Ph.D. Thesis

Architecture for Guaranteed Delay Service in High Speed Networks

by Vinod Peris
Advisor: Armand Makowski

CSHCN Ph.D. 97-1
(ISR Ph.D. 97-3)

Abstract

Title of Dissertation: Architecture for Guaranteed Delay Service in High Speed Networks

Vinod Peris, Doctor of Philosophy, 1997

Dissertation directed by: Professor Armand Makowski
Department of Electrical Engineering

The increasing importance of network connections coupled with the lack of abundant link capacity suggests that the day when service guarantees are required by individual connections is not far off. In this dissertation we describe a networking architecture that can efficiently provide end-to-end delay guarantees on a per-connection basis.

In order to provide any kind of service guarantee it is imperative for the source traffic to be accurately characterized at the ingress to the network. Furthermore, this characterization should be enforceable through the use of a traffic shaper (or similar device). We go one step further and assume an extensive use of traffic shapers at each of the network elements. Reshaping makes the traffic at each node more predictable and therefore simplifies the task of providing efficient delay guarantees to individual connections. The use of per-connection reshapers to regulate traffic at each hop in the network is referred to as a Rate Controlled Service (RCS) discipline in [52]. By exploiting some properties of traffic shapers we demonstrate how the per-hop reshaping does not increase the bound on the end-to-end delay experienced by a connection. In particular, we show that an appropriate choice of traffic shaper parameters enables the RCS discipline to provide better end-to-end delay guarantees than any other service discipline known today.

The RCS discipline can provide efficient end-to-end delay guarantees to a connection; however, by definition it is not work-conserving. This fact may increase the average delay that is observed by a connection even if there is no congestion in the network. We outline a mechanism by which an RCS discipline can be modified to be work-conserving without sacrificing the efficient end-to-end delay guarantees that can be provided to individual connections. Using the notion of service curves to bound the service process at each network element, we are able to provide an upper bound on the buffers required to ensure zero loss at the network element. Finally, we examine how the RCS discipline can be used in the context of the Guaranteed Services specification that is currently in the process of being standardized by the Internet Engineering Task Force.

ARCHITECTURE FOR GUARANTEED DELAY SERVICE IN HIGH SPEED NETWORKS

by

Vinod Peris

Dissertation submitted to the Faculty of the Graduate School of the
University of Maryland at College Park in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
1997

Dedication

To my parents
and
Sangeeta

Acknowledgements

First and foremost I would like to thank my advisor Dr. Armand Makowski for his guidance and support throughout my Ph.D. program. He provided me with the freedom to explore a wide range of subjects some of which have found a place in this dissertation. I greatly cherish the countless discussions, some related to networking and some unrelated ones that will no doubt leave a deep and lasting impression on me. His careful review of several versions of this manuscript have gone a long way in improving the quality of this dissertation.

The seeds from which this work sprouted were sown in the summer of 1994 when I was a Summer Intern in the Broadband Networking group at Watson. I would especially like to thank Dr. Roch Guérin and Dr. Leonidas Georgiadis at I.B.M. T.J. Watson Research Center for watering this seed and providing me with guidance as well as insights into the many practical issues related to networking. I am grateful to all my colleagues at Watson including, Ping Pan, Raju Rajan, Shai Herzog, Dimitris Pendarakis, Erol Basturk, Sanjay Kamat, Robert Engel, Dilip Kandlur, Kumar Sivarajan, Debanjan Saha and Robert Haas who have probably taught me more than I need to know about networking. I take this opportunity to thank Dr. Mark Squillante for the several indepth discussions on multi-processor scheduling and the wonderful summer I spent at I.B.M. T.J. Watson Research Center in 1993.

I am grateful to Dr. Prakash Narayan, Dr. Udaya Shankar, Dr. Mark Shayman and Dr. Leandros Tassiulas for serving on my dissertation committee. I would like to thank Dr. John Baras and Dr. Evaggelos Geraniotis for some stimulating discussions on ATM networks. I am pleased to acknowledge the financial support provided by the Institute for Systems Research which allowed me the luxury of pursuing a Ph.D. degree. This work was supported partially through NSF Grant NSFD CDR-88-03-012 and NASA Grant NAGW-2777. I would also like to acknowledge I.B.M Research for providing a stimulating environment that was most conducive for research.

This work has benefitted enormously from my interactions with fellow students in both Electrical Engineering and Computer Science. These include Lionel Banege, Nol Rananand, Young Kim, Sanjeev Khudanpur, Ramin Rezaiifar, John Bartusek, George Shuttic, Partho Mishra, Dhiraj Sanghi and Pravin Bhagwat.

Last but most certainly the most, I would like to thank my ever patient wife for putting up with me and my computer for days on end. Without her encouragement and support this would certainly not have been possible.

Table of Contents

List of Figures

Chapter 1

Introduction

The words “high-speed” in the title of this thesis are almost archaic now. About five years ago, it was used to qualify networks that had links with capacity in the Megabits/sec range. It is now fairly common for carriers to have OC-3 links (155 Mb/s) in their backbones and a few of them even have OC-12 links (622 Mb/s). The day when they will be replaced by OC-48 links, which carry 2.4 gigabits of traffic every second, is not far away. Currently the link speeds are increasing at a much higher rate than the switch speeds and there is no indication that this trend will change.

So if the link capacity is increasing at this phenomenal rate why is it that we, the end-users, seem to be unhappy with the networking service that we receive? In today’s Internet there are several reasons for this. If we start at the end-station, we right away notice one bottle-neck, the access to the Internet. For example, today, most users connect to their service provider through a measly 28.8 Kbps, or slower speed modem. This definitely does not help. However, it is not the local access that is necessarily the problem. For instance it is far from unusual for a user to wait a long time for the CNN web-page to be down-loaded to her[1] browser. And this could be at work, where a typical user is connected from her workstation to the LAN Switch through a dedicated 10Mbps Ethernet port. The LAN Switch is connected to a router sitting on a 100Mbps shared FDDI ring on which there is another router that connects to the Internet through a T1 (1.5 Mb/s link). Most likely the delay is caused by some congestion in the network. Another possibility is that the CNN web-server is bombarded with hits from several browsers from all over the world and is unable to keep up with the requests. However, given that our focus is on networking, we shall conveniently ignore this possibility with the firm belief that with the rapid advances being made in microprocessor technology, this processing limitation will be quickly overcome.

We can draw some comfort from the fact that this phenomenon is probably experienced at some time or another by almost everyone who has surfed the web. This may be acceptable as far as surfing

[1]the word *her* is used in a gender neutral sense

the web is concerned, but it is certainly not going to be very appealing to people who are using the Internet to conduct financial transactions, or remote surgery or video-conferencing or any other form of interactive communication. One might question whether the Internet is the appropriate network for some these applications. While there are certainly other Wide Area Networking technologies being deployed it is becoming increasingly common for TCP/IP to run on top of these networks. The popularity of Frame Relay and ATM among the major telecommunication carriers are a good example of alternative networking technologies but they are mainly used to provide IP connectivity. If the current trend is any indication of things to come the end-user will run a TCP/IP stack for many, many years.

Soon, it is going to be increasingly important, if not imperative, for the Internet to support some form of service differentiation. There are several people working on providing support for this service differentiation in the Internet. In fact, there are a few working groups in the Internet Engineering Task Force (IETF) that are addressing this problem right now and we briefly touch upon them later in this thesis.

In this thesis, we are primarily interested in building a framework that will allow the network to efficiently provide per-connection end-to-end Quality of Service (QoS) guarantees. There are many ways to provide QoS support in the network depending on the underlying networking infrastructure. At one extreme it is possible to have almost no service differentiation at all. Rather, assuming that there are several flows statistically multiplexed onto the link, we can engineer the network so that the expected buffer occupancies are low. While this does not provide service differentiation, it does provide all the flows with an acceptable level of service. Note that estimating the queue size at a network element requires a fairly accurate statistical model of traffic at each of the nodes in the network, which in itself is a fairly difficult task. On the other hand we can implement complex scheduling policies that by design guarantee a certain QoS to a given set of flows. There are also a large number cases in between, like a simple priority scheduler that does provide some level of service differentiation among the different priority levels, but does not differentiate between flows that have the same priority. Whether one assumes a simple (FIFO) scheduling policy with suitable over-engineering of the network or a complex scheduling capability in the network elements, there are pros and cons that have to be considered. With the former approach, the existing networks can be used with little or no changes at all in the hardware. The caveat is that the QoS guarantees cannot be made on an individual flow basis. For a single flow to have better quality of service, it will be necessary to provide this service to all the flows that share a common path with this flow. In other words it has to be ensured that there are no bottlenecks along the way. On the other hand, while the latter approach does have the capability to provide QoS on a per-flow basis, it suffers from the problem of increasing the complexity of the router/switch. A few years ago, this complexity may have been hard to realize in hardware. However, currently there are several chips in the marketplace that can perform a fairly sophisticated amount of scheduling [17, 4]. Although this technology is not in widespread use we are of the firm belief that in the near future most, if not all, networking equipment will be capable of service differentiation. In the subsequent chapters of this thesis we examine how we can efficiently support end-to-end delay and throughput guarantees in an internet, assuming that there is some degree of scheduling support available in the networking equipment. In the remainder of this chapter we provide a brief overview of some of the aspects related to the provision of end-to-end delay guarantees, with the hope that it will motivate and position the work done as part of this thesis.

1.1 Traffic Regulation

In order to provide per-connection service guarantees, it is important for the user-specified traffic characteristics to be enforceable at the network ingress. This is required to prevent misbehaving flows from adversely affecting the service guarantees that have been provided to the conformant flows. Apart from policing traffic to ensure that it complies with the user-specifications, there is sometimes a need to modify the traffic stream without dropping any packets so that the resultant stream is conformant with a pre-specified traffic characterization. We use the term *traffic regulator* to describe devices that perform this general function. Traffic regulators are used to shape traffic at the network elements. In general a traffic regulator assigns an eligibility time to an incoming packet and queues it until it becomes eligible. Different regulators are obtained based on the way the eligibility times are computed. One of the first traffic regulators to be implemented was the Leaky Bucket [47]. The Leaky Bucket, also called (σ, ρ)-regulator [13], ensures that the output stream satisfies a certain pre-specified (σ, ρ) traffic descriptor [13]. Both the ATM and the Internet standard bodies specify traffic descriptors of this form. Another type of regulator that was used in the Tenet work at Berkeley is the $(X_{min}, X_{ave}, I, S_{max})$ regulator which ensures that the output of the regulator satisfies the $(X_{min}, X_{ave}, I, S_{max})$ characterization [6].

There is another type of regulator called the Delay-Jitter type of regulator [52] which ensures that the output traffic of the regulator is identical to the traffic pattern at the entry into the network. This type of regulator requires some information to be carried in the packet so that the traffic pattern at the network ingress can be recreated at network elements that are deep inside the network. This is particularly inefficient for ATM networks where the small ATM cell size results in a relatively large control overhead.

It is not surprising that the Leaky Bucket (or Token Bucket as it is sometimes called) is the regulator most used in practice. One of the reasons for its popularity is that it requires only a single state variable to be maintained for each connection. The eligibility time computation is very simple and only depends on the state variable and the arrival time of the packet. In fact both the ATM and the Internet standardization bodies require the use of a Leaky Bucket to police and/or reshape traffic on a per-connection basis. In this thesis we look at a generalized form of Leaky Bucket regulator and examine some fundamental properties of this device. *In particular we look at the relationship between reshaping and scheduling, and how a traffic shaper can be used at all network elements to improve the efficiency of scheduling mechanisms.*

1.2 Service Disciplines

When packets from a flow arrive at the network element, a service discipline is required to arbitrate between the different packets waiting to be transmitted on the link. One of the goals of scheduling is to provide predictable delays (or more precisely, delay bounds) for packets of different flows. Some desirable aspects of service disciplines are [53]:

1. *Schedulable Region*: Based on the scheduling policy and the traffic characterization used, it should be possible to determine the number of flows that can be carried by a link without violating any of their delay requirements.

2. *Efficiency*: The schedulable region should be large enough so that several calls can be carried by the link. Efficiency can be defined on a relative basis. Basically, a link scheduler is considered to be at least as efficient as another if it can carry all the traffic of the other scheduler without sacrificing any of the delay requirements.

3. *Protection*: The service discipline should afford some level of protection to conformant flows, even in the presence of flows that do not conform to their specified characteristics.

4. *Implementation Complexity*: The scheduling policy must be relatively simple to implement. With the ever increasing link speeds (OC-192) the amount of time available for the scheduling function is quite limited.

5. *Signalling*: In order to select the appropriate level of scheduling at each of the nodes there has to be some signalling protocol that operates throughout the network. Upgrading the signalling infrastructure for the entire network is a complex task which is sometimes even more difficult than upgrading the network hardware itself. A scheduling policy that requires minimal signalling support is definitely desirable.

Several service disciplines with many of the above mentioned properties have been proposed in the literature. In particular, in recent years there has been a proliferation of scheduling policies aimed at providing per-connection performance guarantees [18, 20, 24, 25, 37, 56, 54]. A comprehensive review can be found in [53]. The performance of various service disciplines has also been extensively studied in the past in the context of cpu-scheduling and hard real-time systems [33, 45]. In general, it is not easy to extend results that provide delay guarantees at a single node to the multiple node case, since the traffic characterization at the output of a node can be quite different from the traffic characteristics at its input. It is possible, however, to obtain a characterization of the traffic at the output of the scheduler [13, 32, 51]. If the delay at a single node can be computed then traffic characteristics at the output of the scheduler can be used to compute the delay at the downstream node and this can be successively applied at all nodes along the path of the connection. One drawback to this approach is that typically the bounds on the traffic characteristics grow with the number of hops on the path, resulting in a rather conservative estimate of the end-to-end delay.

Service disciplines can broadly be classified into two categories:

1. Work conserving service disciplines
2. Non work-conserving service disciplines

A system is said to be work-conserving if the link is never idle when there are packets waiting to be transmitted. If, even for a brief period of time, it is possible for packets to be queued in the system while the link is idle, such a system is referred to as non work-conserving. Sometimes the nature (or simplicity) of the scheduling algorithm causes the lack of work-conservation. For example both the Stop-and-Go [24] as well as the Hierarchical Round Robin [29] service disciplines are non work-conserving. They are both frame-based, in the sense that the time slots on the link are grouped into frames. Since there are specific rules regarding the frames in which packets can be placed it is possible for some frames to have idle slots even though there are packets waiting to be transmitted. The main drawback of this type of scheduling is that the frame size determines both the granularity of bandwidth allocation as well as the minimum delay bound that can be achieved. However one advantage of the frame-based formulation is that there is inherently a control on the amount of jitter that can be introduced by the network.

The Rate Based Scheduling policies like the Virtual Clock [56], or the many variants of Generalized Processor Sharing (GPS) [37] are work-conserving service disciplines. They operate by assigning some form of a rate guarantee to each flow. The rates are used to determine a priority ordering on the packets that are waiting to be transmitted. The basic idea is that packets are inserted into a sorted priority queue based on the time at which they arrive and the amount of bandwidth that they have reserved. The scheduling policy selects the packet with the highest priority for transmission on the link. Depending on the way in which the relative priority is computed, several different service disciplines arise. Some examples are Packetized Generalized Processor Sharing (PGPS) [37], Self-Clocked Fair Queueing (SCFQ) [25], Start-time Fair Queueing (SFQ) [27], Worst-case Fair Weighted Fair Queueing (WF^{2}Q) [8], etc. One of the main complexities of these types of service disciplines is that they require insertion into a priority queue, which in the worst case requires $O(\log N)$

operations, where N denotes the number of packets in the queue [31]. As far as end-to-end delay bounds are concerned, the PGPS service discipline is the most efficient service discipline known today. In [37], Parekh obtained the end-to-end delay bounds for the PGPS discipline and these serve as a benchmark for comparison with other scheduling policies.

It may be advantageous to combine some aspects of both the frame-based and the rate-based scheduling policies. The Rate Controlled Service (RCS) Discipline, first proposed by Zhang as a Rate Controlled Static Priority Discipline [52], makes such an attempt. This approach combines the non work-conserving nature of some of the frame-based service disciplines with the dynamic priority structure of the rate-based service disciplines. The idea is to have a regulator that reshapes the traffic at each node in order to ensure that the traffic offered to the scheduler arbitrating local packet transmissions conforms to specific characteristics. The regulators are typically used to enforce the same traffic parameter control as the one performed at the network access point, which is based on the parameters negotiated during connection establishment. Reshaping makes the traffic at each node more predictable and therefore, simplifies the task of guaranteeing performance to individual connections. Since it is possible that packets are held in the system until they are eligible for transmission, this service discipline is intrinsically non work-conserving in nature. The main advantages of an RCS discipline, especially when compared to GPS, are flexibility, lower buffer requirements at intermediate nodes, and typically simpler implementation [53]. However, as pointed out in [55] one of the main drawbacks of the "naive" RCS discipline is its inability to provide as good end-to-end delay bounds as the GPS service discipline. That is why it has often been argued that despite its potentially greater complexity, a GPS-based service discipline like PGPS, should be the solution of choice for providing performance guarantees to individual connections (see for example [12]).

Our work focuses on the provision of guaranteed delay service in a network. While the PGPS service disciplines and its variants can be used to provide delay guarantees, they are intrinsically coupled to the rate that is reserved. For example, if a flow requires a small end-to-end delay guarantee, then it must reserve a relatively large rate for itself. This coupling between the rate and the delay can lead to inefficiencies, particularly when dealing with low bit-rate flows. Also, the work-conserving nature of the GPS-based service disciplines has the potential to introduce large amounts of jitter into the stream. Moreover, the PGPS service discipline as proposed by Parekh [37] is fairly complex to implement since it involves an online simulation of the corresponding fluid-flow model to determine the order in which the packets are to be served. This task can be quite difficult to accomplish in hardware. Many of the variants of PGPS like SCFQ and SFQ simplify the implementation of the service discipline, albeit at the cost of slightly looser delay bounds. *In this thesis we focus on the RCS discipline and show how with the right choice of traffic regulators, it can provide better end-to-end delay guarantees than any other service discipline known today.*

1.3 End-to-end Service

It is not enough to identify a service discipline that provides good end-to-end delay guarantees. For it to be practical, it is imperative to specify a connection setup protocol that picks the different parameters that are required for efficient operation at each network element. In general, a connection may traverse network elements that are not manufactured by the same vendor. Thus several vendors must agree on a "standard" protocol to setup and tear down connections. For several people to agree on something is hard; having several companies agree on a standard is harder still. Thus there are very few standards that are really implemented today. Rather than invent a new setup protocol and try to convince the rest of the world to implement it, we tried to work within the existing framework that is being defined by the Integrated Services (IntServ) working group of the IETF [42].

The Internet is rapidly getting overloaded and the need for service differentiation becomes more and more important with every passing day. The IETF has adopted a two-tier model to provide service differentiation through the setup of QoS connections. One aspect is the reservation establishment and tear-down protocol, and the other is the selection of QoS parameters. These two parts are handled by two separate working groups in the IETF, namely the RSVP (Resource reSerVation Protocol) and the IntServ working groups.

RSVP is a receiver oriented protocol in the sense that the reservation is made by the receiver [9]. One of the primary goals of RSVP is to efficiently support IP multicast wherein a sender typically does not know the identity of the receivers of a multicast session. Therefore, it is the receivers who make the reservation by sending a message that retraces the path back to the sender. RSVP only specifies the signalling and setup of the connection. The actual QoS parameters depend on the type of service requested. The IntServ working group specifies the different service types as well as the parameters that need to be specified. Currently, there are two possible service types – Controlled Load and Guaranteed Service. Controlled Load simply provides a qualitative service guarantee as opposed to the more quantitative delay guarantees provided by the Guaranteed Service Specification. Since we are dealing with absolute delay guarantees in this thesis, the Guaranteed Service Specification is the one which is more applicable to us. *We investigate how the Rate Controlled Service Discipline can be efficiently used to support the Guaranteed Service specification. In addition, we motivate the specification of a new service called the Committed Rate Service that will improve the efficiency of the network.*

1.4 Thesis Outline

In this thesis we propose a framework and mechanism for providing per-connection guaranteed delay service in an internet. It is assumed that the user specifies its traffic characteristics at the time of connection setup and requests a certain delay guarantee. The network may choose to reject the connection if it deems that there are insufficient resources to provide such a guarantee. The delay guarantee only holds as long as the user specified traffic characterization is valid.

We start with a definition of the network model and then describe the traffic characterization that we use throughout this thesis. Since we are dealing with absolute delay guarantees, we use a deterministic traffic characterization called a "traffic envelope" that can be easily enforced on a per-connection basis. We make a brief review of the state of the art in scheduling policies and comment on some of their features and drawbacks. We then demonstrate by way of examples how the burstiness of a connection can increase as a result of scheduling. This prompts us to reshape traffic at each hop in order to smooth out some of the bursts that have been introduced by the upstream nodes.

Traffic can be shaped on a per-connection basis using traffic shapers. In Chapter 3, we discuss several properties of traffic shapers that are key to some of the service disciplines that are examined later in this thesis. We consider the extreme case of reshaping traffic at every node in the network and compute a per-connection bound on the end-to-end delay.

In Chapter 4, we describe the Rate Controlled Service Discipline (RCS) which consists of a traffic regulator (also called traffic shaper) and a scheduler. We look at each of these components in detail and derive some general guidelines on the choice of shaper parameters in the context of providing tight end-to-end delay guarantees. In addition, we look at the tradeoff between the scheduler delay guarantees and the shaper parameters in some specific cases.

In Chapter 5 we compare the end-to-end delay bounds for the RCS discipline with the best known delay bounds for the GPS discipline. There we demonstrate how a naive use of the RCS discipline, where the traffic shapers have the same envelope as the input traffic, performs much worse than the GPS service discipline. However, with appropriate traffic shaping, we show that the RCS discipline can outperform the GPS discipline.

RCS disciplines, by definition are not "work-conserving", i.e., packets may be queued even if the link is idle. This can increase the average delay that is experienced by a flow, but has the advantage of making the traffic more predictable, thus reducing the buffers that are needed downstream to ensure zero packet loss. In Chapter 6 we describe some variations on RCS disciplines that are work-conserving, yet provide the same end-to-end delay guarantees as RCS disciplines. We discuss some of the buffer implications of these schemes as well.

In Chapter 7 we consider the "big picture" and examine how the service disciplines that we propose can be used in the context of the Internet. We describe how the RCS discipline can be used without requiring any changes to the recently proposed Guaranteed Service Specification of the IETF [42]. More specifically, we describe how the traffic shaper parameters for the regulators, and the delay guarantees at each hop can be determined from the reservation made by the receiver. In addition, we go on to describe a Committed Rate Service that can be synergetically supported by the RCS discipline with the bandwidth left over from the Guaranteed Service flows. We conclude with a summary of our contributions, as well as a preview of some future work in this area.

Some significant assumptions as well as notational conventions are displayed in boxes in the relevant sections and can be found on pages 13, 30, 37 and 49.

Chapter 2

Network Model

In this chapter we first define the network model considered in this thesis, along with the associated service model that the end-user needs to specify its traffic characteristics. We briefly review some of the existing scheduling policies that have been proposed in the literature to provide a guaranteed delay service. We demonstrate the need for reshaping by examining the burstiness introduced into a traffic stream as a result of output contention at the link. While reshaping clearly has its benefits in smoothing traffic at each hop, thus preventing an accumulation of bursts, it potentially adds to the delay that may be observed by any packet. In the next chapter, we show how this additional delay introduced by the reshapers does not contribute to the worst case end-to-end delay that a packet may experience.

2.1 Network Model

The network is modeled as a directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ is a non-empty set of nodes and $\mathcal{E}$ is a set of directed edges. Each node in the graph represents a switching element and the directed edges represent transmission links, with the transmission taking place in the direction of the edge. A duplex link is modeled as a pair of oppositely directed edges. Each switch can have several input and output links, and packets are routed from an input to an output link, based on information that is available in the packet header. In general, packets can be queued in the switch, either at the input or at the output, or both. It is clear that if packets are only queued at the input there is an inherent loss of throughput due to the blocking caused by "Head of the Line" packets [28]. While output queueing does not suffer from this problem, it requires the switching fabric to operate at a much higher speed than that of the input links. Most switch vendors today implement some form of output queueing, with the switch operating at a higher speed than the input links. In this thesis we assume that each switch is output queueing, with no internal blocking that there is no variable delay taking place inside the switch. Thus, each output link at the switch

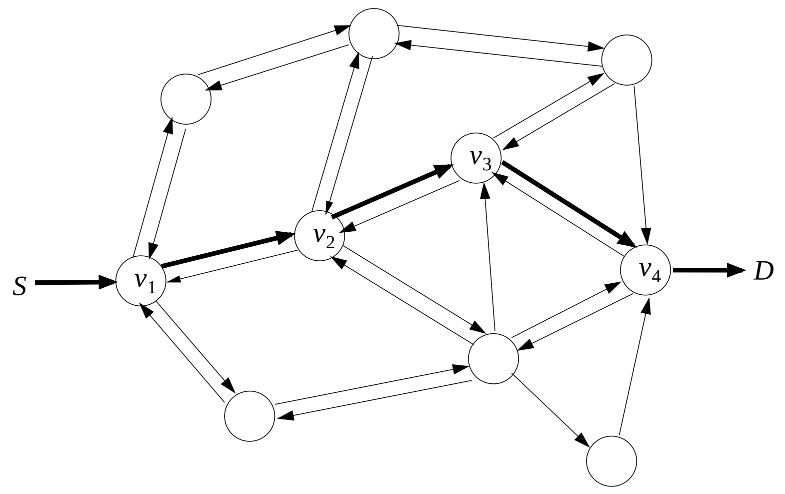

Figure 2.1: Graph representing a network

can be modeled as an independent multiplexer with a scheduling policy that determines the order of packet transmissions on the link. In the context of high-speed networks it is usually assumed that once a packet is selected for transmission it is completely transmitted. In other words once a packet has begun transmission it cannot be preempted by another packet. A scheduling policy with this property is referred to as a *non-preemptive* scheduling policy and we are mainly interested in such policies.

A path from node S to node D in the graph shown in Figure 2.1 represents a sequence of distinct nodes $S, v_1, v_2, \ldots, v_M, D$, such that (S, v_1), $(v_1, v_2), \ldots,$ $(v_{M-1}, v_M), (v_M, D)$ are directed edges in $\mathcal{E}$. In this thesis, we typically focus on the nodes that are on the path of a single connection, and without loss of generality, assume that they are numbered from 1 to M, with 0 and $M+1$ denoting the source and destination respectively. At connection set-up, a path is selected from source to destination and the connection traffic traverses this path. The process of selecting a suitable path is referred to as routing, and this is a separate problem not addressed here.

2.2 Input Traffic Model

Traditionally, traffic has been characterized as a stochastic process, some examples are the Poisson model for telephony, the ON-OFF fluid model for voice, Markov modulated processes for video, etc. These models may be appropriate if one is interested in the behavior of several connections that are statistically multiplexed together. However, they may be quite inaccurate at characterizing traffic from a single source. In addition some of these models are too complex and do not lend themselves to

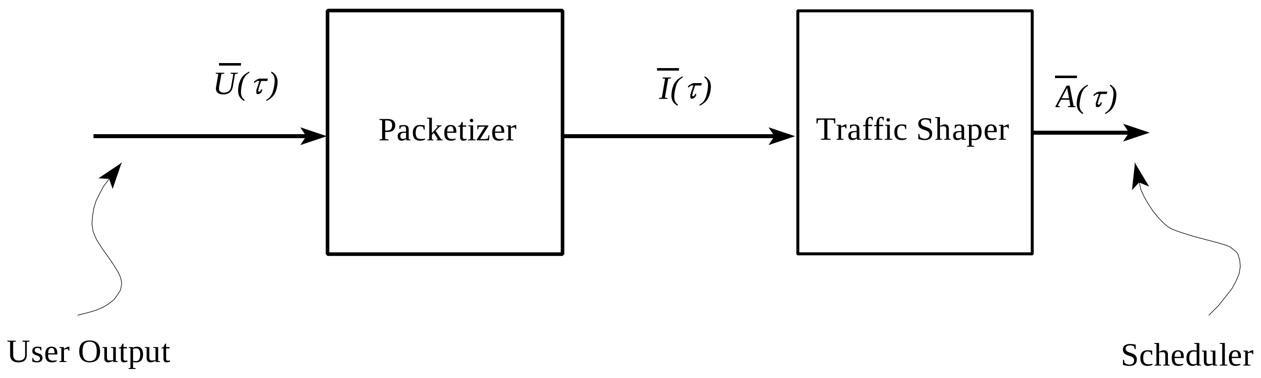

Figure 2.2: Connection Traffic Flow

analysis particularly for multi-hop connections. Recently, several simpler deterministic models have been proposed that specify an envelope or bound on the traffic of a source as opposed to accurately characterizing it. The (σ, ρ) characterization in [13] has been very popular in the literature as well as in practice, with major networking standards like those for the Asynchronous Transfer Mode (ATM) and the Internet Protocols (IP) specifying traffic descriptors based on this characterization. One of the main advantages of the (σ, ρ) characterization is the fact that it allows traffic from individual connections to be regulated to ensure conformance with a given traffic descriptor. The traffic model that we use is a generalization of the (σ, ρ) characterization, and is described next.

We take the position that bit generation is a continuous process and traffic arriving on the link is modeled as a fluid, with the understanding that a unit volume of fluid corresponds to a single bit in the real network. Let $U(t)$ denote the volume of traffic that arrives at the network ingress in the interval $[0, t]$. We assume that the mapping $t \to U(t)$ is non-decreasing and right continuous on IR_+, and that there exists a mapping $\overline{U} : \mathrm{IR}_+ \to \mathrm{IR}_+$ such that

$$U(t+\tau) - U(t) \leq \overline{U}(\tau), \qquad t \geq 0, \tau \geq 0.$$

The mapping $\overline{U}$ is called an *envelope* of U. This idea of using an envelope to bound the burstiness of a deterministic traffic stream was first introduced by Cruz [13]. The envelope function is clearly not unique. Chang [10, p. 915], showed that given an envelope $\overline{U}$ for a traffic stream U, another envelope $\overline{U}'$ for U can be generated, which is both *increasing* and *sub-additive* ($\overline{U}(\tau_1) + \overline{U}(\tau_2) \geq \overline{U}(\tau_1 + \tau_2)$, $\tau_1, \tau_2 \geq 0$). A partial ordering on the envelope functions can be defined as follows: Given two envelope functions $\overline{U}_1$ and $\overline{U}_2$, we write $\overline{U}_1 \preceq \overline{U}_2$ (or $\overline{U}_2 \succeq \overline{U}_1$) whenever $\overline{U}_1(\tau) \leq \overline{U}_2(\tau)$ for all $\tau \geq 0$. In terms of this ordering, Chang [10] developed the notion of a minimum envelope

$\overline{U}^*$ for a traffic stream U, namely

$$\overline{U}^*(\tau) = \sup_{t \geq 0} \{U(t+\tau) - U(\tau)\}, \quad \tau \geq 0.$$

It can be easily verified that the minimum envelope $\overline{U}^*$ is increasing and sub-additive. In general, the minimal envelope function for a given traffic stream is not easy to compute, and therefore we do not assume that its minimal envelope is known. However, without loss of generality, we assume that the envelope, $\overline{U}$ is both sub-additive and increasing. Henceforth, in this thesis the word *envelope*, by itself, simply denotes a mapping from $\mathbb{R}_+ \to \mathbb{R}_+$ that is both sub-additive and increasing.

The traffic arriving on the link is viewed as a stream of bits which we model as a continuous fluid flow. However in most if not all real networks, a complete packet is received before it is processed by the network element[1]. We assume that this conversion of the bit stream into a packet stream is performed by an abstract element called a *Packetizer* shown in Figure 2.2. The Packetizer accepts a stream of bits as input, and outputs a complete packet when the last bit of the packet is received. The notion of a packetizer is introduced for the sole purpose of modeling the fact that the network device operates on complete packets, and not on a bit-by-bit basis.

