{3.95}$$

where $AT_i^k + \tau$ is the beginning time of the next frame and a_i is the service quantum of the connection in switch i, which is defined to be the maximum number of packets that can be served on the connection within each frame of size T_m. In the static priority scheduler, the delay bound associated with level m is T_m, $1 \leq m \leq n$. If a connection traverses a level-m RJ_h regulator, it is assigned to the priority level m in the scheduler. This shows the coupling between delay and bandwidth allocation in HRR — in RCSP, a connection can be assigned to any priority level regardless of its rate parameters.

3.4.2 Regulator Tradeoffs

Regulator Type	ET_i^k
DJ	$AT_0^k + Ahead_{i-1}^k + (\overline{\pi}_i - \pi_i^k)$
	$ET_{i-1}^k + \overline{d}_{i-1} + \overline{\pi}_i^k$
DJ_e	$AT_i^k + Ahead_{i-1}^k$
	$ET_{i-1}^k + \overline{d}_{i-1} + \pi_i^k$
DJ_s	$ET_{i-1}^k + T^m + \overline{\pi}_i^k$
	$AT_i^k + Ahead_{i-1}^k + \theta$
RJ	$\max(ET_i^{k-1} + Xmin_j, ET_i^{k-\lfloor \frac{I}{Xave_j} \rfloor + 1} + I, AT_i^k)$
RJ_h	$\max(AT_i^k, ET_i^{k-a_i} + T_j^m)$

Table 3.2: Definitions of Eligibility Times in Different Regulators

Table 3.2 shows the definitions of eligibility times in different regulators. Notice that there are two equivalent definitions of eligibility times for each of the DJ, DJ_e and DJ_s regulators. As can be seen from the table, there are two types of regulators: delay-jitter controlling regulators, DJ, DJ_e, and DJ_s, where the eligibility time of a packet at a switch is defined with respect to the eligibility time of the *same* packet at the *previous* switch; and rate-jitter controlling regulators, RJ and RJ_h, where the eligibility time of a packet at a switch is defined with respect to *earlier arriving* packets at the *same* switch.

There are three considerations in deciding whether to use a rate-jitter controlling regulator or a delay-jitter controlling regulator: relative implementation complexity, services offered to the clients, and effects on the design of the network.

The first consideration is the relative complexity of implementation . There are two types of cost associated with implementing a regulator: computing the eligibility time for each packet, and holding packets if necessary. As will be shown in the next section, holding packets can be implemented efficiently by using a calendar queue mechanism. Thus, the only difference between a delay-jitter controlling regulator and a rate-jitter controlling regulator is in the computation of the eligibility time. To implement delay-jitter controlling regulators, there must be some mechanism for synchronization between consecutive switches. In Stop-and-Go, the physical links are framed and the framing structures at the two ends of a link are the same. In Jitter-EDD, one more quantity is needed to compute the eligibility time of a packet than in a rate-jitter controlling regulator, i.e., the amount of time the packet was ahead of schedule in the previous switch. This quantity can be calculated in the previous switch and stamped into the packet header as proposed in [86]. In RCSP, the same mechanism as in Jitter-EDD can be used if the link delay is constant. If links can have variable delays, synchronized clocks are needed to implement delay-jitter controlling regulators [25]. In an ATM network where each packet is only 53 bytes, the timestamping of each packet may be too expensive. Thus, synchronization, either at the link level or at the switch level, needs to be implemented. In a fast packet network, or an internetwork, where packet size is in the order of kilobytes, timestamping causes relatively small overhead. Note that synchronization among hosts and gateways in the current Internet is becoming more and more common due to the widespread use of synchronization protocols like the Network Time Protocol (NTP) [60, 61, 62] and TEMPO [39]; thus the implementation of delay-jitter control comes almost for free.

The second consideration relates to the services that can be provided to the clients. Delay-jitter control within the network provides bounded-delay-jitter service for communication clients for little additional cost. Rate-jitter control is not sufficient to provide bounded-delay-jitter service. This is illustrated by the following simulation results. The network being simulated is shown in Figure 3.9. All the links are 10 Mbps. The connection being measured traverses nodes (1, 3, 5, 6) with a guaranteed end-to-end delay bound of 48 ms. Additional connections traversing (0, 1, 3), (2, 3, 5), and (4, 5, 6) are established to introduce cross traffic.

Figure 3.10 gives the results of two experiments that show the delay distributions of a rate-jitter controlled connection and a delay-jitter controlled connection. As can been seen, packets on both the rate-jitter controlled connection and the delay-jitter controlled

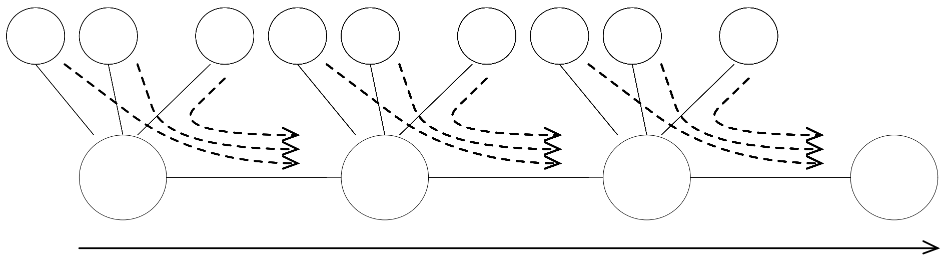


Figure 3.9: Network being simulated

connection meet the end-to-end delay bound. The delay jitter on the rate-jitter controlled connection is about 32 ms, whereas the delay jitter on the delay-jitter controlled connection is only about 10 ms. That is, the delay jitter is about three times larger on the rate-jitter controlled connection than on the delay-jitter controlled connection. This is due to the accumulation of traffic pattern distortions at each node in the case of rate-jitter control. If the measured connection were to traverse more nodes, the delay jitter for the rate-jitter controlled connection would be even larger, while the delay jitter for the delay-jitter controlled connection would be little affected. Another observation is that the average delay is much smaller for the rate-jitter controlled connection than for the delay-jitter controlled connection. For some clients, which need both low average delay and bounded delay services, a rate-jitter controlled connection is a better choice. For other clients, especially playback applications like video and audio applications, where the performance indices are the delay bound and the delay-jitter bound instead of the average delay, a delay-jitter controlled connection is a better choice. Notice that the simulation scenario in Figure 3.9 is very simple. More realistic scenarios may include more complex topologies or networks with traffic loops. The bounded-delay-jitter property is *proven* to hold for networks with delay-jitter controlling regulators even in these more complex settings. For networks with rate-jitter controlling regulators, a more complex topology only introduces more interactions between traffic streams, thereby resulting in larger delay jitter. Therefore, the simple network topology used in the simulation does not mean that the result is not general – on the contrary, it shows that delay jitter can be large for rate-jitter controlled connections even in a network with a very simple topology.

The third consideration in deciding whether to use a rate-jitter controlling regulator or a delay-jitter controlling regulator is the effect on network design. Using delay-jitter controlling regulators will *completely* reconstruct traffic patterns at each regulator; the traf-

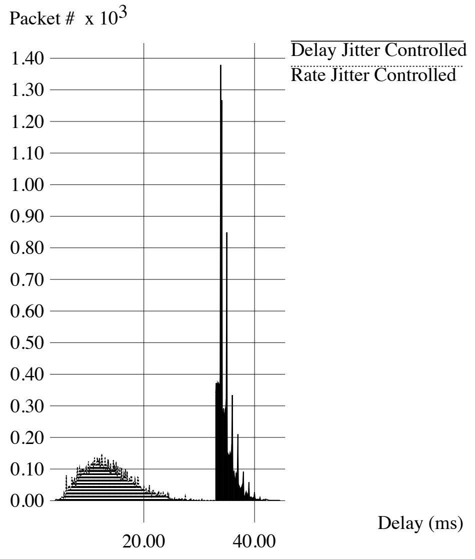


Figure 3.10: Effects of rate-jitter and delay jitter controls

fic pattern at the entrance to each scheduler will be *exactly* the same as that at the entrance to the network. Thus, if we can characterize the statistical properties of a source, the same statistical properties will hold throughout the network. By using this observation, we can provide end-to-end statistical guarantees as described in Chapter 4. Using rate-jitter controlling regulators only *partially* reconstructs traffic patterns at each regulator; some statistical properties of the traffic may be lost, and therefore it may be difficult to provide statistical guarantees.

3.4.3 Scheduler Tradeoffs

In this chapter, we have so far discussed two types of schedulers, First-Come-First-Served (FCFS), and Static Priority (SP). Among the non-work-conserving disciplines proposed in the literature, HRR, Stop-and-Go, and RCSP all use static priorities. Jitter-EDD uses Earliest-Due-Date.

As discussed in Chapter 1, we need to consider four aspects of a service discipline: protection of real-time connections, relative complexity of implementation, flexibility in allocating different delay bounds and bandwidths to different connections, and efficiency in utilizing the output link.

In a rate-controlled service discipline, the protection of real-time connections against other real-time connections is achieved by the rate-controller, and the protection of real-time connections against non-real-time traffic is achieved by servicing real-time packets at a higher priority that that of non-real-time traffic. Thus, the scheduler does not need a separate mechanism to enforce protection of real-time connections.

As for implementation complexity, FCFS is by far the simplest discipline. This is the reason why most switches today implement the FCFS discipline. Static Priority is more complex than FCFS, but insertion and deletion operations in Static Priority can all be implemented by constant numbers of steps. A slightly more complex discipline, Hierarchical Round Robin, has been implemented in custom VLSI to run at 1.3 Gbps as part of the Xunet project [36]. In this implementation, the scheduler can also be programmed to implement the Static Priority policy. Thus, it has been demonstrated that Static Priority can be implemented to run at very high speeds even with today's technology. Earliest-Due-Date is the most complex algorithm among the three in terms of implementation complexity. As mentioned before, the insertion operation needs $O(log(n))$ steps, where n is the number of packets in the queue. It has yet to be demonstrated whether Earliest-Due-Date, or other schedulers based on sorted priority queues, can be implemented at very high speeds.

In terms of the flexibility in allocating different bandwidths and delay bounds to different connections, the separation of the rate-control and the delay-control functions in rate-controlled service disciplines decouples the values these indices can take. The rate-controller is responsible for allocating bandwidth. For a scheduler, only its flexibility in allocating different delay bounds needs to be considered. Since a FCFS scheduler can only have one delay bound, it is unlikely to be acceptable for an integrated-services networks since different applications have diverse performance requirements. An EDD scheduler can allocate a continuous spectrum of delay bounds to different connections; thus, it is the most flexible among the three schedulers. A Static Priority scheduler can only allocate a fixed number of delay bounds; however, in practice, this is usually sufficient.

As to the efficiency in utilizing the output link, we have shown in this chapter that a FCFS scheduler can provide a deterministic delay bound even when the sum of the peak rates of all the connections is greater than the link speed. For homogeneous sources, it can achieve a reasonably high average utilization. For heterogeneous sources, however, the delay bound has to satisfy the most stringent requirement among all the connections, and will therefore underutilize the network. A Static Priority scheduler can provide multiple deterministic delay bounds even when the sum of the peak rates of all the connections is greater than the link speed. It can achieve a reasonably high average utilization for real-time traffic in for both homogeneous and heterogeneous sources. The admission control conditions given for Earliest-Due-Date in [31] require that the sum of the peak rates of all the connections not exceed the link speed. This is a sufficient condition, but not a necessary one. As shown in Section 3.3, with these admission control conditions for EDD, even FCFS schedulers can achieve a higher average utilization. However, it has been proven in [57] that EDD is optimal in the sense that, for any sequence of tasks, if they are schedulable under any scheduling policy, they will also be schedulable under EDD. Thus, the admission control conditions used for FCFS and Static Priority can also be used for EDD, in which case only one or a fixed number of delay bounds can be allocated. Therefore, EDD can achieve at least the same average utilization as FCFS or Static Priority. The problems of deriving the necessary conditions for bounding delays for an EDD scheduler and studying the improvement of average utilization under such a condition are not addressed in this dissertation.

In summary, Static Priority provides a good balance between simplicity of implementation and flexibility in allocating delay bounds. Further, it achieves a reasonably high average utilization for real-time traffic in for both homogeneous and heterogeneous sources. Thus, Rate-Controlled Static Priority, or RCSP, should be considered a good candidate for the service discipline for high speed integrated-services networks.

3.4.4 Generalization

The class of rate-controlled service disciplines is quite general. In the previous two subsections, we show that various regulators and schedulers can be used in rate-controlled service disciplines. Most of the non-work-conserving disciplines that have been proposed in the literature either belong to this class or can be implemented with a rate-controlled service discipline by choosing an appropriate rate-controller and an appropriate scheduler. Different combinations of regulators and schedulers will result in more service disciplines, as shown in Figure 3.11.

3.5 Implementation

In this section, we present one implementation of the RCSP queueing discipline. We believe that this implementation is simple enough to run at very high speeds.

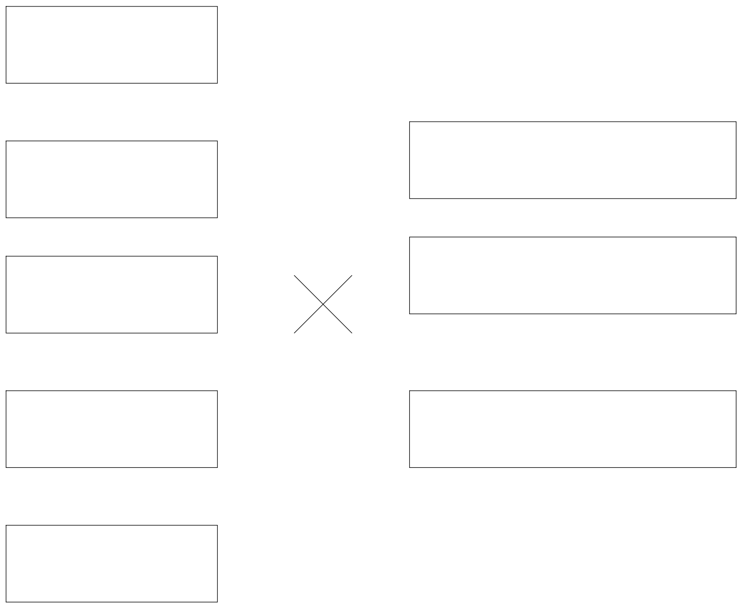


Figure 3.11: Generality of Rate-Controlled Service Disciplines

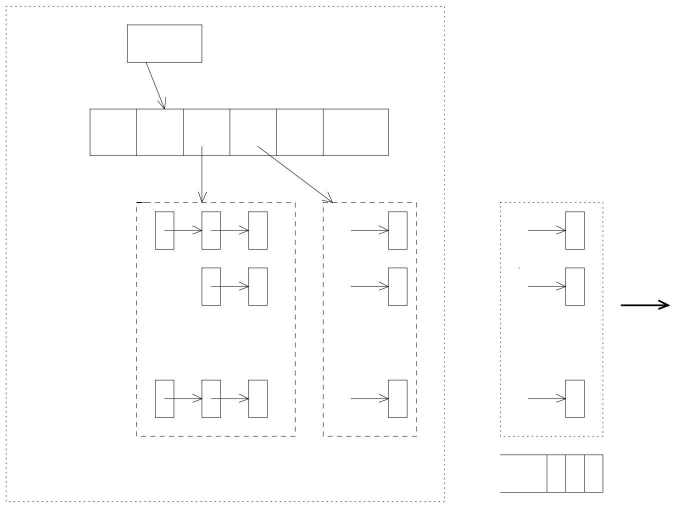


Figure 3.12: Implementation of RCSP

We have shown that a RCSP server has two components, a scheduler and a rate controller. The scheduler consists of multiple prioritized FCFS queues, and the rate controller consists of a set of regulators corresponding to each connection. Notice that the conceptual decomposition of the rate controller into a set of regulators does not imply that there must be multiple physical regulators in an implementation; a common mechanism can be shared by all logical regulators. Each regulator has two functions: computing the eligibility times for incoming packets on the corresponding connection, and holding packets till they become eligible. Eligibility times for packets from different connections are computed using the same formula (as described in Section 3.2) with different parameters; holding packets is equivalent to managing a set of timers; the mechanism for managing timers, which is a calendar queue [10, 84], can be shared by all regulators.