In our fluid model, the input to the Packetizer is a continuous stream of fluid. On receiving the complete packet, the packetizer outputs a volume of fluid equal to the length of the packet. Thus the output process from the packetizer makes discrete jumps at the instants in time when the complete packet is received. Let $I(t)$ denote the volume of traffic that is output by the packetizer in the interval $[0,t]$. If L denotes the maximum length of a packet, it is clear from the operation of the packetizer that

$$U(t) - L < I(t) \leq U(t), \quad t \geq 0.$$

Therefore,

$$\begin{aligned} I(t+\tau) - I(t) &\leq U(t+\tau) - U(t) + L \\ &\leq \overline{U}(\tau) + L \quad =: \quad \overline{I}(\tau), \quad t, \tau \geq 0. \end{aligned}$$

Because $\overline{U}$ is assumed to be sub-additive and increasing, it follows that $\overline{I}$ is also sub-additive and increasing.

Throughout this thesis we use $\overline{I}$ to denote the traffic envelope characterizing the source traffic at the network ingress (at the output of the first packetizer). Also, unless explicitly stated otherwise, we assume that the maximum packet size for all connections is the same and is denoted by L. Consequently, we must have $\overline{I}(0) \geq L$.

[1]We do not assume cut-through routing, since it is practical mainly for large packet sizes and relatively low-speed links

2.3 Service Model

The basic service model to invoke a Guaranteed Delay service is roughly the same in both the ATM and the Internet environments: First, the end-user provides a characterization of its traffic by means of its traffic envelope, and its end-to-end delay requirements. The network accepts the user's call based on the availability of resources along the path of the connection, and on the required end-to-end delay guarantees. Once the connection has been accepted, the network guarantees that the specified end-to-end delay bounds will be met provided the user's traffic does not violate its specified characterization.

If the user traffic does not conform to the characterization specified at connection setup time, then there are several possible options, many of which are currently being debated in the Integrated Services working group of the IETF [1]. Some of them include:

- Police the traffic, i.e., check for compliance, and drop all non-conforming traffic at the edge of the network.
- Police the traffic at each switch inside the network and drop all non-conforming traffic. This is fairly drastic as traffic which was conformant at the edge of the network can become non-conformant as a result of scheduling at the switch output. However, this approach has been suggested for cases where a switching element in the network, does not trust another switching element at the edge of the network to have appropriately regulated the traffic.
- Police traffic at the edge of the network, and mark all non-conforming traffic as Best Effort, i.e., a lower priority service. This, however, can result in some packets being transmitted out of order.
- Reshape non-conforming traffic at the edge of the network. While this scheme increases the delay for possible conforming traffic that arrives later, it prevents packets from arriving out of sequence.
- Reshape non-conforming traffic at each switch. In this case it is important to account for the additional delay that is introduced by reshaping.

It is not clear as to which one of these policies is better or worse. In some cases the choice of a policy may well be application dependent. For example, an audio or video playback application may have little use for packets that are late, and so it may be best to police and drop the excess traffic. However, for a TCP connection it is definitely a bad idea to drop packets since it will cause a retransmission of several packets and a reduction in the TCP window-size, resulting in an overall loss of throughput. Thus, connections using TCP-like protocols may prefer to have their traffic reshaped. In this thesis, we do not dwell further on the issue of policing, and make the assumption that the source traffic obeys the specified characterization at the edge of the network, and that once the connection is accepted by the network, its delay guarantees have to be met, without any packet loss.

2.4 Scheduling Policies

The nature of the scheduling policy employed at the output links, greatly impacts the ability of the network to provide efficient end-to-end delay bounds. In addition, the traffic characteristics of a connection, may be drastically altered depending on the scheduling policy used. First, we quickly describe a few well known scheduling policies, and comment on some of their features or drawbacks in the context of providing per-connection end-to-end delay guarantees.

2.4.1 First In First Out

The First In First Out (FIFO) policy, is one of the most basic scheduling policies, and requires that only a single queue be maintained at the output link. Each arriving packet is enqueued at the tail of the queue, and the scheduler picks the packet from the head of the queue and transmits it on the output link. Under the FIFO policy it is impossible to provide different QoS guarantees to each connection, since the scheduler does not distinguish between the packets of different connections.

2.4.2 Fixed Priority Scheduler

This policy offers some ability to provide QoS guarantees to different classes of traffic, each being associated with a given *static* priority. The link-multiplexer maintains a separate queue for each priority, and serves the packets in each priority according to the FIFO policy described earlier. However, packets in the lower priority queues, are served, only when all the higher priority queues are empty. Different variants of the policy exist, depending on whether or not the transmission of a lower priority packet can be interrupted by the arrival of a higher priority packet.

2.4.3 Earliest Deadline First

The Earliest Deadline First (EDF) policy computes a deadline for each packet that is given by the sum of its arrival time and the delay guarantee associated with its connection. The scheduler always selects for transmission the packet with the smallest deadline; hence the name. The EDF scheduler is a particular *dynamic priority* scheduler where the priority of the packet increases with the amount of time spent in the system. Thus, if a packet with a large delay guarantee is queued for a long time it can depart earlier than a packet with a small delay guarantee that has just arrived. This ensures that packets with loose delay requirements, obtain a better service than they would in a fixed priority scheduler, without sacrificing the delay guarantees provided to the packets with tighter delay requirements. For packet networks a non-preemptive type of service is typically assumed and the resulting policy is abbreviated as NPEDF. It is known that for any packet arrival process, where a deadline is associated with each packet, the EDF policy is optimal in terms of minimizing the maximum lateness of packets [19]. Here, lateness is defined as the difference between the deadline of a packet and the time that it is actually transmitted on the link.

2.4.4 Stop-and-Go

The Stop-and-Go service discipline [24] is based on the notion that traffic on the link can be divided into fixed length frames. The size of each frame, denoted by T, is fixed for the entire network. A key property of the Stop-and-Go service discipline is that for any flow, packets that are in one frame at the source remain in a single frame all along the path. This is accomplished by ensuring that at each network element a complete frame is received before any of the packets in that frame are sent out, as shown in Figure 2.3. The mapping between the frames received on the input links and those sent out on the output links is fixed. If the per-frame traffic on the input links is known, one can easily compute the traffic on the output links as well.

Each flow is characterized in terms of the amount of traffic it generates in a frame of size T. Consider an output link at some network element: If the per-frame sum of traffic from all connections passing through this output link is less than the frame size, then it is possible to satisfy the key property of the Stop-and-Go discipline outlined above. Thus the schedulability check at any particular link in the network is a simple sum of all the traffic characterizations of all the flows traversing that link.

The delay encountered at each network element can be bounded above by the length of a frame, as well as the offset between the incoming and the outgoing frames. Even though for a single network element all the frames do not need to be synchronized, it is important that there is no drift between the frame clocks on incoming and outgoing frames on different links.

Additionally, the framing structure introduces a coupling between the delay bound at a switch and the frame size T. Since the delay is directly proportional to the frame size, a small value of T is desirable. However, for a maximum packet size of L, the minimum bandwidth that needs to be reserved is L/T, which may be larger than what the connection really desires. In other words it is not possible for a connection to obtain tight delay guarantees and reserve a low bandwidth at the same time.

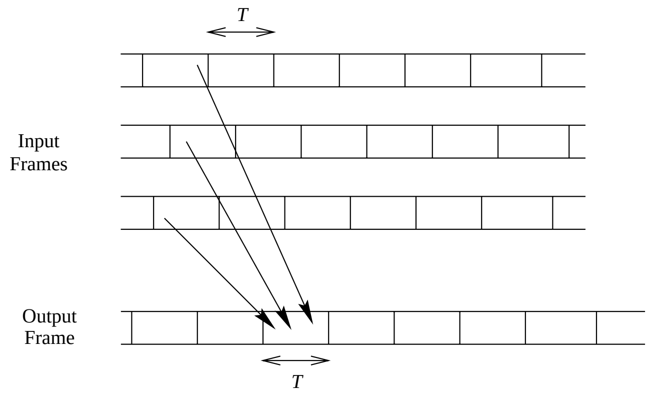

Figure 2.3: Operation of the Stop and Go scheduling policy

Another reason for setting a large frame size would be to allow a reasonable utilization of the link. Note that in the Stop-and-Go service discipline it is necessary for a packet to be confined to a single frame, i.e., packets cannot straddle frame boundaries. Since fractional packets cannot be sent in a frame, a significant amount of link bandwidth may be wasted in every frame. A large frame size will amortize this waste over a larger period in time, thereby improving the overall link utilization.

2.4.5 Fluid Fair Queueing

The Generalized Processor Sharing (GPS) service discipline and its many variants have received a lot of attention recently both in the research community as well as with the Switch/Router vendors. The reason for its popularity is the fact that tight bounds can be obtained for the end-to-end delay for a connection. In addition, GPS provides a certain amount of isolation between the different connections that are multiplexed on a link, thereby preventing any single connection from adversely affecting the performance of the other connections.

Assume there a total of N flows multiplexed onto a single link, with the flows being numbered from 1 to N. The basic idea behind GPS is to associate a weight ϕ_i with flow i, $i = 1, 2, \ldots, N$, and to share the link resources among active flows in proportion to their weights. If r denotes the link speed, then flow i is guaranteed to receive a clearing rate of at least $\frac{\phi_i}{\sum_{j=1}^{N} \phi_j} r$, $i = 1, 2, \ldots, N$. However, at any point in time, there may not be packets from all the flows, waiting to be transmitted on the link. Thus, there can be some unutilized link bandwidth that can be shared among the flows that are back-logged (active flows). The GPS scheduler shares this excess capacity among the back-logged flows in proportion to their respective weights. This should be contrasted with a static rate allocation as in TDMA or FDMA, where it is not possible to make use of the bandwidth unutilized by a flow.

The GPS scheduler or Fluid Fair Queueing (FFQ) scheduler as it is sometimes called is a theoretical construct that is defined for a fluid traffic model. The GPS scheduler server each fluid flow at a rate that is proportional to the weight assigned to the flow. With the weights defined as before, let $C(t)$ denote the set of flows that are back-logged at time $t \geq 0$. Then at time $t \geq 0$, flow i is served at the rate of $r_i(t)$ given by

$$r_i(t) = \begin{cases} \frac{\phi_i}{\sum_{j \in C(t)} \phi_j} r & i \in C(t), \\ 0 & \text{otherwise.} \end{cases}$$

Clearly in practice, real communication networks don't have fluid flows and the purpose of the GPS scheduler is to construct a reference model that can be used to determine the order of packet transmissions in the real world. A packetized version of GPS abbreviated as PGPS – also referred to as Weighted Fair Queueing (WFQ) [18] – is defined using GPS as a reference [37]. In the reference system the packet arrival process is replaced by a fluid flow with the jth packet of size l_j corresponding to a volume l_j of fluid. We say that a packet has been transmitted in the GPS reference system, when the fluid volume corresponding to that packet has been completely served. The PGPS scheduler tries to imitate the workings of the reference GPS scheduler. To do so, it simulates the reference GPS scheduler with the fluid arrival process as described above and computes the departure times of the packets in the reference system. If there are several packets waiting to be transmitted out on the link, the PGPS scheduler picks the packet that would have departed the earliest in the reference GPS system. In [37] it is shown that the PGPS scheduler transmits packets no more than L/r time units later than the reference GPS scheduler. Thus, delay bounds that are derived for the reference GPS scheduler, which is simpler to analyze, can be easily extended to obtain delay bounds for the PGPS scheduler. Currently, the best known end-to-end delay bounds are obtained when PGPS schedulers are assumed at each link [26]. This is the reason why we extensively compare the framework presented in this thesis with PGPS in subsequent chapters.

One advantage of the PGPS scheduler is that it has certain "fairness" properties. Roughly speaking fairness is the notion that during periods of congestion, the scheduler will guarantee each back-logged flow a proportional share of its service. A precise definition of fairness can be found in [25]. The reference GPS scheduler is typically used as a benchmark for fairness and by definition the PGPS scheduler is bound to be reasonably fair. Another advantage of the PGPS scheduler is that it lends itself to a fairly simple analysis, and tight end-to-end delay bounds can be computed on a per-flow basis [37].

In [37] it is shown that if the input traffic is characterized by the traffic envelope in (2.1), then tight end-to-end delay bounds can be obtained for the GPS discipline. These efficient bounds are obtained by analyzing the delay dependencies that are present among the various nodes along the path of the connection. In particular, it can be shown that if a flow experiences the worst case delay at a single GPS node along the path, it will not encounter any more delay at the subsequent nodes [37]. The main drawback of the PGPS discipline is the complexity involved in

1. simulating the reference GPS system, and
2. scheduling the appropriate packet departures.

In addition, there is an inverse relationship between the end-to-end delay bounds and the weight or bandwidth that is reserved at each switch. Thus, a low bandwidth connection that requires a tight delay bound is forced to reserve a large bandwidth pipe all along its path. On the other hand, the GPS service discipline provides the best end-to-end delay guarantees known so far, particularly for traffic that is characterized by a traffic envelope. There are several variations of the GPS service discipline like Worst-case Fair Weighted Fair Queueing (WF^{2}Q) [8], which improves on the fairness aspect, or Self-Clocked Fair Queueing (SCFQ) [25], which lends itself to a simpler implementation, but at the cost of looser delay bounds.

2.5 Increase in Burstiness as a Result of Scheduling

When several traffic streams are multiplexed onto a single link, they interfere with each other causing a potential “clumping” together of packets from the same stream at the output. This clumping effect, increases the burstiness of the stream as it passes through the scheduler. For example, consider the four connections A, B, C and D that are multiplexed onto a single link using a FIFO scheduler as shown in Figure 2.4. Each of the connections is assumed to send packets at regular intervals, so that there is a fixed spacing between the packets. At the output it can be observed that the burst length of connection A is now double what it was at the input. What is even more disturbing is the fact that as the connection passes through more and more hops, it can potentially become more and more bursty.

The “clumping effect” is even more pronounced for the lower priority connections of a Priority Scheduler. This is illustrated in Figure 2.5 which depicts a priority scheduler with two priority levels. The high priority connections, A, B, C and D are periodic as before, but the single low priority connection E is assumed bursty, with only a single burst consisting of three packets being shown in the figure. Note that the three packets of the burst are not back to back at the input to the scheduler, because of the spacing that is imposed by the peak rate limitation of connection E, which is lower than the link speed. The resulting output is also shown in Figure 2.5, where it can be observed that connection E now has three packets back-to-back, resulting in a higher peak rate

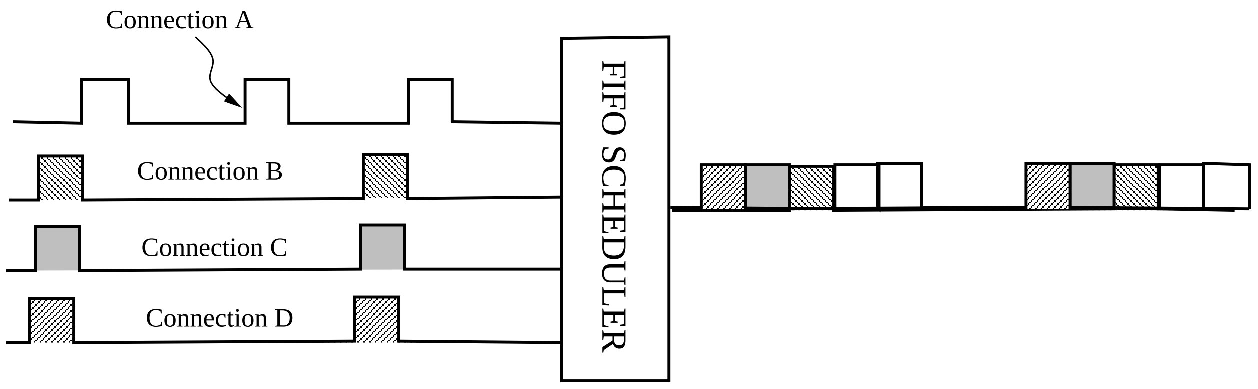

Figure 2.4: Increase in Burstiness at a FIFO scheduler

at the output. Thus even though in this case, the burst size of connection E remains the same, the effect of the priority scheduler is manifested in the increased peak rate of connection E. With multiple priorities, the clumping effect, in general, gets worse for the lower priority classes, as they are subjected to the compounded impact of all the higher priority traffic.

2.5.1 Bounds on the Output Traffic

As demonstrated in the previous section, the traffic of a connection can become more and more bursty as it traverses the network. However, it may still be possible to characterize the output traffic based on all the traffic that is input to the multiplexer. One of the basic approaches to providing end-to-end delay guarantees is to first obtain a delay bound at a single multiplexer, given the characterization of all the input traffic. Next, if a bound on the characterization of the output traffic is available, this can be used to characterize the traffic that will be input to the next switch on the path of the connection. The delay at the next switch can then be computed based on this traffic characterization and so on. This operation can be successively applied at all the switches along the path of the connection. The individual delay bounds obtained for the connection at each of the switches on its path can be summed up to obtain an end-to-end delay bound for the connection. There are several different ways to characterize the output traffic, based on the characterization of the input traffic, and we describe a couple of them.

2.5.2 Traffic characterization at the output of a multiplexer

Assume that the output link is served by a *work conserving* service discipline, i.e., the link is never idle when there are packets waiting to be transmitted. Let the input traffic be characterized by an

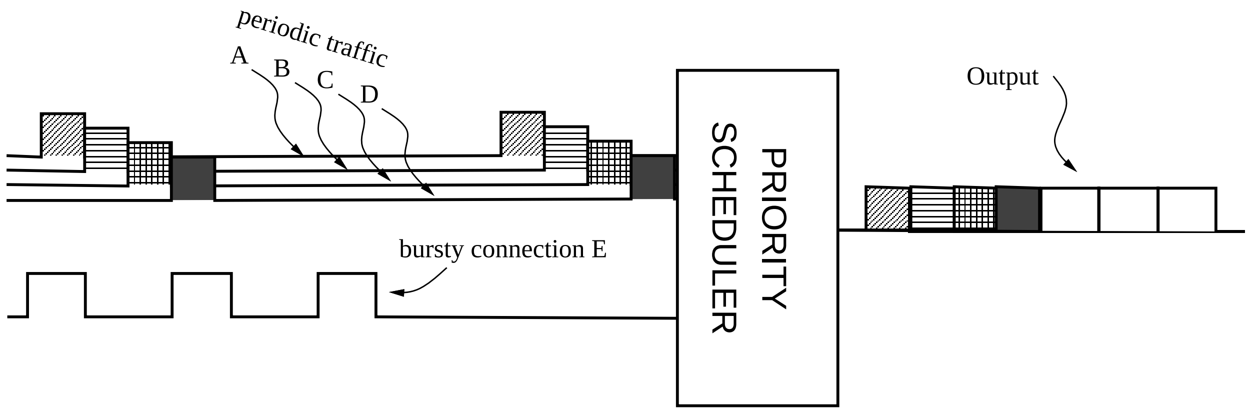

Figure 2.5: Clumping effect at a priority scheduler

affine envelope

$$\overline{I}(\tau) = \sigma + \rho\tau, \quad \tau \geq 0. \tag{2.1}$$

Further, assume that the maximum delay experienced by the connection at this multiplexer can be bounded above by D. In [13] it is shown that the output traffic can be characterized by an envelope $\overline{O}(\tau) = \sigma + \rho D + \rho\tau$, $\tau \geq 0$, and now $\overline{O}(\tau)$ can be used to obtain a delay bound at the next multiplexer. Notice that the bound on the burst size which was σ, at the input to the multiplexer, is now $\sigma + \rho D$. In other words the characterization of the connection traffic gets progressively more and more bursty as it traverses the network. Thus, the end-to-end delay bounds get worse and worse as the number of hops increase. So far we have focused on the approach where the traffic from a connection is characterized at each switch along its path, and delay bounds at each of the switches are independently computed. It is possible that this is a rather pessimistic assumption, for once traffic is significantly delayed at a switch, it may not be delayed as much at other switches along its path. In other words, there is a dependency on the traffic characteristics of a connection at each hop along its path, which in general, is fairly difficult to account for. However this has been done for a few scheduling policies, notably the GPS policy, and fairly tight end-to-end delay bounds have been obtained for GPS, both for the deterministic [37] and stochastic traffic models [57]. However, these bounds only apply for the case of acyclic networks, and for the case of networks where there are cycles, certain restrictions have to be applied [37, 57].

2.5.3 Feedback Networks

To appreciate the negative effects of a feedback network, consider the example of the ring network given in [13, 37]. The ring has M nodes labeled $0, 1, 2, \ldots, M-1$, with each node having two inputs

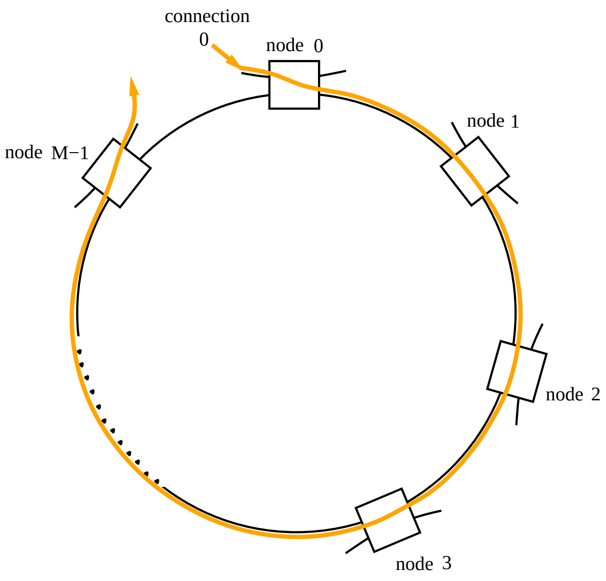

Figure 2.6: Ring network with symmetric connections, with only a single connection shown

and two outputs. Connection m enters the network at node m and departs the network at node $m + M - 1(\text{mod})M$ (denoted by $m \oplus (M-1)$), $m = 0, 1, \ldots, M-1$. This is illustrated in Figure 2.6 where only a single connection (connection 0) is shown. Note that this topology is not uncommon, with good examples including Token Ring or FDDI networks.

For simplicity, we assume a fluid traffic model with each connection being characterized by the same affine envelope process $A(\tau) = \sigma + \rho\tau$, $\tau \geq 0$. The network is shown in Figure 2.6, along with the path of a single connection. Now, assume that each of the nodes (say m) gives a low priority to traffic originating at that node (connection m), and a high priority to all other traffic. Alternatively, if GPS is the scheduling policy used, we can assume that a very small weight (≈ 0) is given to the traffic from connection m, while all the other connections have a weight of 1 at node m. When the input and output links are operating at the same speed r, it is clear that at any node, only the low priority connection will experience any delay. Thus the traffic characteristics of all connections i, $i \neq m$, are unchanged at node m.

Let $\overline{A_i^m}(\tau) = \sigma_i^m + \rho$, $\tau \geq 0$, be an envelope for connection i at the input to node m, with $m, i = 0, 1, \ldots, M-1$. Then by Remark 3 after Theorem 4.5 in [13], the burstiness of connection i

traffic at the input to node $i \oplus 1$ is bounded above by

$$\sigma_i^{i\oplus 1} = \sigma + \rho \left(\frac{\sum\limits_{k \neq i, i\oplus 1} \sigma_k^i}{r - \sum\limits_{k \neq i, i\oplus 1} \rho} \right), \quad i = 0, 1, \ldots, M-1, \tag{2.2}$$

and this bound is tight if any single node is considered in isolation. From the symmetry of the problem, it is clear that $\sigma_k^i = \sigma_k$ for all $k \neq i, i \oplus 1$. Thus we can solve (2.2) to obtain

$$\sigma_k = \frac{\sigma\,(r - (M-2)\rho)}{r - 2(M-2)\rho}. \tag{2.3}$$

For large M, (2.3) blows up when ρ approaches $\frac{r}{2M}$, thereby restricting the total link utilization to 1/2. In [37] it is conjectured that this result is rather conservative, since each node has been analyzed in isolation, and the delay dependencies in the traffic from the different connections may actually prevent the bound in (2.2) from being achieved. This is a still an open problem, and the issue of characterizing the traffic inside a network where there is potential for feedback can be fairly cumbersome. In addition, guaranteeing stability when there is feedback, often comes at the price of a lower link utilization.

Chapter 3

Traffic Shapers

Characterizing source traffic has always been a difficult problem. The literature abounds with models, some tailored to specific types of sources like audio, video, etc. while there are others that apply to sources in general. However most, if not all of them, are inherently approximate, and it is never easy to justify results based on these models.

In this thesis we follow a slightly different approach. Rather than estimate traffic characteristics at the source we force it to satisfy certain pre-defined characteristics, a process we call *reshaping.* While this in itself is not new, we go a step further and recommend traffic reshaping at *every* network element. Shaping traffic before it is presented to the link scheduler makes for a more predictable arrival process at the link scheduler. An efficient scheduler can then provide tight delay bounds on a per-connection basis. Before we get too far ahead, let us precisely define what we mean by a traffic shaper.

3.1 Traffic Shaper Definition

The function of the traffic shaper is to smooth potential bursts of traffic that may arrive at its input. It does so by delaying the incoming packets, so that the output of the shaper is bounded by a particular envelope function, say $\overline{A}$, which is referred to as the shaper envelope. The traffic shaper outputs packets in their order of arrival with each packet being released at the earliest time that allows $\overline{A}$ to be an envelope of the shaper output stream. More precisely, if s_i denotes the arrival time of the ith packet at the shaper and f_i denotes the time of its departure, we have $f_1 = s_1$ and

$$f_i = \min \left\{ t \geq s_i : \overline{A}(t - f_j) \geq \sum_{k=j}^{i} l_k,\ j = 1, 2, \ldots i - 1 \right\}, \quad i = 2, 3, \ldots \tag{3.1}$$

where l_i denotes the size of the ith packet released by the packetizer in Figure 2.2, $i = 1, 2, \ldots$. Note that it is necessary for the shaper envelope $\overline{A}$ to satisfy $\overline{A}(0) \geq L$, so that a packet of size

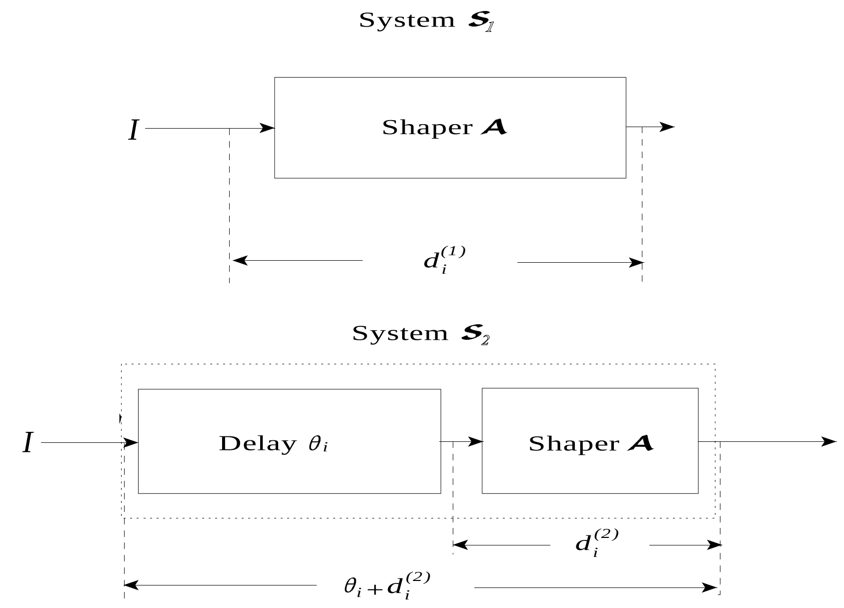

Figure 3.1: The Systems $\mathcal{S}_1$ and $\mathcal{S}_2$

L can pass through the shaper, and this is implicitly assumed in all shaper envelopes considered throughout this thesis. Traffic shapers exhibit a monotonicity property with respect to the arrival process, which is the subject of the next lemma.

3.2 Monotonicity properties of Traffic Shapers

System $\mathcal{S}_1$ consists of a traffic shaper $\mathcal{A}$, while system $\mathcal{S}_2$ consists of a "delay" subsystem and an identical shaper $\mathcal{A}$ connected in series as shown in Figure 3.1. The delay subsystem delays the ith arriving packet by an arbitrary amount $\theta_i \geq 0$, $i = 1, 2, \ldots$, and then delivers it to the shaper $\mathcal{A}$.

Lemma 3.1 *Assume that packets arrive to systems $\mathcal{S}_1$ and $\mathcal{S}_2$ according to the same arrival process I. If $d_i^{(1)}$ and $d_i^{(2)}$ denote the delay of the ith packet in the traffic shaper in systems $\mathcal{S}_1$ and $\mathcal{S}_2$, respectively, then*

$$d_i^{(1)} \leq d_i^{(2)} + \theta_i, \qquad i = 1, 2, \ldots$$

that is, the delay of every packet in system $\mathcal{S}_1$ is smaller than its corresponding delay in system $\mathcal{S}_2$.

Proof. Let $\overline{A}$ denote the envelope of the shaper in systems $\mathcal{S}_1$ and $\mathcal{S}_2$. Also, let $s_i^{(k)}$ denote the arrival time of the ith packet at the shaper of system $\mathcal{S}_k$ and let $f_i^{(k)}$ denote its departure time,

$k = 1, 2$. By definition, $s_i^{(2)} = s_i^{(1)} + \theta_i$ with $\theta_i \geq 0$, and therefore it suffices to show that

$$f_i^{(1)} \leq f_i^{(2)}, \quad i = 1, 2, \ldots$$

The proof proceeds by induction on $i = 1, 2, \ldots$ Since $\overline{A}(0) \geq L$, the first packet leaves the shaper instantaneously in both systems, i.e., $f_1^{(1)} = s_1^{(1)}$ and $f_1^{(2)} = s_1^{(2)}$, and we have $f_1^{(1)} \leq f_1^{(2)}$.

Now, assume that

$$f_i^{(1)} \leq f_i^{(2)}, \quad i = 1, 2, \ldots, m. \tag{3.2}$$

for some $m = 1, 2, \ldots$. From (3.1) we can compute the departure time of the $(m+1)$st packet in system $\mathcal{S}_1$ as

$$\begin{aligned} f_{m+1}^{(1)} &= \min\{t \geq s_{m+1}^{(1)} : \overline{A}(t - f_i^{(1)}) \geq \sum_{k=i}^{m+1} l_k,\ i = 1, 2, \ldots m\} \\ &\leq \min\{t \geq s_{m+1}^{(2)} : \overline{A}(t - f_i^{(1)}) \geq \sum_{k=i}^{m+1} l_k,\ i = 1, 2, \ldots m\} \qquad (3.3) \\ &\leq \min\{t \geq s_{m+1}^{(2)} : \overline{A}(t - f_i^{(2)}) \geq \sum_{k=i}^{m+1} l_k,\ i = 1, 2, \ldots m\} \qquad (3.4) \\ &= f_{m+1}^{(2)}, \end{aligned}$$

where (3.3) holds because $s_{m+1}^{(2)} \geq s_{m+1}^{(1)}$, and (3.4) follows from the non-decreasing nature of the shaper envelope $\overline{A}$ and the induction hypothesis (3.2). Hence, (3.2) holds for the $(m+1)$st packet and the induction step is completed. ■

Lemma 3.1 is an important property of the traffic shapers considered in this thesis and is key to establishing the general end-to-end delay bounds for RCS disciplines obtained in Corollary 3.2. Another form of monotonicity that traffic shapers exhibit is with respect to the shaper envelopes, and is considered next. In general, we use the notation $\overline{A}_n$ to denote the envelope of shaper $\mathcal{A}_n$. By definition, a traffic shaper has an envelope function, and so with a slight abuse of notation, we sometimes write $\mathcal{A}_1 \preceq \mathcal{A}_2$, to denote the ordering of the shaper envelopes, *viz.* $\overline{A}_1 \preceq \overline{A}_2$.