Figure 3.12 shows the proposed implementation. Each of the real-time queues is implemented as a linked list. The operations performed on a linked list are: deleting the packet at the head (when the next packet is chosen for transmission) and appending a packet or linked list of packets at the tail (when a packet arrives or the clock ticks; see below).

The rate controller is implemented using a modified version of a calendar queue. A calendar queue consists of a clock and a calendar, which is a pointer array indexed by time; each entry in the calendar points to an array of linked lists indexed by priority levels. The clock ticks at fixed time intervals. Upon every tick of the clock, the linked lists in the array indexed by the current time are appended at the end of the scheduler's linked lists: packets from the linked list of one priority level in the rate-controller are appended to the linked list of the same priority level in the scheduler.

Upon the arrival of each packet, the eligibility time of the packet, ET, is calculated; if $\lfloor \frac{ET}{Tick} \rfloor$ is equal to the current clock time, where $Tick$ is the clock tick interval, the packet is appended at the end of the corresponding real-time queue of the scheduler; otherwise, the packet is appended at the corresponding linked list at the calendar queue entry indexed by $\lfloor \frac{ET}{Tick} \rfloor$.

As can be seen, the data structures used in the proposed implementation are simple: arrays and linked lists; the operations are all constant-time ones: insertion at the tail of a linked list and deletion from the head of a linked list. We believe that it is feasible to implement this in a very high speed switch.

3.6 Summary

In this chapter, we have presented a class of non-work-conserving disciplines called rate-controlled service disciplines. A rate-controlled service discipline has two components: a rate-controller and a scheduler. After the rate-controller limits the distortion of the traffic introduced by load fluctuations inside the network, the scheduler orders the packets for transmission. The end-to-end delay of a packet in a network with rate-controlled servers consists of the following components: waiting times in the schedulers, holding times in the rate-controllers and the link delays.

One of the most important results described in this chapter was Theorem 3.1, which stated that the end-to-end delays of all the packets on a connection can be bounded, as long as the delays on links and the delays at each of the schedulers can be bounded. Thus, although holding times may increase the average delay of packets on a connection, they do not increase the end-to-end delay bounds. The key step in establishing this theorem is Lemma 3.1, which considers the case of two rate-controlled servers connected in cascade.

Another important result is the improvement of the Tenet admission control algorithms for deterministic service [31]. In the previous version of the algorithms, the deterministic test ensures that the sum of the peak rates of all deterministic connections is less than the link speed. This condition will result in a low average link utilization if the peak-to-average-rate ratio is high for real-time traffic. In Section 3.3, we have shown that deterministic service can be provided even when the condition is not satisfied. We gave conditions to bound delays for FCFS and Static Priority schedulers, and presented numerical examples. The new admission control algorithms result in a multifold increase in the number of admitted connections.

We showed that the key property of rate-controlled service disciplines is to separate the server into two components: a rate-controller and a scheduler. Such a separation have several advantages that make rate-controlled service disciplines suitable for supporting guaranteed performance communication in a high speed networking environment:

- End-to-end performance bounds can be obtained in a network of arbitrary topology. Unlike most work-conserving disciplines, which can provide bounds only in feed-forward networks and some restricted classes of feed-back networks, rate-controlled disciplines provide bounds in arbitrary networks.

- Unlike most existing solutions, which assume *constant* link delays between switches, we need only to assume *bounded* link delays when rate-controlled disciplines are used. This is particularly important in an internetworking environment, where switches are connected by subnetworks. The delays of packets traversing subnetworks must be *bounded*, but may be *variable*.
- Given this separation of functionality into rate control and packet scheduling, we can have arbitrary combinations of rate-control policies and packet scheduling disciplines.
- By having a server with two components, we can extend results previously obtained for a single scheduler to a networking environment. Any scheduler that can provide delay bounds to connections at a single switch can be used in conjunction with a rate-controller to form a rate-controlled server, which can then be used to provide end-to-end performance bounds in a networking environment.
- Separation of rate-control and delay-control functions in the design of a server allows decoupling of bandwidth and delay bound allocation to different connections. Most existing solutions have the drawback of coupling the bandwidth/delay bound allocation — allocating of a lower delay bound to a connection automatically allocates a higher bandwidth to the connection. Such solutions cannot efficiently support low delay/low bandwidth connections.
- Unlike work-conserving disciplines which require the reservation of more buffer space at the down stream switches traversed by a connection, rate-controlled disciplines also have the advantage of requiring evenly distributed buffer space at each switch to prevent packet loss.

We have shown that rate-controlled service disciplines provide a general framework under which most of the existing non-work-conserving disciplines such as Jitter-EDD [86], Stop-and-Go [38] and Hierarchical Round Robin [44] can be naturally expressed. We discussed the tradeoffs of various rate-controllers and schedulers. One discipline in this class called Rate-Controlled Static Priority (RCSP) is particularly suitable for providing performance guarantees in high speed networks. It achieves both flexibility in the allocation of bandwidth and delay bounds to different connections, as well as simplicity of implementation.

Chapter 4

Rate-Controlled Service Disciplines: Statistical Analysis

In the previous chapter, we have described a new class of service disciplines that can provide end-to-end per-connection delay and delay jitter bounds. The bounds are deterministic, in the sense that, *even in the worst case*, the performance guarantees are satisfied for *all* packets. Although deterministic services are important for certain applications, for example, network-based medical imaging application, they require more resources. To provide deterministic delay guarantees, we have to compute the delay for pathological worst case arrival patterns, where packets from all the connections are arriving at the link at the maximum rate. With deterministic service, the stochastic properties of traffic sources cannot be exploited to achieve statistical multiplexing gains. Further, when the sources are bursty, providing deterministic service results in lower average utilization of the network by guaranteed performance traffic. Alternatively, many applications such as voice and video conferencing can tolerate certain losses of data without significantly affecting the quality of the output delivered. *Statistical service*, in which *probabilistic* or *statistical* performance bounds are guaranteed, can be used to support these applications. The advantage of providing statistical service is that average utilization of the network by guaranteed performance traffic can be increased by exploiting statistical multiplexing.

There are two important aspects of the problem of providing guaranteed statistical services: developing probabilistic models to characterize traffic sources, and developing techniques to provide end-to-end probabilistic bounds in a network environment.

In the literature, many models have been proposed for video or audio traffic sources. Among the most popular ones are the on-off model for voice sources [8, 9] and more sophisticated models based on Markov or other renewal processes for video sources [59]. A good survey for the probabilistic models for voice and video sources is presented in [65]. There are two important limitation to such traffic models. First, in an integrated-services network, traffic sources are heterogeneous and will not in general conform to one model. If the traffic source does not conform to the analytical model, no statistical guarantees can be made. Also, neither the parameters nor the accuracy of the model is generally known. They are difficult to determine beforehand, and are usually not trying to represent a "statistical worst case" that is necessary to provide statistical guarantees. Second, the above models cannot capture varying statistical properties of the traffic over time intervals of different length. It is therefore important to investigate traffic models that can both represent statistical worst case and capture the *interval-length dependent* property of traffic sources.

In [51], Kurose proposes modeling a source with a family of random variables that bound the number of packets sent over various time intervals. In this paper, we extend this model so that the random variable that bounds the number of packets sent over an interval of length t is an explicit function of t. This model can thus be used to characterize interval-length dependent behavior; for example, the observation that, over longer intervals, the total number of packets sent by a source is more likely to correspond to the source's long-term average rate, while on a shorter time scale the source is more likely to send according to its peak rate.

Having a traffic model for sources only solves part of the problem. In a networking environment, packets from different connections are multiplexed at each of the switches. Even if the traffic can be characterized at the entrance to the network, complex interactions among connections will destroy many properties of the traffic inside the network, and the traffic model at the source may not be applicable inside the network. Thus, even if performance guarantees may be offered for a single switch, it may be impossible to provide end-to-end performance guarantees.

In this chapter, we address these two aspects of the problem by using interval-dependent stochastic traffic models to characterize traffic sources, and rate-controlled service disciplines inside the network to reconstruct traffic patterns. By using interval-dependent stochastic traffic models and rate-controlled service disciplines, we can provide per-connection end-to-end statistical guarantees on throughput, delay, and delay-jitter in a network of *arbitrary* topology. Again, as in the case of deterministic service, rate-controlled service disciplines allow our results to be quite general. Unlike most existing solutions, which work only in feed-forward and some restricted classes of feedback networks, our results hold in arbitrary networks. Also, unlike most existing solutions which assume *constant* link delays between switches, we need only assume *bounded* link delays. This is particularly important in an internetworking environment, where switches are connected by subnetworks. The delays of packets traversing subnetworks must be *bounded*, but may be *variable*.

The remainder of the chapter is organized as follows. In Section 4.1, the interval-dependent source model is developed. Section 4.2 investigates multiplexing the discrete model of Section 4.1 for both homogeneous and heterogeneous sources. In addition, the switch utilization for various model parameters is investigated along with other trends involved with providing statistical performance guarantees to various sources. The Rate-Controlled Static-Priority (RCSP) scheduler is used to provide different performance guarantees to different connections. Finally, in Section 4.3, the results of Section 4.2 are extended to the network to provide end-to-end per-connection statistical performance guarantees.

4.1 Interval Dependent Source Model

In this section, we extend the Tenet's deterministic $(Xmin, Xave, I, Smax)$ traffic model to a stochastic traffic model. The proposed traffic model is based on Kurose's recent use of stochastic bounding models instead of stochastic processes to characterize traffic sources.

4.1.1 Stochastic Processes vs. Bounding Models

The literature contains a wide variety of analytic approaches that model traffic sources by some stochastic process, calculate the aggregate process, and then solve for quantities in a switch such as the steady-state buffer distribution. Though such techniques provide valuable insights into certain classes of problems, such analyses are often difficult to extend to integrated-services networks for at least the following problems:

- Homogeneous traffic sources - this is not the case in integrated-services networks.
- Model fitting — for a given stochastic model of traffic (as opposed to a bounding distribution), some sources will not fit the model. In such a case, no statistical guarantees

can be derived.

- Aggregate results — we need per-connection analyses to provide different services to different applications.
- Average results — a statistical real-time service usually needs stronger guarantees than those on mean parameters.
- Computationally intensive results - admission control tests are performed per connection and must be fast.
- Single switch analysis — results are often confined to a single switch and cannot be extended to the network because of the generally intractable problem of determining the transformation a switch performs on the traffic of each connection.
- No priorities — often, results only hold for single priority FCFS queueing. Again, B-ISDN will carry a wide range of traffic types, not only voice or only data.

Because of such difficulties with complex stochastic models, a great deal of attention has been given to analyzing deterministic traffic models that provide some means of *bounding* a source's peak and average bandwidth over an averaging interval [19, 31, 38, 58]. Such models are not only practical, but they also result in an analysis that does not suffer from many of the problems mentioned above. Specifically, the analysis can provide end-to-end *per-stream* bounds in priority queueing service disciplines. One drawback to such models is that they cannot characterize many of the statistical properties of the source, and can only be used to provide deterministic performance bounds, not statistical performance bounds.

Recently, Kurose proposed a general framework for characterizing traffic sources [51]. In such a framework, source S_j is characterized by a family of pairs $\{(R_{t_1,j}, t_1), (R_{t_2,j}, t_2), (R_{t_3,j}, t_3)...\}$, where $R_{t_i,j}$ is a random variable that is *stochastically larger* than the number of bits generated over any interval of length t_i by source S_j. A random variable X is said to be stochastically larger than a random variable Y (notationally, we write $X \succeq_{st} Y$) if and only if $Prob(X > x) \geq Prob(Y > x)$ for all x. Instead of actually modeling the exact arrival process of the source, Kurose's model stochastically *bounds* the number of transmitted bits in intervals of different length. In [89, 90], a stochastic extension to Cruz's deterministic model [19] is proposed that provides end-to-end stochastic bounds.

However, this model does not take into account the interval dependencies considered in this chapter. Moreover, it is applied to a network of work-conserving servers, and therefore, as will be discussed in Section 4.3, the results are restricted to a certain class of networks.

In Chapter 3, we used the deterministic traffic model $(Xmin, Xave, I, Smax)$ proposed in [31] to characterize source. In this chapter, we propose the extension of the deterministic model to a probabilistic model within Kurose's framework. We extend Kurose's model to make the bounding random variables explicit functions of the interval length in order to better characterize the properties of the source. The are two general requirements for the interval-dependent bounding random variable:

$$\begin{aligned} R_{t_n} + R_{t_m} &\succeq_{st} R_{t_n+t_m} && (4.1) \\ \frac{E(R_{t_n})}{t_n} &\leq \frac{E(R_{t_m})}{t_m} \quad if \;\; t_n > t_m && (4.2) \end{aligned}$$

The first property is stochastic sub-additivity. The second property requires that the mean rate over smaller time intervals be greater than the mean rate over larger time intervals. In the next two subsections, we present two stochastic traffic bounding models that capture the interval-dependent properties of a source.

4.1.2 Interval-Dependent Traffic Model

In this section, we introduce a discrete-valued family of random variables with a parameterized binomial distribution to bound the number of packets that can be generated by a source in intervals of different length. Note that this is not to say that the underlying random process is binomial, rather that a binomial random variable is used to bound the process. By choosing different parameters for the random variable, it is possible to bound different processes with complicated distributions.

For the binomial bounding random variable, let the j^{th} source, denoted by S_j, be characterized by the following:

$$S_j \sim \{(R_{t,j}, t) \mid t \geq 0\} \qquad (4.3)$$

$R_{t,j}$ stochastically bounds the total number of packets that can arrive on connection j during any interval of length t, and is assumed to have a binomial distribution with parameters $M_{t,j}$ and $p_{t,j}$, which are given by the following equations:

$$M_{t,j} = \lceil t/Xmin_j \rceil \qquad (4.4)$$

$$p_{t,t} = \begin{cases} \frac{Xmin_j}{1-e^{\gamma}}((\frac{1}{Xmin_j} - \frac{1}{Xave_j})e^{-\frac{t/I_j}{\gamma}} + \frac{1}{Xave_j} - \frac{1}{Xmin_j}e^{-\frac{t}{I_j}}) & t \le I_j \\ \frac{Xmin_j}{Xave_j} & t > I_j \end{cases} \tag{4.5}$$

Equations (4.4) and (4.5) express $M_{t,j}$ and $p_{t,j}$ using four parameters: $Xmin_j$, $Xave_j$, I_j, and γ, where $Xmin_j \le Xave_j < I_j$ and $\gamma > 0$. The formulas are chosen so that some desired properties as will be described below hold. This parameterization extends the Tenet deterministic model to a stochastic traffic model within Kurose's framework. The stochastic representation above captures a more detailed interval-dependent behavior of a source with the $M_{t,j}$ and $p_{t,j}$ parameters. In such a model, $M_{t,j}$ and $M_{t,j} \cdot p_{t,j}$ stochastically bound the maximum and the mean number of packets that can arrive during an interval of length t. $\frac{E(R_{t,j})}{t}$ is a stochastic bound on the mean rate during an interval of length t, and is given by:

$$\frac{E(R_{t,j})}{t} = \frac{M_{t,j} \cdot p_{t,j}}{t} \tag{4.6}$$

$$= \begin{cases} \frac{1}{1-e^{\gamma}}((\frac{1}{Xmin_j} - \frac{1}{Xave_j})e^{-\frac{t/I_j}{\gamma}} + \frac{1}{Xave_j} - \frac{1}{Xmin_j}e^{-\frac{t}{I_j}}) & t \le I_j \\ \frac{1}{Xave_j} & t > I_j \end{cases} \tag{4.7}$$

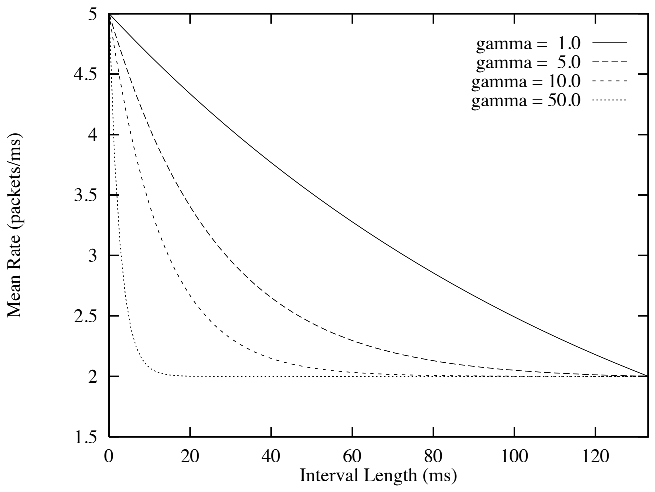


Figure 4.1: Effect of γ on mean rate

In these equations, γ is a client-specified parameter that controls how rapidly the mean rate over an interval approaches the long-term average rate as the interval length gets larger. A larger γ means that the speed with which this rate is approached is faster. This is illustrated in Figure 4.1.

It is easy to verify that the following properties hold:

1. The maximum value of the source's mean rate over any interval is greater than $\frac{1}{Xave_j}$, and less than $\frac{1}{Xmin_j}$, i.e.,

$$\frac{1}{Xave_j} \le \frac{E(R_{t,j})}{t} \le \frac{1}{Xmin_j} \tag{4.8}$$

   where $E(R_{t,j}) = p_{t,j} M_{t,j}$.

2. The property in equation (4.2) is satisfied since the mean rate over a longer interval is no more than the mean rate over a shorter interval. As well, one can verify that equation (4.1) is satisfied.

3. If two sources S_{j_1} and S_{j_2} have the same $Xave$, I, and γ, but $Xmin_{j_1} > Xmin_{j_2}$, then

$$\frac{E(R_{t,j_1})}{t} \le \frac{E(R_{t,j_2})}{t} \quad \forall\, t > 0. \tag{4.9}$$

   I.e., if two connections have the same long term (with interval length no less than I) bounding average rate ($\frac{1}{Xave}$) [1], the bounding mean rate over any interval is greater for the connection with the higher peak rate ($\frac{1}{Xmin}$). This property is illustrated in Figure 4.2. In the figure, the vertical axis is the bounding average rate, and the horizontal axis is the length of the interval over which the average rate is computed. Curves for three sources are plotted. The three sources have the same I of 133 ms and same $Xave$ of 0.5 packets/ms, but have different $Xmin$'s. It can be seen that a curve with a smaller $Xmin$ is always above a curve with a larger $Xmin$. The same property holds in the deterministic model. In fact, in the deterministic model, with fixed $Xave$, a smaller $Xmin$ means a burstier traffic.

t (ms)	$M_{t,j}$	$p_{t,j}$	$\frac{E(R_{t,j})}{t} Smax_j$ (Mbps)
4	40	0.889655	3.558622
20	200	0.602784	2.411138
60	600	0.376289	1.505155
120	1200	0.334655	1.338619

Table 4.1: Effect of Intervals On Bounding Variable Parameters

The following numerical example illustrates the interval-dependent property of this traffic model. Consider a connection with $Xmin = 0.1$ ms, $Xave = 0.3$ ms, $I = 133$ ms,

[1] The unit here is *packets/sec.*

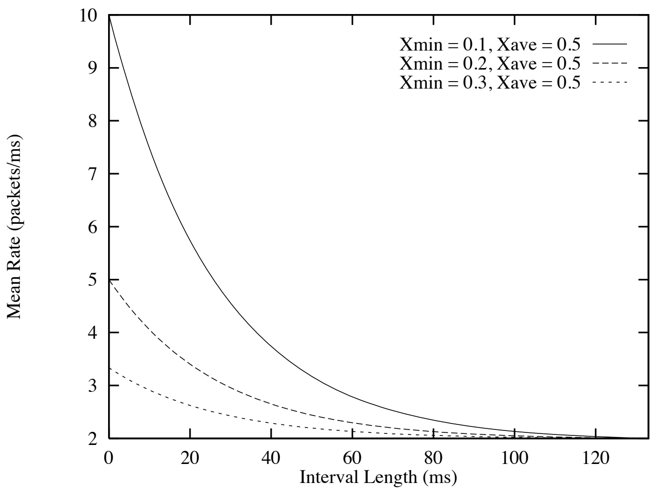


Figure 4.2: Effect of burstiness on mean rate

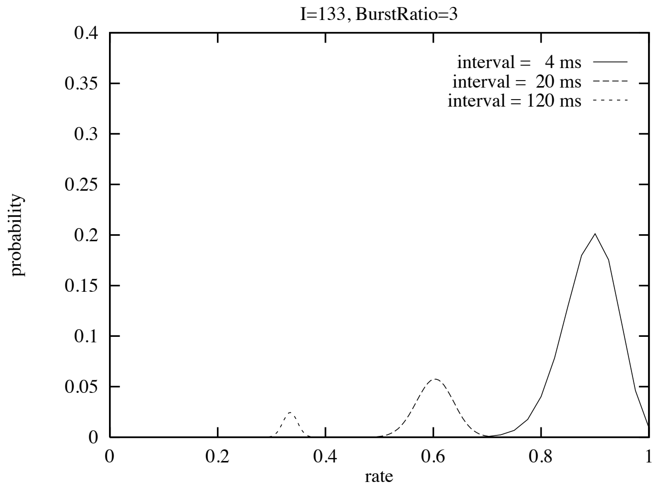


Figure 4.3: Distribution of a discrete rate-bounding random variable

$Smax = 400$ bits and $\gamma = 5.0$. It is easy to derive from the specification that the connection has a peak rate of 4 Mbps and a long-term average rate of 1.33 Mbps. Table 4.1 shows the values of the parameters of the bounding random variable over intervals of different lengths, and Figure 4.3 shows the probability distribution of the normalized number of bits sent in intervals of length 4, 20, and 120 msec (normalized with respect to the peak rate). As can be seen, the rate over a shorter interval is more likely to be near the peak rate, while the rate over a longer interval is more likely to be near the average rate.

4.2 Multiplexing Interval-Dependent Sources

In this section, we analyze the delay characteristics of connections whose traffic is described by the interval-dependent traffic model for a Static Priority scheduler. The technique of extending one server analysis to a networking environment is discussed in Section 4.3. A Static Priority scheduler has the advantage of being both flexible, in that it can offer a multiple number of different delay bounds, and simple, so that it can run at very high speeds. We present numerical examples and consider both a heterogeneous and a homogeneous mix of sources.

As discussed in Chapter 3, the scheduler in an RCSP server consists of a number of prioritized real-time packet queues and a non-real-time packet queue. Packets at priority level 1 have the highest priority. A channel is assigned to a particular priority level at the channel's establishment time, and all packets on the channel are then inserted into the real-time packet queue at that priority level. Multiple channels can be assigned to the same priority level. The scheduler services packets using a non-preemptive static priority policy which chooses packets in FCFS order from the highest-priority non-empty queue. Non-real-time packets are serviced only when there are no real-time packets.

There is a delay bound $\overline{d^m}$ associated with each priority level m. For a connection associated with priority level m, we are interested in calculating the delay-bound violation probability: $Prob\{d^m > \overline{d^m}\}$.

Proposition 4.1 *Let $\overline{d^1}, \overline{d^2}, \ldots, \overline{d^n}$ ($\overline{d^1} < \overline{d^2} < \cdots < \overline{d^n}$) be the respective delay bounds associated with the n priority levels in a Static Priority scheduler. Assume that C_q is the set of connections at level q, and the j^{th} connection in C_q has the traffic specification*

$$\{(R_{\overline{d^1},j}, \overline{d^1}), (R_{\overline{d^2},j}, \overline{d^2}), (R_{\overline{d^3},j}, \overline{d^3}) \ldots\}.$$

Also assume that the link speed is l, and the size of the largest packet that can be transmitted onto the link is $\overline{Smax}$. We have

$$Prob\{d^m > \overline{d^m}\} \leq Prob\{\sum_{q=1}^{m} \sum_{j \in C_q} R_{\overline{d^q},j} \times Smax_j + \overline{Smax} \geq \overline{d^m} l\} \tag{4.10}$$

The proposition shows that the tail distribution of the sum of the bounding random variables for all the connections with the same or higher priorities can be used to provide an upper bound for the delay-bound violation probability. The result applies to bounding random variables with any distribution.

In the following sections, we present numerical examples using the discrete traffic model developed in Section 4.1.2. We consider the cases of both homogeneous and heterogeneous sources.

4.2.1 Homogeneous Sources

As in Section 4.1.2, we assume that $R_{\overline{d^q},j}$ has a binomial distribution with parameters $M_{\overline{d^q},j}$ and $p_{\overline{d^q},j}$, which are defined according to (4.4) and (4.5). For homogeneous sources,

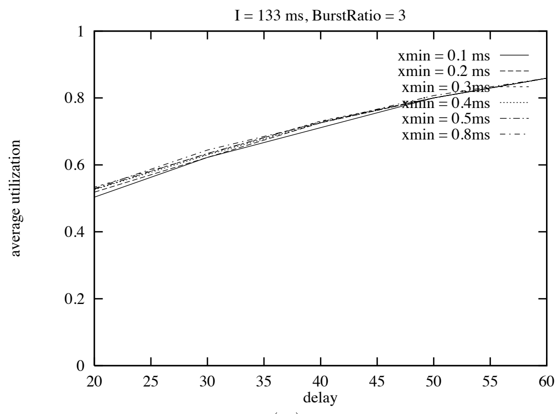
$$M_{\overline{d^q}} = M_{\overline{d^q},j} \quad \forall j \tag{4.11}$$

$$p_{\overline{d^q}} = p_{\overline{d^q},j} \quad \forall j \tag{4.12}$$

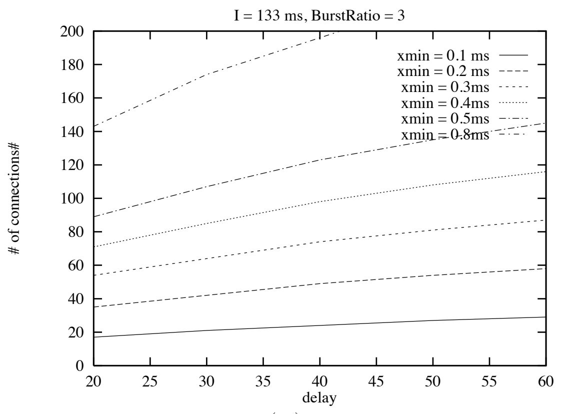
Assuming independence among the connections, $\sum_{q=1}^{m} \sum_{j \in C_q} R_{\overline{d^q},j}$ has a binomial distribution with parameters $\sum_{q=1}^{m} J_q M_{\overline{d^q}}$ and q, where J_q is the number of connections at priority level q, or $\mid C_q \mid$.

The following examples show the relationships between the average utilization μ and the delay bound under various conditions. The link transmission rate is 45 Mbps, $\gamma = 6.0$, and the delay overflow probability is 0.1%. The parameters under consideration are $Xmin$, I, and $BurstRatio$. In each of the following examples, we fix two of the three parameters, vary the third, and show a family of μ vs. d curves.

In Figure 4.4, the averaging interval I and the burst ratio $\frac{Xave}{Xmin}$ are fixed. Varying the value of $Xmin$ allows us to investigate the effect of the peak rate of homogeneous connections on average utilization. The plots show that the peak rate has little effect on the average utilization for homogeneous sources. For example, with $I = 166$ ms, $BurstRatio$ $= 5$, and $d = 20$ ms, the switch can accept either 20 connections with $Xmin = 0.1$ ms (peak rate of 4 Mbps), or 83 connections with $Xmin = 0.4$ ms (peak rate of 1 Mbps). I.e., the switch accepts 4 times more connections when the peak rate is reduced by a factor of 4.



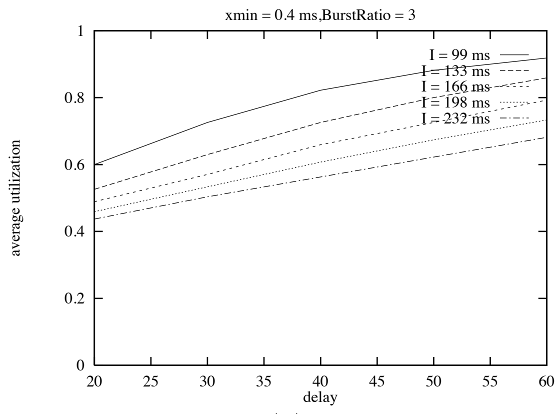
(a)



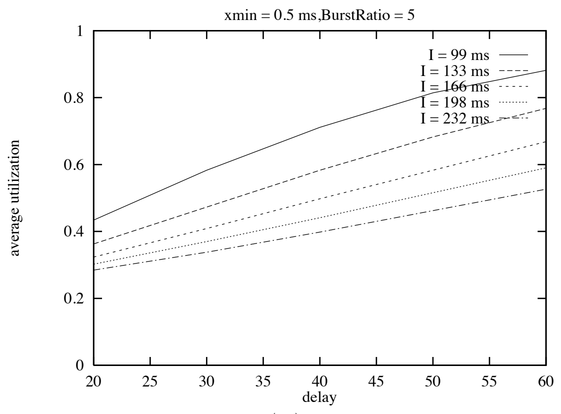
(b)

Figure 4.4: Effect of peak rate

In Figure 4.5, the minimum packet spacing $Xmin$, and the peak-to-average rate ratio $\frac{Xave}{Xmin}$ are fixed, and I is varied. As shown, a longer averaging interval I results in a lower average utilization of the link. Recall that in the deterministic deterministic case, a source can send a maximum $\frac{I}{Xave}$ consecutive packets with $Xmin$ spacing. A larger I means a larger maximum burst length, which results in lower average utilization of the link.



(a)



(b)

Figure 4.5: Effect of averaging interval

Figure 4.6 shows how $BurstRatio$ affects statistical multiplexing. Figure 4.6 (a) shows that, for a given performance requirement, a smoother traffic is easier to multiplex and results in a higher average utilization of the link. Additionally, Figure 4.6 (b) shows that, although the average utilization is lower for bursty traffic, the peak utilization, a measure of the statistical multiplexing gain, is higher. This matches the intuition that higher burst ratios provide more opportunity for statistical multiplexing but result in lower network utilization when a specific performance guarantee is to be satisfied.

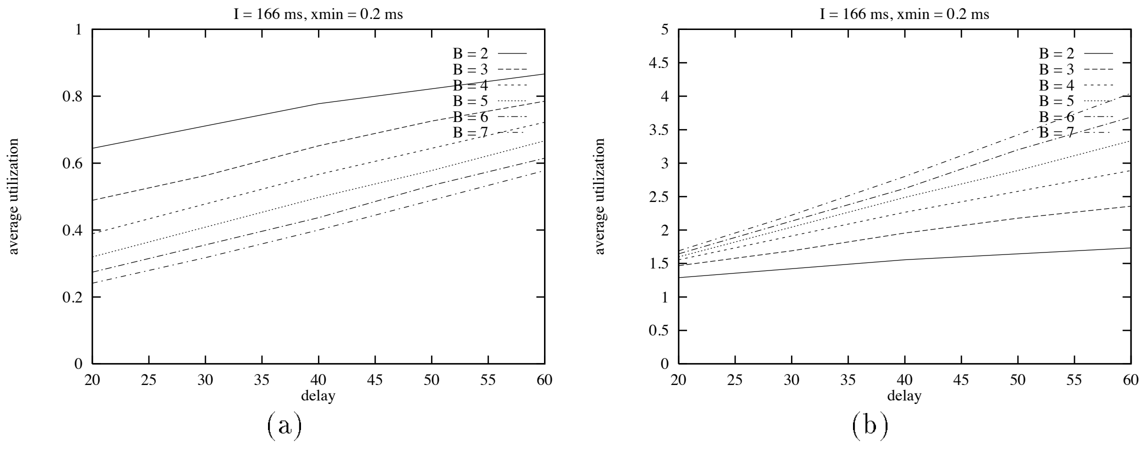


Figure 4.6: Effect of burst ratio

4.2.2 Heterogeneous Sources

This section presents an investigation into the effects of interactions between heterogeneous traffic sources.

Since there is not a closed-form solution for the distribution of the sum of binomial random variables with different parameters, the resulting distribution may be calculated by convolving the individual probability distribution functions. This convolution can be efficiently implemented with the Fast Fourier Transform. Numerical algorithms from [77] were used.