Lemma 3.2 *Assume that packets from the same arrival process I, arrive to shapers $\mathcal{A}_1$ and $\mathcal{A}_2$, with $\mathcal{A}_2 \preceq \mathcal{A}_1$. If $d_i^{(k)}$ denotes the delay of the ith packet in the shaper $\mathcal{A}_k$, $k = 1, 2$ then*

$$d_i^{(1)} \leq d_i^{(2)}, \qquad i = 1, 2, \ldots,$$

i.e., the delay of every packet through shaper $\mathcal{A}_1$ is smaller than its corresponding delay through shaper $\mathcal{A}_2$.

Proof. The proof is by induction and is similar to that of Lemma 3.1. Let s_i denote the arrival time, and let $f_i^{(k)}$ denote the departure times of the ith packet at the shapers $\mathcal{A}_k$, $k = 1, 2$. In both instances, because $\overline{A}^k(0) \geq L, k = 1, 2$, the first packet leaves the shaper instantaneously, i.e., $f_1^{(1)} = f_1^{(2)} = s_i$. Now, assume that

$$f_i^{(1)} \leq f_i^{(2)}, \quad i = 1, 2, \ldots, m \tag{3.5}$$

for some $m = 1, 2, \ldots$. From (3.1), we compute the departure time for the $(m+1)$st packet from shaper $\mathcal{A}_1$ as

$$\begin{aligned}
f_{m+1}^{(1)} &= \min\{t \geq s_{m+1} : \overline{A}_1(t - f_i^{(1)}) \geq \sum_{k=i}^{m+1} l_k,\ i = 1, 2, \ldots m\} \\
&\leq \min\{t \geq s_{m+1} : \overline{A}_1(t - f_i^{(2)}) \geq \sum_{k=i}^{m+1} l_k,\ i = 1, 2, \ldots m\} && (3.6) \\
&\leq \min\{t \geq s_{m+1} : \overline{A}_2(t - f_i^{(2)}) \geq \sum_{k=i}^{m+1} l_k,\ i = 1, 2, \ldots m\} && (3.7) \\
&= f_{m+1}^{(2)}, && (3.8)
\end{aligned}$$

where (3.6) follows from the induction hypothesis (3.5), and (3.7) is a consequence of the assumption $\mathcal{A}_2 \preceq \mathcal{A}_1$. Hence (3.5) holds for the $(m+1)$st packet and the induction step is completed. ■

3.3 Practical Traffic Shapers

While the operation of a traffic shaper has been well defined, it is clear from a practical point of view that this definition is unusable, because the complete history of packet departures has to be checked before determining subsequent packet departures. In a high speed network it is imperative that the computation to determine the departure time of a packet, be performed in less than a few microseconds. To address this concern, we now consider a more restrictive class of shapers that lend themselves to relatively simple implementations.

The simplest form of traffic shaper is the Leaky Bucket (or Token Bucket) which was first introduced as a regulatory device in [47]. The Leaky Bucket is a simple device which can be described as follows:

- Tokens accumulate at a fixed rate (ρ) into a bucket that can accommodate a fixed number (σ) of tokens.
- A packet is allowed to leave the Leaky bucket if there is a token with which it can be paired. Each departing packet decreases the number of tokens in the bucket by one. If there are no tokens in the bucket the packet is queued until there is a token for it to depart.

The above regulator is fairly simple to implement as it requires a single state variable that represents the number of tokens present in the bucket. They are relatively simple to implement in hardware

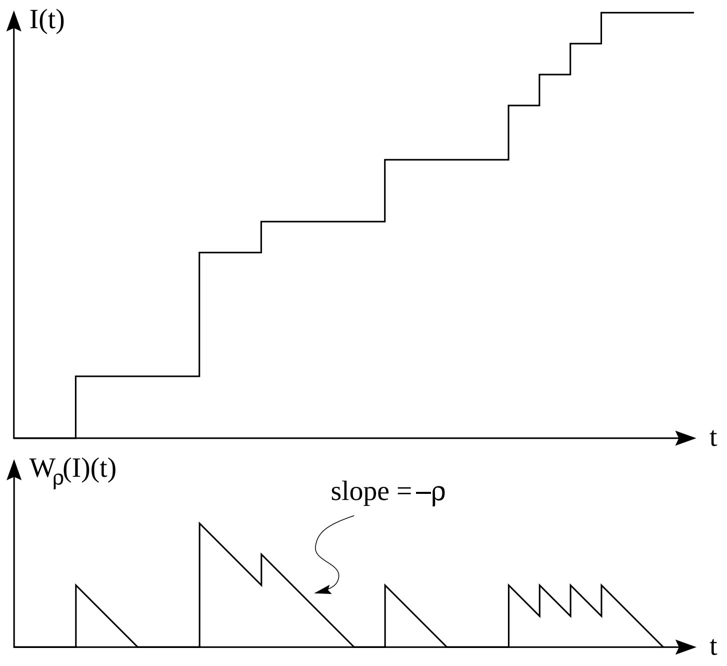

Figure 3.2: Example illustrating the computation of $W_\rho(I)(t)$

and most vendors' ATM adapters offer support for Leaky Bucket regulation. Leaky Buckets are an integral part of both IP (IntServ) and ATM standards [42, 2].

The Leaky Bucket described above is very similar to the (σ, ρ) regulator of Cruz [13], which can be described in terms of the backlog in a hypothetical queue served at the rate ρ : If traffic I is fed to a queue that is served at the rate of ρ, then the backlog in this queue, denoted by $W_\rho(I)$, is given [7] by

$$W_\rho(I)(t) := \max_{0 \le s \le t} \{I(t) - I(s) - \rho(t - s)\} \quad t \ge 0. \tag{3.9}$$

The (σ, ρ)-regulator releases the ith packet (arriving at time s_i) at the earliest time $f_i \ge s_i$ such that the regulator output traffic A satisfies the condition

$$W_\rho(A)(f_i) \le \sigma. \tag{3.10}$$

By substituting (3.9) in (3.10) it is easily verified that the (σ, ρ)-regulator is a shaper with envelope $\overline{A}$ given by

$$\overline{A}(\tau) = \sigma + \rho\tau, \quad \tau \ge 0.$$

The condition $\sigma \ge L$ is required to ensure that a maximum sized packet passes through the shaper.

The (σ, ρ)-regulator described here differs from the one defined in [13] in two minor respects:

1. Packets are entering and exiting the regulator instantaneously and not at some finite bit-rate, and
2. The length of the ith packet exiting the regulator at time f_i is included in the calculation of $W_\rho(A)(f_i)$.

Note that s_i and f_i are defined as the times when the *last* (not the *first* as in [13]) bit of the ith packet enters and exits the regulator, respectively. However, with $d_i := f_i - s_i$, denoting the delay that the ith packet experiences in the regulator, the analysis in [13, Appendix E] can be repeated with minor modifications to show that,

$$d_i = \frac{1}{\rho}\left[W_\rho(I)(s_i) - \sigma\right]^+, \quad i = 1, 2, \ldots, \tag{3.11}$$

where $x^+ \equiv \max\{x, 0\}$.

An attractive property of the (σ, ρ)-regulators is that they can be connected in series to obtain traffic-shapers with more general envelopes. First, we state Theorem 5.1 of [14] without proof, which only requires minor modifications to that found in [14] to account for our definition of the (σ, ρ)-regulator.

3.4 Series connection of Traffic Shapers

Theorem 3.1 ([14]) *Consider a series of (σ_k, ρ_k)-regulators, $k = 1, \ldots, K$, and let s_i denote the arrival time of the ith packet to the first regulator (the (σ_1, ρ_1)-regulator) and let f_i denote its departure time from the last regulator (the (σ_K, ρ_K)-regulator). Then the delay encountered by packet i is given by*

$$f_i - s_i = \max_{k=1,2,\ldots,K}\left\{\frac{1}{\rho_k}\left(W_{\rho_k}(I)(s_i) - \sigma_k\right)^+\right\}, \qquad i = 1, 2, \ldots \tag{3.12}$$

In addition, we have

$$A(t+\tau) - A(t) \le \min_{k=1,2,\ldots,K}\{\sigma_k + \rho_k\tau\}, \quad t, \tau \ge 0, \tag{3.13}$$

where A denotes the traffic stream at the output of the (σ_K, ρ_K)-regulator.

Corollary 3.1 *A series of (σ_k, ρ_k)-regulators, $k = 1, \ldots, K$, is a traffic shaper with envelope $\overline{A}$ given by*

$$\overline{A}(\tau) = \min_{k=1,2,\ldots,K}\{\sigma_k + \rho_k\tau\}, \quad \tau \ge 0. \tag{3.14}$$

Proof. In order to show that the series of regulators is a shaper with envelope $\overline{A}$ we need to show that each packet exiting the shaper, is delayed by the minimal amount necessary in order to satisfy (3.13). Consider the delay d_i encountered by the ith packet, and assume that the maximum in (3.12) is attained for some ℓ, $\ell = 1, \ldots K$, i.e.

$$d_i = \frac{1}{\rho_\ell} \left(W_{\rho_\ell}(I)(s_i) - \sigma_\ell \right)^+ .$$

From (3.11), we know that this is the delay experienced by the ith packet in a (σ_ℓ, ρ_ℓ)-regulator. But by definition,

$$\overline{A}(\tau) \leq \sigma_\ell + \rho_\ell \tau, \quad \tau \geq 0,$$

and so from Lemma 3.2, we conclude that d_i is the minimum possible delay for packet i, in any shaper with envelope $\overline{A}$. Since the choice of packet i was arbitrary, it is clear that this series of K regulators delays each packet by no more than a shaper with envelope $\overline{A}$. ■

Note that (3.12) does not depend on the order in which the regulators are arranged. In fact, the K regulators in series are equivalent to K regulators in parallel, with the understanding that a packet is released only when it clears all K regulators.

In the remainder of this thesis, we restrict ourselves to shapers whose envelope is a concave, increasing (i.e., $\overline{A}(t_1) < \overline{A}(t_2)$ whenever $t_1 < t_2$), piecewise linear function with finite number of slopes on $\mathbb{R}_+$.

Our interest in these types of shapers stems from the fact they constitute generalizations of shapers adopted by the the Internet [44] and ATM standards [2]. Also, from Corollary 3.1, it follows that these shapers can be easily implemented using a series of (σ, ρ)-regulators. From (3.13) we can obtain an upper bound on the packet delays in the shaper, when the traffic envelope of the input process to the shaper is known. Taking into account (3.9) in (3.12), we have

$$\begin{aligned} d_i &\leq \max_{k=1,2,\ldots,K} \left\{ \frac{1}{\rho_k} \left(\max_{0 \leq s \leq s_i} \left\{ \overline{I}(s_i - s) - \rho_k(s_i - s) - \sigma_k \right\} \right)^+ \right\} \\ &\leq \max_{k=1,2,\ldots,K} \left\{ \left(\max_{\tau \geq 0} \left\{ \frac{\overline{I}(\tau) - \sigma_k - \rho_k \tau}{\rho_k} \right\} \right)^+ \right\} && (3.15) \\ &= D(\overline{I} \| \mathcal{A}), && (3.16) \end{aligned}$$

where we have set

$$D(\overline{I} \| \mathcal{A}) := \max_{\tau \geq 0} \left\{ \left(\max_{k=1,2,\ldots,K} \left\{ \frac{\overline{I}(\tau) - \sigma_k}{\rho_k} \right\} - \tau \right)^+ \right\} . \qquad (3.17)$$

The quantity $D(\overline{I} \| \mathcal{A})$ only depends on the input traffic envelope $\overline{I}$ and the shaper envelope $\overline{A}$. Also, $D(\overline{I} \| \mathcal{A})$ is indeed a tight upper bound on the maximum delay experienced by any packet in

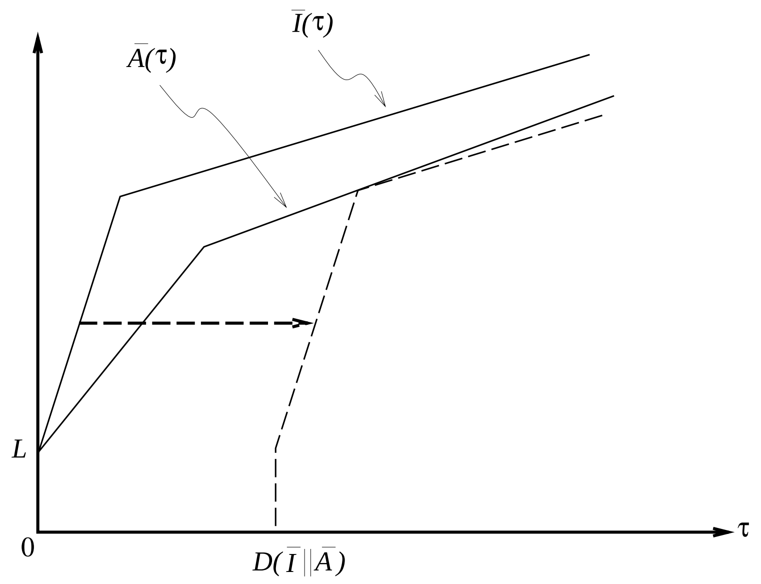

Figure 3.3: Graphical computation of $D(\overline{I}||\mathcal{A})$.

the shaper. This can be seen by assuming that the input traffic has infinitesimal packets (L=0) arriving at the maximum rate allowed by its traffic envelope, i.e.

$$I(t) = \overline{I}(t) \quad t \geq 0. \tag{3.18}$$

From 3.18), (3.9) and (3.11) we conclude that the delay experienced in the shaper is indeed $D(\overline{I}||\mathcal{A})$.

We can write (3.17) in another form that will be useful in the sequel. From (3.14), the range of $\overline{A}$ is $[\min_k \sigma_k, \infty)$ and its inverse is given by

$$\overline{A}^{(-1)}(y) = \max_{k=1,\cdots,K} \left\{ \frac{y - \sigma_k}{\rho_k} \right\}, \quad y \geq \min_k \sigma_k. \tag{3.19}$$

Extending the definition of $\overline{A}^{(-1)}$ by setting $\overline{A}^{(-1)}(y) = 0$ whenever $0 \leq y < \min_k \sigma_k$, we see from (3.17) and (3.19) that

$$D(\overline{I}||\mathcal{A}) = \max_{\tau \geq 0} \left\{ \left(\overline{A}^{(-1)} \left(\overline{I}(\tau) \right) - \tau \right)^+ \right\}. \tag{3.20}$$

Graphically, (3.20) represents the maximum horizontal distance between $\overline{I}(\tau)$ and $\overline{A}(\tau)$, as illustrated in Figure 3.3.

When the traffic entering shaper $\mathcal{A}$ is the output of a shaper $\mathcal{A}_1$ with envelope $\overline{A}_1$, we also use the notation $D(\mathcal{A}_1||\mathcal{A}) \equiv D\left(\overline{A}_1||\mathcal{A}\right)$. If $\overline{I} \preceq \overline{A}$, then from (3.15) we have $D\left(\overline{I}||\mathcal{A}\right) = 0$ which implies that no packet is delayed in shaper $\mathcal{A}$. In particular we have

$$D(\mathcal{A}||\mathcal{A}) = 0. \tag{3.21}$$

Consider next two shapers $\mathcal{A}_1$ and $\mathcal{A}_2$ in series. Corollary 3.1 implies that this arrangement is equivalent to a traffic shaper $\mathcal{A}_3$ with envelope

$$\overline{A}_3(\tau) = \min\left\{\overline{A}_1(\tau), \overline{A}_2(\tau)\right\}, \quad \tau \geq 0. \tag{3.22}$$

By equivalence, we mean that for any input traffic stream, the delay of every packet from the time it enters $\mathcal{A}_1$ to the time it exits $\mathcal{A}_2$ is identical to the delay of the packet in $\mathcal{A}_3$. We use the notation $\mathcal{A}_1 \wedge \mathcal{A}_2$ to denote the series connection of shapers $\mathcal{A}_1$ and $\mathcal{A}_2$.

3.5 Minimal shaper envelopes

We consider next, the problem of constructing the "smallest" shaper that can provide a specified delay bound for the input traffic with envelope $\overline{I}$. Specifically, given $d \geq 0$, we want to construct a shaper $\mathcal{A}(d)$ such that $D\left(\overline{I}\|\mathcal{A}(d)\right) \leq d$, with the additional requirement that $\mathcal{A}(d) \preceq \mathcal{A}$ for any shaper $\mathcal{A}$ satisfying $D\left(\overline{I}\|\mathcal{A}\right) \leq d$. We assume that $\overline{I}$ is an increasing, concave, piecewise linear function with a finite number of slopes. Later on, we explain how these assumptions on the input traffic envelope, essentially, do not entail any loss of generality.

For convenience, let

$$\overline{U}(\tau) \equiv \overline{I}(\tau) - L, \quad \tau \geq 0. \tag{3.23}$$

Since $\overline{I}$ is a concave, increasing, piecewise linear function with a finite number of slopes, the same is true for $\overline{U}$. Assume $\overline{U}$ has K slopes, denoted by ρ_k, $\rho_k > \rho_{k+1}$, $k = 1, \ldots, K-1$. Set $\tau_1 = 0$ and let τ_k be such that at point $(\tau_k, \overline{U}(\tau_k))$ the slope of the envelope $\overline{U}$ changes from ρ_{k-1} to ρ_k, $k = 2, \ldots, K$. We can then write $\overline{U}$ in the form (see Figure 3.4)

$$\overline{U}(\tau) = \min_{k=1,\ldots,K} \left\{\delta_k + \rho_k \tau\right\}, \quad \tau \geq 0 \tag{3.24}$$

where $\delta_1 = \overline{U}(0)$ and

$$\delta_k = \delta_{k-1} + \tau_k(\rho_{k-1} - \rho_k), \quad k = 2, \ldots, K. \tag{3.25}$$

According to (3.23) and (3.24) the envelope $\overline{I}$ can be written as

$$\overline{I}(\tau) = L + \min_{k=1,\ldots,K} \left\{\delta_k + \rho_k \tau\right\}, \quad \tau \geq 0.$$

Now, assume that traffic with envelope $\overline{I}$ is reshaped by shaper $\mathcal{A}$ with envelope

$$\overline{A}(\tau) = L + \min_{j=1,\ldots,J} \left\{\delta'_j + \rho'_j \tau\right\}, \quad \tau \geq 0.$$

According to (3.17), $D\left(\overline{I}\|\mathcal{A}\right) = \infty$ when $\min_{j=1,\ldots,J}\left\{\rho'_j\right\} < \rho_K$, while $D\left(\overline{I}\|\mathcal{A}\right) \leq \delta_K/\rho_K$ whenever $\min_{j=1,\ldots,J} \rho'_j \geq \rho_K$, and it suffices to restrict our attention to the range $0 \leq d \leq \delta_K/\rho_K$. For the next theorem, it will be helpful to refer to Figure 3.4.

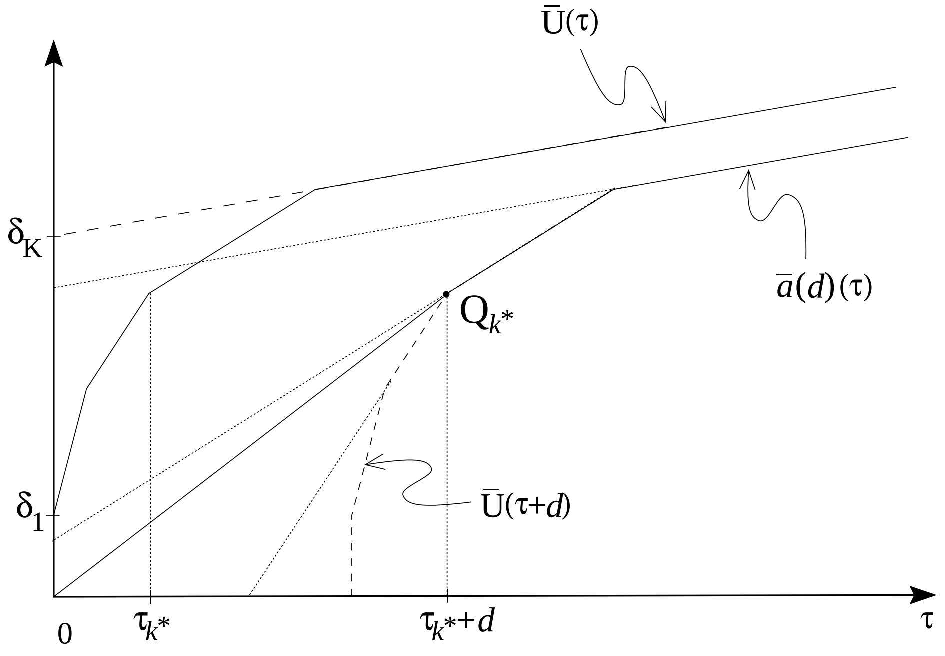

Figure 3.4: Construction of the Smallest Envelope Function

Theorem 3.2 *Let* $0 \le d \le \delta_K/\rho_K$*, and define*

$$k^* := \min_{k=1,\ldots,K} \left\{ k : \overline{U}(\tau_k) \;-\rho_k(\tau_k + d) \ge 0 \right\}.$$

Then, the envelope of the smallest shaper $\mathcal{A}(d)$ *such that* $D\left(\overline{I}\|\mathcal{A}(d)\right) \le d$*, is given by*

$$\overline{A}(d)(\tau) = L + \overline{a}(d)(\tau), \quad \tau \ge 0,$$

where

$$\overline{a}(d)(\tau) = \begin{cases} \dfrac{\overline{U}(\tau_{k^*})}{(\tau_{k^*}+d)}\tau & \text{if} \quad 0 \le \tau < \tau_{k^*} + d, \\ \overline{U}(\tau - d) & \text{if} \quad \tau \ge \tau_{k^*} + d. \end{cases}$$

Proof. From Figure 3.4, it can be seen that k^* is the smallest index k such that the line with slope ρ_k, passing through the point $Q_k = \left(\tau_k + d, \overline{U}(\tau_k)\right)$ has a non-negative y-intercept. The index k^* always exists because

$$\begin{aligned} \overline{U}(\tau_K) - \rho_K(\tau_K + d) &= \delta_K + \rho_K\,\tau_K - \rho_K(\tau_K + d) \\ &= \delta_K - \rho_K\, d \;\ge\; 0. \end{aligned}$$

Next, we show that $\overline{A}(d)$ corresponds to a shaper envelope function. For this, it suffices to show that $\overline{A}(d)$ is concave on $\mathbb{R}_+$, which will follow by construction, if we show that $\frac{\overline{U}(\tau_{k^*})}{(\tau_{k^*}+d)} \ge \rho_{k^*}$; but this is a consequence of the definition of k^*. To show that $D\left(\overline{I}\|\mathcal{A}(d)\right) \le d$, recall that according to (3.20) we can write

$$D\left(\overline{I}\|\mathcal{A}(d)\right) = \max_{\tau\ge 0}\left\{\left(\,\overline{A}^{(-1)}(d)\left(\overline{I}(\tau)\right) - \tau\right)^+\right\},$$

and that by construction we have $\overline{A}^{(-1)}(d)\left(\overline{I}(\tau)\right) - \tau \le d$ for all $\tau \ge 0$.

Finally, we need to show that $\overline{A}(d) \preceq \overline{A}$ for any other shaper $\mathcal{A}$ such that $D\left(\overline{I}\|\mathcal{A}\right) \le d$. To see this, observe that if $\overline{A}(d)(\tau) > \overline{A}(\tau)$ for some $\tau \ge \tau_{k^*} + d$, then $\overline{A}^{(-1)}\left(\overline{A}(d)(\tau)\right) > \tau$. Also, by construction

$$\overline{A}(d)(\tau) = \overline{I}(\tau - d), \quad \tau \ge \tau_{k^*} + d.$$

Therefore, from (3.20) we have

$$\begin{aligned} D\left(\overline{I}\|\mathcal{A}\right) &\ge \overline{A}^{(-1)}\left(\overline{I}(\tau-d)\right) - (\tau - d), \quad \tau \ge \tau_{k^*} + d \\ &> \tau - (\tau - d) = d, \end{aligned}$$

a contradiction. We conclude that

$$\overline{A}(d)(\tau) \le \overline{A}(\tau) \quad \tau \ge \tau_{k^*} + d. \tag{3.26}$$

Now, from the definition of traffic shapers,

$$\overline{A}(d)(0) = L \le \overline{A}(0). \tag{3.27}$$

From the concavity of the shaper envelope $\overline{A}$ we know that for $0 \le \lambda \le 1$,

$$\begin{aligned} \overline{A}(\lambda(\tau_{k^*} + d)) &\ge \lambda\overline{A}(\tau_{k^*} + d) + (1-\lambda)\overline{A}(0) \\ &\ge \lambda\overline{A}(d)(\tau_{k^*} + d) + (1-\lambda)\overline{A}(d)(0) \qquad (3.28) \\ &= \overline{A}(d)(\tau_{k^*} + d), \qquad (3.29) \end{aligned}$$

where (3.28) follows from (3.27) and (3.26), and (3.29) is a result of the construction, i.e.,

$$\overline{A}(d)(\tau) = L + \frac{\overline{U}(\tau_{k^*})}{\tau_{k^*} + d} \quad 0 \le \tau \le \tau_{k^*} + d.$$

■

As mentioned earlier, in this thesis we are mainly interested in traffic envelopes that are concave, increasing, piecewise linear functions with finite number of slopes. Given such an envelope, and the maximum shaper delay that can be tolerated, the construction in Theorem 3.2 can be used to compute the envelope for the smallest shaper. However, it turns out that Theorem 3.2 can be extended to apply to a larger class of traffic envelopes, *viz* the set of sub-additive, increasing, piecewise linear functions with a finite number of slopes. Such functions can approximate arbitrarily closely any nondecreasing function $\mathbb{R}_+ \to \mathbb{R}$ in the sense of the Skorohod metric [38, Chapter VI].

Theorem 3.3 *Let $\widehat{I}$ be a non-decreasing piecewise linear function with a finite number (say K) of slopes such that $\lim_{\tau\to\infty}\widehat{I}(\tau)/\tau > 0$. Now if $\overline{I}$ denotes the minimal concave function such that*

$$\overline{I}(\tau) \geq \widehat{I}(\tau) \quad \tau \geq 0.$$

and $\mathcal{A}(d)$ denotes the envelope of the smallest shaper such that $D\left(\overline{I}\|\mathcal{A}(d)\right) \leq d$, then $\mathcal{A}(d)$ is indeed the smallest shaper such that $D\left(\widehat{I}\|\mathcal{A}(d)\right) \leq d$.

Proof. Note that $\overline{I}$ is increasing, piecewise linear with a finite number of slopes and we can use Theorem 3.2 to construct the minimal envelope $\overline{A}(d)$.

Now, consider another shaper $\mathcal{A}$ such that $D(\widehat{I}\|\mathcal{A}) \leq d$, and assume that there exists some $\tau' \geq 0$, such that

$$\overline{A}(\tau') < \overline{A}(d)(\tau'). \tag{3.30}$$

Interpreting the inequality $D(\widehat{I}\|\mathcal{A}) \leq d$, using (3.20) (alternatively, see Figure 3.3), we know that

$$\widehat{I}(\tau) \leq \overline{A}(\tau + d), \quad \tau \geq 0. \tag{3.31}$$

By assumption, $\overline{A}$ is a concave function; however, $\overline{I}$ is the minimal concave function such that $\overline{I}(\tau) \geq \widehat{I}(\tau)$ for all $\tau \geq 0$. Therefore,

$$\overline{I}(\tau) \leq \overline{A}(\tau + d), \quad \tau \geq 0.$$

Using (3.20) again, we conclude that $D(\overline{I}\|\mathcal{A}) \leq d$. Now, using (3.17) we have

$$\begin{aligned} D\left(\overline{I}\|\mathcal{A}(d) \wedge \mathcal{A}\right) &= \max\left\{D\left(\overline{I}\|\mathcal{A}(d)\right),\ D(\overline{I}\|\mathcal{A})\right\} \\ &\leq d. \end{aligned}$$

But the shaper $\mathcal{A}(d) \wedge \mathcal{A}$ has envelope

$$\overline{A}_\wedge(\tau) = \min\left\{\overline{A}(d)(\tau),\ \overline{A}(\tau)\right\}. \tag{3.32}$$

From (3.30) the envelope $\overline{A}_\wedge$ is strictly smaller than $\overline{A}(d)$, thereby contradicting the optimality of $\mathcal{A}(d)$. ■

3.6 Reshaping traffic at every hop

If we reshape traffic at the source we have an efficient characterization of the traffic at the first network element along its path. This characterization can be used by the first network element to accurately estimate the amount of resources that it needs to allocate for a given flow. However, the

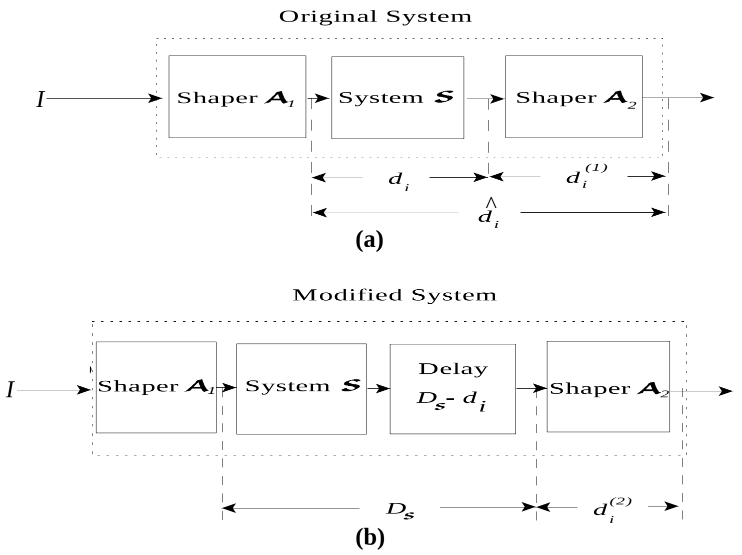

Figure 3.5: Original and Modified System used in the proof of Theorem 3.4

scheduling in the first network element perturbs the traffic and it can no longer be efficiently characterized at the input to the second network element along its path. This problem is exacerbated at network elements that are further downstream.

One way to provide accurate characterization of the traffic at each hop in the network is by reshaping the traffic at each hop. The main issue to consider here, is whether the shaper delays have to be added to the end-to-end delay guarantees that are obtained by summing up the single hop delay bounds. In [55] it is shown that the shaper delays do not add to the worst case end-to-end delays, for a connection that can be characterized by the $(X_{\text{min}}, X_{\text{ave}}, I)$ traffic model, where X_{min} is the minimum inter-arrival time between two packets, and X_{ave} is the minimum average inter-arrival time of packets measured over any interval of length I. It turns out that shaper delays do not add to the bounds on the end-to-end delays for more general models of traffic as well. More specifically, in the next theorem we show that if the connection traffic is reshaped at every hop to its bounding envelope process $\overline{A}$, then the end-to-end delay guarantee is simply the sum of the delay guarantees provided by the schedulers at each hop. In particular, the shapers do not contribute to the worst-case end-to-end delay bound.

Theorem 3.4 *Consider the system depicted in Figure 3.5(a). Assume that the output of traffic shaper $\mathcal{A}_1$ enters a system $\mathcal{S}$ where it is known that the delay experienced by these packets is bounded above by $D_{\mathcal{S}}$. The output of system $\mathcal{S}$ enters shaper $\mathcal{A}_2$. The total delay $\widehat{d_i}$ that packet i experiences* *from the time it exits $\mathcal{A}_1$ to the time it exits $\mathcal{A}_2$ is bounded above by*

$$\widehat{d}_i \leq D_{\mathcal{S}} + D(\mathcal{A}_1 \| \mathcal{A}_2), \quad i = 1, 2, \ldots \tag{3.33}$$

Proof. Let d_i be the delay of packet i in system $\mathcal{S}$, and let $d_i^{(1)}$ be its delay in $\mathcal{A}_2$. The total delay experienced by packet i, from the time it exits shaper $\mathcal{A}_1$ to the time it exits shaper $\mathcal{A}_2$ is denoted by $\widehat{d}_i = d_i + d_i^{(1)}$. Consider next a modified system where a delay system that delays the ith packet by $\theta_i = D_{\mathcal{S}} - d_i$, is inserted between $\mathcal{S}$ and $\mathcal{A}_2$ (see Figure 3.5). Now let $d_i^{(2)}$ denote the delay of packet i in $\mathcal{A}_2$ under this new arrangement. Note that $\theta_i \geq 0$ by the definition of $D_{\mathcal{S}}$. Applying Lemma 3.1 we conclude that

$$d_i^{(1)} \leq D_{\mathcal{S}} - d_i + d_i^{(2)},$$

or equivalently,

$$\widehat{d}_i \leq D_{\mathcal{S}} + d_i^{(2)}.$$

Observe now that since the delay of every packet between its entrance time to $\mathcal{S}$ and its exit from the delay system is $d_i + \theta_i = D_{\mathcal{S}}$, the traffic entering shaper $\mathcal{A}_2$ when the delay system is inserted, is a time-shifted version of the traffic exiting $\mathcal{A}_1$, and therefore has envelope $\overline{A}_1$. Hence, $d_i^{(2)} \leq D(\mathcal{A}_1 \| \mathcal{A}_2)$ and the proof of (3.33) is completed. ■

From (3.21) we know that when the shapers $\mathcal{A}_1, \mathcal{A}_2$ are identical, $D(\mathcal{A}_1 \| \mathcal{A}_2) = 0$, i.e., in this case reshaping does not introduce extra delays. Also, from the proof we see that any shaper that has the property of Lemma 3.1 satisfies Theorem 3.4 as well. In particular, the shaper of [55] can easily be seen to satisfy Lemma 3.1. We refer to the combination of the per-connection shapers and the scheduler as a Rate-Controlled Service (RCS) discipline [55].

We can now apply Theorem 3.4 to provide end-to-end delay guarantees to a connection that passes through a network of nodes that use the RCS discipline.

In the remainder of this thesis we follow the convention that the subscript $_n$ on a variable signifies a reference to a particular connection (connection n). Similarly we use the superscript ^{m} to identify the network element along the path of the connection. For example, $\overline{A}_n^m$ denotes the traffic shaper envelope for connection n at the network element m hops away from the source.

Corollary 3.2 *Assume connection n has a traffic envelope of $\overline{I}_n$ at the input to the network and passes through M network nodes, numbered from 1 to M, with $M+1$ denoting the destination. Let $\mathcal{A}_n^m$ denote the envelope of the traffic shaper for connection n at node m. Then the end-to-end delay experienced by a packet from connection n satisfies the following guaranteed upper bound*

$$D_n = D(\overline{I}_n \| \mathcal{A}_n^1) + \sum_{m=1}^{M-1} D\left(\mathcal{A}_n^m \| \mathcal{A}_n^{m+1}\right) + \sum_{m=1}^{M} D_n^m + \sum_{m=1}^{M} T^{(m,m+1)}, \tag{3.34}$$

where D_n^m denotes the scheduler delay bound for connection n at node m and $T^{(m,m+1)}$ denotes propagation delay on the link $(m, m+1)$.

Proof. Apply Theorem 3.4 with the system $\mathcal{S}$ consisting of both the scheduler at node m and the link $l = (m, m+1)$, and the shapers $\mathcal{A}_1 \equiv \mathcal{A}_n^m$, and $\mathcal{A}_2 \equiv \mathcal{A}_n^{m+1}$. We conclude that the delay that a packet from connection n experiences between the time it exits shaper $\mathcal{A}_n^m$ and the time it exits $\mathcal{A}_n^{m+1}$ is bounded above by,

$$D_n(m, m+1) = D_n^m + T^{(m,m+1)} + D\left(\mathcal{A}_n^m \| \mathcal{A}_n^{m+1}\right). \tag{3.35}$$

Taking (3.35) into account we obtain

$$\begin{aligned} D_n &= D(\overline{I}_n \| \mathcal{A}_n^1) + \sum_{m=1}^{M-1} D_n(m, m+1) + D_n^M + T^{(M,M+1)} \\ &= D(\overline{I}_n \| \mathcal{A}_n^1) + \sum_{m=1}^{M-1} D\left(\mathcal{A}_n^m \| \mathcal{A}_n^{m+1}\right) + \sum_{m=1}^{M} D_n^m + \sum_{m=1}^{M} T^{(m,m+1)}. \end{aligned}$$

■

It is important to note that the delay bounds D_n^m depend on the choice of the traffic shapers $\mathcal{A}_n^m$. Therefore, one should not conclude from (3.34) that the end-to-end delay guarantees are minimized by choosing $\overline{I}_n$ as the envelope for all the traffic shapers so that $D(\overline{I}_n \| \mathcal{A}_n^1) = D\left(\mathcal{A}_n^m \| \mathcal{A}_n^{m+1}\right) = 0$. In fact, as we will see in the next section, this choice may be quite inappropriate.

As in the policies proposed in [55], the delay bound in (3.34) is basically a sum of the worst case delays at *each* node along the path of a connection. However, an individual packet may not encounter the worst case delays at each node. Therefore, one may suspect that these bounds are overly pessimistic and lead to inefficient allocations when compared to bounds for other disciplines which take into account delay dependencies between nodes along the path. In Chapter 5 we compare the end-to-end delay bounds for the RCS discipline with those for the GPS discipline and show that with the appropriate choice of shaper envelopes, the RCS discipline can provide as good if not better delay bounds.

3.7 Summary

In this chapter we define a traffic shaper, which is a fundamental building block for the Rate Controlled Service (RCS) discipline. We start with a generic traffic shaper and derive some basic monotonicity properties of traffic shapers. These properties are key to proving the end-to-end delay bounds for RCS disciplines.

From a practical point of view we need to further restrict the space of traffic shapers so that they can be efficiently implemented, preferably in hardware. The Leaky Bucket or (σ, ρ)-regulator is the most popular in the networking world and we mainly consider traffic shapers that can be realized by a series concatenation of (σ, ρ)-regulators. We derive a bound on the delay experienced by a connection given the input traffic and shaper envelopes. Subsequently, we describe a construction on how to obtain the "minimal" traffic shaper given the input traffic envelope and a pre-specified delay bound.

We conclude this chapter by examining Rate Controlled Service disciplines where each individual connection is reshaped at every hop along its path and derive an end-to-end delay bound for the connection based on local delay bounds at the schedulers at each network element. In the sequel we will examine how we can choose shapers and schedulers so that per-connection end-to-end delay bounds can be efficiently provided by the network.

Chapter 4

Scheduler and Shaper Parameters

The idea to regulate traffic on a per-connection basis at each node, before considering it eligible for scheduling has appeared in different forms in the literature. The Stop and Go service discipline [24] recognized the value of delaying the transmission of some packets in order to provide a more predictable end-to-end behavior for all connections. Subsequently the Delay-EDD policy – the EDF policy is sometimes referred to as Earliest Due Date (EDD) – was proposed in [20]. The idea was to use the EDF policy at each node, with the deadlines being assigned with respect to the time that the packet is expected to arrive based on its traffic characterization, rather than its actual arrival time at the node. The Jitter-EDD policy in [48] operated along the same lines, with the difference that the traffic pattern at the ingress into the network was recreated at each node along the path of the connection. This was achieved by placing the difference between a packet's deadline and its actual transmission time at the node, in the packet header. In the downstream node the packet was delayed by exactly the amount of time specified in its header before it was considered eligible for scheduling. In [54] it was shown that with $(X_{\text{min}}, X_{\text{ave}}, I, S_{\text{max}})$ regulators a Static Priority scheduling algorithm could be used to provide end-to-end delay guarantees that were the sum of the delay bounds at each of the nodes along the path of the connection. Finally, the term Rate-Controlled Service (RCS) disciplines was used in [55] to denote the broad class of scheduling disciplines that use regulators at each hop to shape the traffic of each connection, before it is considered eligible for transmission on the link.

4.1 Rate Controlled Service Disciplines

As shown in Figure 4.1, RCS disciplines are composed of two parts:

1. A traffic regulator (also called a rate controller), and
2. A scheduling policy.

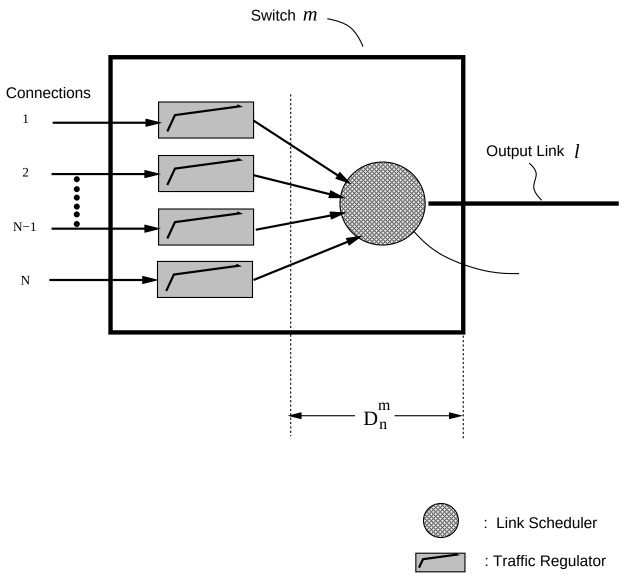

Figure 4.1: Components of the RCS Discipline

Traffic that arrives on the input link, has to first pass through a regulator before it is considered eligible for scheduling. In practice an entire packet is typically received, and then a simple check is performed to determine the eligibility time of this packet. If the regulator is a leaky bucket, then the packet is eligible right away if there are tokens in the leaky bucket, otherwise the eligibility time is set to the arrival of the next token. If the regulator is one of the more general shapers described in Chapter 3, the eligibility time is determined from (3.1). In this thesis, we focus on RCS disciplines that have regulators of the kind described in Chapter 3.

If the eligibility time of a packet is less than or equal to the current time at any output link, then we refer to it as an *eligible packet.* Since the traffic shaping is done on a per-connection basis, at any given time there arises the possibility that several packets are eligible, in which case an arbitration mechanism is required to decide the order of transmission of these eligible packets. This is where the scheduling policy comes in, and any scheduling policy can be used in conjunction with reshapers.

In general the performance of RCS disciplines depend both on the choice of the scheduling policies as well as the per-connection traffic shapers. Since the per-connection shapers decouple one scheduler from the next we can look at the performance of the scheduler at a single node in isolation. We would like to have scheduling policies that lend themselves to the provision of tight delay guarantees on a per-connection basis. In this chapter we investigate the performance of some simple scheduling in terms of the delay guarantees that can be provided to connections with pre-specified traffic envelopes. Once we have picked a scheduling policy, we are still left with the choice of traffic shaper envelopes that will be used to reshape traffic from each connection. We will see later on in this chapter that the shaper envelopes play a significant role in the performance of the RCS disciplines. In this chapter we also establish some guidelines that can be used to select appropriate shaper envelopes so that tight end-to-end delay guarantees can be provided to the connection.

4.2 Service Curves

Just as the traffic envelope characterizes the arrival process, it is possible to define a *service curve* to describe the service process. The notion of service curve to provide a lower bound on the service received by a single connection was introduced by Parekh in [37], and further generalized by Cruz in [15, 39].

Let $I_n(t)$ denote the amount of traffic input to a network element by connection n on $[0, t]$ and let $S_n(t)$ denote the amount of that connection's traffic served by the network element in the interval $[0, t]$. We set $S_n(0) = 0$ and assume both mappings $t \to I_n(t)$ and $t \to S_n(t)$ to be right continuous and non-decreasing on $\mathrm{I\!R}_+$.

Definition 4.1 *A network element is said to provide a service curve* $\overline{S}_n : \mathrm{I\!R}_+ \to \mathrm{I\!R}_+$ *to connection* n*, if at any time* $t > 0$ *there exists* s*,* $0 \leq s \leq t$*, such that* $S_n(t) - I_n(s) \geq \overline{S}_n(t - s)$*.*

Whereas the traffic envelope provides an *upper* bound on the amount of traffic that can arrive from a given connection in any interval of time, the service curve provides a *lower* bound on the amount of service provided to a connection over certain intervals of time. Of course, the lower bound on the service received is contingent on the availability of packets to serve.

With the service curve $\overline{S}_n$ defined above, the worst-case delay D_n, experienced by connection n is given by [15, 39]

$$D_n = \max_{t \geq 0} \left\{ \min_{\tau \geq 0} \left\{ \tau : \overline{I}(t) \leq \overline{S}_n(t + \tau) \right\} \right\}. \tag{4.1}$$

In other words, D_n is simply the horizontal distance between the traffic envelope and the service curve for that connection as illustrated in Figure 4.2. For many scheduling policies, the delay guarantees that can be provided to a connection can be elegantly computed using a service curve formulation, as will be seen in the following sections.

4.3 Delay bounds for different scheduling policies

In this section we look at the performance of some of the scheduling policies that were described in Chapter 2 with respect to the provision of delay guarantees. We consider a single output link

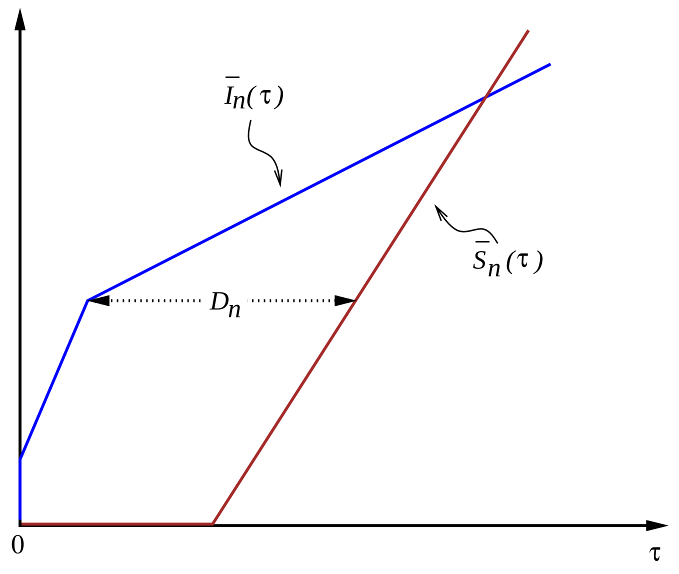

Figure 4.2: Delay bound calculation using a Service Curve

that operates at the rate r, and assume that there are N connections where connection n traffic is bounded by a traffic envelope $\overline{A}_n$, $n = 1, 2, \ldots N$. For convenience we set $\overline{A}_n(t) = 0,\ t < 0,\ n = 1, 2, \ldots N$.

4.3.1 Delay bounds for a FIFO scheduler

The FIFO scheduling policy, does not discriminate between traffic from different connections, and so they all have the same worst-case delay bounds. We can readily derive a service curve $\overline{S}^{\mathrm{FIFO}}$ that bounds from below the service provided to the aggregate traffic that is input to the FIFO scheduler.

Let $\overline{S}^{\mathrm{FIFO}}(\tau) = r\tau, \tau \geq 0$. It can be readily verified from Definition 4.1 that $\overline{S}^{\mathrm{FIFO}}$ is indeed the service curve for the aggregate traffic. A bound on the maximum packet delay D^{FIFO} in the FIFO scheduler is then obtained from (4.1) as

$$D^{\mathrm{FIFO}} = \max_{t \geq 0} \left\{ \min_{\tau \geq 0} \left\{ \tau : \sum_{n=1}^{N} A_n(t) \leq r(t + \tau) \right\} \right\}. \tag{4.2}$$

From (4.2) it is clear that a single bursty connection can result in poor delay bounds for all the connections. Thus the FIFO policy is not suitable for providing per-connection delay guarantees. In computing (4.2), we assume sufficient buffers to accommodate all the packets that may accumulate during any busy period. If the amount of buffers is actually limited to X, then the delay guarantee is

$$D^{\text{FIFO}} = \min\left\{\frac{X}{r}, \max_{t\geq 0}\left\{\min_{\tau\geq 0}\left\{\tau : \sum_{n=1}^{N} A_n(t) \leq r(t+\tau)\right\}\right\}\right\}. \qquad (4.3)$$

For a network supporting a wide variety of services, it will be necessary to dimension the buffers so that the smallest delay requirement can be met. However, a small buffer can result in a significantly high packet loss behavior, unless the link speed is comparable to the sum of the peak rates of all the connections that are multiplexed onto the link. But this will result in a severe underutilization of the link.

4.3.2 Delay bounds for Static Priority scheduler

For the Static Priority scheduler, let us assume that there are K distinct priorities, with priority 1 denoting the highest priority. The delay for the highest priority is similar to that in a FIFO scheduler, and is given by (4.3), where the summation is carried out over the top priority ($k = 1$) connections only. The delay bound for the queues with priority $k > 1$, depends only on the traffic from the priorities, $1, \ldots, k$. Also, because the priority scheduler does not distinguish between different connections with the same priority, the delay bound will be the same for all connections of the same priority.

Let $\mathcal{C}_k$ denote the set of connections with priority k, $k = 1, \ldots, K$. In any interval $[t, t+\tau]$, we know that the amount of traffic at the output link from all of the $\mathcal{C}_k$ connections is bounded above by $\sum_{n\in\mathcal{C}_k} \overline{A}_n(\tau)$. In computing the delay bound for priority k traffic we can ignore the traffic from the priority $k+1, \ldots, K$ since they will be served only in the absence of traffic from higher priorities. In the worst case, the traffic from a connection with priority k that arrives in the interval $[t, t+\tau]$, is served only after $\sum_{j=1}^{k-1} \sum_{n\in\mathcal{C}_j} \overline{A}_n(\tau)$ amount of traffic from the higher priorities are served.

Thus, if priority k traffic is back-logged in the interval $[t, t+\tau]$, it is guaranteed to receive a service of at least $\left[r\tau - \sum_{j=1}^{k-1} \sum_{n\in\mathcal{C}_j} \overline{A}_n(\tau)\right]^+$ in the same interval. In other words, the aggregate priority k traffic is guaranteed a service curve of

$$\overline{S}_k(\tau) = \left[r\tau - \sum_{j=1}^{k-1} \sum_{n\in\mathcal{C}_j} \overline{A}_n(\tau)\right]^+, \quad k = 2, \ldots K.$$

Note that the service curve $\overline{S}_k$ is guaranteed to the aggregate of all the connections in priority class k. The delay bound D_k^{PRI}, for the priority k traffic is given by the maximum horizontal distance between the traffic envelope for the aggregate traffic in class k, $\sum_{n\in\mathcal{C}_k} \overline{A}_n$, and the service curve guaranteed to class k traffic, $\overline{S}_k$, namely

$$D_k^{\mathrm{PRI}} = \max_{t\geq 0}\left\{\min_{\tau\geq 0}\left\{\tau : \sum_{n\in\mathcal{C}_k}\overline{A}_n(\tau)\leq \overline{S}_k(t+\tau)\right\}\right\}, \quad k=2,\ldots,K. \tag{4.4}$$

This bound is indeed tight as can be seen by assuming that traffic from each of the connections arrives at the maximum rate allowed by their envelopes. Unlike the FIFO policy, the Static Priority policy does allow for some level of differentiation in terms of the delay guarantees that can be provided to connections. However, addition of a connection in any priority affects the delay guarantees of all the connections with the same priority as well as those that have a lower priority.

4.3.3 Delay bounds for the GPS scheduler

Delay bounds for the GPS policy and its derivatives like Self-Clocked Fair Queueing (SCFQ) have been obtained in [37, 25]. For the single node case the delay bound for the GPS policy can be readily obtained using the service curve formulation. Recall that for the GPS policy each of the connections is assigned a weight. Let ϕ_n denote the weight that is assigned to connection n, and without loss of generality we assume that the weights are normalized with respect to the link speed, i.e., $\sum_{n=1}^N \phi_n = r$.

From the definition of GPS (see Section 2.4.5) connection n is guaranteed a service rate of at least ϕ_n. Therefore

$$S_n(\tau) = \phi_n\tau, \quad \tau\geq 0$$

is a service curve for connection n. Again from (4.1) we conclude that a bound on the delay experienced by connection n in the GPS scheduler is given by

$$D_i^{\mathrm{GPS}} = \max_{t\geq 0}\left\{\min_{\tau\geq 0}\left\{\tau : \overline{A}_n(t)\leq \overline{S}_n(t+\tau)\right\}\right\}. \tag{4.5}$$

4.3.4 Delay bounds for the EDF scheduler

In [21] it is shown that the EDF policy is delay optimal, i.e., if any policy can guarantee the delay requirements of a set of connection with specified traffic envelopes, then the EDF policy can. Thus, the EDF policy can at least provide the delay guarantees of all the policies described so far. For communication networks it is typically assumed that a non-preemptive scheduling policy is used, and so we are mainly interested in the NPEDF policy.

The operation of the NPEDF policy is defined in terms of packets, and so the delay guarantee that can be provided to a connection depends on the maximum packet size, which we denote by L. Let us denote the delay bound for connection n by D_n^{EDF}, with the connections being indexed such that $D_1^{\mathrm{EDF}} \leq D_2^{\mathrm{EDF}} \leq \cdots \leq D_N^{\mathrm{EDF}}$. Then the following condition [21],

$$\sum_{n=1}^{k} \overline{A}_n(\tau - D_n^{\mathrm{EDF}}) \leq r\tau - L, \quad \tau \geq 0,\ k = 1, \ldots, N, \tag{4.6}$$

is both necessary and sufficient for a feasible schedule to exist. The proof for the sufficiency of the above condition is fairly involved and can be found in [21]. However, its necessity can be easily explained. If all packets from connection n have been served before their deadline, then at any time τ, at least $A_n(\tau - D_n^{\mathrm{EDF}})$ amount of connection n traffic must have been served, $n = 1, \ldots, N$. However, in time τ the total amount of traffic that the link could have transmitted is $r\tau$. Therefore, it is necessary that

$$\sum_{n=1}^{N} A_n(\tau - D_n^{\mathrm{EDF}}) \leq r\tau, \quad \tau \geq 0.$$

Now, in the worst case the traffic from each connection can be exactly its traffic envelope, i.e. $A_n(\tau) = \overline{A}_n(\tau), n = 1, \ldots, N$. The reason for the additional L term in (4.6), is explained by the non-preemptive nature of the scheduling policy.

The feasibility condition (4.6) needs to be verified for all $\tau \geq 0$, which in general is a fairly daunting task. However, for most practical traffic shapers the feasibility condition can be reduced to a few checks. For instance, when the traffic envelopes are of the form $A_n(\tau) = L + \delta_n + \rho_n\tau$, the feasibility check (4.6) reduces to

$$\min\{k+1, N\}L + \sum_{n=1}^{k} \delta_n \leq D_k^{\mathrm{EDF}} \left(r - \sum_{n=1}^{k-1} \rho_n \right) + \sum_{n=1}^{k-1} \rho_n D_n^{\mathrm{EDF}},\ k = 1, \ldots, N. \tag{4.7}$$

It is simpler to check (4.7), and we will use this form of the feasibility check for the NPEDF scheduler, later on in this Chapter.

4.3.5 Example of delay guarantees with the priority, GPS and EDF schedulers

It is instructive to graphically compares the delay guarantees for EDF, GPS and priority scheduling policies on an example. Consider 3 connections with envelopes, $\overline{A}_1, \overline{A}_2$, and $\overline{A}_3$ as shown in Figure 4.3, with respect to a normalized link speed of 1. For the priority scheduler, we assume that each connection is in a separate priority class, with 1 being the highest priority. For the purpose of comparison with the EDF policy we assume the packet size L to be arbitrarily small. We focus our attention on connection 3, since the delay guarantees for connection 1 and 2 are fairly close for both the EDF and priority scheduler, and they are roughly 6 and 18 units of time, respectively. For the GPS scheduler, we need to assign weights $\phi_n, n = 1, 2, 3$, so that the required delay guarantees are met. An assignment of $\phi_1 = 0.7$, and $\phi_2 = 0.27$ will result in delay guarantees for connection 1

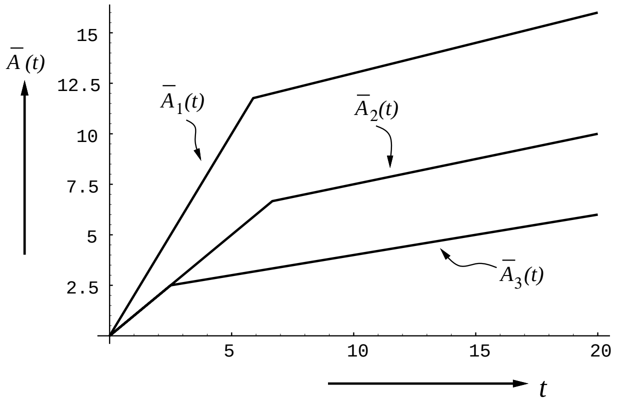

Figure 4.3: Traffic envelopes for the 3 connections.

and 2 that are close to 6 and 18 units of time respectively. This leaves connection 3 with, at best, a weight of $\phi_3 = 0.03$.

The delay guarantee for connection 3 with the priority scheduler, is obtained by first computing the service curve for priority 3 connections, and then computing the horizontal distance between the service curve and the traffic envelope for connection 3. This is shown in Figure 4.4, with the delay bound $D_3^{\mathrm{PRI}} = 36$ units of time. For the GPS scheduler, the only difference is in the computation of the service curve, which is described in Section 4.3.3. After that, the delay is obtained in a similar manner as for the priority scheduler and is illustrated in Figure 4.5. The delay bound for connection 3 with the GPS discipline is $D_3^{\mathrm{GPS}} = 31.35$ units of time. However, if an EDF scheduler is used, a delay guarantee, $D_3^{\mathrm{EDF}} = 23$, can be guaranteed to connection 3, and this can be verified from the feasibility check in (4.7). This feasibility check is also graphically illustrated in Figure 4.6.

A major drawback of the priority scheduler, is the fact that the delay guarantee for priority k does not take into account the specific values of the delay guarantees for the traffic with priorities $1, 2, \ldots, k-1$. From (4.4) or Figure 4.4 it can be seen that the delay guarantee for a connection in class k depends only on the envelopes of the traffic with priority $1, 2, \ldots, k$. Thus, if the delay requirement of some class k connection is larger than what is achievable with the traffic that is present in the higher priorities, the class $k+1$ traffic is unable to take advantage of this fact, simply because the scheduler itself does not make this distinction.

While the previous example demonstrated the ability of the EDF policy to provide small delay guarantees, it is not clear how efficient are the corresponding end-to-end delay guarantees. One of the advantages of the GPS policy, as demonstrated in [37], is that tight end-to-end delay bounds can be computed. If the EDF policy is simply used at each node along the path, then it is *not*

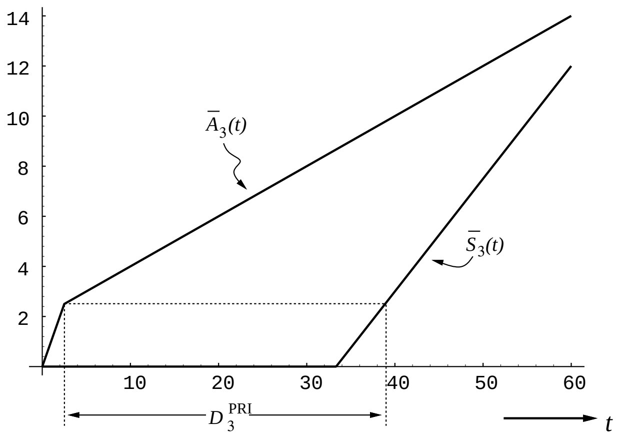

Figure 4.4: Best delay bound for connection 3, using priority scheduler. ($D_3^{\mathrm{PRI}} = 36$).

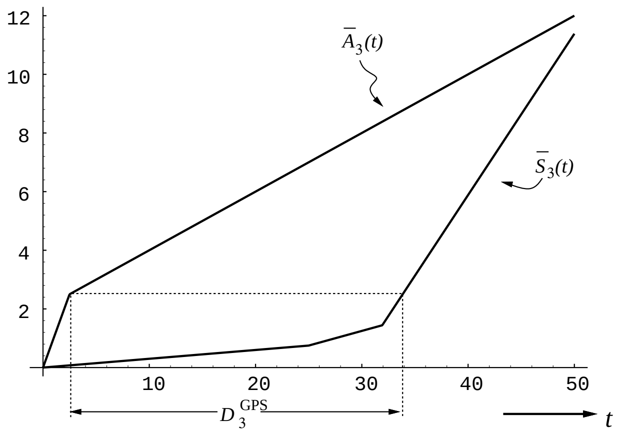

Figure 4.5: Delay bound for connection 3, using GPS scheduler. ($D_3^{\mathrm{GPS}} = 31.35$).

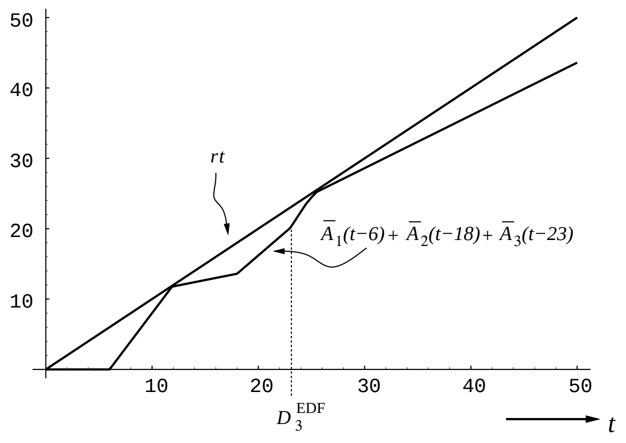

Figure 4.6: Best delay bound for connection 3, using EDF scheduler ($D_3^{\mathrm{EDF}} = 23$).

possible to compute the feasibility of the deadlines at downstream nodes, since the envelope of the traffic at the downstream node cannot be efficiently computed. However, if the EDF policy is used as the scheduler for RCS disciplines, then by virtue of the reshaping and Theorem 3.4, end-to-end delay guarantees can be computed on a per-connection basis. Furthermore, the choice of the shaper envelopes used in these RCS disciplines, greatly affects the end-to-end delays that can be guaranteed. In the next section we address some properties of traffic shapers that affect the end-to-end delay guarantees that can be provided by RCS disciplines.

Since the EDF policy can provide the same or better delay guarantees than all other policies, in the rest of this thesis, whenever we use the term RCS discipline, we assume that the EDF scheduling policy is used.

4.4 Traffic Shaper Parameters

In the previous section we considered the different scheduling policies that can be used as the scheduler for the RCS discipline and chose the EDF policy as best suited for providing delay guarantees. In this section we look at how we can choose shaper envelopes that will provide efficient end-to-end delay guarantees. The framework of Section 3.6 provides the flexibility of specifying different shaper envelopes at different nodes along the path of a connection. However, it is not clear that there is any benefit in having different shaper envelopes for the same connection at different nodes. In this section we show, that for the same connection it does not pay to have different shapers at different nodes. Recall from Section 3.6 that we use the notation $\mathcal{A}_n^m$ to denote the shaper used for connection n at the network element which is m hops away from the source. First, we make the simple but important observation.

Lemma 4.1 *Consider the link scheduler at the node m and let D_k^m denote the delay bound guaranteed to connection $k, k = 1, \ldots, N$ at node m. If for connection n, we replace the shaper $\mathcal{A}_n^m$ with $\mathcal{B}_n^m$, with $\mathcal{B}_n^m \preceq \mathcal{A}_n^m$, then the delay guarantee of D_k^m for connection $k, k = 1, \ldots, N$ still holds.*

Proof. Because $\overline{B}_n^m(\tau) \leq \overline{A}_n^m(\tau)$, $\tau \geq 0$, it follows that $\overline{A}_n^m$ is also an envelope for the traffic exiting $\mathcal{B}_n^m$. By definition, D_k^m remains an upper bound on the delay of any connection k traffic as long as connection n still has envelope $\overline{A}_n^m$. ■

Another simple observation can be made with regards to a series connection of traffic shapers.