For simplicity, we consider different mixtures of two different sources. The algorithm, however, can calculate the case of arbitrary heterogeneous sources at no extra computational cost. A Class 1 source has $Xmin = 0.2$ ms, $Xave = 0.4$ ms, $I = 198$ ms, and a fixed packet size of 400 bits. A Class 2 source has a lower bandwidth, but has a greater burst ratio: $Xmin = 0.5$ ms, $Xave = 2.5$ ms, the same $I = 198$ ms and the same fixed packet size of 400 bits. Thus, a Class 1 source has a burst ratio of 2 with a peak rate of 2 Mbps and an average rate of 1 Mbps, while a Class 2 source has a burst ratio of 6 with a peak rate of 400 kbps and an average rate of 66.7 kbps. The delay overflow probability is 0.1%, and γ is 6.0.

Figure 4.7 shows the maximum numbers of Class 1 and Class 2 connections that can be accepted under different delay constraints. A point (n_1, n_2) on the curve means that if there are n_1 Class 1 connections traversing the link, at most n_2 Class 2 connections can be accepted over the same link. As in the case of homogeneous sources, more connections can be accepted when the delay bound is larger.

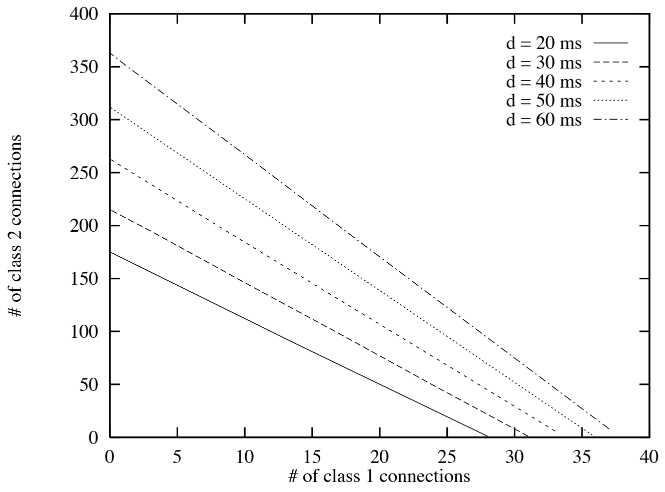
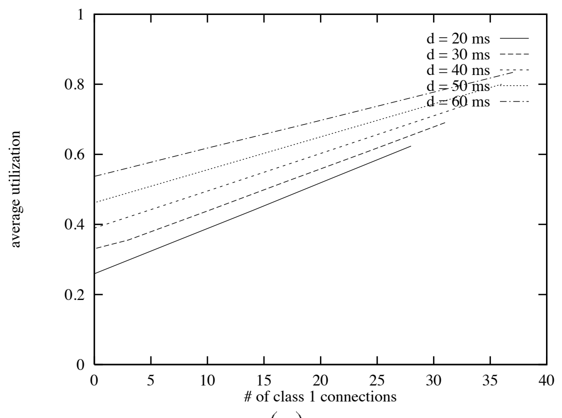
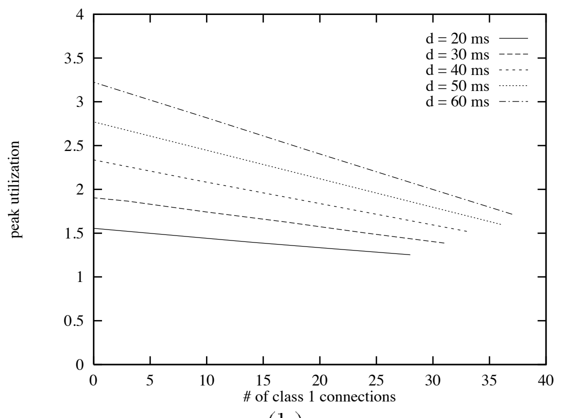


Figure 4.7: Maximum numbers of connections that can accepted

Figure 4.8 shows the average and peak utilizations of the link under different mixtures of Class 1 and Class 2 connections. Each number n_1 on the horizontal axis in the figures represents a 2-tuple (n_1, n_2) as defined in Figure 4.7. Since the traffic of a Class 2 connection is burstier than that of a Class 1 connection, a larger number of Class 1 connections means that the mixture has a larger fraction of Class 1 traffic, and is therefore less bursty. Figure 4.8 (a) shows similar characteristics to the case of homogeneous sources shown in Figure 4.6 (a): less bursty traffic is easier to multiplex and results in a higher average utilization of the link. Figure 4.8 (b) reconfirms the results we obtained for homogeneous sources (in Figure 4.6 (b)): although the average utilization is lower for burstier traffic, the peak utilization, a measure of the statistical multiplexing gain, is higher. Again, the intuition is that higher burst ratios provide more opportunity for statistical multiplexing but result in lower network utilization.



(a)



(b)

Figure 4.8: Average and peak utilizations with different mixtures

4.3 Providing End-To-End Statistical Guarantees

In the previous section, we presented the conditions for bounding the delay overflow probability in a Static Priority scheduler, and gave numerical examples to illustrate the results. In this section, we extend the analysis from a single scheduler to a network of switches.

In a networking environment, packets from different connections are multiplexed at each switch. Even if the traffic can be characterized at the entrance to the network, complex interactions among connections will distort the traffic pattern and destroy many properties of the traffic inside the network. Thus, the traffic model at the source may not be applicable inside the network.

One solution to this problem is to characterize the traffic pattern distortion inside the network, and derive the traffic characterization at the entrance to each switch from the characterizations of the source traffic and the traffic pattern distortion. This approach, taken in [19, 6, 68, 51], has several limitations.

First, it only applies to networks with *constant-delay* links. Constant-delay links have the desirable property that the traffic pattern at the receiving end of the link is the same as that at the transmitting end of the link. This property is important for these solutions because they use the characterization of the output traffic from a scheduler as the characterization of the input traffic to the next-hop scheduler. However, in an internetworking environment, links connecting switches may be subnetworks such as ATM or FDDI networks. Though it is possible to bound delay over these subnetworks, the delays for different packets will be *variable.* Thus, these solutions do not apply to an internetworking environment.

Second, most of the solutions characterize traffic in networks with *work-conserving* service disciplines [2]. Characterizing the traffic pattern inside the network is equivalent to solving a set of multi-variable equations [20, 68, 51]. In a feedback network, where traffics on different connections form traffic loops, the resulting set of equations may be unsolvable. Thus, most of these solutions apply only to feed-forward networks or a restricted class of feedback networks.

Finally, in networks with *work-conserving* service disciplines, even in situations in which the characterization of traffic is possible everywhere, the characterization inside the network usually corresponds to a burstier traffic than that at the source. This is independent of the traffic model being used. To see that this is the case, let us consider two traffic models, the deterministic bounding model by Cruz [19], and the stochastic bounding model by Kurose [51]. In [19], a source is said to satisfy the (σ, ρ) model if, during any time interval of length u, the amount of its output traffic is less than $\sigma u + \rho$. In this model, σ is the maximum burst size, and ρ is the average rate. If the traffic on connection j is characterized by (σ_j, ρ_j) at the entrance to the network, its characterization will be

$$(\sigma_j + \sum_{i'=1}^{i-1} \rho_j \overline{d}_{i',j}, \rho_j) \tag{4.13}$$

at the entrance to the $i-th$ switch along the path, where $\overline{d}_{i',j}$ is the local delay bound for the connection at the $i'-th$ switch. Compared to the characterization of the source traffic, the maximum burst size in (4.13) increases by $\sum_{i'=1}^{i-1} \rho_j \overline{d}_{i',j}$. This increase of burst size grows roughly linearly along the path.

In [51], If the j^{th} connection is characterized by $\{(R_{t_1,j}, t_1), (R_{t_2,j}, t_2), (R_{t_3,j}, t_3)...\}$ at the entrance to the network, its characterization will be

$$\{(R_{t_1+\sum_{i'=1}^{i-1} b_{i',j}}, t_1), (R_{t_2+\sum_{i'=1}^{i-1} b_{i',j}}, t_2), (R_{t_3+\sum_{i'=1}^{i-1} b_{i',j}}, t_3), ...\} \tag{4.14}$$

at the $i'-th$ switch, where $b_{i'}$ is the maximum busy period at switch i'. The same random variable that bounds the maximum number of packets over an interval at the entrance of the network now bounds the maximum number of packets over a much *smaller* interval at switch i. I.e., the traffic is burstier at switch i than at the entrance.

[2] With a work-conserving discipline, a link is never idle when there are packets waiting in its queue [93].

In both the (σ_j, ρ_j) and $\{(R_{t_1,j}, t_1), (R_{t_2,j}, t_2), (R_{t_3,j}, t_3)...\}$ analyses, the burstiness of a connection's traffic accumulates at each hop along the path from source to destination.

Another solution to the traffic pattern distortion problem, which we adopt in our approach, is to reconstruct the traffic pattern at each switch with rate-controlled service disciplines.

As shown in Chapter 3, rate-controlled service disciplines have the following properties:

(1) If a connection's traffic satisfies certain traffic characteristics at the entrance to the network, with the use of the appropriate rate-controllers the same characteristics will be satisfied by the traffic at the entrance to each of schedulers along the path. One type of rate-controller, the delay-jitter controlling regulator, completely reconstructs the original traffic pattern at each switch. If a connection traverses a path of rate-controlled servers with delay-jitter controlling regulators, the traffic pattern at the entrance to each of the schedulers is *exactly the same* as that at the entrance to the network. This allows us to perform delay analysis at each scheduler using the *same* traffic characterization.

(2) The end-to-end delay of a packet in a network of rate-controlled servers with delay-jitter controlling regulators can be bounded if the local delay can be bounded at each of the schedulers and each of the links.

Proposition 4.2 *Consider a connection passing through n switches connected in cascade where $\overline{\pi}_i$ is the upper bound on the delay of the link between the $i-1^{st}$ and the i^{th} switch. Assume that the scheduler of the i^{th} switch can guarantee that the delays of all the packets on the connection are bounded by $\overline{d_i}$ as long as the connection's input traffic to the scheduler satisfies traffic characteristics Θ. If the traffic on the connection obeys Θ at the entrance to the first switch, the end-to-end delay for every packet is bounded by $\sum_{i=1}^{n} \overline{d}_i + \sum_{i=2}^{n} \overline{\pi}_i$, and the delay-jitter is bounded by $\overline{d}_{i_n}$ if delay-jitter controlling regulators are used.*

The conclusion of Proposition 4.2 is more general than those of Theorem 3.1. Instead of requiring the connection to obey the $(Xmin, Xave, I, P)$ specification, any traffic specification Θ can be used. However, the proposition only applies to networks where delay-jitter controlling regulators are used.

Proposition 4.2 is quite general. It applies to networks with arbitrary topology, both feedback (aggregate traffic forming loops) and feed-forward networks, internetworks with variable but bounded link delays, and networks with rate-controlled servers that have different schedulers.

Properties (1) and (2) are significant. Property (1) means that we can analyze the delay characteristic of each scheduler along a path using the *same* traffic characteristics of the original source. The traffic characteristics need not be the ones discussed in this chapter. For example, if a connection can be characterized by a MMPP at the entrance to the network, it can be characterized by the *same* MMPP at each of the schedulers. Property (2) means that we can combine the delay analysis of each individual scheduler and obtain the end-to-end delay characteristics of a connection. If we assume that any packet missing the local delay bound at a scheduler is dropped immediately, the end-to-end delay overflow probability bound Z can be decomposed into local delay overflow probability bounds z_i, where $Z = \prod_{i=1}^{n} z_i$ and n is the total number of switches along the path traversed by the connection. This is done in [31].

In summary, with rate-controlled service disciplines and analyzing delay characteristics using the same traffic characterization at each scheduler will achieve higher network utilization than using work-conserving disciplines and characterizing the traffic inside the network. This is due to the fact that, in the latter case, a connection's traffic characterization must become more bursty with each hop. Additionally, using rate-controlled service disciplines allows us to obtain end-to-end performance bounds in much more general networking environments than previous solutions allowed.

4.4 Summary

In this chapter, we have presented a new approach to providing end-to-end statistical guarantees in packet-switching networks. We believe that there are two important aspects to the problem of providing guaranteed statistical services: developing bounding probabilistic models to characterize traffic sources, and developing techniques to provide end-to-end statistical bounds in a network environment.

As for the traffic model, we argue that traffic sources can be described in an interval-dependent way; i.e. a traffic source in general has different characteristics in intervals of different lengths. We present an interval-dependent stochastic traffic model that extends the Tenet deterministic traffic model $(Xmin, Xave, I, Smax)$ within Kurose's framework [51]. The model stochastically bounds the number of bits sent over time intervals of different lengths and requires that these distributions are explicit functions of the interval length. The model differs from previous stochastic models in two important aspects:

1. instead of *modeling* the source using some stochastic process like MMPP [59], our model uses random variables to stochastically *bound* the number of bits transmitted in different intervals;

2. instead of using one model to capture the traffic characteristics in any interval, we use a family of random variables with parameters that are explicit functions of the interval length.

We use binomial distribution for the bounding random variables. Other distributions can be used also. In [94], a continuous model based on two-state markov is presented to characterize the source.

As for enforcing end-to-end statistical bounds in a network environment, we first provided a local statistical bound in a single scheduler, we then extended it to a networking environment by using rate-controlled service disciplines. In the single node case, we analyzed the multiplexing of sources served by a Static Priority scheduler, considering both homogeneous and heterogeneous sources. We showed that the improvement of network utilization due to statistical multiplexing increases as the burst ratio becomes higher; however, the overall average network utilization is lower when the burst ratio is higher. By using delay-jitter control, we extend these results to a network to provide end-to-end per-connection statistical performance guarantees. As in the case of deterministic service, the results hold in general networking environments with arbitrary topology, both feedback and feed-forward networks, internetworks with variable but bounded link delays, and networks with rate-controlled servers that have different schedulers. Compared to the approach of using work-conserving disciplines and characterizing traffic patterns inside the network, rate-controlled service discipline allows *more* connections to be admitted in *more general* networking environments.

Chapter 5

The Real-Time Internet Protocol

In Chapters 3 and 4, we focused on the analysis of service disciplines using deterministic and statistical techniques. In this chapter, we describe our effort to implement the proposed algorithms in an internetworking environment.

The predominant protocols used in current networks and internetworks are those in the TCP/IP protocol suite [74, 75, 76]. This protocol suite was designed to support data communication. Some of the most important goals were: survivability in the face of failure, multiple types of transport service, and support for varieties of networks [16]. Based on these design goals, the protocol suite adopted a connectionless architecture. The network layer protocol, the Internet Protocol, or IP [75], provides only a best-effort datagram service.

As discussed in Chapter 1, new applications such as video conferencing, scientific visualization and medical imaging have stringent performance requirements in terms of delay, delay jitter, bandwidth and loss rate. There is an increasing demand to support these applications in an internetworking environment.

The Tenet Group at University of California at Berkeley and the International Computer Science Institute has designed and implemented a new protocol suite that provides guaranteed performance service in an internetworking environment [26, 27]. The protocol suite consists of five protocols: three data delivery protocols and two control protocols. The three data delivery protocols are the Real-Time Internet Protocol (RTIP), the Real-time Message Transport Protocol (RMTP) [85, 95] and the Continuous Media Transport Protocol (CMTP) [88, 28]. RTIP is the network layer protocol, while RMTP and CMTP are two transport layer protocols that provide message-oriented and stream-oriented transport services, respectively, on top of RTIP. The two control protocols are the Real-Time

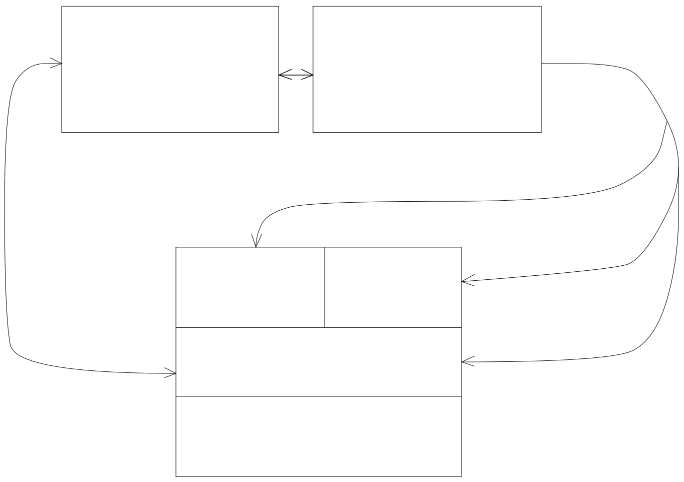


Figure 5.1: Tenet Protocol Suite

Channel Administration Protocol (RCAP) [7], and the Real-Time Control Message Protocol (RTCMP). RCAP is responsible for establishment, tear-down and modification of the real-time channels, while RTCMP is responsible for control and management during data transfers. Figure 5.1 shows the architecture of the protocol suite.