Lemma 4.2 *Let $\mathcal{A}_1 \wedge \mathcal{A}_2$ denote the series connection of two shapers $\mathcal{A}_1$ and $\mathcal{A}_2$. Then,*

$$D\left(\overline{I} \| \mathcal{A}_1 \wedge \mathcal{A}_2\right) \leq D\left(\overline{I} \| \mathcal{A}_1\right) + D\left(\mathcal{A}_1 \| \mathcal{A}_2\right). \tag{4.8}$$

Proof. While (4.8) can be obtained by a brute force substitution in (3.17), we follow a more intuitive proof. Recall from (3.22) that $\mathcal{A}_1 \wedge \mathcal{A}_2$ is a traffic shaper. Also recall that $D\left(\mathcal{A}_1 \| \mathcal{A}_2\right)$ is a tight upper bound on the maximum delay encountered in the shaper $\mathcal{A}_1 \wedge \mathcal{A}_2$, and is achieved when the input traffic arrives at the maximum rate allowed by the traffic envelope, i.e.,

$$I(t) = \overline{I}(t), \quad t \geq 0.$$

When this maximum delay of $D\left(\mathcal{A}_1 \| \mathcal{A}_2\right)$ is experienced, let $d_{\mathcal{A}_1}$ and $d_{\mathcal{A}_2}$ denote the corresponding delay encountered in shapers $\mathcal{A}_1$ and $\mathcal{A}_2$ respectively. Therefore,

$$d_{\mathcal{A}_1} + d_{\mathcal{A}_2} = D\left(\mathcal{A}_1 \| \mathcal{A}_2\right). \tag{4.9}$$

But,

$$d_{\mathcal{A}_1} \leq D\left(\overline{I} \| \mathcal{A}_1\right), \tag{4.10}$$

and

$$d_{\mathcal{A}_2} \leq D\left(\mathcal{A}_1 \| \mathcal{A}_2\right). \tag{4.11}$$

Combining (4.9), (4.10) and (4.11) we directly obtain (4.8) and the proof is completed. ■

Theorem 4.1 *Consider connection n that traverses nodes $1, 2, \ldots M$ and let π denote an RCS discipline that uses shapers $\mathcal{A}_n^m$, and guarantees scheduler delay D_n^m, at node m, $m = 1, \ldots, M$. The RCS discipline π' that uses the same scheduling policy at all nodes as π, but with the shapers*

$$\mathcal{B}_n^m = \bigwedge_{m=1}^{M} \mathcal{A}_n^m \equiv \mathcal{B}, \quad m = 1, \ldots, M \tag{4.12}$$

can provide to all connections, the same end-to-end delay guarantees as π.

Proof. First, observe by (3.22) that

$$\mathcal{A}_n \succeq \mathcal{A}_1 \wedge \mathcal{A}_2, \quad n = 1, 2. \tag{4.13}$$

Now from (4.13) and (4.12) we have $\mathcal{A}_n^m \succeq \mathcal{B}_n^m$ and by Lemma 4.1, π' can therefore guarantee the same scheduling delays to all connections. Since the shapers remain the same for any connection $k \neq n$, it follows that for those connections, policy π' guarantees the same end-to-end delays as π. Consider next connection n, whose envelope at the first shaper is denoted by $\overline{I}_n$. Let the end-to-end delay guarantee for connection n, provided by π and π' be denoted by D_n^{π} and $D_n^{\pi'}$ respectively. From Corollary 3.2 we know that

$$D_n^{\pi} = D(\overline{I}_n \| \mathcal{A}_n^1) + \sum_{m=1}^{M-1} D\left(\mathcal{A}_n^m \| \mathcal{A}_n^{m+1}\right) + \sum_{m=1}^{M} D_n^m + \sum_{m=1}^{M} T^{(m,m+1)}. \tag{4.14}$$

Taking into account the fact that $D\ \mathcal{B}_n^m \| \mathcal{B}_n^{m+1}) = D(\mathcal{B} \| \mathcal{B}) = 0$, we conclude from (3.34) that

$$D_n^{\pi'} = D\left(\overline{I}_n \| \mathcal{B}\right) + \sum_{m=1}^{M} D_n^m + \sum_{m=1}^{M} T_n^{(m,m+1)}. \tag{4.15}$$

From (4.8) we also observe that

$$D\left(\overline{I}_n, \mathcal{B}\right) \leq D\left(\overline{I}_n \| \mathcal{A}_n^1\right) + \sum_{m=1}^{M-1} D\left(\mathcal{A}_n^m \| \mathcal{A}_n^{m+1}\right). \tag{4.16}$$

From (4.14), (4.14) and (4.14) we conclude that we conclude that $D_n^{\pi'} \leq D_n^{\pi}$. ■

According to Theorem 4.1 we can restrict our attention to disciplines with identical shapers at all nodes. In the remainder of this thesis, we consider RCS disciplines that for any given connection use *identical* shapers at each node, i.e., $\mathcal{A}_n^m = \mathcal{A}_n$. Under that assumption, the end-to-end delay guarantee for connection n becomes

$$D_n = D(\overline{I}_n \| \mathcal{A}_n) + \sum_{m=1}^{M} D_n^m + \sum_{m=1}^{M} T^{l_m}. \tag{4.17}$$

A word of caution is warranted, as one should not conclude from (4.17) that the end-to-end delay guarantees are minimized by choosing $\overline{I}_n$ as the envelope for all the traffic shapers. While this choice will certainly result in $D(\overline{I}_n \| \mathcal{A}_n) = 0$, the scheduler delay bounds (D_n^m) may increase because they depend on the choice of the traffic shapers $\mathcal{A}_n^m$. Finding the shaper envelope that is optimal, in terms of providing minimum end-to-end delay guarantees, is a complex problem for general shaper envelopes and multi-hop connections. For the simpler case where only (σ, ρ)-regulators are used as shapers, we obtain a result that says that if there are "sufficiently many" hops, then it is beneficial to shape the traffic to a smaller envelope than $\overline{I}_n$. First, we define the performance measure of interest in a more precise manner.

4.4.1 Schedulable Regions

The ability of a discipline to provide end-to-end delay guarantees to a given set of connections, is best quantified by the notion of *schedulable region.* Assume that there are N_T connections in a communication network, with the same scheduling discipline, π, operating on all the links in the network. The input traffic of connection n has envelope function $\overline{I}_n$, and traverses path P_n of the network, $n = 1, 2, \ldots, N_T$. In other words the connection n is characterized by the tuple (I_n, P_n), $n = 1, \ldots, N_T$. The vector

$$\mathbf{D} = (D_1, \ldots, D_{N_T})$$

is said to be *schedulable* under discipline π if the delay bound D_n can be guaranteed under π for all packets of connection n, $n = 1, \ldots, N_T$. The *schedulable region* of discipline π is the set of all vectors $\mathbf{D}$ that are schedulable under π. Note that the schedulable region of a service discipline depends on the envelope functions $\overline{I}_n$ and the connection paths P_n, $n = 1, \ldots, N_T$. The schedulable region is defined in terms of delay bounds that can be guaranteed *a priori*, based on the knowledge of the input traffic envelope and the connection paths. These bounds are an integral part of the service discipline and may in fact be significantly worse than the delays actually experienced by packets. From the point of view of admission control, it is irrelevant if in the actual operation of a policy smaller delays are observed, since what is required at the time of connection establishment, is to know whether the delay bounds can be *guaranteed* or not.

Let

$$\mathcal{P}_n = \{(I_n, P_n), n = 1, \ldots, N_T\}$$

denote the set of connections in the network. We say that service discipline π_1 is *at least as good as* discipline π_2, if the schedulable region of π_1 is a superset of π_2, for *any* given set of connections $\mathcal{P}$. If, in addition, the schedulable region of π_1 is a *strict* superset of π_2 we say that π_1 is *better* than π_2.

4.5 Effect of traffic shaper envelopes on the end-to-end delay guarantees

In Theorem 4.1 we saw that if we had different traffic shapers along the path of a connection we could replace all of them by the "smallest" one. Substituting a smaller traffic shaper at a node ensured that the scheduler delay guarantees at that node would still be valid. In this section we look at the more general tradeoff between shaper envelopes and the scheduler delay guarantees that can be provided at a node. A smaller shaper will in general allow the schedulers to provide tighter delay guarantees at each of the nodes along the connection's path. However, reshaping the connection traffic to a smaller envelope will incur an additional delay at the first reshaper that the connection encounters.

With a general traffic envelope as defined in Section 2.2 it is hard to make quantitative statements about this trade-off, so we focus on a simpler traffic envelope given by

$$\overline{I}_n(\tau) = L + \delta_n + \rho_n \tau, \quad \tau \geq 0, n = 1, \ldots, N_T.$$

Also we restrict the space of traffic shapers ($\mathcal{A}_n^m$) to those with envelopes of the form

$$\overline{A}_n^m(\tau) = L + \delta_n^m + \rho_n^m \tau, \quad m = 1, \ldots, M, n = 1, \ldots, N_T.$$

We wish to find out what effect, if any, the choice of the traffic shaper parameters has on the end-to-end delay that can be guaranteed to a connection.

Consider the parameters of the shaper envelope for some connection i at node m. Clearly, we cannot choose $\rho_i^m < \rho_i$ since it is possible for the source to inject traffic into the network at an average rate of ρ_i, thereby causing a net accumulation of traffic in the shaper at node m. Of course, we can have $\rho_i^m > \rho_i$, but from (4.7) it follows that this will only result in an increase in the delay guarantees that can be made to the other connections and will therefore reduce the schedulable region. Choosing $\delta_i^m > \delta_i$ will result in a similar decrease in the schedulable region.

The effects of choosing $\delta_i^m < \delta_i$ are not so clear. If we only consider node m, the other connections may benefit from the smaller envelope that we have chosen for connection n. However, in doing so, connection i has incurred a potential delay of $\frac{\delta_i - \delta_i^m}{\rho_i}$ at the first shaper along the path. For the network to be able to provide the same end-to-end delay guarantee D_i to connection i, it is necessary that some or all of the nodes along the path of connection i now provide smaller delay guarantees to connection i.

Choose $f_i^m, 0 \leq f_i^m \leq 1,\ m = 1, \ldots, M$ such that

$$\sum_{m=1}^{M} f_i^m = 1,$$

and assume that a fraction f_i^m of the shaper delay $\frac{\delta_i - \delta_i^m}{\rho_i}$ is accounted for at node m. In other words, the local delay guarantee at node m for connection i, is reduced from D_i^m to $D_i^m - \frac{f_i^m(\delta_i - \delta_i^m)}{\rho_i}$. We need to verify whether this reduction in the delay guarantee for connection i still results in a feasible schedule at node m. Without loss of generality we assume that there are a total of N_m connections at node m numbered from $1, 2, \ldots, N_m$, which are ordered such that their delay guarantees are

$$D_1^m \leq D_2^m \leq \ldots \leq D_{N_m}^m. \tag{4.18}$$

Let r^m denote the link speed at node m. From (4.7) we observe that the following inequalities need to be verified to determine if there is a feasible schedule at node m:

$$\begin{aligned}
(k+1)\,L + \sum_{n=1}^{k} \delta_n \quad &\leq \quad D_k^m \left(r^m - \sum_{n=1}^{k-1} \rho_n \right) + \sum_{n=1}^{k-1} \rho_n D_n^m, \\
&\qquad\qquad k = 1, \ldots, l-1, \qquad (4.19) \\
(l+1)\,L + \sum_{n=1}^{l-1} \delta_n + \delta_i^m \quad &\leq \quad \left(D_i^m - \frac{f_i^m(\delta_i - \delta_i^m)}{\rho_i} \right) \left(r^m - \sum_{n=1}^{l-1} \rho_n \right) \\
&\qquad + \sum_{n=1}^{l-1} \rho_n D_n^m, \qquad (4.20) \\
(k+2)\,L + \sum_{n=1}^{k} \delta_n + \delta_i^m \quad &\leq \quad D_k^m \left(r^m - \sum_{n=1}^{k-1} \rho_n - \rho_i \right) + \sum_{n=1}^{k-1} \rho_n D_n^m \\
&\qquad + \rho_i D_i^m - f_i^m(\delta_i - \delta_i^m), \;\; k = l, \ldots, i-1, \qquad (4.21) \\
\min\{k+1, N_m\}\,L + \sum_{\substack{n=1 \\ n \neq i}}^{k} \delta_n + \delta_i^m \quad &\leq \quad D_k^m \left(r^m - \sum_{n=1}^{k-1} \rho_n \right) + \sum_{n=1}^{k-1} \rho_n D_n^m \\
&\qquad - f_i^m(\delta_i - \delta_i^m), \quad k = i+1, \ldots, N_m. \qquad (4.22)
\end{aligned}$$

where the index l is chosen such that $D_{l-1}^m < D_i^m - \frac{f_i^m(\delta_i - \delta_i^m)}{\rho_i} \leq D_l^m$ and it is assumed that $i < N_m$. A few words about these inequalities will be helpful in understanding them. Note that by changing D_i^m to $D_i^m - \frac{f_i^m(\delta_i - \delta_i^m)}{\rho_i}$, we may be changing the relative ordering of the deadlines, and that is why the inequality for D_i^m is moved up to the l^{th} position. If $i = N_m$, there are a few minor changes in (4.19)-(4.22) and for the sake of brevity we assume that $i < N_m$, for the rest of the discussion; however, the results are true for $i = N_m$ as well.

Theorem 4.2 *If $f_i^m \leq \rho_i / (r^m - \sum_{n=1}^{l-1} \rho_n)$, then a feasible schedule still exists at node m.*

Proof. We need to show that any vector $(D_1^m, D_2^m, \ldots, D_{N_m}^m)$ that satisfies (4.7), also satisfies the inequalities (4.19)-(4.22). For clarity we divide the proof into specific cases based on the range of index k.

Case (i) $k = 1, \ldots, l-1$.
There is no difference between the inequalities (4.7) and (4.19) and so there is nothing to be shown here.

Case (ii) $k = l, \ldots, i-1$.
In order to verify (4.21), it is sufficient to show that the following inequality

$$\begin{aligned}
(i+1)L + \sum_{n=1}^{i-1} \delta_i + \delta_i^m \quad &\leq \quad D_k^m \left(r^m - \sum_{n=1}^{k-1} \rho_n - \rho_i \right) + \sum_{n=1}^{k-1} \rho_n D_n^m + \rho_i D_i^m \\
&\qquad - f_i^m(\delta_i - \delta_i^m) \qquad (4.23)
\end{aligned}$$

holds for $k = l \ldots, i-1$. To show that (4.23) is true for $k = l$, consider (4.30), i.e.,

$$\begin{aligned}
&(i+1)L + \sum_{n=1}^{i-1} \delta_n + \delta_i^m \\
&\quad \leq \left(D_n^m - \frac{f_i^m(\delta_i - \delta_i^m)}{\rho_n} \right) \left(r^m - \sum_{n=1}^{l-1} \rho_n \right) + \sum_{n=1}^{l-1} \rho_n D_n^m \\
&\quad = \left(D_n^m - \frac{f_i^m(\delta_i - \delta_i^m)}{\rho_i} \right) \left(r^m - \sum_{n=1}^{l-1} \rho_n - \rho_i \right) + D_i^m \rho_i - f_i^m(\delta_i - \delta_i^m) \\
&\qquad\qquad + \sum_{n=1}^{l-1} \rho_n D_n^m \\
&\quad \leq D_l^m \left(r^m - \sum_{n=1}^{l-1} \rho_n - \rho_i \right) + \sum_{n=1}^{l-1} \rho_n D_n^m + D_i^m \rho_i - f_i^m(\delta_i - \delta_i^m).
\end{aligned}$$

We proceed by induction on $k = l, \ldots, i-2$. Assume that

$$\begin{aligned}
(i+1)L + \sum_{n=1}^{i-1} \delta_i + \delta_i^m \quad \leq \quad & D_j^m \left(r^m - \sum_{n=1}^{j-1} \rho_n - \rho_i \right) + \sum_{n=1}^{j-1} \rho_n D_n^m + \rho_i D_i^m \\
& - f_i^m(\delta_i - \delta_i^m) \quad k = l, \ldots, j, \qquad (4.24)
\end{aligned}$$

holds for some $j = l, \ldots, i-2$. Then,

$$\begin{aligned}
&(i+1)L + \sum_{n=1}^{i-1} \delta_n + \delta_i^m \\
&\quad \leq D_j^m \left(r^m - \sum_{n=1}^{j-1} \rho_n - \rho_i \right) + \sum_{n=1}^{j-1} \rho_n D_n^m + \rho_i D_i^m - f_i^m(\delta_i - \delta_i^m) \\
&\quad = D_j^m \left(r^m - \sum_{n=1}^{j} \rho_n - \rho_i \right) + \sum_{n=1}^{j} \rho_n D_n^m + \rho_i D_i^m - f_i^m(\delta_i - \delta_i^m) \\
&\quad \leq D_{j+1}^m \left(r^m - \sum_{n=1}^{j} \rho_n - \rho_i \right) + \sum_{n=1}^{j} \rho_n D_n^m + \rho_i D_i^m - f_i^m(\delta_i - \delta_i^m), \qquad (4.25)
\end{aligned}$$

where (4.25) follows from (4.18). Hence, the induction hypothesis (4.24) holds for $k = j+1$ and the induction step is completed.

Case (iii) $k = i$.

To verify (4.20), we start with $k = i$ in (4.7), namely

$$(i+1)L + \sum_{n=1}^{i} \delta_i \leq D_i^m \left(r^m - \sum_{n=1}^{i-1} \rho_n \right) + \sum_{n=1}^{i-1} \rho_n D_n^m. \qquad (4.26)$$

Adding $\delta_i^m - \delta_i$ to both sides we obtain,

$$\begin{aligned}
(i+1)L + \textstyle\sum_{n=1}^{i-1} \delta_n + \delta_i^m & \\
\leq \; & D_n^m \left(r^m - \sum_{n=1}^{i-1} \rho_n \right) + \sum_{n=1}^{i-1} \rho_n D_n^m - (\delta_i - \delta_i^m) && (4.27) \\
\leq \; & D_i^m \left(r^m - \sum_{n=1}^{i-1} \rho_n \right) + \sum_{n=1}^{l-1} \rho_n D_n^m + \sum_{n=l}^{i-1} \rho_n D_i^m \\
& \qquad - (\delta_i - \delta_i^m) && (4.28) \\
= \; & D_i^m \left(r^m - \sum_{n=1}^{l-1} \rho_n \right) - (\delta_i - \delta_i^m) + \sum_{n=1}^{l-1} \rho_n D_n^m \\
= \; & \left(D_i^m - \frac{(\delta_i - \delta_i^m)}{r^m - \sum_{n=1}^{l-1} \rho_n} \right) \left(r^m - \sum_{n=1}^{l-1} \rho_n \right) + \sum_{n=1}^{l-1} \rho_n D_n^m && (4.29) \\
\leq \; & \left(D_i^m - \frac{f_i^m(\delta_i - \delta_i^m)}{\rho_i} \right) \left(r^m - \sum_{n=1}^{l-1} \rho_n \right) + \sum_{n=1}^{l-1} \rho_n D_n^m && (4.30)
\end{aligned}$$

Inequality (4.28) follows from (4.27) and (4.18), and (4.30) follows from (4.29) because

$$f_i^m \leq \frac{\rho_i}{r^m - \sum_{n=1}^{l-1} \rho_n}.$$

Since the deadline of connection i was reduced, we must have $l \leq i$ and so (4.20) follows directly from (4.30).

Case (iv) $k = i+1, \ldots, N_m$.
Since $0 \leq f_n^m \leq 1$, (4.22) follows directly from (4.7). ■

It is possible for bursts to be completely smoothed out of the traffic by the first hop traffic shaper, introducing an up front reshaping delay. However, this additional delay can be amortized over the nodes traversed by the connection since the smoother traffic will enable the schedulers to provide better delay guarantees. If there are sufficient number of hops to amortize the reshaping delay then efficient end-to-end delay guarantees can still be provided and Theorem 4.3 quantifies the number of hops that are required.

Theorem 4.3 *If* $\sum_{m=1}^{M} \rho_i / r^m \geq 1$, *the schedulable region of the RCS discipline* π *is not reduced if the shapers for connection* i *are restricted to ones with envelopes such that* $\delta_i^m = 0$, $m = 1, 2, \ldots M$.

Proof. Assume that the vector $(D_1, D_2, \ldots, D_{N_T})$ is schedulable under π. Now consider connection i and assume that it has shapers $\mathcal{A}_i^m$ with envelopes

$$\overline{A}_i^m(\tau) = L + \delta_i^m + \rho_i \tau, \quad \tau \geq 0, \; m = 1, \ldots, M,$$

and is provided the end-to-end delay guarantee of D_i. Let D_i^m denote the delay guarantees for connection i at the individual nodes along its path. Hence,

$$D_i = \sum_{m=1}^{M} D_i^m.$$

Now, let

$$\overline{A}_i^*(\tau) = L + \rho_i \tau, \quad \tau \geq 0$$

and assume that the shapers for connection i are modified such that they have the same envelope $\overline{A}_i^*$ at each node along the path. Then, in the worst case connection i may encounter an additional delay of $\frac{\delta_i}{\rho_i}$ at the first node. Using Theorem 4.2 we can amortize this extra delay over the nodes along the path of connection i.

In particular we know from Theorem 4.2 that node m can now provide connection i with the delay guarantee of $D_i^m - \frac{\rho_i}{r^m}\left(\frac{\delta_i}{\rho_i}\right)$, $m = 1, \ldots, M$. Hence, the end-to-end delay guarantee that can be provided to connection i is

$$\begin{aligned} \frac{\delta_i}{\rho_i} + \sum_{m=1}^{M}\left(D_i^m - \frac{\rho_i}{r^m}\left(\frac{\delta_i}{\rho_i}\right)\right) &= D_i + \frac{\delta_i}{\rho_i}\left(1 - \sum_{m=1}^{M}\frac{\rho_i}{r^m}\right) \\ &\leq D_i. \end{aligned}$$

■

4.6 Summary

Rate Controlled Service disciplines are mainly composed of a Traffic Shaper and a scheduler. In this chapter we investigate each of these components in turn. First we consider a few different scheduling policies and examine their performance in terms of providing end-to-end delay bounds. We find the EDF policy to be best suited for providing tight delay guarantees and consider it to be the scheduler of choice for RCS disciplines.

In the second part of this chapter we consider the implications of having different traffic shaper envelopes at each of the nodes along the path of a connection. It turns out that it is better to have the same traffic shaper for the connection at all the nodes that it traverses.

Previously we examined the two components of RCS disciplines – the shapers and the schedulers – in isolation. Clearly there is some interrelation between the two, but in general, it is difficult to account for both of them in computing efficient end-to-end delay guarantees. We were able to obtain some quantitative results by narrowing the problem space and considering traffic envelopes of the form $\overline{A}(\tau) = \sigma + \rho\tau$, $\tau \geq 0$. We found that if the connection traverses a "sufficient" number of hops it is advantageous to smooth out all the bursts before traffic is admitted into the network.

Chapter 5

Comparison with GPS

The GPS discipline has received a lot of attention in the recent literature. Part of the reason for this is the fact that tight end-to-end delay bounds have been obtained for the GPS discipline. Thus, it has been argued that the GPS discipline is the scheduling discipline of choice for the provision of end-to-end delay guarantees in packet networks. In the next section, we demonstrate how with the choice of suitable shaper envelopes, an RCS discipline can provide the same end-to-end delay bounds as a GPS discipline. In addition, we show that the RCS discipline can accept connections with delay requirements that cannot be accepted by GPS. This demonstrates the advantage of RCS over GPS in providing efficient end-to-end delay guarantees, as well as provides some insight into the choice of traffic shapers for the provision of efficient end-to-end delay guarantees.

In this chapter, we assume for comparison purposes that the traffic of connection n entering the first node packetizer has envelope $\overline{U}_n(\tau) = \delta_n + \rho_n\tau$. Therefore, the envelope of the traffic that enters the first traffic shaper is $\overline{I}_n(\tau) = L + \delta_n + \rho_n\tau$. We also assume that connection n traverses nodes $m = 1, 2, \ldots, M$ and that the propagation delays are all zero. For definitions and notation relating to GPS we refer the reader to [35, 36]. Let $C^{m,l}$ denote the set of connections that pass through the output link l of node m. Denoting the speed of this link as $r^{m,l}$, we will assume the stability condition

$$\sum_{n \in C^{m,l}} \rho_n \leq r^{m,l}$$

throughout the remainder of this section .

5.1 Achieving GPS Delay Guarantees (Simple Case)

The GPS policy operates by allocating weight ϕ_n^m for connection n at node m. These weights are used to determine the rate at which traffic from connection n is served when a set $B^{m,l}$ of connections is back-logged at the output link l of node m through which connection n passes.

Specifically, the service rate of connection n is given by

$$g_n^m = \frac{\phi_n^m}{\sum_{k \in B^{m,l}} \phi_k^m} r^m,$$

where for simplicity in notation we denote $r^{m,l}$ as r^m when there is no possibility of confusion. PGPS is a non-preemptive policy that tracks GPS. In general the procedure developed in [35] to obtain delay bounds given the weights ϕ_n^m is complicated and imposes certain restrictions on these weights. Moreover, the practically more important inverse procedure of specifying appropriate weights, that satisfy predetermined delay bounds, is even more cumbersome. However, a simple bound can be obtained in the special case of non-preemptive PGPS, where $\phi_n^m = \rho_n$ at all nodes through which the connection passes. Since each connection is served in proportion to this weight at each node along its path, the service discipline is sometimes called Rate Proportional Processor Sharing (RPPS) discipline. Specifically, the end-to-end delay bound D_n^* obtained under non-preemptive RPPS is given [36, 26] by

$$D_n^* = \frac{\delta_n + ML}{\rho_n} + \sum_{m=1}^{M} \frac{L}{r^m}. \tag{5.1}$$

From formula (5.1) we can already see the weakness of the RCS disciplines relative to RPPS, if the traffic shapers for connection n at every node have envelopes that are identical to the input envelope $\overline{I}_n$. In this case $D(\overline{I} \| \mathcal{A}_n) = 0$ and since propagation delays are assumed zero, from (4.17) we obtain

$$D_n = \sum_{m=1}^{M} D_n^m.$$

At node m, even if the entire link bandwidth of r^m was somehow dedicated to connection n, the scheduler delay bound D_n^m is no less than $(\delta_n + L)/r^m$, and the end-to-end delay bound guaranteed by the RCS discipline satisfies the inequality

$$D_n \geq \sum_{m=1}^{M} \frac{\delta_n}{r^m} + \sum_{m=1}^{M} \frac{L}{r^m}.$$

Since δ_n can be much larger than L, the bounds provided by the RCS discipline under the scenario considered here can be much worse than those obtained under RPPS. For example, assume that all the link speeds are the same, i.e., $r^m = r,\ m = 1, \ldots, M$. If $\delta_n = 50L$ and $\rho_n = 0.8r$, we have

$$\frac{D_n}{D_n^*} \geq \frac{40.8 \times M}{50 + 1.8 \times M}.$$

Therefore, when $M = 2$ we already have $D_n/D_n^* \geq 1.52$, and for large M, $D_n/D_n^* \geq 22.67$. This discrepancy is due to the fact that the bounds for RPPS take into account delay dependencies at the various nodes, while the bounds for the RCS disciplines are based on independently summing the worst case bounds at each node.

The previous example notwithstanding, we show next that we can design RCS disciplines that provide the same delay guarantees as RPPS by employing traffic shapers with envelopes that are, in general, different from that of the input traffic.

We design the RCS discipline π as follows. For each link we use the NPEDF scheduling policy. We choose the same traffic shaper $\mathcal{A}_n$ for connection n at each node along its path with envelope given by

$$\overline{A}_n(\tau) = L + \rho_n \tau, \quad \tau \geq 0.$$

Assume that connection n is routed through output link l at node m and let r^m denote the speed of this link. For connection n, we specify the delay bounds at node m for the NPEDF scheduling policy as

$$D_n^m = L/\rho_n + L/r^m. \tag{5.2}$$

As we now show, these bounds can be guaranteed by the NPEDF policy at every node. Consider output link l at node m. Denote by N the total number of connections multiplexed on this link, and index the connections by $i_1, i_2, \ldots, i_N$ such that $D_{i_1}^m \leq D_{i_2}^m \leq \ldots \leq D_{i_N}^m$. We only need to verify (4.7). Using (5.2) we have

$$\begin{aligned} D_{i_k}^m \left(r^m - \sum_{n=1}^{k-1} \rho_{i_n} \right) + \sum_{n=1}^{k-1} \rho_{i_n} D_{i_n}^m &= L\frac{r_m - \sum_{n=1}^{k-1} \rho_{i_n}}{\rho_{i_k}} + L\frac{r^m - \sum_{n=1}^{k-1} \rho_{i_n}}{r^m} \\ &\quad + (k-1)L + L\frac{\sum_{n=1}^{k-1} \rho_{i_n}}{r^m} \\ &= L\frac{r^m - \sum_{n=1}^{k-1} \rho_{i_n}}{\rho_{i_k}} + kL \\ &\geq (k+1)L, \end{aligned}$$

where the last inequality follows from the stability condition $\sum_{n=1}^{N} \rho_n \leq r^m$. By design, the traffic shapers have $\delta_n^m = 0$ and therefore (4.7) is verified.

We now proceed to derive the end-to-end delay bounds for the connections. Recall that we have assumed zero propagation delays, so from (4.17) we obtain

$$D_n = D\left(\overline{I}_n \| \mathcal{A}_n\right) + \sum_{m=1}^{M} D_n^m.$$

For the delay $D\left(\overline{I}_n \| \mathcal{A}_n\right)$, using (3.15), we have

$$D\left(\overline{I}_n \| \mathcal{A}_n\right) = \max_{\tau \geq 0} \left\{ \frac{\overline{I}_n(\tau) - L - \rho_n \tau}{\rho_n} \right\} = \frac{\delta_n}{\rho_n}.$$

Therefore, taking into account (5.2) we now obtain

$$\begin{aligned} D_n &= \frac{\delta_n}{\rho_n} + \sum_{m=1}^{M} \frac{L}{\rho_n} + \sum_{m=1}^{M} \frac{L}{r^m} \\ &= \frac{\delta_n + ML}{\rho_n} + \sum_{m=1}^{M} \frac{L}{r^m}. \end{aligned} \tag{5.3}$$

Since (5.1) is identical to (5.3) we see that the proposed RCS discipline π can guarantee the same end-to-end delays as RPPS.

From the previous argument we see that if the delay bounds in (5.1) are required by the connections in the network, then the RCS discipline π can be used. It provides the flexibility of easily specifying other delay bounds, whereas the bounds in RPPS are tied to the rate ρ_n of a connection. In addition, since reshaping is performed at each node, buffer requirements will typically be lower than those of RPPS as will be seen in Chapter 6.

If the end-to-end delay requirements of connection n are smaller than (5.1), a slightly more general version of RPPS can be used. Rather than providing a rate of ρ_n to connection n, better delay performance can be obtained by giving it a rate of $g_n \geq \rho_n$ at each node. The end-to-end delay bound is then given by

$$D_n^* = \frac{\delta_n + ML}{g_n} + \sum_{m=1}^{M} \frac{L}{r^m}. \tag{5.4}$$

The previous analysis still applies with very little modification and can be used to specify an RCS discipline that guarantees the bounds in (5.4). In this case, all traffic shapers have envelopes $\overline{A}_n^m(\tau) = L + g_n\tau$ and the delay guarantees at the scheduler of node m are

$$D_n^m = L/g_n + L/r^m.$$

The intuition behind choosing traffic shapers of this kind is as follows. If the RPPS discipline guarantees a clearing rate of g_n to connection n, then somewhere along the path, say at node m, the connection n may only receive a service rate of g_n. This congested link behaves like a traffic shaper that has an envelope of $\overline{A}_n^m(\tau) = L + g_n\tau$. Based on Theorem 4.1, we know that for an RCS discipline it is beneficial to choose the "smallest" shaper at all the nodes, so that they can all take advantage of the smaller traffic envelope. Since in RPPS the smallest rate that a connection can be given at any node is g_n, a natural choice for the shaper envelopes of the RCS discipline is then $L + g_n\tau$.