Several other algorithms or protocols have been proposed to provide Quality of Service in an internetworking environment. Among them are the Stream Protocol Version II (ST-II) [83], the Multipoint Congram-oriented High-performance Internet Protocol (MCHIP) [70], the Flow Protocol [96], a proposal for predicted service [17, 98], and a proposal for resource management in datagram networks [32]. However, none of these protocols provide both guaranteed bounded-delay and bounded-delay-jitter services.

In this chapter, we focus on the network layer data delivery protocol RTIP. RTIP provides a host-to-host, simplex, sequenced, unreliable, and guaranteed-performance packet service. Some of the functions of RTIP are rate control, jitter control and packet scheduling. We have implemented RTIP in Ultrix on DECstation 5000 workstations and in HP/UX on HP9000 workstations in an FDDI environment [1, 2]. The implementation has been ported to other platforms and environments. The operating system and workstation platforms includes SunOS on SPARC and Sun 4/280 workstations, and IRIX on SGI IRIS-4D workstations. Besides FDDI, RTIP has also been ported to such environments as Xunet-2 [36], a wide-area internetwork consisting of FDDI and ATM networks; SequoiaNet [29], a wide-area T1 and T3 network; and a local-area HIPPI network[14] including the RAID-II high performance storage server [71, 53, 47].

The service disciplines we used in these implementations were Delay-Earliest-Due-Date, Jitter-Earliest-Due-Date and Rate-Controlled Static Priority. For the latter two rate-controlled service disciplines, we implemented both rate-jitter controlling regulators and delay-jitter controlling regulators.

The rest of the chapter is organized as follows: Section 5.1 specifies the environment assumed by the Tenet Real-Time Protocol Suite; Section 5.2 describes the services, functions, protocol data unit format, and interfaces between a RTIP module and other modules in the system; Section 5.3 describes the implementation of RTIP in a BSD Unix-like environment; Section 5.4 presents measurements of the RTIP implementation on DECstations over a local FDDI testbed; and Section 5.5 summarizes the chapter.

5.1 Environment

The Tenet protocol suite provides guaranteed performance services in an internetworking environment.

We assume an internetwork consisting of a number of nodes connected together by links. A node may be either a host or a gateway. Only hosts are allowed to be the end-points of communication at the network layer. All nodes have Internet or IP addresses [75]. A link can either be a physical link or a subnetwork. The delay on the links must be boundable [26].

We assume that the nodes are store-and-forward nodes. The RCAP and RTIP protocols must be implemented in all the hosts participating in real-time communications and all the gateways which real-time channels traverse. The RCAP protocol should also be implemented inside subnetworks that cannot provide bounded-delay services without having resources managed explicitly by RCAP; it need not be implemented inside a subnet if the delay within it can be bounded by the subnet. At least one of the RMTP and CMTP protocols needs to be implemented in the hosts participating in real-time communications. Two hosts can communicate with each other if they both implement the same real-time transport protocol.

We assume that the loss rates on physical links or within subnetworks are low, and that packets are not (or only infrequently) misordered by links or subnetworks.

To provide delay-jitter-bounded service, we assume that the nodes in the network have synchronized clocks. The accuracy of clock synchronization is a lower bound on the delay-jitter bounds that can be guaranteed by RTIP.

Finally, the performance guarantees are obeyed only when there are no hardware or software failures in the nodes or links involved.

5.2 RTIP Design

RTIP is the network layer data delivery protocol in the Tenet protocol suite. In this section, we will describe the services, functions and interfaces provided by RTIP.

5.2.1 RTIP Services

RTIP provides a host-to-host, simplex, sequenced, unreliable, and guaranteed-performance packet service. All the data are transferred on a simplex channel from the sending client to the receiving client in packets. A packet is not guaranteed to be delivered; it may be dropped. For packets not dropped, they are delivered to the receiving client in the same order as they were sent by the sending client; the relative positions of the dropped packets are indicated. The client data are not checksummed; they may get corrupted due to transmission errors. If the sending client sends packets neither larger than the maximum packet size negotiated during channel setup time, nor faster than the specified maximum rate, packets delivered are guaranteed to meet the delay and delay jitter requirements with certain probability, which is specified by the client during connection establishment time.

5.2.2 RTIP Functions

Operating at each host and gateway participating in the real-time communication, RTIP performs the following functions:

- *rate control:* RTIP monitors the packet traffic arriving from each channel, verifies that it does not violate the traffic specification, and, if necessary, enforces compliance. This is achieved by using rate-based service disciplines [93] and managing buffer space on a per-connection basis. The traffic model is $(Xmin, Xave, I, Smax)$. There are two forms of rate control: if the node is implemented as a *rate-allocating server* [93], packets on a connection that violates its traffic specification are transmitted as fast as possible provided that performance guarantees for packets on other connections sharing the same server are not violated; if the node is implemented as a *rate-controlled server*, packets on a connection that violate its traffic specification will be held regardless of whether there is spare capacity in the server [93]. Since buffer space is allocated on a per connection basis, if the violation of the traffic specification for one channel persists, packets from that channel will eventually be dropped as the buffer space allocated to it will overflow. The packet dropping policy is not specified in the protocol;

- *jitter control:* for a channel requiring bounded-jitter service, RTIP is responsible for eliminating the jitter accumulated by a packet in the previous segments of the channel's path. This is achieved by calculating an eligibility time for each packet, and holding the packet until that time before scheduling it for transmission;
- *packet scheduling:* RTIP schedules packet transmissions according to a policy;
- *data transfer:* RTIP transfers packets to the next node along the path or passes it to the appropriate upper layer protocol at the destination;
- *statistics collection:* RTIP optionally collects per-node per-channel packet statistics at each node, and end-to-end per-channel packet statistics at the destination host.

Notice that certain functions that are usually associated with other network layer protocols are not supported in RTIP. In particular, RTIP does not perform packet fragmentation and reassembly. This function is performed by higher layer protocols. This separation of functions between network layer protocol and upper layer protocols is possible because RTIP is connection-oriented: it uses fixed routing and has a connection establishment phase before the data transfer begins. Fixed routing ensures that there is a fixed Maximum Transmission Unit (MTU) [1] during the life time of the connection; the connection establishment phase allows the path MTU to be discovered before the data transfer. This is not true with a connectionless network, where algorithms have to be designed to discover the path MTU during the data transmission phase [63].

[1]A path MTU is the largest size of a packet that does not require fragmentation anywhere along the path from the source to the destination.

5.2.3 Protocol Data Unit Format

Each RTIP packet has a fixed length (20 bytes), 32-bit aligned packet header. It has been suggested by previous research that adopting a fixed length and word-aligned header format simplifies protocol implementation and speeds up protocol execution, therefore is appropriate for high-speed networking protocols [13].

The header format for a RTIP packet is shown in figure 5.2.

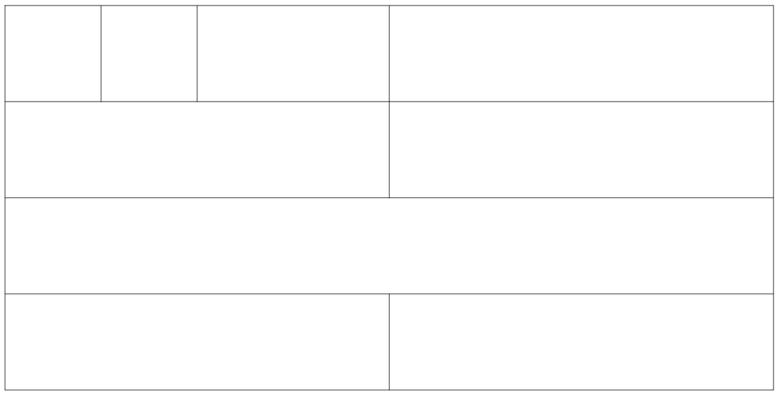


Figure 5.2: RTIP packet header format

`RTIP` is a 4-bit field that is used by lower-layer modules to identify RTIP packets.

`Ver` is the RTIP version number.

`Local Channel Identifier` is a 16-bit number that uniquely identifies the channel at the next downstream node along the path. This number is chosen by RCAP at channel establishment time. A real-time channel is uniquely identified by the combination of its local channel identifier at the source node and the IP address of the source node.

`Packet Length` is a 16-bit number denoting the length of the packet in bytes. The maximum packet size is 64 Kilobytes.

`Packet Sequence Number` is a 16-bit number that increases monotonically by one at every new packet packet transmitted on the same channel;

`Time Stamp` is a 32-bit number representing time with a granularity of 2^{-16} seconds. The content of the timestamp depends on the particular jitter control scheme used. It is either the time when the packet was received by the RTIP module at the sending host, or the amount of time the packet was ahead of schedule in the previous node.

`Header Checksum` is a 16-bit CRC computed over all the fields in the RTIP header.

5.2.4 RTIP interfaces

At both the hosts and gateways, the RTIP protocol entity interfaces with the underlying data link protocol entity and the RCAP protocol entity. At the hosts, RTIP also interfaces with upper layer protocol entities.

Interface with Upper Layer Protocols

The following interface primitives, written in C for convenience, are exported to protocols immediately above RTIP. At the source, the interface is:

```
RtipSend (
        u_short  localChannelID,
        caddr_t  packet,
        u_short  packetLength,
        TIME     eligibleTime )
```

where `localChannelID` is the local channel identifier for the channel at the source, `packet` is a pointer indicating the location of the data to be transferred, `packetLength` specifies the packet's length in bytes, `eligibleTime` is the time at which the packet should become schedulable for transmission.

At the destination, the interface is:

```
RtipIndicate (
        u_short localChannelID,
        caddr_t packet,
        u_short packetLength,
        u_short packetSequenceNum)
```

where `localChannelID` is the local channel identifier for the channel at the destination, `packet` is the pointer indicating the location of the data received, `packetLength` specifies the packet's length in bytes, `packetSequenceNum` is the sequence number of the received packet.

The delay of a RTIP packet is defined as the time difference between the `eligibleTime` used in the send operation and the time instant the `rtipIndication` routine was called by the RTIP module at the receiver.

Interfaces With Lower Layer Protocols

The RTIP protocol entity interfaces with the lower data link layer protocol entity at each node.

There are two interface function calls: one call from the RTIP module to the data link module when a packet is eligible to be sent, and one upcall from the data link module to the RTIP module when a packet is received.

They are respectively:

```
InterfaceOutput(
        ifnet          *interface,
        caddr_t        packet,
        u_long         channelType,
        u_long         deadline,
        struct Address *nextHopAddress)
```

and

```
RtipReceive(
        ifnet          *interface,
        caddr_t        packet)
```

where `interface` specifies the output or input interface, `packet` points to the packet, `channelType` specifies the channel type (either deterministic or statistical), `deadline` gives the deadline of the packet, and `nextHopAddress` is the data link layer address of the next hop node.

Interface With RCAP

RTIP exports interfaces to RCAP to set up and release channel states. `SetChannelSpec` is used by RCAP to set up the state for a channel at a node.

```
SetChannelSpec(
    u_short                localChannelID,
    u_short                downStreamLocalChannelID,
    boolean                jitterControl,
    struct  sockaddr_in    downStreamName,
    struct  TrafficSpec    trafficSpec,
    struct  PerfSpec       perfSpec,
    struct  BufferSpec     bufferSpec)
```

where `localChannelID` is the local channel identifier of the channel at the current node, `downStreamLocalChannelID` is the local channel identifier of the channel at the downstream node (`downStreamLocalChannelID` is 0 if the current node is the destination of the channel), `jitterControl` specifies whether there should be jitter control on this channel, `downStreamName` is the IP address of the down stream node, `trafficSpec` specifies the traffic characteristics of the channel, `perfSpec` specifies the local requirement of the channel, and `bufferSpec` specifies the buffer space that need to be reserved.

`ReleaseChannel` is used by RCAP to release the state associated with a channel.

```
ReleaseChannel(u_short      localChannelID)
```

where `localChannelID` is the local channel identifier of the channel.

5.3 Implementation

The transport layer protocol RMTP and the networking layer protocol RTIP have been implemented in Ultrix on DECstation 5000 workstations and in HP/UX on HP9000/7000 workstations. Both Ultrix and HP/UX are Unix-like operating systems [79] with networking software derived from BSD Unix [55, 54].

In BSD Unix, the communication system is logically divided into two layers: the interprocess communication (IPC) subsystem, and the network communication subsystem. The IPC subsystem, layered above the network communication subsystem, provides sockets, a unified abstraction for endpoints of communication. The network communication subsystem is further divided into three layers: transport layer, network layer, and network-interface layer. The transport and network layer software is protocol-dependent. In BSD Unix, the network system is designed so that multiple *protocol families* may coexist. The network-interface layer is mainly concerned with driving the transmission media involved and performing any necessary link-level protocol encapsulation and decapsulation.

The software structure of the RMTP and RTIP implementation is shown in Figure 5.3. As shown in the figure, while RCAP is implemented in user space, RMTP and RTIP are implemented in the kernel and co-exist with TCP, UDP and IP. Through the socket

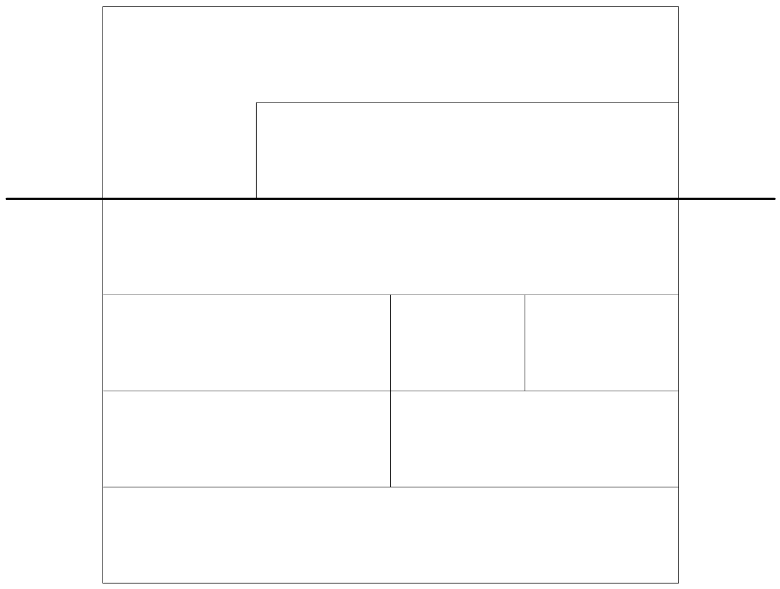


Figure 5.3: Software structure of RMTP/RTIP

layer software, RMTP and RTIP export interfaces to RCAP for control purposes and to user programs for data transfer purposes.

Protocols previously implemented in the BSD Unix provide only best-effort service. In most implementations, e.g., in HP-UX and in Ultrix, there is a buffer pool shared among all the connections, and the queueing discipline used is First-Come-First-Served. To guarantee performance requirements, the implementation of RTIP needs to have priority queueing mechanisms, as well as per-connection buffer management and rate control.

5.3.1 Programming Interface

RMTP and RTIP export programming interfaces to both user programs and RCAP agents. The following example illustrates how a user program sends data using the Tenet protocol suite.