In addition to being able to provide the same bounds as RPPS, the RCS discipline also has the advantage of allowing additional connections to be accepted, albeit with looser delay requirements. Specifically, the schedulability check for RPPS is now $\sum_{l \in C_n^m} g_l \leq r^m$, $m = 1, \ldots, M$, where C_n^m

denotes the set of connections that are multiplexed on the same link as connection n at node m. This implies that some amount of bandwidth, namely $r^m - \sum_{l \in C_n^m} \rho_l$, cannot be utilized by RPPS. However, this bandwidth can be used by an RCS discipline to accept additional connections that require relatively larger end-to-end delay guarantees. At the end of this chapter we provide a specific example of this benefit of RCS disciplines over the more general GPS disciplines.

5.2 Achieving GPS Delay Guarantees

In [36, Section VIII], tight bounds on per connection packet delays are developed for GPS under a fairly general assignment of weights, ϕ_n^m, called Consistent Relative Session Treatment (CRST). These bounds are achieved in certain node configurations, and even in the special case of RPPS, they can be much tighter than those provided by (5.1). However, the calculation of the bounds is much more cumbersome as they take into account the effect of all the other connections along a connection's path. We will show that even with these tight bounds, an RCS discipline can be designed that guarantees the same end-to-end delay bounds.

To simplify the discussion and to avoid obscuring the main idea of the argument, we assume a continuous flow model. As far as the design of traffic shapers is concerned, this basically amounts to setting $L = 0$. Before proceeding with the design of the RCS discipline, recall the delay bound for the GPS discipline provided in Section 4.3.3. First, the service curve, $\overline{S}_n$ was obtained in terms of the simpler traffic envelopes and then the delay bound in (4.5) was obtained in terms of this service curve. As stated in [35] for the GPS discipline the service curve $\overline{S}_n(t)$ is a piece-wise linear function, convex in the range $[0, t^B]$ where t^B is the end of the first busy period of connection n, when all the N connections are greedy. In this range, $\overline{S}_n$ is characterized by the pairs (s_k, b_k), $k = 1, \ldots, k$, where s_k is the slope of the kth segment, b_k its duration, and k_n is the number of line segments in $\overline{S}_n$. By the convexity of $\overline{S}_n$ we have that

$$s_1 \leq s_2 \leq \ldots \leq s_{k_n}.$$

For our purposes, the case where $\overline{A}_n(\tau) = \min\{c_n\tau, \delta_n + \rho_n\tau\}$, with $c_n \geq \rho_n$ and $\delta_n \leq \delta_n^*$ will be of interest. For convenience, we summarize in the next lemma two specific cases of (4.5) that are useful in the remainder of this section.

Lemma 5.1 *Assume that N connections are multiplexed on a link that is using the GPS scheduling policy, with connection n having the envelope $\overline{A}_n(\tau) \leq \delta_n^* + \rho_n^*\tau$, $\tau \geq 0$, $n = 1, 2, \ldots N$. Furthermore, assume that $\overline{A}_n(\tau) = \min\{c_n\tau, \delta_n + \rho_n\tau\}$, $\tau \geq 0$.*

1. *If $s_1 \geq c_n$, then $D_n^* = 0$.*

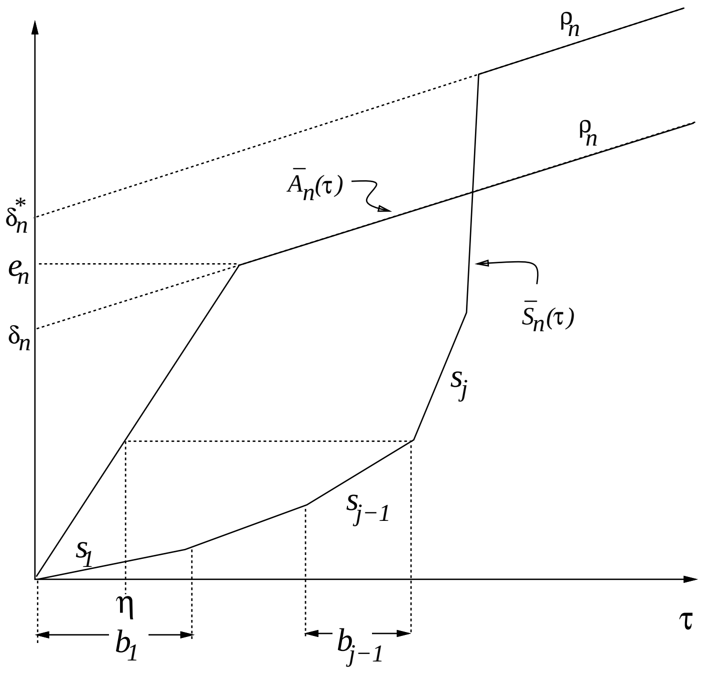

Figure 5.1: Delay bound for the GPS scheduling policy

2. *If* $s_k < c_n$, $k = 1, \ldots, j-1$, $s_j \geq c_n$, *and*

$$\overline{S}_n\left(\sum_{k=1}^{j-1} b_k\right) \leq e_n := (c_n\delta_n)/(c_n - \rho_n),$$

then $D_n^* = \sum_{k=1}^{j-1} b_k - \eta$, *where* $\eta = \overline{S}_n\left(\sum_{k=1}^{j-1} b_k\right)\Big/ c_n$.

Proof. The first part of the lemma follows by observing that $s_1 \geq c_n$ implies $\overline{A}_n(\tau) \leq \overline{S}_n(\tau)$ and therefore

$$\min_{t \geq \tau}\left\{t : \overline{S}_n(t) \geq \overline{A}_n(\tau)\right\} = \tau.$$

A geometric interpretation of the second part is given in Figure 5.1. ■

The development of GPS bounds for connection n is based on the Universal Service Curve (USC) for that connection [36, Section VIII]. Just as $\overline{S}_n$ characterizes the service that connection n receives at a single node, the USC of a connection characterizes the end-to-end service that it receives. We summarize here the method by which the USC is obtained when all the nodes use a GPS discipline [36] .

1. Under a CRST weight assignment, an algorithm is developed by which an envelope function, $\delta_n^m + \rho_n\tau$, is guaranteed for every connection n traffic entering node m [36, page 142]. For

our purposes, it is important to note that

$$\delta_n^1 = \delta_n, \quad \delta_n^m \geq \delta_n, \quad m = 2, \ldots, M. \tag{5.5}$$

2. Given envelope functions of the form $\delta_\nu^m + \rho_\nu \tau$, for any connection ν that is multiplexed with connection n at node m (ν and n are on the same output link at node m), the service function $\overline{S}_n^m(\tau)$ for connection n, is calculated. Let (s_k^m, b_k^m), $k = 1, \ldots, k_n^m$, be the set of slopes that characterize $\overline{S}_n^m(\tau)$.

3. The USC $\widehat{S}_n(\tau)$ for connection n is given by the formula

$$\widehat{S}_n(\tau) = \min\left\{G_n^M(\tau), \overline{I}(\tau)\right\}, \tau \geq 0,$$

   where $G_n^M(\tau) = \infty$ for $\tau > \sum_{m=1}^M \sum_{k=1}^{k_n^m} b_k^m$, and for $\tau \leq \sum_{m=1}^M \sum_{k=1}^{k_n^m} b_k^m$, $G_n^M(\tau)$ is composed of the segments (s_k^m, b_k^m), $m = 1, \ldots, M$, $k = 1, \ldots, k_n^m$ of $\overline{S}_n^m(\tau)$, arranged in a nondecreasing order of slopes [36, page 144]. We denote by $(\widehat{s}_k, \widehat{b}_k)$, $k = 1, \ldots, \sum_{m=1}^M k_n^m$ this nondecreasing order.

Let k_q be such that $\widehat{s}_{k_q} \geq \rho_n$, and

$$\widehat{s}_k < \rho_n, \quad k = 1, \ldots, k_q - 1. \tag{5.6}$$

We are now ready to design an RCS discipline that is at least as good as GPS. Consider first the design of traffic shapers. Recall from the beginning of this chapter that for the purpose of comparison with GPS we assume that the envelope of connection n traffic entering the first traffic shaper is of the form $\overline{I}_n(\tau) = \delta_n + \rho_n\tau$ ($L = 0$). For connection n, at each node m on the path, we choose traffic shapers that have the same envelope, i.e., $\overline{A}_n^m(\tau) = \min\{c_n\tau,\ \delta_n + \rho_n\tau\}$. To specify how the parameter c_n is picked, we need to distinguish between two classes of connections.

1. **Class (a).** Connection n belongs to this class when

$$\widehat{S}_n\left(\sum_{k=1}^{k_q-1} \widehat{b}_k\right) < \delta_n, \tag{5.7}$$

   where the USC $\widehat{S}_n$ is defined as above. In this case, the delay bound for connection n traffic under GPS is given by the solution of the equation [37, p. 136] (see Figure 5.2.i),

$$D_n^* : \widehat{S}(D_n^*) = \delta_n.$$

   Let $k^* \geq k_q$, be the index of the slope of the USC at time D_n^*. If at time D_n^* there is a change in slope, then define k^* as the index of the smaller of the two slopes (in fact either slope would work). We set $c_n = \widehat{s}_{k^*}$.

2. **Class (b).** Connection n belongs to this class when

$$\widehat{S}_n\left(\sum_{k=1}^{k_q-1}\widehat{b}_k\right) \geq \delta_n,$$

in which case, the delay bound for connection n traffic under GPS [37, p. 136] (see Figure 5.3.i) is given by

$$D_n^* = \sum_{k=1}^{k_q-1}\widehat{b}_k - \frac{\widehat{S}_n\left(\sum_{k=1}^{k_q-1}\widehat{b}_k\right) - \delta_n}{\rho_n}.$$

We then set $c_n = \rho_n$.

For connection n, we assign the scheduler delay at node m, D_n^m, to be equal to the maximum delay that would be experienced by the connection under the GPS scheduling policy at that node, when the conditions of Lemma 5.1 are satisfied. The next theorem establishes that with this assignment the resulting in an RCS discipline provides the same end-to-end delay bounds as GPS.

Theorem 5.1 *Consider connection n and let the nodes traversed by this connection be numbered $1, \ldots, M$. The EDF scheduler is employed throughout the network and for connection n at node m we choose a traffic shaper with envelope*

$$\overline{A}_n^m(\tau) = \min\{c_n\tau, \delta_n + \rho_n\tau\}, \quad \tau \geq 0,$$

and assign a delay guarantee D_n^m, $m = 1, \ldots, M$, as follows:

- *If $s_1^m \geq c_n$, then set $D_n^m = 0$.*
- *If $s_k^m < c_n$, $k = 1, \ldots, j_m - 1$, $s_{j_m}^m \geq c_n$, then assign*

$$D_n^m = \sum_{k=1}^{j_m-1} b_k^m - \eta, \text{ where } \eta = S_n^m\left(\sum_{k=1}^{j_m-1} b_k^m\right)\Bigg/ c_n.$$

Then the resulting RCS discipline provides the same delay bounds as GPS to all the connections.

Proof. The proof is a direct consequence of Lemmas 5.2 and 5.3. In Lemma 5.2 we establish that the specified delays can be guaranteed by the EDF policy at each node. Instead of verifying (4.6) to ensure a feasible schedule, it is simpler to argue indirectly as follows: we show that the specified delays are guaranteed when the RCS discipline uses GPS as the scheduling policy at each node. Since EDF is better than GPS in the single node case, it follows that the same delay guarantees can at a minimum be provided when the EDF scheduling policy is employed.

In Lemma 5.3 we establish that the end-to-end delay guarantee of the RCS discipline as given by (4.17), does not exceed D_n^*, the end-to-end delay bound for GPS obtained in Lemma 5.1. ■

Lemma 5.2 *The delay assignment in Theorem 5.1 results in a feasible schedule at each node.*

Proof. According to (5.5), we have $\overline{A}_v(\tau) \leq \delta_v^m + \rho_v \tau$, $\tau \geq 0$, for any connection v that is multiplexed with connection n on the same output link of node m. It is also true that $c_n \geq \rho_n$. This follows by definition for a connection in class (b). For a connection in class (a), observe that because of (5.6) and the fact that $\widehat{s}_k$, $k = 1, 2, \ldots$, is nondecreasing we have $c_n = \widehat{s}_{k^*} \geq \widehat{s}_{k_q} \geq \rho_n$. Applying Lemma 5.1 (where we replace $\delta^* \leftarrow \delta_n^m$), we conclude that the delay $D_n^m = 0$ can be guaranteed under the GPS policy for any node m for which $s_1^m \geq c_n$. For a node m, where $s_k^m < c_n$, $k = 1, \ldots, j_m - 1$, $s_{j_m}^m \geq c_n$, we apply part 2 of Lemma 5.1 and, therefore, we first need to show that

$$S_n^m \left(\sum_{k=1}^{j_m - 1} b_k^m \right) \leq \frac{c_n \delta_n}{c_n - \rho_n}.$$

This is trivially true for a connection in class (b) since $c_n \delta_n / (c_n - \rho_n) = \infty$. If connection n belongs to class (a), observe that from the definition of $\widehat{s}_{k^*}$, j_m and $\widehat{S}_n(\tau)$, we have (see Figure 5.2),

$$\begin{aligned} S_n^m \left(\sum_{k=1}^{j_m - 1} b_k^m \right) &\leq \widehat{S}_n \left(\sum_{k=1}^{k^*-1} \widehat{b}_k \right) \\ &\leq \delta_n \\ &\leq \frac{\widehat{s}_{k^*} \delta_n}{\widehat{s}_{k^*} - \rho_n}. \end{aligned}$$

■

Thus, we have established that in both cases (a) and (b), the specified delay bound can be guaranteed at node m.

Lemma 5.3 *The end-to-end delay guarantee of the RCS discipline as given by (4.17), does not exceed the end-to-end delay bound D_n^*, for GPS.*

Proof. Recall that the input traffic envelope $\overline{I}_n$ for connection n is given by

$$\overline{I}_n(\tau) = \delta_n + \rho_n \tau, \quad \tau \geq 0.$$

Therefore from (3.17), we obtain

$$D\left(\overline{I}_n \| \mathcal{A}_n\right) = \frac{\delta_n}{c_n}.$$

Therefore, it suffices to show that

$$\frac{\delta_n}{c_n} + \sum_{m=1}^{M} D_n^m = D_n^*, \tag{5.8}$$

where D_n^* is obtained from Lemma 5.1.

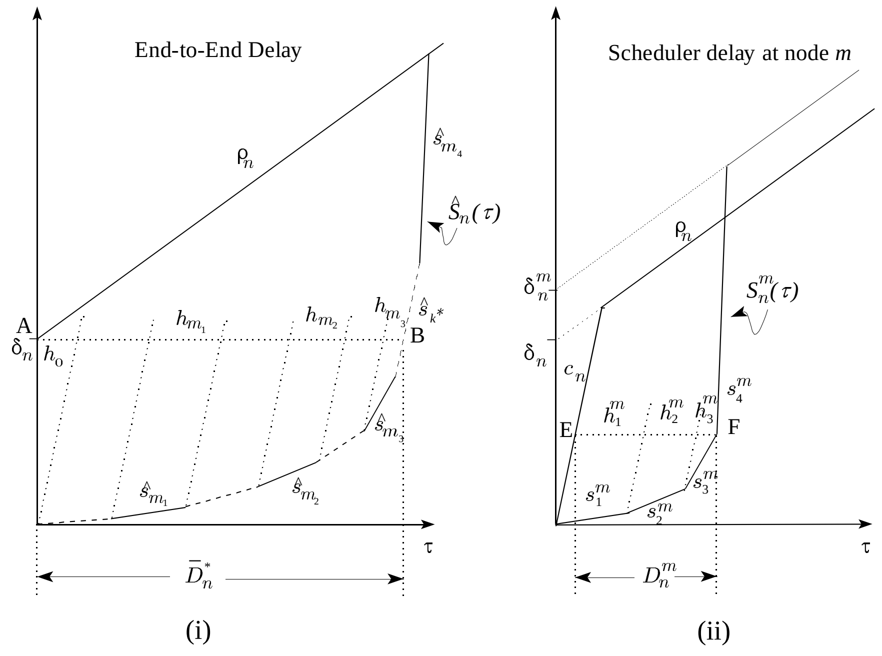

Figure 5.2: Delay Decomposition of a Class (a) Connection

Let M_0 be the set of nodes for which $D_n^m > 0$, so that $\sum_{m=1}^{M} D_n^m = \sum_{m\in M_0} D_n^m$. Assume first that connection n belongs to class (a). Observe that the set of slopes $\hat{s}_k$, $k = 1,\ldots,k^*-1$, can be partitioned into subsets F_m for each index m in M_0, where

$$F_m = \{\hat{s}_l : \hat{s}_l = s_k^m, \ \text{ for some } k = 1,\ldots,j_m - 1\}.$$

We denote by m_k the index l for which $\hat{s}_l = s_k^m$, i.e., $\hat{s}_{m_k} = s_k^m$. For the rest of the discussion, it is best to use geometric arguments. Referring to Figure 5.2(i), draw lines with slope $\hat{s}_{k^*}$ from all the points in $\hat{S}_n(\tau)$ where the slope changes and remains less than $\hat{s}_{k^*}$. These lines intersect segment AB (corresponding to the delay D_n^*) and divide it into segments of length h_k, $0 \le k \le k^*-1$, where segment h_k corresponds to slope $\hat{s}_k$, $1 \le k \le k^*-1$. Denote by h_{m_k} the segment that corresponds to $\hat{s}_{m_k}$. Since by construction $h_0 = \delta_n / c_n$, we then have

$$D_n^* = \frac{\delta_n}{c_n} + \sum_{m\in M_0} \sum_{k=1}^{j_m-1} h_{m_k}. \tag{5.9}$$

Similarly, in Figure 5.2(ii), draw lines with slope $\hat{s}_{k^*}$ from all the points in $S_n^m(\tau)$ where the slope changes and remains less than $\hat{s}_{k^*}$. These lines intersect segment EF (corresponding to the delay D_n^m) and divide it into segments h_k^m, $1 \le k \le j_m - 1$ (in the figure we have $j_m - 1 = 3$). We can

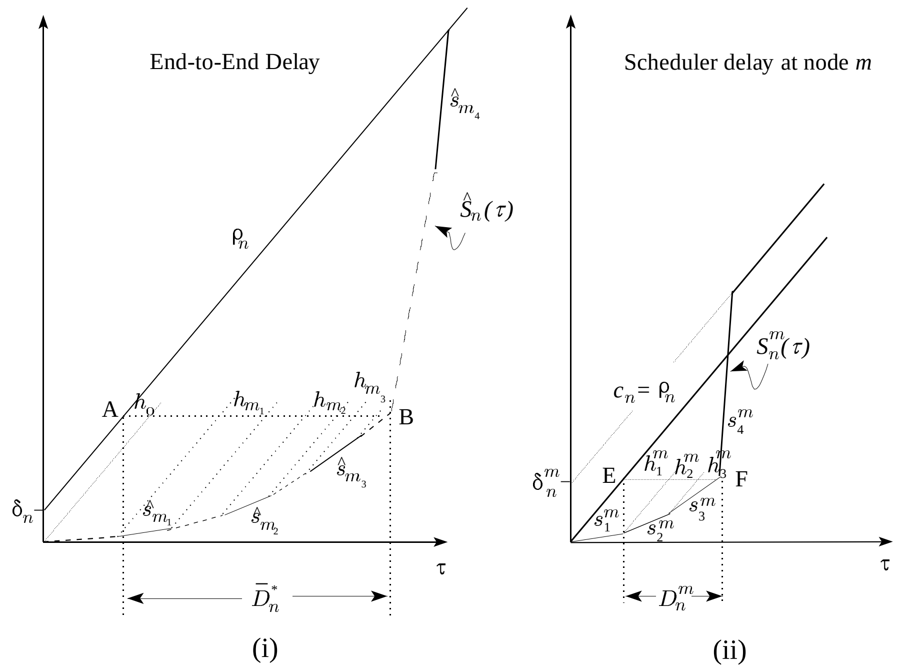

Figure 5.3: Delay Decomposition of a Class (b) Connection

then write,

$$D^m_n = \sum_{k=1}^{j_m-1} h^m_k. \tag{5.10}$$

Using the facts $\widehat{s}_{m_k} = s^m_k$ and $\widehat{b}_{m_k} = b^m_k$, it can be easily seen that $h_{m_k} = h^m_k$. Taking into account (5.9) and (5.10), we conclude the correctness of (5.8).

Similar arguments can be made for a connection that belongs to class (b). The main difference is that we now draw lines with slope ρ_n. Figure 5.3 illustrates the construction in this case. ■

These derivations established that an RCS disciplines can be constructed that provides the same delay bounds as GPS, but the arguments used were more involved than for the simpler case of RPPS. As a result, it is much harder to gain some insight into why and how this is achieved. A possible (and partial) explanation is that the reshaping peak rate c_n for connection n, should be set to the service rate in the Universal Service Curve of the GPS policy, that corresponds to the maximum delay value. Using a larger value will not help since service, and hence reshaping, at that rate will be encountered. Using a smaller value will result in higher delays.

In the course of the previous argument, we showed that the delay guarantees provided by a pure GPS policy can also be achieved by an RCS discipline working with worst case delays at each node, where the scheduling policy at each node is GPS. If we replace GPS with the (simpler) EDF scheduling policy at each node, we are not only assured that we can still guarantee the GPS end-to-end delays, but we also create a service discipline that is better than GPS. This is due to the fact that in the single node case, EDF is better than GPS [21]. That is, there are delay vectors that can be guaranteed by EDF but cannot be guaranteed by GPS no matter what weights are chosen. For example, consider a link of capacity r, where two connections are multiplexed and $\overline{I}_n(\tau) = \delta_n + \rho_n\tau$, $n = 1, 2$, with $\rho_1 + \rho_2 \leq r$. Using (4.7) with $L = 0$, we can see that the delays that can be guaranteed by the EDF policy are

$$D_1 = \frac{\delta_1}{r}, \quad D_2 = \frac{\delta_2}{r-\rho_1} + \frac{\delta_1}{r},$$

For GPS on the other hand, it can be seen from the construction in [35, Section VI.C], that in order to guarantee $D_1^* = \delta_1/r$ we need to specify $\phi_2 = 0$, and then the minimum guaranteed delay for connection 2 is

$$D_2^* = \frac{\delta_2}{r-\rho_1} + \frac{\delta_1}{r-\rho_1}. \tag{5.11}$$

The difference between the GPS and EDF delay guarantees for connection 2 is

$$D_2^* - D_2 = \frac{\rho_1\delta_1}{r(r-\rho_1)},$$

which can be quite large. Similar examples can be given for the packetized model when comparing PGPS to NPEDF. The better bounds of EDF in this simple example, are essentially a reflection of the fact that, in the single node case, EDF is the optimal policy. This is in part due to EDF's ability to, unlike GPS (or its variants), decouple delay and rate guarantees. In the above example, this difference is expressed in the $\delta_1/(r-\rho_1)$ term of (5.11). This term reflects the behavior of GPS, which serves all new packets of connection 1 at rate r, irrespective of the fact that they may have just arrived and, therefore, are in no danger of being excessively delayed. In contrast, the EDF policy exploits this knowledge to improve the delay guarantee it gives to connection 2. In the multiple node case, the benefit of decoupling delay and rate guarantees is still obtained, while the problem of summing up worst case node delays has been alleviated by suitably reshaping the connection traffic.

5.3 Summary

Tight end-to-end delay bounds have been previously obtained for GPS and its many derivatives. Until now these were the best available end-to-end delay guarantees. In this chapter we first demonstrated how a naive use of the RCS discipline where the shaper envelopes were chosen to be identical to the input traffic envelope, could lead to rather poor end-to-end delay guarantees.

However, we showed that with the right choice of traffic shapers the RCS discipline can provide better end-to-end delay guarantees than the GPS discipline.

Chapter 6

Buffer Requirements and Work-Conserving Extensions

In the previous chapter we demonstrated how an RCS discipline with an EDF scheduler can provide tight end-to-end delay guarantees to individual connections. There is a large body of literature in the recent past devoted to the study of scheduling policies and their end-to-end delay performance[25, 27, 35, 46, 50]. However, a delay bound is only part of the story, and is relevant only if there are sufficient buffers to hold the traffic that is delayed. If a network element is buffer constrained then it may well be that the size of the buffer determines the maximum amount of delay that can be experienced by a flow. In the framework adopted in this thesis, we assume that the network, before accepting a connection, ensures that it has sufficient buffers in order not to lose any packets of that connection. In the first half of this chapter, we look at the buffers required at each network element to ensure that no packets are lost. Since the RCS discipline consists of a shaper and a scheduler, we need to compute these buffer requirements, both at the shaper as well as the scheduler. Another QoS parameter that is closely related to delay is jitter. In this chapter we briefly investigate the natural benefits afforded by the RCS discipline in terms of limiting the jitter experienced by a connection.

The many benefits and properties of the RCS discipline, mostly arise from the fact that traffic is regulated at each hop. Regulating traffic at each hop, however, comes at a price in that packets may be delayed in the regulator even though the link is idle. The per-connection traffic reshapers and the link scheduler can be combined, and together viewed as a single server system, where the service process corresponds to the transmission of the packet on the link. A single server system is said to be *work-conserving*, if the server is never idle when there are packets in the system. It is clear that the system in consideration here, is not work-conserving since the server may be idle, i.e. the link could be idle, even when there are packets in the system. This non-work-conserving nature of the RCS discipline might increase the average delay that is experienced by packets of all connections. In the latter part of this chapter we examine some simple modifications to the RCS discipline to make it work-conserving without compromising the end-to-end delay guarantees that can be provided.

6.1 Buffer Requirements

In Chapter 5 we briefly introduced the notion of a service curve to lower bound the service received by a connection at a network element. This service curve was then used to compute a bound on the delay experienced by a connection at a network element. Service curves of individual network elements can be *convolved* to compute a network-wide service curve which is then used to elegantly compute the end-to-end delay bounds for a connection. The idea of using service curves to bound the service process first appeared in [37] and further development of the theory of service curves can be found in [39, 15, 11]. In this section we use the service curve formulation to compute the per-connection buffer requirements at both the shaper and the scheduler. First, we define the term buffer requirement.

Consider connection n and recall that $I_n(t)$ denotes the volume of connection n traffic that is input to a network element in the interval $[0, t]$. Let $O_n(t)$ denote the volume of connection n traffic that is output by the network element in the interval $[0, t]$. We assume both mappings $t \to I_n(t)$ and $t \to O_n(t)$ to be non-decreasing and right continuous on $\mathbb{R}_+$. Then the buffer requirement B_n for connection n is defined as

$$B_n \equiv \max_{t \geq 0} \{I_n(t) - O_n(t)\}. \tag{6.1}$$

If traffic from connection n has a separate queue at the network element and is guaranteed B_n amount of buffer, then it is guaranteed to never overflow its buffer. In practice we do not know the exact nature of the arrival process I_n nor the output process O_n. However, we might know the traffic envelope of the input and a service curve that describes the service process. The next theorem summarizes some results on service curves, the proofs of which are to be found in [39, 15].

Theorem 6.1 [16] *Assume that traffic with an envelope of $\overline{I}_n$ is input to a network element which guarantees it a service curve of $\overline{S_n}$. Then, for connection n, the following bounds can be obtained:*

- *The buffer requirement B_n to ensure that there is no cause for buffer overflow is given by*

$$B_n = \max_{\tau \geq 0} \left\{\overline{I}_n(\tau) - \overline{S}_n(\tau)\right\}^+ \tag{6.2}$$

- *The maximum delay bound D_n is given by*

$$D_n = \max_{\tau \geq 0} \min \left\{\alpha : \alpha \geq 0 \text{ and } \overline{I}_n(\tau) \leq \overline{S}_n(\tau + \alpha)\right\}^+ \tag{6.3}$$

- *An envelope, $\overline{O}_n$, for the connection n traffic at the output of the network element is given by,*

$$\overline{O}_n(\tau) = \max_{\alpha \geq 0} \left\{ \overline{I}_n(\tau + \alpha) - \overline{S}_n(\alpha) \right\}^+, \quad \tau \geq 0. \tag{6.4}$$

In other words, the delay bound for connection n is the maximum horizontal distance from the traffic envelope $\overline{I}_n$ to the service curve $\overline{S}_n$. Similarly, the buffer requirement for connection n is given by the maximum vertical distance from $\overline{S}_n$ to $\overline{I}_n$.

Theorem 6.2 **[16]** *Assume now, that connection n traffic passes through a series of M network elements, each of which guaranteeing it a service curve of $\overline{S}_n^m$, $m = 1, 2, \ldots M$. The entire set of network elements can be viewed as a single entity that guarantees a service curve $\overline{S}_n = \overline{S}_n^1 \otimes \overline{S}_n^2 \cdots \otimes \overline{S}_n^M$, where the operator $\otimes$ is defined as follows:*

$$\overline{S}_n^1 \otimes \overline{S}_n^2(\tau) = \min_{0 \leq \alpha \leq \tau} \left\{ \overline{S}_n^1(\alpha) + \overline{S}_n^2(\tau - \alpha) \right\}, \quad \tau \geq 0. \tag{6.5}$$

The proof for Theorem 6.2 can be found in [16]. Now, let us try and apply some of these results in our setup which assumes that the RCS discipline is used at all nodes in the network. As usual, we focus on a single connection, n, with input traffic envelope $\overline{I}_n$. Without loss of generality we assume that connection n traverses nodes $1, 2, \ldots M$. Let r^m denote the link speed at node m, $m = 1, \ldots, M$. Recall that we use the same traffic shaper with envelope $\overline{A}_n$ at all nodes along the path. Because $\overline{A}_n$ is piece-wise linear, increasing and concave we can write it in the form

$$\overline{A}_n(\tau) = \min_{k=1,\ldots,K} \{\delta_k + \rho_k \tau\}, \tau \geq 0, \tag{6.6}$$

with $\delta_1 \leq \delta_2 \leq \cdots \leq \delta_K$. Furthermore, we assume that

$$\delta_1 \leq r_m \tag{6.7}$$

which is a reasonable assumption if the input and output links operate at the same speed, since traffic can never arrive at a rate that is higher than the link speed. We can now compute the buffer requirements for connection n at node m.

Theorem 6.3 *The buffer requirement $B_{\mathcal{S},n}^m$ for connection n at the scheduler at node m, is given by*

$$B_{\mathcal{S},n}^m = \overline{A}_n(D_n^m), \quad m = 1, \ldots, M.$$

The buffer requirement $B_{\mathcal{A},n}^m$ for the shaper at node m, is given by

$$B_{\mathcal{A},n}^m = \begin{cases} \max\limits_{\tau \geq 0} \left\{ \overline{I}(\tau) - \overline{A}_n(\tau) \mathbf{1}[\tau > 0] \right\}^+ & m = 1, \\ \overline{A}_n(D_n^{m-1}) & m = 2, \ldots, M, \end{cases} \tag{6.8}$$

where

$$\mathbf{1}[\tau > x] \equiv \begin{cases} 0 & \textit{if } \tau \leq x \\ 1 & \textit{if } \tau > x. \end{cases}$$

Proof. The RCS discipline at any node m consists of two parts, the shaper and the scheduler. For clarity we drop the subscript n from the service curves, but it should be emphasized that the service curves are guaranteed to a particular connection. Assuming that a feasible schedule exists, the EDF policy at node m guarantees that all packets from connection n will be transmitted before the local delay guarantee of D_n^m. Therefore, the EDF policy guarantees connection n a service curve [39] of

$$\overline{S}_{\mathcal{S}}^m(\tau) = \begin{cases} 0 & \text{if } \tau \leq D_n^m \\ \infty & \text{if } \tau > D_n^m. \end{cases} \tag{6.9}$$

Using (6.2) we readily compute the buffer requirements for connection n at the scheduler at node m to be $\overline{A}_n(D_n^m)$.