The user program shown in Figure 5.4 first invokes the `socket` system call to open an RMTP socket. Next, the program calls `RcapEstablishRequest` to establish the real-time channel. As part of the request, the user specifies the traffic specification and end-to-end performance requirements in `ParametersBlock`. The remaining arguments are `lcid`, used to hold a return value, and `Destination`, the destination IP address. `RcapEstablishRequest` is an RCAP library function; when invoked, it contacts the RCAP agent at the sender's

```
DataSock = socket(AF_INET, SOCK_DGRAM, IPPROTO_RMTP)
RcapEstablishRequest(&ParametersBlock, &lcid, &Destination)
setsockopt(DataSock, IPPROTO_RTIP, RTIP_ASSOC, &lcid, sizeof(u_short))
send(DataSock, msg, len, flags)
```

Figure 5.4: Programming interface for a user program

node and as well as the RCAP agents at all nodes on the path from source to destination. Each RCAP agent along the path reserves local resources for the new request and establishes the real-time channel. Each agent resides in user space and calls

```
setsockopt(ControlSocket, IPPROTO_RTIP, RTIP_SPEC, &RtipSpec,
                              sizeof(struct RtipSpec))
```

upon receiving the RCAP message. With this system call, the local agent sets up a data structure in RTIP using `ControlSocket`, a permanent socket opened for communication between the RCAP agent in user space and the RTIP module in kernel space on the same machine. The local traffic specification, the performance parameters, and the amount of reserved resources are given to the RTIP module via the `RtipSpec`.

If the channel is successfully established, `RcapEstablishRequest` returns a small integer `lcid`, which is the unique local channel identifier of the real-time channel at the source. As shown in the figure, the user program then calls `setsockopt` to associate the opened data socket with the established real-time channel. Finally, the user can start sending data to the data socket using the `send` system call.

A similar interface is provided on the receiving side. There are also other `setsockopt` calls that RCAP uses to communicate with RMTP and RTIP.

5.3.2 Internal Data Structure

For each RTIP connection, there is a data structure called *RtipPcb*, or RTIP protocol control block, associated with it. The RtipPcb structure contains all the relevant information of the connection, and is shown below.

```
struct RtipPcb {
    u_int                 ipb_state;
    struct inpcb          *ipb_inpcb;       /* back pointer to inpcb */
    char                  *ipb_rttpcb;      /* pointer to transport protocol */
    char                  *ipb_so;          /* pointer to socket             */
    struct mbuf           *ipb_buffer;      /* list of rtip buffers          */
    int                    ipb_currentBuf;  /* number of buffers available   */
    int                    ipb_totalBuf;    /* total number of buffers       */
    int                    ipb_pktSize;     /* maximum packet size           */
    struct TrafficSpec     ipb_trafficSpec; /* traffic specification         */
    struct PerfSpec        ipb_perfSpec;    /* performance specification     */
    struct ChannelID       ipb_globalID;    /* source IP + source LCID       */
    struct ForwardingInfo  ipb_forwardingInfo; /* next hop address           */
    struct RtipRunTime     ipb_rtipRunTime;  /* run time state info          */
    struct RtipStat        ipb_stat;         /* statistics                   */
    int                    (*ipb_transportRev)();  /* upper layer receiving  *
                                                    * function               */
    u_short                ipb_lcid;          /* lcid number                 */
    u_char                 ipb_transportType; /* CMTP or RMTP                */
    u_char                 ipb_rtipOptions;   /* options                     */
};
```

5.3.3 Buffer Management

In BSD Unix, a special data structure called *mbuf* is used to manage the memory for interprocess communication and networking subsystems. An ordinary mbuf structure has 128 bytes with 112 bytes reserved for data storage. If a packet has more than 112 bytes of data, multiple mbufs, which are combined as a linked list, are used to store the packet. When packets are large, the overhead of fragmenting data into a number of small buffers and traversing the linked list becomes large. Thus, a second type of mbuf, called *cluster mbuf*, was added to store larger packets. Each cluster mbuf has a pointer to a larger external buffer, where more data can be stored. Mbufs are allocated from a system-wide central pool on a per-packet basis.

To implement per-connection buffer management, a number of cluster type mbufs and the associated external buffers are allocated to each connection at connection establishment time. The number and size of the buffers are specified by RCAP through a `setsockopt` call from RCAP to RTIP. When a packet arrives, the data is copied into the first available buffer. The associated mbuf is then passed through different software modules for processing. After the data has been transmitted or passed to the user level program, the system calls the routine `m_free()` to release the buffer. In the current system, since memory is managed as a central pool, the `m_free()` routine does not need to know which connection this packet belongs to; it just frees the buffer and returns it to the central buffer pool. For per-connection buffer management, the buffer should be returned to the connection's buffer pool. A mechanism is needed to identify which connection the buffer belongs to. One possibility is to change the `mbuf` data structure so that it has one more field to identify the connection, and to change the `m_free()` routine so that it can return the buffer to the connection specified by the new parameter. However, `mbuf` is a very commonly used data structure in the system, and changing it would affect too much existing software. The mechanism used in the implementation of RTIP is a special purpose mbuf that is designed for the Network File System. Such a mbuf has two fields, `mun_clfun`, which is a function pointer, and `mun_clarg`, which is a parameter. When the mbuf is released, the system does not return to the central buffer pool, but calls the function pointed by `mun_clfun` with the parameter `mun_clarg`. For the purpose of RTIP buffer management, `mun_clfun` points to a RTIP buffer management function `RtipFreeBuf`, and `mun_clarg` has the value of the Local Channel Identifier of the connection. The `RtipFreeBuf` function, when called, returns the private buffer associated with the mbuf to the connection's buffer pool.

5.3.4 Rate Controlled Service Discipline

Some of the most important functions of RTIP are rate control, jitter control, and packet scheduling. These functions can be implemented using rate-based service disciplines. We have implemented a number of service disciplines in the prototype; they include Delay-Earliest-Due-Date, Jitter-Earliest-Due-Date and Rate-Controlled Static Priority. For the latter two rate-controlled service disciplines, we implemented both rate-jitter controlling regulators and delay-jitter controlling regulators. The implementation is modular, so that other types of regulators and schedulers can be easily implemented without affecting the rest of the system.

The implementation of a rate-controlled service discipline can be logically divided

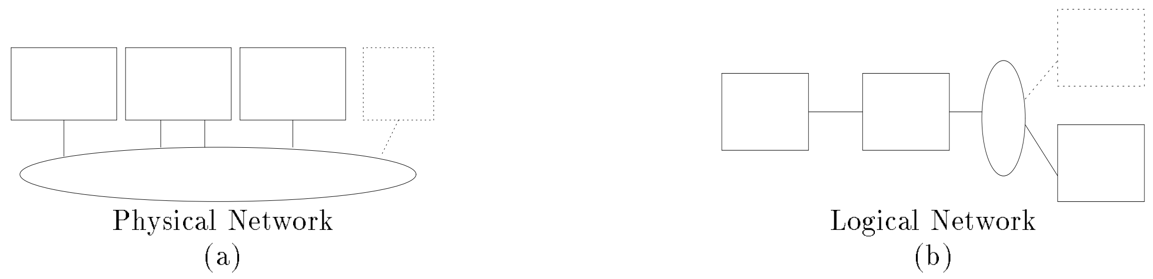
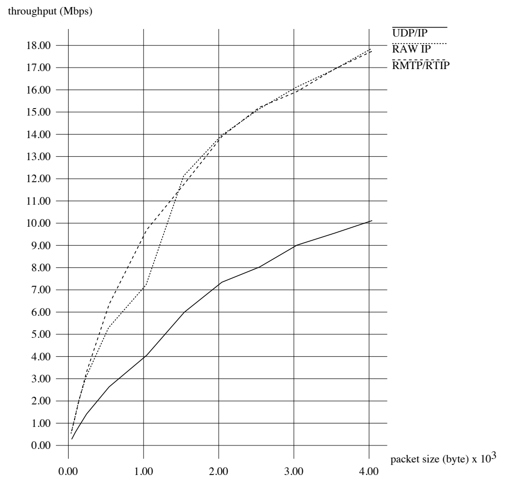


Figure 5.5: Testbed configuration

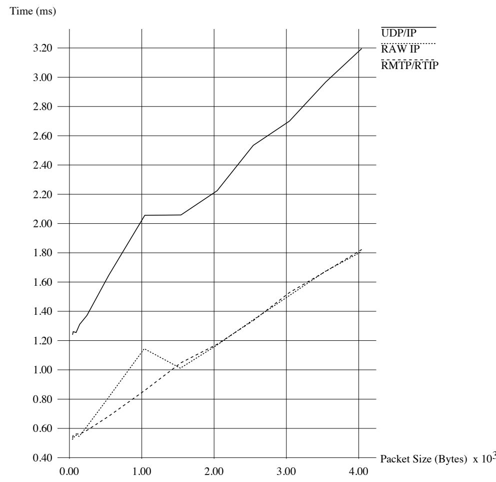
into three tasks: calculating of the eligibility time of each packet, holding the packet if necessary, and enqueueing and dequeueing of packets at the scheduler. The eligibility time of each packet can be calculated using the formula defined in Chapter 3. Holding packets is implemented differently in the gateways than in the hosts. In a source host, where packets are sent from some processes, holding packets can be implemented by putting the transmitting process into sleep. In a gateway, where packets are to be forwarded, there is no context or process associated with packets, and thus the sleeping mechanism cannot be used. Instead, the `timeout` procedure is used to insert the packet into a waiting queue whenever the packet's eligibility time is greater than the current time. `Timeout` is a mechanism provided by Unix to support the execution of events based on timers. It takes three parameters as its arguments: a pointer to a procedure, a pointer to a parameter, and a time period. When the time period expires, the procedure is invoked with the specified parameter. Finally, the implementation of the dequeue and enqueue operations depends on the scheduling policy. In our prototype implementation, the EDD queue is organized as a linked list, while the Static Priority queue is organized as a number of linked lists. Thus, insertion and deletion operations for EDD are $O(N)$ and $O(1)$ time, respectively; insertion and deletion operations for SP are all $O(1)$. All the measurement experiments in the next section were performed using the EDD scheduler. Since in the experiments, the transmission speed was limited by the CPU speed, output queues were not long, and the $O(N)$ insertion operation in EDD did not appreciably affect the performance of RTIP.

5.4 Measurement Experiments

In order to evaluate the performance of the RMTP/RTIP implementation, we set up a local testbed and performed a set of experiments on the testbed [92].



Throughput



Time to Send One Packet

Figure 5.6: Performance of UDP/IP, raw IP and RMTP/RTIP

The physical configuration of the testbed is shown in Figure 5.5(a). Both *theorem* and *lemma* are DECstation 5000/125's, while *faith* is a DECstation 5000/240. *Fake* is a non-existent machine; it is used to be the receiver of artificial network background load. All three workstations are connected to one FDDI ring. While *theorem* and *faith* each have one FDDI interface attached, *lemma* has two FDDI interfaces. The routing tables on the three machines are set up to form the logical network shown in Figure 5.5 (b).

5.4.1 Throughput and Processing Time

Our first experiment compares the throughput of RMTP/RTIP with those of UDP/IP and raw IP. The experiment was performed with one process on *theorem* sending 10,000 packets of equal size in a tight loop to a receiving process on *faith*. No load was applied to either the gateway or the hosts.

Tests were performed with different packet sizes and different underlying protocols. For RMTP/RTIP, the rate parameters (Xmin, Xave, I) were set so that the "reserved rate" was higher than the achievable rate. This was just for experimentation purposes, so that packets could be sent back-to-back without being held by the rate control mechanism. Figure 5.6 shows the results of the experiment. The diagram on the left shows the throughput, and the diagram on the right shows the amount of time to send a packet during a tight sending loop on theorem. It can be seen from the figure that the throughput

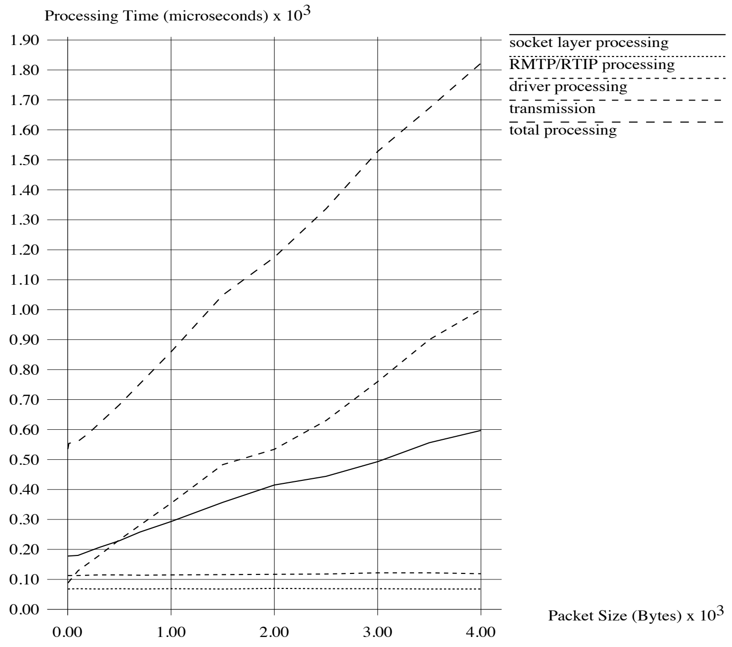


Figure 5.7: Breakdown of processing times in the kernel

achieved using RMTP/RTIP is almost the same as that using raw IP. This result is not surprising. It has been observed in [18] that, for bulk data transfers, the cost of operations on data such as copying or checksumming dominates that of protocol processing. Since neither RMTP/RTIP nor raw IP do checksumming, the operations on data are the same for both of them; thus, the times for sending one packet and the throughput are approximately the same in both cases. The UDP throughput was measured with UDP checksumming turned on, which explains the discrepancy between its curves and the curves of raw IP and RMTP/RTIP.

Figure 5.7 shows the breakdown of the processing time of a RMTP/RTIP packet on a sending DS 5000/125. As can be seen, the RMTP/RTIP protocol processing time is about 68 μs per packet, and the driver software processing time is about 114 μs per packet. Socket level processing includes copying the data from user space into kernel space; transmission includes copying the packet from the kernel memory into the interface adaptor and transmitting the packet onto the FDDI ring. Both types of processing involve data movement and account for a larger fraction of the total processing time.

5.4.2 Effects of Rate Control

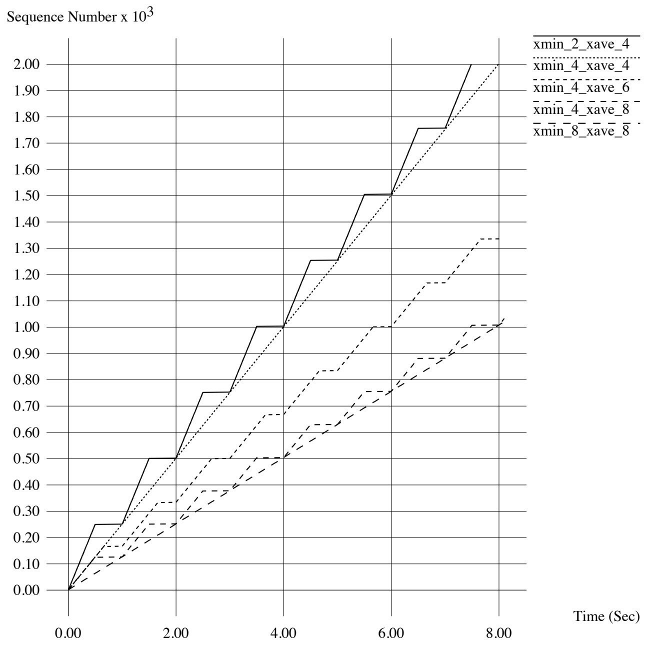


Figure 5.8: Effects of rate control at the source in RMTP/RTIP

One of the functions of RMTP is to perform rate control at the source so that the packets are injected into the network at a rate not higher than declared when the channel was established. This experiment was designed to check the effectiveness of rate control.

In the experiment, a process on *theorem* sent packets continuously to a process on *faith* through a real-time channel in a tight loop. The receiver program reads the clock upon receiving each packet.

Figure 5.8 displays the relationship between packet sequences and timestamps. Each of the experiments whose results are reported in the figure was performed independently with different values of $Xmin$ and $Xave$. $Xmin$ and $Xave$ are expressed in milliseconds. The same value of 1 second was used for I in all experiments.

Although the sender program sends packets in a tight loop, the rate control mechanism in the RMTP/RTIP protocol ensures that the source never sends packets faster than what is specified by traffic parameters. As can been seen in the figure, the $(Xmin, Xave, I, Smax)$ traffic restrictions have been indeed satisfied by the rate control mechanism.