From the definition of the traffic shaper in (3.1) it is clear that the shaper guarantees a service curve of

$$\overline{S}_{\mathcal{A}}^m(\tau) = \overline{A}_n(\tau)\mathbf{1}[\tau > 0]. \tag{6.10}$$

To compute the buffer requirements at the first shaper, i.e., $m = 1$ we can directly apply Theorem 6.1 and (6.2) gives the buffer requirements. In order to compute the buffer requirements for the shaper at node $m+1$, $m = 1, \ldots, M-1$, we need to have an envelope for the traffic that is input to this shaper. For this, we first compute the service curve of the combination of the scheduler at node m and the link from node m to node $m+1$. Recall that the link that is traversed by connection n at node m, is assumed to be operating at a constant rate r^m and so the service curve $\overline{S}_{\Gamma}^m$ of the link is given by

$$\overline{S}_{\Gamma}^m(\tau) = r^m\tau, \quad \tau \geq 0. \tag{6.11}$$

Using (6.5) we can obtain a combined service curve for the link and the scheduler at node m, which is

$$\overline{S}_{\mathcal{S}}^m \otimes \overline{S}_{\Gamma}^m(\tau) = r^m(\tau - D_n^m)\mathbf{1}[\tau > D_n^m], \quad \tau \geq 0. \tag{6.12}$$

From (6.4) we compute a traffic envelope $\overline{O}_n^m$ for the connection n traffic exiting node m, namely

$$\begin{aligned} \overline{O}_n^m(\tau) &= \max_{\alpha \geq 0} \left\{ \overline{A}_n(\tau + \alpha) - \overline{S}_{\mathcal{S}}^m \otimes \overline{S}_{\Gamma}^m(\alpha) \right\}^+ && (6.13) \\ &= \max_{\alpha \geq 0} \left\{ \overline{A}_n(\tau + \alpha) - r^m(\alpha - D_n^m)\mathbf{1}[\alpha > D_n^m] \right\}, \quad \tau \geq 0. && (6.14) \end{aligned}$$

Applying Theorem 6.1 we obtain the buffer requirement $B_{\mathcal{A},n}^{m+1}$ for the shaper at node $m+1$

$$\begin{aligned} B_{\mathcal{A},n}^{m+1} &= \max_{\tau \geq 0} \left\{ \overline{O}_n^m(\tau) - \overline{A}_n(\tau)\mathbf{1}[\tau > 0] \right\} \\ &= \max_{\tau \geq 0} \left\{ \max_{\alpha \geq 0} \left\{ \overline{A}_n(\tau + \alpha) - r^m(\alpha - D_n^m)\mathbf{1}[\alpha > D_n^m] \right\} - \overline{A}_n(\tau)\mathbf{1}[\tau > 0] \right\} \end{aligned}$$

$$
\begin{aligned}
&= \max_{\tau \geq 0} \left\{ \max_{\alpha \geq 0} \left\{ \overline{A}_n(\tau+\alpha) - r^m(\alpha - D_n^m)\mathbf{1}[\alpha > D_n^m] - \overline{A}_n(\tau)\mathbf{1}[\tau > 0] \right\} \right\} \\
&= \max_{\alpha \geq 0} \left\{ \max_{\tau \geq 0} \left\{ \overline{A}_n(\tau+\alpha) - \overline{A}_n(\tau)\mathbf{1}[\tau > 0] - r^m(\alpha - D_n^m)\mathbf{1}[\alpha > D_n^m] \right\} \right\} \\
&= \max_{\alpha \geq 0} \left\{ \overline{A}_n(\alpha) - r^m(\alpha - D_n^m)\mathbf{1}[\alpha > D_n^m] \right\} && (6.15) \\
&= \overline{A}_n(D_n^m), && (6.16)
\end{aligned}
$$

where (6.15) follows from the concavity of $\overline{A}_n$ and (6.16) follows from (6.6) and (6.7). ■

The important conclusion to be drawn from this exercise is that with the RCS discipline the buffer requirements at a node are *independent* of the number of hops traversed by the connection. This is in sharp contrast to most of the other rate-based service disciplines like PGPS, SCFQ, SFQ [25, 27, 35], where the buffer requirements keep increasing with the number of hops between the network element and the source. The reason for this benefit lies in the fact that with an RCS discipline traffic is being reshaped at each hop, and so bursts are not allowed to accumulate. In fact, as seen shortly in Section 6.3, if we relax the requirement of holding back packets in the shaper, we lose some of the benefits of reshaping.

6.2 Bounded Jitter

For a large number of applications it is important to bound the jitter, i.e., the variation in the delay that is experienced at the receiver. This is particularly important for audio and video applications that need to be supplied with a constant data stream. Typically these applications maintain a playback buffer that holds the incoming packets and plays them out at a constant rate. The size of this playback buffer depends on the jitter that the network might introduce into the flow. In this section we will briefly examine how RCS disciplines have a nice property of limiting the jitter.

We begin by defining exactly what we mean by jitter. Let a_i denote the time at which the ith packet enters the network, and let f_i denote the time at which it is received at its destination, $i = 1, 2, \ldots$. The jitter experienced by the ith packet is then defined as

$$J_i = |(f_i - f_{i-1}) - (a_i - a_{i-1})|, \quad i \geq 2. \tag{6.17}$$

Jitter is typically of consequence for traffic generated at a constant rate. In this section, we assume packets are generated by the source at a constant rate α and that the packets are of fixed size L. Therefore,

$$a_i = \alpha i, \quad i = 1, 2, \ldots$$

It turns out that the reshapers in this case closely resemble playback buffers, and this is apparent when we look at a very simple property of the reshapers. For convenience we assume a reshaper is located at the destination as well, and assume that f_i denotes the time at which packet i clears the reshaper at the destination. Let D denote a bound on the end-to-end delay for the connection. The delay incurred by the ith packet is denoted by $d_i := f_i - a_i$. The next lemma makes an interesting observation about the individual packet delays.

Lemma 6.1 *Under an RCS discipline, the packet delay d_i, $i = 1, 2, \ldots$ is non-decreasing in i, and is bounded above by D, i.e.,*

$$d_i \leq d_j \leq D, \quad i < j. \tag{6.18}$$

Proof. The upper bound for the delay is simply a consequence of using RCS disciplines. To verify the monotonicity in the packet delays, observe that the reshaper at the destination ensures $f_{i+1} \geq f_i + \alpha$. Therefore,

$$\begin{aligned} d_{i+1} &= f_{i+1} - a_{i+1} \\ &\geq f_i + \alpha - a_{i+1} \\ &= f_i - a_i \qquad (6.19) \\ &= d_i, \end{aligned}$$

where (6.19) follows from the fact that packets are sent at a constant rate, i.e., $a_{i+1} = a_i + \alpha$. ■

Rearranging the terms in (6.17) and using (6.18, we get

$$J_i = d_i - d_{i-1}, \quad i = 2, \ldots \tag{6.20}$$

Now applying Lemma 6.1, we conclude to the following result. result.

Theorem 6.4 *Consider a connection where traffic is generated at a constant rate and assume that it is provided an end-to-end delay guarantee D by an RCS discipline. Then the* **total** *jitter experienced by all the packets of this connection, is bounded above by the end-to-end delay guarantee, i.e.,*

$$\sum_{i=2}^{\infty} J_i \leq D.$$

Proof. From the definition of jitter (6.17) we have

$$\begin{aligned} \sum_{i=2}^{\infty} J_i &= \sum_{i=2}^{\infty} |d_i - d_{i-1}| && (6.21) \\ &= \sum_{i=2}^{\infty} (d_i - d_{i-1}) && (6.22) \\ &= \lim_{i \to \infty} d_i - d_1 && (6.23) \\ &\leq D, && (6.24) \end{aligned}$$

where (6.22) follows from (6.20) and the limit (6.23) exists by monotonicity of d_i. ■

Thus, it is clear that once any packet has experienced the maximum delay guarantee, all subsequent packets experience zero jitter. It should be noted that we are measuring the jitter just after the reshaper at the destination. However, it may well be that the receiver does not have a reshaper. In this case we can apply the result to the last hop node traversed by the connection and the jitter experienced at the receiver is only bounded by the local delay guarantee at the last hop. Another important observation is that the bound on the jitter only requires a reshaper to be present at the receiver and not at the various intermediate nodes along the path. However, if there is only a single reshaper at the receiver, then applying Theorem 6.3 we see that large amount of buffers are needed to ensure that there is sufficient amount of memory to hold the packets until they have cleared the reshaper. Having reshapers at each hop along the path, prevents large bursts from building up because traffic is smoothed out at each hop.

6.3 Work Conserving RCS discipline

As we pointed out earlier, the RCS discipline is *not* work-conserving. So far, when we talked of delay guarantees we have only assumed that the applications require packets to be delivered within a certain amount of time. It is not relevant to the application if in the course of operation, packets are indeed delivered early since what counts is the *a priori* delay bound that is guaranteed to the connection. This is typical for applications like audio and video playback that have a playback buffer and delay packets to ensure a smooth stream of data at the output of the playback buffer.

However, there are also a large class of applications that, in addition to a hard delay guarantee, are interested in receiving packets as early as possible. The RCS discipline as it stands is not work-conserving and so even though it provides excellent end-to-end delay guarantees, the average delays that packets experience could be larger than other work-conserving disciplines. In this section we describe a modification to the RCS discipline that will make it work conserving, without sacrificing the tight end-to-end delay guarantees that it provides.

Consider any network element that uses the RCS discipline. At any instant of time, if there are packets in the system that have not cleared their reshapers, we refer to them as *ineligible* packets. For simplicity assume that we have a synchronous link where fixed size packets are transmitted in fixed length slots (e.g. SONET framing). If the link is idle for one slot, it means that a packet transmission opportunity is lost. Transmitting an ineligible packet in that slot therefore does not have an adverse impact on the delays experienced by the other packets. For the more general case of variable length packets it is not clear, a priori, whether choosing an ineligible packet for transmission whenever the link is idle, affects the delay guarantees of the other connections. Indeed, were a very long ineligible packet selected for transmission when there were no eligible packets in the system, any eligible packet that has to be transmitted shortly thereafter, will have to wait until this ineligible packet is completely transmitted due to the non-preemptive nature of the scheduling policy.

Assuming that the delay guarantees to all connections at a single node are not violated by sending ineligible packets out whenever the link is idle, it is not clear whether the end-to-end delay guarantee of (4.17) still hold. In the rest of this section we describe a simple modification to the RCS discipline that will make it work-conserving without sacrificing the end-to-end delay guarantees that can be provided to individual connections.

To clarify the exposition, we describe a particular implementation of the RCS discipline [30]. Where there is no cause for confusion, we drop the superscript m that identifies the node along the path of the connection. We do not concern ourselves with the individual shapers for each connection; rather, as each packet arrives into the system, the shaper for that connection simply places a time-stamp on the packet that corresponds to the time at which the packet would have left the shaper. We refer to this time-stamp as the *eligibility time* of the packet. At any time t, only packets with eligibility time greater than or equal to t are considered eligible for transmission on the link.

Based on their eligibility time we can divide the packets in the system at any time t, into two sets: $\mathbf{Q}_e(t)$ is the set of *eligible* packets, i.e., packets whose eligibility time is greater than or equal to t, and $\mathbf{Q}_i(t)$ is the set of *ineligible* packets, i.e., those with eligibility time less than t. The link-scheduler only selects packets in $\mathbf{Q}_e$ for transmission on the link, and once a packet has completed its transmission it is removed from $\mathbf{Q}_e$. Since the eligibility time of a packet is computed based on the connection's traffic envelope, it is clear that packets from each connection enter $\mathbf{Q}_e$ in conformance with their respective traffic envelopes. The call admission criteria ensures that a feasible schedule exists for all packets in $\mathbf{Q}_e$.

We now develop a work-conserving discipline π_W, by modifying the link-scheduler in the non-work-conserving RCS discipline π_{NW} as follows:

Operation of Work-Conserving Extension to RCSD (π_W)

1. A non-preemptive priority mechanism is used to arbitrate between the two sets, $\mathbf{Q}_e(t)$ and $\mathbf{Q}_i(t)$, with $\mathbf{Q}_e(t)$ being the one with the higher priority, i.e., at any time t, a packet from $\mathbf{Q}_i(t)$ is selected for transmission only if $\mathbf{Q}_e(t)$ is empty.

2. When $\mathbf{Q}_e(t)$ is non-empty, packets are selected only from $\mathbf{Q}_e(t)$ for transmission, based on the scheduling policy that is used in π_{NW}.

3. When $\mathbf{Q}_e(t)$ is empty, if there is a packet in $\mathbf{Q}_i(t)$ it is transmitted. If there are several packets in $\mathbf{Q}_i(t)$, any policy can be used to select a packet for transmission.

Let us concentrate on a single node, say m, and for the moment assume that there are N connections multiplexed on a single outgoing link. Traffic from connection n is conformant with a traffic envelope of $\overline{A_n}$, and is guaranteed a local delay bound of D_n^m, $n = 1, \ldots, N$, by the service discipline π_{NW}. First, we establish that even with π_W, the scheduler delay bounds that were obtained for π_{NW} remain valid.

Theorem 6.5 *Under discipline π_W, packets of any connection n, $n = 1, \ldots, N$, are not delayed by more than D_n^m at the scheduler at node m.*

Proof. D_n^m must be larger than the transmission time of a packet from connection n, and so packets that start transmission before they become eligible can never miss their deadline. So we need to show that once a packet becomes eligible it does not miss its deadline. If $\mathbf{Q}_e(t)$ is non-empty at time t, then define a $\mathbf{Q}_e$-busy period to be the largest closed interval $[t_s, t_f]$, $t_s \leq t \leq t_f$, such that $\mathbf{Q}_e(t) \neq \phi$ for any t in the interval $[t_s, t_f]$. All the eligible packets begin transmission in $\mathbf{Q}_e$-busy periods, and so it suffices to show that the packets transmitted in $\mathbf{Q}_e$-busy periods do *not* miss their respective deadlines.

We say that an ineligible packet has *arrived at the scheduler*, when it is promoted to $\mathbf{Q}_e$ or when it is selected for transmission. Let $[t_s, t_f]$ be a $\mathbf{Q}_e$-busy period. If an eligible packet starts transmission at t_s, then the traffic of all connections arriving at the scheduler in $[t_s, t_f]$ are conformant with their respective traffic envelopes. By definition, the operation of the scheduler in π_W during a $\mathbf{Q}_e$-busy period is identical to that in π_{NW}, and therefore it follows that eligible packets do not miss their deadlines.

Now, on the other hand, it is possible that at time t_s an ineligible packet from some connection j, is being transmitted. Let p_0 denote this ineligible packet and let t_0 denote the time at which p_0 begins transmission. We are done if we can show that all the packets that were transmitted in the interval $[t_0, t_f]$ arrived at the scheduler in conformance with their respective traffic envelopes.

Let $A_n(t)$ denote the amount of traffic from connection n that is promoted to $\mathbf{Q}_e$ in the interval $[0, t]$. Let $\widehat{A}_n(t)$ denote the connection n traffic that arrives at the scheduler in the interval $[0, t]$. As $\widehat{A}_n(t)$ includes the ineligible packets that are transmitted we have $\widehat{A}_n(t) \geq A_n(t)$. For $n = 1, 2, \ldots, N$, we need to show that

$$\widehat{A}_n(\tau_2) - \widehat{A}_n(\tau_1^-) \leq \overline{A}_n(\tau_2 - \tau_1), \quad t_0 \leq \tau_1 \leq \tau_2 \leq t_f,$$

where $A_n(\tau_1^-) := \lim_{\epsilon \downarrow 0} A_n(\tau_1 - \epsilon)$.

By the definition of $\overline{A}_n$, we have

$$A_n(\tau_2) - A_n(\tau_1) \leq \overline{A}_n(\tau_2 - \tau_1), \quad 0 \leq \tau_1 \leq \tau_2,$$

while from the definition of a $\mathbf{Q}_e$-busy period, only a single ineligible packet p_0 is transmitted in

$[t_0, t_f]$ and that packet is from connection j. Therefore,

$$\widehat{A}_n(\tau_2) - \widehat{A}_n(\tau_1^-) \leq \overline{A}_n(\tau_2 - \tau_1), \quad t_s \leq \tau_1 \leq \tau_2 \leq t_f, \quad n \neq j.$$

Since, in addition $\widehat{A_n}(t_s^-) - \widehat{A_n}(t_0) = 0$ for $n \neq j$, we have,

$$\widehat{A}_n(\tau_2) - \widehat{A}_n(\tau_1^-) \leq \overline{A}_n(\tau_2 - \tau_1), \quad t_0 \leq \tau_1 \leq \tau_2 \leq t_f, \quad n \neq j. \tag{6.25}$$

Consider next connection j, and let t_0^l denote the local eligibility time of packet p_0. If $t_0^l \geq t_f$, then clearly

$$\widehat{A}_j(\tau_2) - \widehat{A}_j(\tau_1^-) \leq L \leq \overline{A}_j(0), \quad t_0 \leq \tau_1 \leq \tau_2 \leq t_f,$$

since no more packets from connection j will be transmitted in $[t_0, t_f]$. Now suppose $t_0 \leq t_0^l < t_f$. Then, all other packets of connection j will arrive to the scheduler only after t_0^l. For the case when $t_0 \leq \tau_1 \leq t_0^l$ and $t_0^l \leq \tau_2 \leq t_f$, we have

$$\widehat{A_j}(\tau_2) - \widehat{A_j}(\tau_1^-) \leq \overline{A}_j(\tau_2 - t_0^l) \leq \overline{A}_j(\tau_2 - \tau_1).$$

The other cases can be similarly verified. ■

Thus, for the single node case, we have shown that transmitting packets before they have cleared their respective reshapers does not adversely affect the other traffic passing through that node. However, we cannot directly apply Theorem 3.4 to conclude that the end-to-end delay guarantees are still met, since we are not actually reshaping the traffic at each hop. Further, it is not true that every packet in the work-conserving system will see a smaller end-to-end delay than it would in the corresponding non-work conserving system, so we cannot make a pathwise comparison either. Still, it turns out that the end-to-end delay bound for π_W is the same as that for π_{NW} and this is the content of the next theorem.

Theorem 6.6 *Discipline π_W can provide the same end-to-end delay bounds as π_{NW}, i.e., if the nodes traversed by a connection are numbered as $1, 2, \ldots, M$, then the end-to-end delay guarantee for connection n is given by*

$$D_n = D(\overline{I}_n \| \mathcal{A}_n^1) + \sum_{m=1}^{M-1} D\left(\mathcal{A}_n^m \| \mathcal{A}_n^{m+1}\right) + \sum_{m=1}^{M} D_n^m + \sum_{m=1}^{M} T^{(m,m+1)}. \tag{6.26}$$

Proof. For clarity in the exposition we assume that the propagation delays $T^{(m,m+1)}$, $m = 1, \ldots, M$, are zero. We focus our attention on connection n and let p_i denote the ith packet of connection n. We denote by $t_i^{l,m}$, the time at which p_i will be eligible for transmission at node m; so that $t_i^{l,m}$ is the time at which packet p_i would leave shaper $\mathcal{A}_n^m$ in conformance with the traffic envelope $\overline{A}_n^m$. The actual time that p_i leaves $\mathbf{Q}_i^m$ (to be transmitted on the link or promoted to $\mathbf{Q}_e^m$) is denoted by $t_i^{a,m}$. If the link is idle, the packet may be transmitted before it becomes eligible, i.e., $t_i^{a,m} \le t_i^{l,m}$. The departure time of the ith packet from the scheduler is denoted as $t_i^{d,m}$. Similarly, let $t_i^{a,0}$ be the arrival time of p_i at the first traffic shaper on its path, and let $t_i^{d,M}$ be the time it arrives at its destination. Since $t_i^{a,M} \le t_i^{l,M}$ we can write

$$\begin{aligned} t_i^{d,M} - t_i^{a,0} &\le t_i^{l,M} - t_i^{a,0} + t_i^{d,M} - t_i^{a,M} \\ &= \sum_{m=1}^{M-1} \left(t_i^{l,m+1} - t_i^{l,m} \right) + t_i^{l,1} - t_i^{a,0} + t_i^{d,M} - t_i^{a,M}. \end{aligned} \tag{6.27}$$

By definition, $t_i^{l,1} - t_i^{a,0} \le D\left(\overline{I}_n \| \mathcal{A}_n^1\right)$, and $t_i^{d,M} - t_i^{a,M} \le D_n^M$ according to Theorem 6.5, and so we have

$$t_i^{d,M} - t_i^{a,0} \le D\left(\overline{I}_n \| \mathcal{A}_n^1\right) + \sum_{m=1}^{M-1} \left(t_i^{l,m+1} - t_i^{l,m} \right) + D_n^M. \tag{6.28}$$

Comparing (6.28) with (6.26) and noting that we have assumed the propagation delays, $T^{(m,m+1)}$ to be zero, it suffices to show that

$$t_i^{l,m+1} - t_i^{l,m} \le D_n^m + D(\mathcal{A}_n^m \| \mathcal{A}_n^{m+1}) \quad m = 1, \ldots, M-1. \tag{6.29}$$

Let $\mathcal{S}^m$ be the system consisting of the scheduler at node m. Consider the modified system which is same as the work conserving system operating under π_W except for a delay system inserted between $\mathcal{S}^m$ and shaper $\mathcal{A}_n^{m+1}$ as shown in Figure 6.1. The delay system delays p_i by $\theta_i := D_n^m + t_i^{l,m} - t_i^{d,m}$; therefore, p_i departs the delay system at time $\hat{t}_i^{d,m} := D_n^m + t_i^{l,m}$. First we verify that $\theta_i \ge 0$. Indeed,

$$\begin{aligned} \theta_i &= D_n^m + t_i^{l,m} - t_i^{d,m} \\ &\ge D_n^m + t_i^{a,m} - t_i^{d,m} \qquad (6.30) \\ &\ge 0 \qquad (6.31) \end{aligned}$$

with (6.30) following from the fact that packets never depart the shaper later than they are supposed to, i.e., $t_i^{l,m} \ge t_i^{a,m}$, and (6.31) following from Theorem 6.5. Let $\hat{t}_i^{l,m+1}$ be the eligibility time of p_i in the modified system. From Lemma 3.1, we conclude that

$$t_i^{l,m+1} - t_i^{d,m} \le \hat{t}_i^{l,m+1} - t_i^{d,m}. \tag{6.32}$$

Adding $t_i^{d,m} - t_i^{l,m}$ to both sides of (6.32) we have

$$\begin{aligned} t_i^{l,m+1} - t_i^{l,m} &\le \hat{t}_i^{l,m+1} - t_i^{l,m} \\ &= \hat{t}_i^{l,m+1} - \hat{t}_i^{d,m} + \hat{t}_i^{d,m} - t_i^{l,m}. \end{aligned} \tag{6.33}$$

Since $\hat{t}_i^{d,m} = t_i^{l,m} + D_n^m$, $i = 1, 2, \ldots$, it follows that the traffic exiting the delay system has envelope $\overline{A}_n^m$, whence

$$\hat{t}_i^{l,m+1} - \hat{t}_i^{d,m} \le D(\mathcal{A}_n^m \| \mathcal{A}_n^{m+1}). \tag{6.34}$$

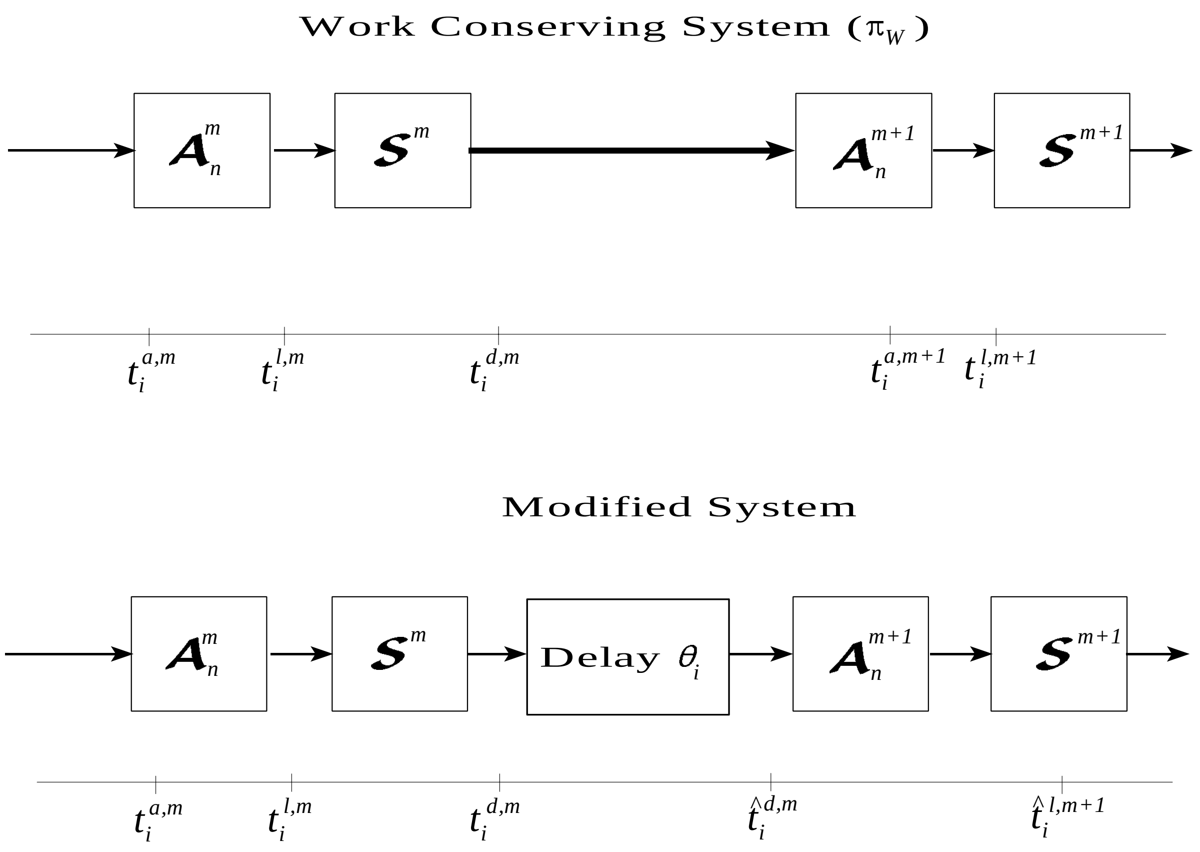

Figure 6.1: Original (work conserving) system and the modified system

Form (6.33) and (6.34) it follows that

$$t_i^{l,m+1} - t_i^{l,m} \leq D(\mathcal{A}_n^m \| \mathcal{A}_n^{m+1}) + D_n^m.$$

When $T_n^{(m,m+1)} > 0$, the same reasoning applies, provided that system $\mathcal{S}^m$ consists of the scheduler at node m, and the link $(m, m+1)$, i.e., the bound on the delay at $\mathcal{S}^m$ is now $D_n^m + T_n^{(m,m+1)}$. ■

We have described a general procedure for making RCS disciplines work conserving regardless of the scheduling discipline that is used – of course, the scheduling discipline, by itself needs to be work conserving. In keeping with this generality, we have not specified the exact order in which packets from the ineligible queue can be served. Based on the scheduling discipline that is used, as well as the behavior that is desired for the flows in general, some specific ordering of the ineligible packets may be appropriate. If the EDF scheduling policy is being used at the link scheduler, then the ineligible packets can also be ordered based on their deadlines. In fact, if the EDF policy is used as the scheduler a simple modification to the RCS discipline can make it work conserving and this is the subject of the next section.

6.3.1 Work Conserving Extension to the RCS discipline that uses an EDF scheduler

When the EDF policy is used at the scheduler, it is possible to come up with a very simple implementation of a work conserving RCS discipline. This implementation might well be simpler than the non-work conserving RCS discipline that uses an EDF scheduler since it it eliminates the need for a separate traffic shaper. All that is required is a slight modification in the computation of the deadlines.

Operation of a Work-Conserving RCSD that uses an EDF scheduling policy (π_{WE})

1. For each connection the system maintains the necessary state to ensure compliance with the traffic envelope. For example, if the traffic characteristics are in terms of a leaky bucket, then the number of tokens in the leaky bucket is maintained. This state is used to determine the eligibility time of the next arriving packet.

2. As each packet arrives into the system a timestamp (deadline) is placed on the packet. This timestamp is the sum of the packet's eligibility time and the local delay guarantee for that connection at the network element.

3. The scheduler (EDF) picks the packet with the smallest timestamp (deadline) and transmits it on the link. Ties are broken, arbitrarily.

It is clear that the above system is work-conserving since the link is never idle if there are packets in the system. However, Theorem 6.5 cannot be used to conclude that the packets do not miss their deadlines since with π_{WE} there may be cases where ineligible packets with small delay guarantees get transmitted even though there are eligible packets with larger delay guarantees waiting in the system. Nevertheless, it turns out that the deadlines are indeed met as we now show.

Theorem 6.7 *If the feasibility conditions for the EDF policy* (4.6) *are met, then under* π_{WE} *all packets are transmitted before their deadlines.*

Proof. Recall that $A_n(t)$ denotes the amount of traffic that has arrived from connection n until time t. Let $O_n(t)$ denote the amount of traffic from connection n that has left the system by time t. The connection n backlog at time t is denoted by $B_n(t) := A_n(t) - O_n(t)$. Therefore, at any time t, the total backlog in the system is given by $B(t) := \sum_{n=1}^{N} B_n(t)$.

Assume that a packet missed its deadline, and let t_f be the deadline of the first packet that missed its deadline. Now, let t_s denote the start of the busy period containing t_f, i.e.

$$t_s := \min\{t : t \leq t_f \text{ and } B(\tau) > 0,\ t \leq \tau \leq t_f\}.$$

Furthermore, let t_h be the start time of the last packet transmitted in the interval $[t_s, t_f]$ that had a deadline greater than or equal to t_f. Let $t_{h'}$ denote the time at which this packet completed its transmission. If no such packet exists, set $t_{h'} = t_h = t_s$. Now, let $X[t_{h'}, t_f]$ denote the total amount of traffic that is in the system during the interval $[t_{h'}, t_f]$ and has a deadline less than or equal to t_f. Note that $X[t_{h'}, t_f]$ includes the packet whose deadline t_f was missed.

By the definition of $t_{h'}$ and t_f, it follows that only packets with deadlines in $[t_{h'}, t_f]$ are served in the same interval, which is actually part of a busy period; therefore

$$X[t_{h'}, t_f] > r(t_f - t_{h'}), \tag{6.35}$$

where r denotes the speed of the link. The strict inequality in (6.35) is due to the fact that $X[t_{h'}, t_f]$ includes the packet with the missed deadline that was not transmitted by time t_f.

Observe that only those connections that are not back-logged at time t_h (i.e. n such that $B_n(t_h) = 0$) can contribute to $X[t_{h'}, t_f]$. This is because the EDF policy, by definition, selects the packet with the smallest deadline for transmission, and so if at time t_h there were packets in the system from back-logged connections, they must have all had deadlines larger than t_f. Therefore,

$$\begin{aligned} X[t_{h'}, t_f] &\leq \sum_{n:B_n(t_h)=0} \overline{A}_n(t_f - D_n - t_h) \qquad (6.36)\\ &\leq \sum_{n=1}^{N} \overline{A}_n(t_f - D_n - t_h) \end{aligned}$$

$$\leq \quad r(t_f - t_h) - L \tag{6.37}$$

$$\leq \quad r(t_f - t_{h'}), \tag{6.38}$$

where (6.36) holds because $\overline{A}_n(t_f - D_n - t_h)$ is the maximum amount of connection n traffic that can arrive in the interval $[t_h, t_f]$ and be assigned a deadline no larger than t_f. Also, (6.37) follows from the feasibility check for the EDF policy as given in (4.6), and (6.38) follows from the fact that $r(t_{h'} - t_h) \leq L$, the maximum packet size. Since (6.38) is in direct contradiction to (6.35) we conclude that our assumption was incorrect and therefore no packet misses its deadline. ■

The Work-Conserving extensions to the RCS discipline described in this chapter, should reduce the average delay that is experienced by packets, since packets are no longer held up when the link is idle. However, this improvement does come at a price, *viz.* traffic is no longer guaranteed to conform to a traffic envelope at any point along its path. Therefore, traffic from a connection can get more and more bursty as it progresses through the network and the buffers required to ensure lossless service will increase. In the next section, we compute a bound on the buffer requirements at each network element if the π_{WE} discipline is used.