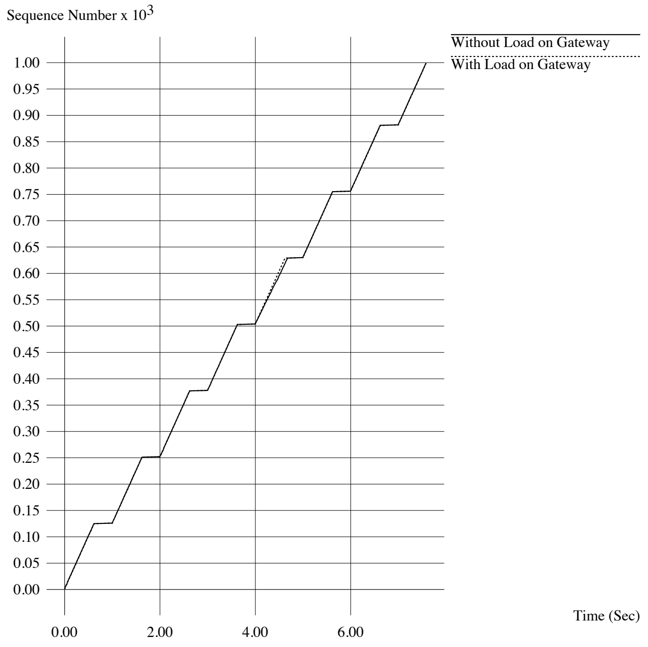


Figure 5.9: Effect of gateway load on real-time channel

5.4.3 Effects of Gateway Load

This experiment was designed to examine the effect of gateway load on real-time channels. In the experiment, a process on *theorem* sent packets to a process on *faith* via a real-time channel. We timestamped each packet and recorded its sequence number at the receiving end. Two tests were performed: gateway unloaded and gateway loaded. To load the gateway, a couple of processes were created on the gateway machine *lemma*, and each of them sent raw IP packets in a tight loop to the non-existing machine *fake*. Figure 5.9 shows both the loaded and the unloaded case. We can see that the load on the gateway did not affect visibly the performance of the real-time channel.

A similar experiment was performed for UDP. Under load, a significant fraction of UDP packets were dropped at the gateway.

5.4.4 Co-existence of Real-Time Channels

The previous experiment shows that RMTP/RTIP traffic is not affected by the presence of IP traffic. A more interesting case is that of multiple real-time channels.

We did three tests: channel 1 active alone, channel 2 active alone, and channel 1

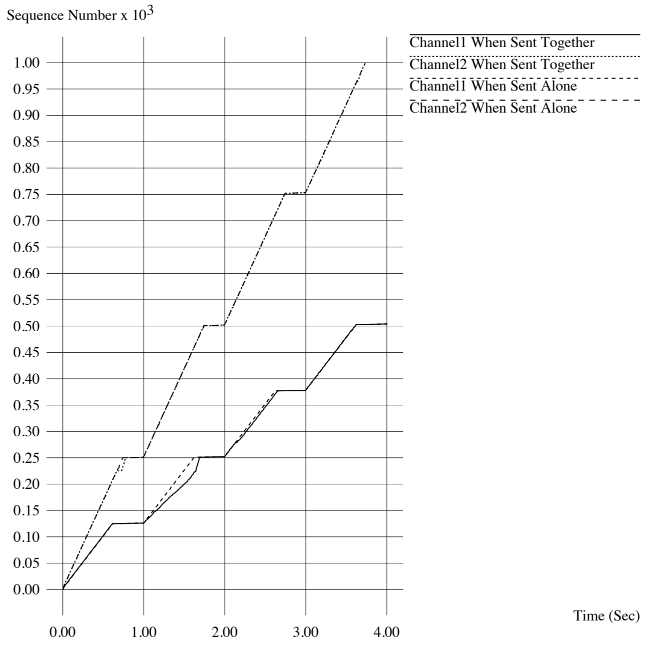


Figure 5.10: Interference between two real-time channels

and channel 2 active simultaneously. We timestamped each packet and recorded its sequence number at the receiving end in all the tests. Figure 5.10 shows the sequence number vs. time graph for all three cases. As can been seen in the figure, for both channel 1 and channel 2, the patterns of packet arrival are almost the same regardless of whether the other channel is active.

5.5 Summary

In this chapter, we have presented an implementation of rate-controlled service disciplines in an internetworking environment. We described the design and implementation of an internetwork layer protocol called Real-Time Internet Protocol or RTIP. RTIP is the network layer data delivery protocol within Tenet real-time protocol suite, and is the first protocol of its kind that provides host-to-host bounded-delay and bounded-delay-jitter services in an internetworking environment.

We have implemented RTIP in Ultrix on DECstation 5000 workstations and in HP/UX on HP9000/7000 workstations. The service disciplines used in the implementation are rate-controlled service disciplines. A measurement study has been conducted to evaluate the performance of the prototype implementation. The study shows that the throughput obtained by using RMTP/RTIP is comparable to that obtained by using raw IP; the rate control mechanism in RMTP/RTIP effectively enforces the traffic specification of communication clients; the scheduling mechanism in RTIP protects real-time channels so that their the performance is not affected by the presence of IP traffic or other real-time channels in the network.

Our research has suffered from some limitations, which should be addressed in the future:

- The current testbed is a local area network. We would like to perform similar measurements in an internetwork such as the Xunet-2 [36] testbed and the SequoiaNet [81, 29].
- All the experiments were performed with synthetic loads. In the future, we would like to experiment with real applications such as video conferencing or scientific visualization.
- The testbed consists of only DECstations. A heterogeneous environment would be more interesting.
- The current version of the Tenet protocol suite supports only unicast communications. Research is under way to design and implement a new Tenet real-time protocol suite called Suite 2 [27]. Suite 2 will support multi-party interactive multimedia applications [64] more efficiently by using multicast as the basic abstraction.

Chapter 6

Summary and Future Work

6.1 Summary

In this dissertation, we studied the issues and tradeoffs that are to be faced when designing service disciplines for integrated-services packet-switching networks.

In Chapter 1, we argued that the design of integrated-services networks is fundamentally different from that of traditional networks, and that the service discipline is one of the most important mechanisms in such a design. Whereas traditional communication networks offer one type of service that supports one type of applications [1], integrated-services networks will have to support applications with diverse traffic characteristics and performance objectives. While current packet-switching data networks provide a highly flexible but only best-effort service, and circuit-switching voice networks provide guaranteed but extremely inflexible services, integrated-services networks have to support *flexible* and *guaranteed* services. To be flexible, integrated-services networks use *packet-switching* technology. To provide guaranteed services, a *proactive* network control approach is needed. The service discipline at the switching nodes in an integrated services packet-switching network is one of the most important components of a proactive network control architecture.

In Chapter 2, we presented a taxonomy and a framework for studying service disciplines in integrated-services networks. Given the framework, we discussed issues and tradeoffs in the design of service disciplines for packet-switching integrated-services net-

[1]In data networks, multiple types of service are offered at the transport layer (e.g., reliable stream service as provided by TCP and unreliable datagram service as provided by UDP) and at the application layer (remote login, file transfer, and electronic mail). Here we are concerned with the network layer service. Only one type of service is offered in current data networks (e.g., best effort datagram service in the Internet.

works, and showed the limitations of several existing solutions. The existing solutions are based on either a time-framing strategy, or a sorted priority queue mechanism. Time-framing schemes suffer from the dependencies they introduce between the queueing delay and the granularity of bandwidth allocation; sorted priority queues are more complex, and are difficult to implement at high speeds. Moreover, these solutions were designed for simple networks, where switches are connected by constant-delay links. In an internetworking environment, links connecting switches may be subnetworks, which have bounded but *variable* link delays. It is important to design service disciplines that can provide performance guarantees in both simple networks and internetworks.

In Chapter 3, we proposed a new class of service policies called *rate-controlled service disciplines*. The key feature is the separation of the server into two components: a rate-controller and a scheduler. After the rate-controller limits the distortion of the traffic introduced by load fluctuation inside the network, the scheduler orders the packets for transmission. This class of service disciplines may be non-work-conserving, i.e. a server may be idle even when there are packets to be transmitted. Non-work-conserving disciplines were seldom studied in the past. In Chapter 3, we showed that rate-controlled service disciplines have several advantages that make them suitable for supporting guaranteed performance communication in a high speed networking environment:

1. End-to-end deterministic performance bounds can be obtained in a network of arbitrary topology. Unlike most work-conserving disciplines, which can provide bounds only in feed-forward networks and some restricted classes of feed-back networks, rate-controlled disciplines provide bounds in arbitrary networks.

2. Unlike most existing solutions, which assume *constant* link delays between switches, we need only assume *bounded* link delays when rate-controlled disciplines are used. This is particularly important in an internetworking environment, where switches are connected by subnetworks. The delays of packets traversing subnetworks may be *bounded* but*variable.*

3. Given this separation of functionality into rate control and packet scheduling, we can have arbitrary combinations of rate control policies and packet scheduling disciplines.

4. By having a server with two components, we can extend results previously obtained for a single scheduler to a networking environment. Any scheduler that can provide

delay bounds to connections at a single switch can be used in conjunction with a rate-controller to form a rate-controlled server, which can then be used to provide end-to-end performance bounds in a networking environment.

5. Separation of rate-control and delay-control functions in the design of a server allows decoupling of bandwidth and delay bound allocations to different connections. Most existing solutions have the drawback of coupling the bandwidth/delay bound allocation. Providing a lower delay bound to a connection automatically allocates a higher bandwidth to the connection. Such solutions cannot efficiently support low-delay/low-bandwidth connections.

6. Unlike work-conserving disciplines, which require the reservation of more buffer space at the downstream switches traversed by a connection, rate-controlled disciplines also have the advantage of requiring an evenly distributed buffer space at each switch to prevent packet loss.

We showed that rate-controlled service disciplines provide a general framework under which most of the existing non-work-conserving disciplines, such as Jitter-EDD [86], Stop-and-Go [38] and Hierarchical Round Robin [44], can be naturally described. We discussed the tradeoffs of various rate-controllers and schedulers. One discipline in this class, called Rate-Controlled Static Priority (RCSP), is particularly suitable for providing performance guarantees in high speed networks. It achieves both flexibility in the allocation of bandwidths and delay bounds to different connections and simplicity of implementation.

In Chapter 3, we also presented new admission control algorithms for deterministic service that can achieve a high average link utilization even when the traffic is bursty. In the previous formulation of the Tenet Scheme, deterministic service required that the sum of the peak rates of all deterministic connections be less than the link speed. This condition results in a low average link utilization if the peak-to-average-rate ratio is high for real-time traffic. We showed that this condition is too restrictive, and that deterministic service can be provided even when the condition is not satisfied. We gave conditions to bound delays for FCFS and Static Priority schedulers. The new admission control algorithms result in a multifold increase in the number of admitted real-time channels when the traffic is bursty.

In Chapter 4, we studied the problem of providing end-to-end statistical guarantees in networks with rate-controlled service disciplines. We believe that there are two important aspects to the problem of providing guaranteed statistical services: developing appropriate probabilistic models to characterize traffic sources, and developing techniques to provide end-to-end statistical bounds in a network environment.

To model traffic, we argued that traffic sources can be thought of as showing interval-dependent behaviors, i.e. a traffic source has different characteristics in intervals of different lengths. We presented an interval-dependent stochastic traffic model, which extends the Tenet deterministic traffic model $(Xmin, Xave, I, Smax)$ within Kurose's framework [51]. The model stochastically bounds the number of bits sent over time intervals of different lengths and requires that these distributions be explicit functions of the interval length. The model differs from previous stochastic models in two important aspects:

1. instead of modeling the source using some stochastic process like MMPP [59], our model uses random variables to stochastically *bound* the number of packets produced in different intervals;

2. instead of using one model to capture the traffic characteristics during intervals of any length, our model uses a family of random variables with parameters that are explicit functions of the interval's length.

To provide end-to-end statistical bounds in a network environment, we first provided a local statistical bound in a single scheduler, and then extended the results to a networking environment by using rate-controlled service disciplines. In the single node case, we analyzed the multiplexing of sources served by a Static Priority scheduler considering both homogeneous and heterogeneous sources. By using delay-jitter control, we extended these results to a network to provide end-to-end per-connection statistical performance guarantees. As in the case of deterministic service, the results hold in general networking environments with arbitrary topology, both feedback and feed-forward networks, internetworks with variable but bounded link delays, and networks with rate-controlled servers that have different schedulers. Compared to the approach of using work-conserving disciplines and characterizing traffic patterns inside the network, the use of rate-controlled service disciplines allow *more* connections to be admitted in *more general* networking environments.

In Chapter 5, we presented an implementation of the proposed algorithms in an internetworking environment. We described the design and implementation of an internetwork layer protocol called Real-Time Internet Protocol or RTIP, which provides host-to-host guaranteed performance service in an internetworking environment. RTIP is the network layer data delivery protocol in the Tenet real-time protocol suite. We implemented RTIP in Ultrix on DECstation 5000 workstations and in HP/UX on HP9000/7000 workstations. The service disciplines used in the implementation are rate-controlled service disciplines. A measurement study has been conducted to evaluate the performance of the prototype implementation. The study showed that the throughput obtained by using RMTP/RTIP is comparable to that obtained by using raw IP; the rate control mechanism in RMTP/RTIP effectively enforces the traffic specification of communication clients; the scheduling mechanism in RTIP protects a real-time channel so that its performance is not affected by the presence of IP traffic or other real-time channels in the network.

6.2 Future Work

Although we have made important progress in understanding issues and tradeoffs in the design of service disciplines for integrated-services packet-switching networks, a number of issues still need to be explored further.

We have shown that having tighter bounding techniques for delay-bound calculations can improve the average link utilization. We gave the conditions for FCFS and Static Priority schedulers. It is yet to be established how tight these bounds are. Also, we know that an Earliest Due Date scheduler is more flexible than a Static Priority scheduler; future work should investigate the conditions for EDD, and quantify the resulting benefits in terms of the improvement to average link utilization by comparing EDD and SP.

We have argued that the Rate-Controlled Static Priority (RCSP) discipline can be implemented at very high speeds. This claim can be verified only by a real implementation. Although we have implemented the algorithm in software in an internetworking environment, a high speed implementation in an ATM environment, where the packet size is small and many more packets need to be switched for the same transmission speed, is more challenging. A VLSI implementation of RCSP at gigabits per second speed using current technology will be able to demonstrate the simplicity of RCSP.

In Chapter 4, we presented an interval-dependent stochastic traffic model to characterize source traffic. It is unclear how clients can specify the parameters used in the model. Collecting realistic traffic traces and understanding the relationships between traffic model and real data are important areas to be explored.

The experiments we did in Chapter 5 were using synthetic workloads in a local area network environment. More interesting lessons could be learned if experiments were performed using realistic workloads in a wide area network. In particular, we could better understand the delay and delay jitter characteristics of connections with or without real-time protocols.

Bibliography

[1] Fiber distributed data interface (FDDI) - token ring media access control (MAC), ANSI X3.139, 1987.

[2] ANSI FDDI station management standard, ANSI X3T9.5, revision 5.1, September 1989.

[3] Gopal Agrawal, Baio Chen, Wei Zhao, and Sadegh Davari. Guaranteeing synchronous message deadline with the timed token protocol. In *Proceedings of IEEE International Conference on Distributed Computing Systems*, June 1992.

[4] Gopal Agrawal, Biao Chen, and Wei Zhao. Local synchronous capacity allocation schemes for guaranteeing message deadlines with timed token protocol. In *Proceedings of IEEE INFOCOM'93*, San Francisco, CA, April 1993.

[5] H. Ahmadi and W. Denzel. Survey of modern high performance switching techniques. *IEEE Journal on Selected Areas in Communications*, 7(7):1091–1103, September 1989.

[6] Anindo Banerjea and Srinivasan Keshav. Queueing delays in rate controlled networks. In *Proceedings of IEEE INFOCOM'93*, pages 547–556, San Francisco, CA, April 1993.

[7] Anindo Banerjea and Bruce Mah. The real-time channel administration protocol. In *Proceedings of the Second International Workshop on Network and Operating System Support for Digital Audio and Video*, pages 160–170, Heidelberg, Germany, November 1991. Springer-Verlag.

[8] P.T. Brady. A techniques for investigating on-off patterns in speech. *Bell System Technical Journal*, 44:1–22, January 1965.

[9] P.T. Brady. A statistical analysis of on-off patterns in 16 conversations. *Bell System Technical Journal*, 47:73–91, January 1968.

[10] Randy Brown. Calendar queues: A fast O(1) priority queue implementation for the simulation event set problem. *Communications of the ACM*, 31(10):1220–1227, October 1988.

[11] CCITT proposed recommendation i.311, June 1991.

[12] H. Jonathan Chao. Architecture design for regulating and scheduling user's traffic in ATM networks. In *Proceedings of ACM SIGCOMM'92*, pages 77–87, Baltimore, Maryland, August 1992.

[13] Greg Chesson. XTP/PE design considerations. In *Proceedings of IFIP Workshop Protocols High-Speed Networks*, pages 27–33, Rüschlikon, Switzerland, May 1989.

[14] I. Chlamtac, A. Ganz, and M. G. Kienzle. An HIPPI interconnection system. *IEEE Transactions on Computers*, 42(2):138–150, February 1993.

[15] Israel Cidon, Jeff Derby, Inder Gopal, and Bharath Kadaba. A critique of atm from a data communication perspective. *Journal of High Speed Networking*, 1(2), March 1993.

[16] David Clark. The design philosophy of the DARPA internet protocols. In *Proceedings of ACM SIGCOMM'88*, pages 106–114, Stanford, CA, August 1988.

[17] David Clark, Scott Shenker, and Lixia Zhang. Supporting real-time applications in an integrated services packet network: Architecture and mechanism. In *Proceedings of ACM SIGCOMM'92*, pages 14–26, Baltimore, Maryland, August 1992.

[18] David D. Clark, Van Jacobson, John Romkey, and Howard Salwen. An analysis of TCP processing overhead. *IEEE Communications Magazine*, June 1989.

[19] Rene L. Cruz. A calculus for network delay, part I : Network elements in isolation. *IEEE Transaction of Information Theory*, 37(1):114–121, 1991.

[20] Rene L. Cruz. A calculus for network delay, part II : Network analysis. *IEEE Transaction of Information Theory*, 37(1):121–141, 1991.

[21] Rene L. Cruz. Service burstiness and dynamic burstiness measures: A framework. *Journal of High Speed Networks*, 1(2), 1992.

[22] Alan Demers, Srinivasan Keshav, and Scott Shenker. Analysis and simulation of a fair queueing algorithm. In *Journal of Internetworking Research and Experience*, pages 3–26, October 1990. Also in Proceedings of ACM SIGCOMM'89, pp 3-12.

[23] Domenico Ferrari. Real-time communication in packet-switching wide-area networks. Technical Report TR-89-022, International Computer Science Institute, Berkeley, California, May 1989.

[24] Domenico Ferrari. Client requirements for real-time communication services. *IEEE Communications Magazine*, 28(11):65–72, November 1990.

[25] Domenico Ferrari. Design and applications of a delay jitter control scheme for packet-switching internetworks. In *Proceedings of the Second International Workshop on Network and Operating System Support for Digital Audio and Video*, pages 72–83, Heidelberg, Germany, November 1991. Springer-Verlag. Also in *Computer Communications* 15(6):367-373, July-August 1992.

[26] Domenico Ferrari. Real-time communication in an internetwork. *Journal of High Speed Networks*, 1(1):79–103, 1992.

[27] Domenico Ferrari, Anindo Banerjea, and Hui Zhang. Network support for multimedia: a discussion of the Tenet approach. Technical Report TR-92-072, International Computer Science Institute, Berkeley, California, October 1992. Also to appear in *Computer Networks and ISDN Systems.*

[28] Domenico Ferrari, Amit Gupta, Mark Moran, and Bernd Wolfinger. A continuous media communication service and its implementation. In *Proceedings of GLOBECOM '92*, Orlando, Florida, December 1992.

[29] Domenico Ferrari, Joe Pasquale, and George Polyzos. Network issues for Sequoia 2000. In *Proceedings of COMPCOM 92*, pages 401–406, San Francisco, CA, February 1992.

[30] Domenico Ferrari and Dinesh Verma. A scheme for real-time channel establishment in wide-area networks. Technical Report TR-89-036, International Computer Science Institute, Berkeley, California, May 1989.

[31] Domenico Ferrari and Dinesh Verma. A scheme for real-time channel establishment in wide-area networks. *IEEE Journal on Selected Areas in Communications*, 8(3):368–379, April 1990.

[32] Sally Floyd. Issues in flexible resource management for datagram networks. In *Proceedings of the 3rd Workshop on Very High Speed Networks*, Maryland, March 1992.

[33] Sally Floyd and Van Jacobson. The synchronization of periodic routing messages. In *Proceedings of ACM SIGCOMM'93*, pages 33–44, San Francisco, CA, September 1993.

[34] Alexander G. Fraser. Designing a public data network. *IEEE Communications Magazine*, 30(10):31–35, October 1991.

[35] Alexander G. Fraser and Paul S. Henry. Transmission facilities for computer communications. *ACM Computer Communication Review*, 22(5):53–61, October 1992.

[36] Alexander G. Fraser, Chuck R. Kalmanek, A.E. Kaplan, William T. Marshall, and R.C. Restrick. Xunet2: A nationwide testbed in high-speed networking. In *Proceedings of INFOCOM'92*, Firenze, Italy, May 1992.

[37] S. Jamaloddin Golestani. Congestion-free transmission of real-time traffic in packet networks. In *Proceedings of IEEE INFOCOM'90*, pages 527–542, San Francisco, California, June 1990. IEEE Computer and Communication Societies.

[38] S. Jamaloddin Golestani. A stop-and-go queueing framework for congestion management. In *Proceedings of ACM SIGCOMM'90*, pages 8–18, Philadelphia Pennsylvania, September 1990.

[39] Riccardo Gusella and S. Zatti. The accuracy of the clock synchronization achieved by TEMPO in berkeley UNIX 4.3BSD. *IEEE Transactions on Software Engineering*, 15(7):847–853, July 1989.

[40] J. Hyman and A. Lazar. MARS: The Magnet II real-time scheduling algorithm. In *Proceedings of ACM SIGCOMM'91 Conference*, pages 285–293, Zurich, Switzerland, September 1991.

[41] J. Hyman, A. Lazar, and C. Pacifici. Real-time scheduling with quality of service constraints. *IEEE Journal on Selected Areas of Communications*, pages 1052–1063, September 1991.

[42] Van Jacobson. Congestion avoidance and control. In *Proceedings of ACM SIGCOMM'88*, pages 314–329, August 1988.

[43] Raj Jain. Congestion control in computer networks: Issues and trends. *IEEE Network Magazine*, pages 24–30, May 1990.

[44] Charles R. Kalmanek, Hemant Kanakia, and Srinivasan Keshav. Rate controlled servers for very high-speed networks. In *IEEE Global Telecommunications Conference*, pages 300.3.1 – 300.3.9, San Diego, California, December 1990.

[45] Chuck. R. Kalmannek and Robert C. Restrick. Xunet 2 queue module, October 1989. AT&T Bell Laboratories internal technical report.

[46] Mark J. karol, Michael G. Hluchyj, and Sam P. Mogan. Input versus output queueing on a space-division packet switch. *IEEE Transactions on Communications*, 35(12):1347–1356, December 1987.

[47] Randy H. Katz, Peter M. Chen, Ann L. Drapeau, Ed K. Lee, Ken Lutz, Ethan L. Miller, Srini Seshan, and David A. Patterson. RAID-II: Design and implementation of a large scale disk array controller. In *Symposium on Integrated Systems*, 1993.

[48] Srinivasan Keshav. *Congestion Control in Computer Networks.* PhD dissertation, University of California at Berkeley, September 1992.

[49] Leonard Kleinrock. *Queueing Systems.* John Wiley and Sons, 1975.

[50] Donald E. Knuth. *The Art of Computer Programming. Volume 3: Sorting and searching.* Addison-Wesley, 1975.

[51] Jim Kurose. On computing per-session performance bounds in high-speed multi-hop computer networks. In *ACM SigMetrics'92*, 1992.

[52] Jim Kurose. Open issues and challenges in providing quality of service guarantees in high-speed networks. *ACM Computer Communication Review*, 23(1):6–15, January 1993.

[53] Ed Lee. RAID-II: A scalable storage architecture for high-bandwidth network file service. Technical Report UCB/CSD 92/672, University of California at Berkeley, 1992.

[54] Samuel J. Leffler, William N. Joy, Robert S. Babry, and Michael J. Karels. Networking implementation notes: 4.3 BSD edition. In *Unix System Manager's Manual*. USENIX Association, April 1986.

[55] Samuel J. Leffler, Marshall Kirk Mckusick, Michael J. Karels, and John S. Quarterman. *The Design and Implementation of the 4.3 BSD UNIX Operating System*. Addison-Wesley Publishing Company, 1989.

[56] Jörg Liebeherr, September 1993. Personal communication.

[57] C.L. Liu and J.W. Layland. Scheduling algorithms for multiprogramming in a hard real-time environment. *Journal of ACM*, 20(1):46–61, January 1973.

[58] Steve Low. *Traffic Control in ATM Networks*. PhD dissertation, University of California at Berkeley, May 1992.

[59] B. Maglaris, D. Anastassiou, P. Sen, G. Karlsson, and J.D. Robbins. Performance models of statistical multiplexing in packet video communications. *IEEE Transaction on Communication*, 36(7):834–844, July 1988.

[60] David Mills. Network Time Protocol (version 2) - specification, and implementation, September 1989. RFC 1119.

[61] David Mills. Internet time synchronization: the Network Time Protocol. *IEEE Transactions on Communications*, 39(10):1482–1493, October 1991.

[62] David Mills. Network Time Protocol (version 3) - specification, implementation and analysis, March 1992. RFC 1305.

[63] Jeffrey C. Mogul and Steve Deering. Path MTU discovery, 1990. RFC1191.

[64] Mark Moran and Riccardo Gusella. System support for efficient dynamically-configurable multi-party interactive multimedia applications. In *Proceedings of Thrid International Workshop on Network and Operating System Support for Digital Audio and Video*, San Diego, CA, November 1992.

[65] Ioannis Nikolaidis and Ian F. Akyildiz. Source characterization and statistical multiplexing in atm networks. Technical Report GIT-CC-92/24, College of Computing, Georgia Institute of Technology, Atlanta, GA 30332-0280, 1992.

[66] Abhay Kumar J. Parekh. *A Generalized Processor Sharing Approach to Flow Control in Integrated Services Networks.* PhD dissertation, Massachusetts Institute of Technology, February 1992.

[67] Abhay Kumar J. Parekh and Robert G. Gallager. A generalized processor sharing approach to flow control - the single node case. In *Proceedings of the INFOCOM'92*, 1992.

[68] Abhay Kumar J. Parekh and Robert G. Gallager. A generalized processor sharing approach to flow control in integrated services networks: The multiple node case. In *Proceedings of the INFOCOM'93*, pages 521–530, San Francisco, CA, March 1993.

[69] Craig Partridge. Isochronous applications do not require jitter-controlled networks, September 1991. RFC 1157.

[70] Guru Parulkar. The next generation of internetworking. *ACM SIGCOMM Computer Communication Review*, January 1990.

[71] David Paterson, Garth Gibson, and Randy Katz. A case for redundant arrays of inexpensive disks (RAID). In *Proceedings of ACM SIGMOD'88*, Chicago, IL, June 1988.

[72] Jon M. Peha. *Scheduling and Dropping Algorithms to Support Integrated Services in Packet-switching Networks.* PhD dissertation, Stanford University, June 1991.

[73] Jon M. Peha and Fouad Tobagi. A cost-based scheduling algorithm to support integrated services. In *Proceedings of IEEE INFOCOM'91*, pages 741–753, Miami, Florida, July 1991.

[74] Jon Postel. User datagram protocol, August 1980. RFC 768.

[75] Jon Postel. Internet protocol, September 1981. RFC 791.

[76] Jon Postel. Transmission control protocol, September 1981. RFC 793.

[77] William H. Press, Brian P. Flannery, Saul A. Teukolsky, and William T. Vetterling. *Numerical Recipes in C: The Art of Scientific Computing.* Cambridge University Press, 1988.

[78] K.K. Ramakrishnan, D.M. Chiu, and Raj Jain. Congestion avoidance in computer networks with a connectionless network layer. In *Proceedings of ACM SIGCOMM'88*, pages 303–313, Stanford, CA, August 1988.

[79] Dennis Ritchie and Ken Thompson. The UNIX time-sharing system. *Communications of ACM*, 7(7):365–375, July 1974.

[80] John Stankovic and Krithi Ramamritham. *Hard Real-Time Systems*. IEEE Computer Society Press, 1988.

[81] Michael Stonebraker. An overview of the Sequoia 2000 project. In *Proceedings of COMPCOM 92*, San Francisco, CA, February 1992.

[82] P. Todorova and Dinesh Verma. Fast establishment of real-time channels. Technical Report TR-89-056, International Computer Science Institute, Berkeley, California, October 1989.

[83] Claudio Topolcic. Experimental internet stream protocol, version 2 (ST-II), October 1990. RFC 1190.

[84] Dinesh Verma. *Guaranteed Performance Communication in High Speed Networks*. PhD dissertation, University of California at Berkeley, November 1991.

[85] Dinesh Verma and Hui Zhang. Design documents for RMTP/RTIP, May 1991. Unpublished internal technical report.

[86] Dinesh Verma, Hui Zhang, and Domenico Ferrari. Guaranteeing delay jitter bounds in packet switching networks. In *Proceedings of Tricomm'91*, pages 35–46, Chapel Hill, North Carolina, April 1991.

[87] R. Wolff. *Stochastic Modeling and the Theory of Queues*. Prentice Hall, 1989.

[88] Bernd Wolfinger and Mark Moran. A continuous media data transport service and protocol for real-time communication in high speed networks. In *Proceedings of the Second International Workshop on Network and Operating System Support for Digital Audio and Video*, pages 171–182, Heidelberg, Germany, November 1991. Springer-Verlag.

[89] Opher Yaron and Moshe Sidi. Calculating performance bounds in communication networks. In *Proceedings of IEEE INFOCOM'93*, pages 539–546, San Francisco, CA, April 1993.

[90] Opher Yaron and Moshe Sidi. Performance and stability of communication networks via robust exponential bounds. *IEEE Transaction on Networking*, 1(3):372–385, June 1993.

[91] Hui Zhang and Domenico Ferrari. Rate-controlled static priority queueing. In *Proceedings of IEEE INFOCOM'93*, pages 227–236, San Francisco, California, April 1993.

[92] Hui Zhang and Tom Fisher. Preliminary measurement of RMTP/RTIP. In *Proceedings of the Third International Workshop on Network and Operating System Support for Digital Audio and Video*, San Diego, CA, November 1992. Springer-Verlag.

[93] Hui Zhang and Srinivasan Keshav. Comparison of rate-based service disciplines. In *Proceedings of ACM SIGCOMM'91*, pages 113–122, Zurich, Switzerland, September 1991.

[94] Hui Zhang and Ed Knightly. Providing end-to-end statistical performance guarantees with interval dependent stochastic models, October 1993. preprint.

[95] Hui Zhang, Dinesh Verma, and Domenico Ferrari. Design and implementation of the real-time internet protocol. In *Proceedings of the IEEE Workshop on the Architecture and Implementation of High Performance Communication Subsystems*, Tucson, Arizona, February 1992.

[96] Lixia Zhang. *A New Architecture for Packet Switched Network Protocols*. PhD dissertation, Massachusetts Institute of Technology, July 1989.

[97] Lixia Zhang. Virtual clock: A new traffic control algorithm for packet switching networks. In *Proceedings of ACM SIGCOMM'90*, pages 19–29, Philadelphia Pennsylvania, September 1990.

[98] Lixia Zhang, Steve Deering, Deborah Estrin, Scott Shenker, and Daniel Zappala. RSVP: A new resource reservation protocol. *IEEE Communications Magazine*, 31(9):8–18, September 1993.