6.3.2 Buffer Requirements for π_{WE}

The buffer requirements for the π_{WE} discipline can be easily computed using the service curve framework. The service-curve guaranteed by the π_{WE} discipline to a connection n at node m, is a combination of the shaper and scheduler service curves, $\overline{S}_{\mathcal{A}}^m$ and $\overline{S}_{\mathcal{S}}^m$, respectively. Using Theorem 6.2 we can obtain the service curve for the combination of the shaper and the scheduler to be

$$\overline{S}_{\mathcal{A}}^m \otimes \overline{S}_{\mathcal{S}}^m(\tau) = \begin{cases} 0 & \text{if } \tau \leq D_n^m \\ \overline{A}_n(\tau - D_n^m) & \text{if } \tau > D_n^m. \end{cases} \tag{6.39}$$

Now, to compute the buffer requirements at node $m+1$, we need to compute the output traffic envelope at node m. Since we do not know the envelope for the traffic that is input to node m, we are forced to compute the service curve from the first node at which we do know the traffic envelope. In other words we need to compute the service curve $\overline{S}^{1,m}$ for the tandem of network elements, $1, 2, \ldots, m$. This is readily computed using (6.5) as

$$\overline{S}^{1,m}(\tau) = \begin{cases} 0 & \text{if } \tau \leq \sum_{i=1}^m D_n^i \\ \overline{A}_n(\tau - \sum_{i=1}^m D_n^i) & \text{if } \tau > \sum_{i=1}^m D_n^i. \end{cases} \tag{6.40}$$

Assuming that the shaper envelope satisfies the constraints (6.6) and (6.7) we can apply almost identical arguments as found in the proof to Theorem 6.3 to obtain the buffer requirement of $\overline{A}_n(\sum_{i=1}^m D_n^i)$ for the shaper at node $m+1$. This is clearly much larger than the corresponding buffer requirement for the non-work conserving RCSD which is $\overline{A}_n(D_n^m)$. Additionally, this has the disconcerting property of growing with the number of hops along the path. Thus, the work-conserving extension to RCS discipline suffers the same drawback as most of other work-conserving policies, like GPS, SCFQ, namely the buffers required to guarantee zero loss to a connection typically increase with the number of hops along the path.

6.4 Summary

In the first half of this chapter we examined some aspects related to buffer requirements and jitter performance of RCS disciplines. By using the service curve framework of [15, 39] we were able to obtain a a bound on the buffer requirements at a network element to ensure that no packets are lost due to buffer overflow. In the next chapter we consider some specific examples that provide some insight into typical buffer requirements and delay bounds. Another performance metric that is closely related to the end-to-end delay is the end-to-end jitter that is experienced by successive packets. In this chapter we showed how the RCS discipline limits the amount of jitter that is experienced by a Constant Bit Rate (CBR) connection.

By the nature of its operation the RCS discipline is non-work conserving. In the second half of this chapter we explored some modification to the RCS discipline that make it work conserving. We described a general modification that can be applied to any RCS discipline, provided that its scheduler component, by itself, is work-conserving. Finally, we described a simple implementation of a work-conserving RCS discipline that uses the EDF scheduler. This implementation is relatively the most simple to implement as it only involves marking a deadline on each incoming packet and serving the packet with the smallest deadline first. Finally, we obtained the per-connection buffer requirements for this implementation as well.

Chapter 7

End-to-End Services

In this chapter we look at the big picture which is the provision of service guarantees in the context of an internet. The Internet today is quite ubiquitous as there are hardly any computers left without some form of connectivity to the "net". Even though the last few years have witnessed a rapid increase in the backbone capacity of the Internet, the growth in the number of users has also been quite substantial. In addition, with the ever increasing popularity of the Web, today's Internet traffic is predominantly composed of http sessions. A typical http session lasts for a very short amount of time – the amount of time it takes to download the contents of a Web-page. Thus Web traffic is intrinsically bursty in nature and it is well known that bursty traffic does not make the best use of the link resources. The rapidly increasing audio-visual content in Web pages is going to further increase the amount of data transferred to load a single Web page. It is common experience to click on a hypertext link and have to wait several minutes before the page is loaded in the Web browser. Most often, it is not the Web-server, but rather the network that is the cause of this delay. In other words, there simply is not enough bandwidth for all users to be satisfied with the service that they receive and this situation is unlikely to end soon.

The Internet is also rapidly becoming more and more commercial, and it is not hard to envision the day when a significant amount of our commerce will be conducted over the Internet. With the banking and financial industries also moving towards the Internet, the need to provide secure and guaranteed service will become imperative. Thus, it is more and more important to be able to provide some service differentiation between the different flows that are carried by the network.

On the other hand, some argue that most applications of the future will be adaptive in nature and will sense the current state of the network and appropriately adjust their traffic parameters to obtain the desired level of service. While this type of behavior may be acceptable today, it will become less and less effective as the amount of traffic and the commercial use of the Internet increases.

An analogy with the highways around most of the densely populated cities of the Eastern U.S. is probably appropriate. As roads get more and more congested around these cities the Transportation Authorities build additional highways and charge a per-vehicle toll for their use. The tollway or turnpike as it is typically called, provides a "better" service at an increased cost to the user. People could simply have increased the amount of time that they allocated for their commute and stuck to the more congested routes. However, quite to the contrary, most people prefer to take the faster route even though they have to pay a rather high toll. Hopefully, the same will be true for the Internet, so that when given a choice, many users will prefer to have some service differentiation, rather than a one-size fits all type of service.

There are several ways to provide service differentiation at the network level. One very simple solution, which was adopted by the ATM Forum for UNI 3.1 was the notion of providing five different service classes [2]. However, with a few different classes of service it is not easy for the user/application to map its requirements onto one of these service classes. The user is typically interested in end-to-end performance and does not really care or know about how packets are handled at the network layer. If there are fixed service classes, the end-to-end service that is obtained will depend not only on the service class that is chosen, but also the number of hops, propagation delays, etc. The end-user or application cannot be expected to factor all these possibilities while choosing a service class, since in most cases it is not even aware of the network level events taking place. For these reasons the UNI 4.0 Signalling for ATM networks has been designed to support the signalling of end-to-end QoS Parameters [3].

The Internet Engineering Task Force (IETF) is a kind of "standards" body for the Internet. One of the working groups of the IETF, known as the IntServ Working Group has specified several new service definitions to support quality-of-service (QoS) guarantees in the Internet [42, 41, 43, 49]. Some of them are the

1. Guaranteed (G) Service,
2. Predictive Delay (PD) Service,
3. Controlled Delay (CD) Service, and
4. Controlled Load (CL) Service.

Of these the two that are close to becoming a proposed standard are the G Service and the CL service. We focus on the G service specification, since this service is meant to provide deterministic end-to-end delay guarantees. In particular, we would like to examine how the RCS discipline can be effectively used to support the G service, without having to change any of the signalling and setup mechanisms that are being standardized today.

7.1 Guaranteed Services

First, we briefly outline the workings of the G service specification [42]. In order to better motivate the specifications in [42], as well as provide a better understanding of the end-to-end signalling that is involved, we use the Resource reSerVation Protocol (RSVP) as an example. RSVP is a resource reservation setup protocol designed for an internet that supports integrated services [9]. A primary design goal for RSVP was to support multicast with receivers being able to add themselves to a multicast session at will. In keeping with this philosophy, RSVP is a receiver-initiated protocol, with the resource reservation being made by the receivers. For simplicity, in this thesis, we consider an RSVP session that has a single sender[1] with multiple receivers in its distribution list. Furthermore, we only describe the reservation aspects of RSVP, and do not venture into describing the "filtering" of packets from different senders, as that will take us too far afield.

We use the term *flow* to refer to the stream of data traffic that is transported from a sender to one or more receivers. *Path* messages are sent from the sender to all the receivers in the multicast distribution list along the default routing path of the internet. These messages contain information about the flow, i.e., a *flowspec* that describes the flow's characteristics in terms of leaky bucket parameters [47]. In addition, they contain an *adspec*, that can be modified by the different network elements (NE) traversed by the *Path* message. A receiver who decides to join an RSVP session needs to send a *Resv* message that specifies the amount of resources that it wishes to reserve for itself. The receiver uses the information in the *flowspec* and the *adspec* previously received to compute the level of resources that it needs to reserve. The *Resv* message sent by the receiver retraces the path of the *Path* message (details are in [9]) and establishes the required reservation.

The role of the G service specification is to allow the receiver to make an intelligent choice about the level of resources it needs to reserve in order to obtain an upper bound on the end-to-end packet delay. The amount of resources to be reserved are a function of:

- **User Characteristics** : This has to do with the *flowspec* of the flow and the end-to-end delay and/or throughput requirement of the receiver.
- **Network Characteristics** : These include factors like the number of hops on the path, the scheduling policy employed at each hop, the end-to-end latency that is present, etc.

7.1.1 User and Network Characteristics

The User Characteristics consist of the *flowspec* which specifies the traffic envelope of the input, and an *RSpec* which indicates the level of resources that have to be reserved for this flow. For

[1]RSVP supports multiple senders and multiple receivers, as long as they are supported by the underlying internet.

now, it suffices to say that the *RSpec* is a rate R, and a slack term S – the full meaning of these terms will become clear when we describe the Network Characteristics. The *flowspec* specifies the characteristics of the flow in terms of leaky bucket parameters. Specifically it consists of the following parameters (σ, ρ, c, L), where σ is the token bucket size, ρ is the token accumulation rate, c is the maximum peak rate of the flow and L is the maximum packet size. In terms of the traffic envelope characterization that is used in this thesis, the *flowspec* parameters basically specify the traffic envelope for the flow, which is given by

$$\overline{I}(\tau) = \max\{L + c\tau, \sigma + \rho\tau\}, \quad \tau \geq 0.$$

The Network Characteristics are signalled to the receiver in the *adspec* element which, in the case of RSVP, is carried in the *Path* messages that traverse the NEs along the path of the flow. At any given time there can be numerous *Path* messages passing through an NE, most of which do not result in any resource reservation Thus, it is necessary that the network characteristics signalled in the *adspec* element be independent of the other *Path* messages that may be passing through the NE. Also, it is possible that a large number of *Resv* messages from different flows reach a particular NE in a rather short span of time. So, basing the Network Characteristics on the current load at the NE can be equally meaningless. The solution adopted by the IntServ Working Group of the IETF is to define a characterization of the NE that is independent of the other flows that are passing through it, and is described next.

A Network element exports parameters C and E[2], that qualify the level of service that it can provide to flows that traverse it. These exported parameters are carried by the *adspec* element and are interpreted in the context of the reserved rate R that a potential receiver might reserve for a flow. Typically, C and E capture the deviation of the service provided by the NE, from a fluid server that is operating at the rate R. More generally, a network element that advertises C and E values, guarantees each flow that has a reserved rate of R a service curve $\overline{S}$ given by

$$\begin{aligned} \overline{S}(\tau) &= \left[\left(\tau - \frac{C}{R} - E\right) R\right]^+ \\ &= [(\tau - E)R - C)]^+, \quad \tau \geq 0. \end{aligned} \tag{7.1}$$

It should be pointed out that the C and E parameters exported by the network element only account for the queueing and transmission delays, and do not account for the propagation delay on the link. If the propagation delay is known, it can be signalled separately, and is not a mandatory requirement of the service specification. The above signalling scheme is called a One Pass With Advertisement (OPWA) scheme since the receiver chooses the level of service based on the advertised C and E values that it has received [40].

[2]Actually this is referred to as the D parameter in the G Services specification. However, we use the term E in this thesis to avoid confusion with our notation for delay guarantees

Consider flow n that has a *flowspec* of $(\sigma_n, \rho_n, c_n, L_n)$. In other words the flow has a traffic envelope of $\overline{I}_n$ given by

$$\overline{I}_n = \max\{L_n + c_n\tau, \sigma_n + \rho_n\tau\}, \quad \tau \geq 0.$$

Without loss of generality assume that the nodes traversed by this flow are numbered $1, \ldots, M$ and let C_n^m and E_n^m be respectively the parameters C and E advertised by node m, $m = 1, \ldots, M$. Now assume that the receiver makes a reservation for rate R_n for this flow. We can compute bounds on the delay and buffer requirements for flow n.

7.1.2 Bounds on the Delay and Buffer Requirements

Let $\overline{S}_n^m$ denote the service curve guaranteed to flow n by NE m. From Theorem 6.2 it follows that a service curve, $\overline{S}_n^{1,m}$, can be calculated for the tandem of NEs $1, 2, \ldots, m$, $m = 1, \ldots, M$ as

$$\overline{S}_n^{1,m} = S_n^1 \otimes \overline{S}_n^2 \otimes \cdots \otimes \overline{S}_n^m.$$

Using (7.1) we obtain,

$$\overline{S}_n^{1,m}(\tau) = \left[\left(\tau - \sum_{j=1}^{m} E_n^j\right) R_n - \sum_{j=1}^{m} C_n^j\right]^+, \quad \tau \geq 0. \tag{7.2}$$

The service curve $\overline{S}_n^{1,m}$ can then be used to compute an upper bound on the delay incurred by a packet of flow n from the time it enters the network until the time it leaves NE m. In addition, $\overline{S}_n^{1,m}$ can be used to calculate the buffer requirements at NE m.

Applying Theorem 6.1 we obtain a closed form expression for an upper bound on the delay incurred by a packet of flow n, until and including the delay at NE m, as

$$\overline{D}_n^{1,m} = \begin{cases} \dfrac{(\sigma_n - L_n)}{R_n}\dfrac{(c_n - R_n)}{(c_n - \rho_n)} + \dfrac{L_n}{R_n} \displaystyle\sum_{j=1}^{m} \left[\dfrac{C_n^j}{R_n} + E_n^j\right] & \text{if } c_n > R_n, \\ \dfrac{L_n}{R_n} + \displaystyle\sum_{j=1}^{m} \left[\dfrac{C_n^j}{R_n} + E_n^j\right] & \text{if } c_n \leq R_n, \end{cases} \tag{7.3}$$

which is the length of the segment IF in Figure 7.1.

In a similar manner, the buffer requirement for the flow n at the NE m (GH in Figure 7.1), is given by

$$B_n^m = L_n + \frac{(c_n - X_n)}{(c_n - \rho_n)}(\sigma_n - L_n) + \sum_{j=1}^{m} \left[\frac{C_n^j}{R_n} + E_n^j\right] X_n, \tag{7.4}$$

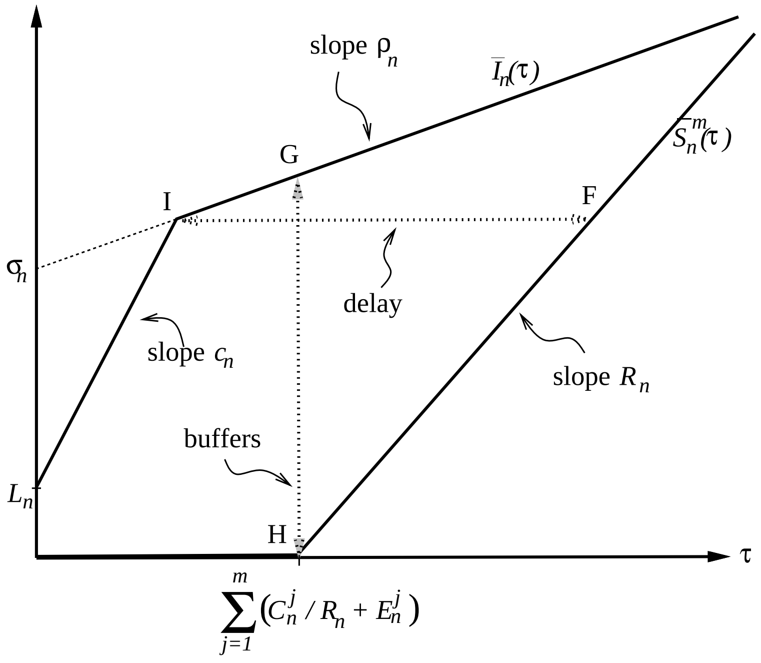

Figure 7.1: Delay and Buffer calculations for a $(\sigma_n, \rho_n, c_n, L_n)$ flow at NE m.

where,

$$X_n = \begin{cases} \rho_n & \text{if } \dfrac{(\sigma_n - L_n)}{(c_n - \rho_n)} \le \sum_{j=1}^{m} \left[\dfrac{C_n^j}{R} + E_n^j \right] \\ R_n & \text{if } \dfrac{(\sigma_n - L_n)}{(c_n - \rho_n)} > \sum_{j=1}^{m} \left[\dfrac{C_n^j}{R_n} + E_n^j \right] \text{ and } c_n > R_n \\ c_n & \text{otherwise.} \end{cases} \tag{7.5}$$

The delay bounds in (7.3) are an intrinsic part of the G Service specification. The assumption here is that the user (receiver) has some idea of the end-to-end delay that it desires. The receiver then chooses a level of service, namely a rate R_n, so that the end-to-end delay bound obtained from (7.3) is what it desires. It is conceivable that the receiver requires a delay guarantee that is so large that even a choice of $R_n = \rho_n$ provides a smaller delay guarantee than it requires. The receiver cannot really choose a rate R_n that is less than the average rate ρ_n since it might lead to a build-up of packets at some node. The G Service specification was modified to allow the receiver to signal a slack term that can be used by the upstream nodes to reduce their resource allocations for this flow. This slack term can be quite effectively used by schedulers which are delay-based like the EDF scheduler, as opposed to rate-based schedulers like GPS. We do not wish to digress into the use of this slack term since it is not directly related to the material in this thesis. For now, suffice it to say that certain rules must be observed when propagating the slack term upstream particularly at split points (of multicast flows) [42].

7.2 RCS Discipline

Our first task is to examine whether the RCS discipline can be used efficiently in the framework of [42]. If so, we need to determine the C and E parameters that can be advertised by the RCS discipline so that the end-to-end delay guarantees as specified in (7.3) can be met.

7.2.1 Parameters exported by the RCS discipline

As mentioned in Section 7.1 the service curve that is guaranteed by a NE is specified in terms of the parameters C and E. An important observation about this service curve is that it depends on the rate R_n reserved by the connection n. The higher the reserved rate R_n, the larger the service curve that is guaranteed to connection n. In contrast, the service curve that is guaranteed by the RCS discipline depends on the deadline that is assigned to the flow as given by (6.12). Therefore we need to map the reserved rate R_n to the deadline that needs to be assigned by the EDF scheduler. It turns out that the mapping that was used in Section 5.1 can be used with a minor modification and that it satisfies the end-to-end delay guarantee of (7.3) as well.

If the following steps are observed:

1. The traffic envelope for the reshaper for this flow is set to

$$\overline{A}_n(\tau) = \min\{\sigma_n + \rho_n \tau, L_n + \min\{c_n, R_n\}\tau\}, \quad \tau \geq 0. \tag{7.6}$$

2. A deadline of $L_n/R_n + \widehat{L}^m/r$ is used at the EDF scheduler for this flow, and

3. the terms C and E exported by NE m are given by:

$$\begin{aligned} C_n^m &= L_n, \\ E_n^m &= \widehat{L}^m/r^m, \end{aligned} \tag{7.7}$$

where $\widehat{L}^m$ denotes the Maximum Transmission Unit (MTU) of the link at node m that flow m is routed on to and r^m denotes the link speed. From the definition of the EDF scheduler, it is clear that the maximum delay encountered by a packet from the time it is released from the reshaper, until it is completely transmitted out on the link, is no more than $L_n/R_n + L^m/r^m$ units of time. Note that (7.7) is obtained by comparing the service curve guaranteed by the RCS discipline and the specification of the terms C and E in the G service specification.

Using (4.6) it can be checked that as long as the sum of the reserved rates (R) of all the flows passing through this NE are less than the link speed, r^m, the deadline assignment in step 2 always results in a feasible schedule.

Applying Corollary 3.2 we obtain an end-to-end delay bound D_n that can be written as

$$D_n = \begin{cases} \dfrac{(\sigma_n - L_n)}{R_n}\dfrac{(c_n - R_n)}{(c_n - \rho_n)} + \displaystyle\sum_{m=1}^{M}\left[\frac{L_n}{R_n} + \frac{\widehat{L}^m}{r^m}\right] & \text{if } c_n > R_n, \\ \displaystyle\sum_{m=1}^{M}\left[\frac{L_n}{R_n} + \frac{\widehat{L}^m}{r^m}\right] & \text{if } c_n \leq R_n, \end{cases} \tag{7.8}$$

where $\widehat{L}^m$, and r^m, are respectively, the MTU and speed of the link that is traversed by flow n at node m, $m = 1, \ldots, M$. It can be readily verified that (7.8) results in a slightly tighter bound on the end-to-end delay than what is obtained by substituting (7.7) in (7.3).

In the next example, we illustrate these calculations using a reasonably representative set of flows. Also we demonstrate how with the G service specification regardless of the scheduling policy employed, a large amount of bandwidth may not be utilized. Subsequently, we outline how with an RCS discipline this excess bandwidth can be utilized to support another type of service called the Committed Rate (CR) service [5]. Finally, we show how with a simple extension, the WFQ policy can also efficiently support the CR service.

7.2.2 Example

As an example, consider an OC-3 (155 Mb/s) output link at an NE. For simplicity, we assume that the G service traffic at this link is comprised of only the 3 types of flows that are listed in Table 7.1. For voice, we assume a standard 64Kb/s constant bit rate flow, while for Stored Video, we use typical values from MPEG traces of a movie like Star Wars, that roughly correspond to 3 Mb/s average rate and a burst size of around 100 Kbytes [34]. The Video Conference flow we consider, has an average rate of about 1.5 Mb/s and a maximum burst size of 10 Kbytes. We assume that the peak rate for both types of Video flows is only limited by the speed of the media to which the source is attached (say 10-base T Ethernet), i.e., 10Mb/s. The maximum packet size for voice is limited to 100 byte packets, while for both the Video flows, we limit the packet size to 1500 bytes (Ethernet MTU). With each of the flows, there is an associated end-to-end delay requirement, that needs to be translated into a reservation rate R. In order to make this translation, it is necessary to make some assumptions about the propagation delay and the number of hops traversed by each of the flows. For simplicity, we make identical assumptions for each of the flows, namely assume that each of the flows traverses 5 hops, with a total propagation delay of 20ms. We also assume that the MTU on all links traversed by the flows is 1500 bytes. The *flowspecs* for each of the flows are listed in Table 7.1.

Subtracting the propagation delay from the end-to-end delay requirement, for each of the flows, we obtain the allowable end-to-end queueing delays. Substituting the end-to-end queueing delay in (7.8), we can solve for the rate, R, that needs to be reserved for each flow. Table 7.2 lists the flows' delay requirements along with the corresponding rate, R, that needs to be reserved for each of them.

Traffic Type	L kB	σ kB	ρ Mb/s	c Mb/s
64 Kb/s Voice	0.1	0.1	0.064	0.064
Video Conference	1.5	10	0.5	10
Stored Video	1.5	100	3	10

Table 7.1: Flow characteristics for 3 types of flows.

Traffic Type	e2e Delay (ms)	R (Mb/s)
64 Kb/s Voice	50	0.162
Video Conference	75	2.32
Stored Video	100	6.23

Table 7.2: End-to-end delay requirements and the rate reserved for each flow

If the entire 155 Mb/s of bandwidth could be used for G service traffic and the NEs were using the RCS discipline with EDF as the scheduler, then from (4.6) it can be verified that the following mix of traffic is feasible: 200 Voice flows, 26 Video Conference flows, 10 Stored Video flows.

Figure 7.2 graphically depicts the schedulability check of (4.6) at NE m for the traffic mix listed above. The straight line passing through the origin has a slope of 155 Mb/s, corresponding to the speed of the OC-3 link. The other curve is the sum of all the traffic envelopes, i.e.,

$$\sum_{n \in \text{flows}} \overline{A}_n(\tau - D_n^m),$$

where the scheduler deadline, D_n^m, is given by the $L_n/R_n + \widehat{L}^m/r^m$ value corresponding to flow n. It is clear from Figure 7.2, that quite a bit of link bandwidth remains available for flows that do not have too stringent delay requirements. In particular, observe that the sum of the reserved rate for the above sample traffic mix is around 155 Mb/s which is the link capacity. However, if you consider the average throughput, it is only 55.8 Mb/s, which means that only a third of the link is utilized. This poor utilization is not peculiar to the type of traffic mix that we have chosen. On the contrary, any flow that has a reserved rate R significantly larger than its average

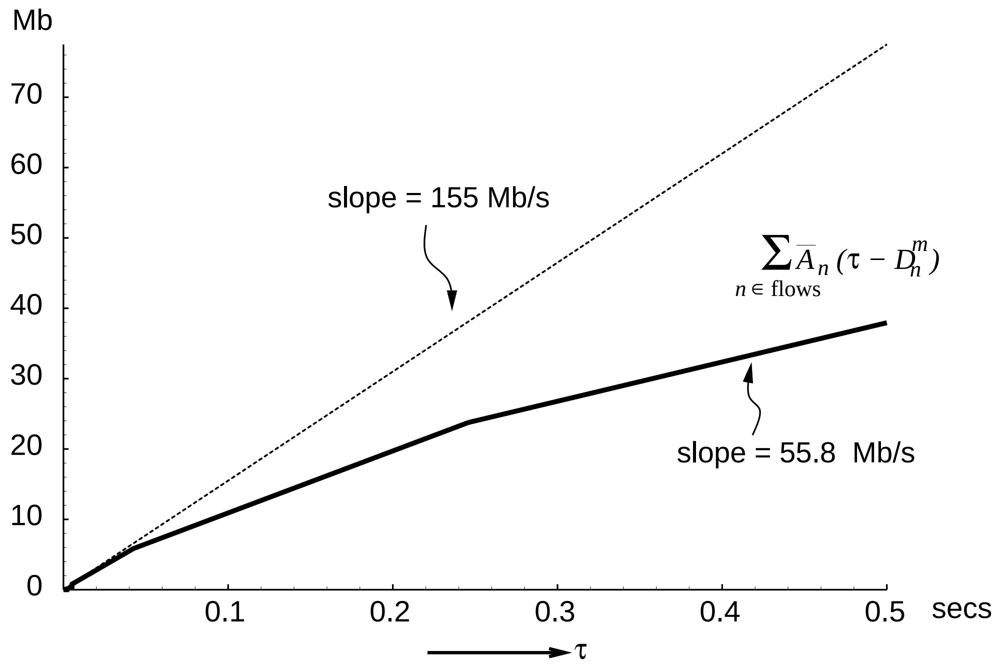

Figure 7.2: Schedulability check for EDF for traffic mix in Table 7.1.

rate, ρ, contributes to the low utilization of the link. For the flows that we have considered in our example, the reserved rate R varies with the maximum packet size that is assumed for each flow. Table 7.3 lists the rate R that needs to be reserved for each flow, assuming different maximum packet sizes, with everything else remaining the same. However, note that we assume that there is no fragmentation, i.e., the MTU on all the links is large enough to accommodate the maximum packet size of all the flows. This is unlikely to be true in practice for large packet sizes, e.g., 50kB. In such cases, fragmentation would take place so that the maximum packet size to consider would then be the MTU on the path of the flow.

Going back to the example, it is possible to add a flow with a deadline of 111 ms, and a throughput as high as 99 Mb/s, without affecting the maximum delays seen by the other flows that have been considered so far. It is this available capacity that we feel should be utilized by a new service, that would provide the same rate guarantees as the G service, but with a much looser delay guarantee. We refer to this as the Commmitted Rate (CR) service, and in the next section, describe it in some detail.

Max. Pkt Size L (kB)	Reservation Rate R (Mb/s)		
	64 Kb/s Voice	Video Conf.	Stored Video
0.1	0.16	1.40	5.91
0.5	0.81	1.66	6.00
1.0	1.62	1.99	6.11
1.5	2.43	2.32	6.23
5.0	8.35	4.87	7.07
10.0	17.50	9.15	8.36
25.0	50.96	24.71	16.31
50.0	140.37	57.01	35.77

Table 7.3: Variation of the Reserved Rate with the packet size

7.3 Committed Rate service

In this section, we describe a service that provides rate guarantees to flows, but unlike the G service, does not provide explicit delay guarantees. More precisely, a user making a Committed rate (CR) reservation for x Mb/s will be able to obtain a throughput of x Mb/s when measured over a fairly large period of time. This differs from a best effort type of service in that the user is *guaranteed* a certain amount of bandwidth that it will receive. No explicit guarantee is made as to the packet loss, but the buffers in the NEs should be engineered so that the packet loss can be made to be fairly small.

There are a number of applications that would benefit from a CR service. For example, an http session may be satisfied with such a kind of service guarantee, where the requested bandwidth depends on the size of the file that is being transmitted. For a large file, say a compressed movie clip, it would be preferable to have a much larger "bandwidth pipe" so that the entire file can be received in a few seconds. On the other hand it would be wasteful to have such a large pipe for a small file that contained only ascii text. With the G service, bandwidth and delay guarantees are coupled, and therefore it is not appropriate for applications that require only bandwidth guarantees. This is because such a coupling is typically more expensive in terms of resources than if only bandwidth guarantees are provided. In general, the CR service is suitable for most applications that like to have bandwidth guarantees, but for whom the G service would be an overkill.

At this point, it is worthwhile to contrast the CR service with the Controlled Load (CL) service proposed in [49]. The CL Service, requires the user to provide a *flowspec* that may be used for

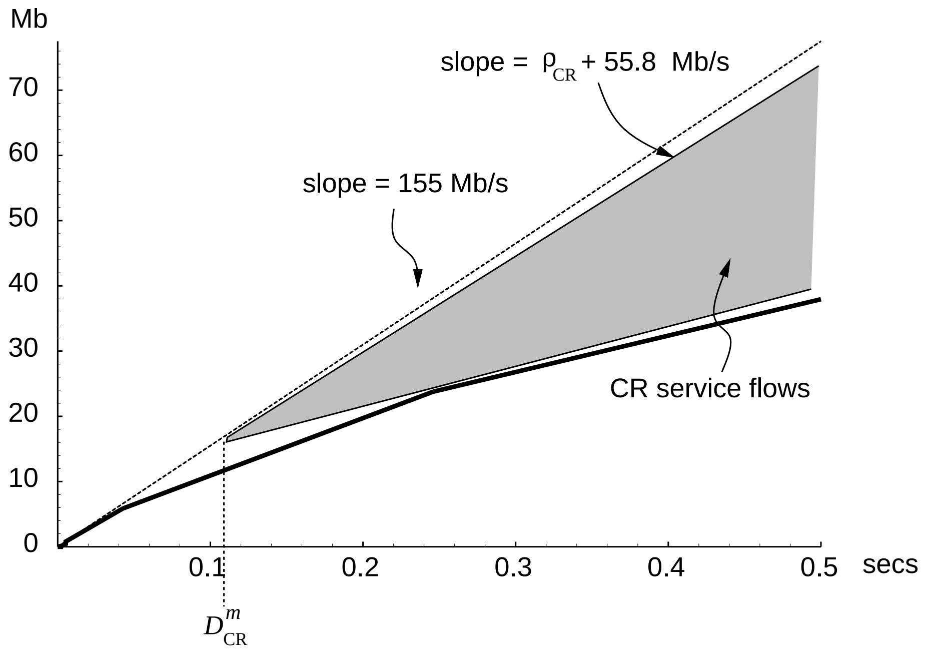

Figure 7.3: Schedulability check for EDF for traffic mix in Table 7.1 with the additional CR service flows.

admission control, but once a flow is accepted, it expects to see an “unloaded” network [49]. In other words, each flow will experience fairly small delays (and losses) as long as it conforms to its *flowspec*. The CR service on the other hand does not promise small end-to-end delays. Instead it only guarantees a certain throughput over some reasonable period of time. As a result, the delays experienced by a CR flow can temporarily be fairly large. This is because in order to use most of the bandwidth left available by the G service but without impacting the delay guarantees of the G service, it is necessary that the CR service be willing to tolerate occasional large delays. However, because of the worst case nature of the G service guarantees, instances where many G service packets are present and need to be sent out ahead of the CR service packets, should be relatively rare.

While we have demonstrated the availability of link bandwidth to support the CR service, it is not clear a priori whether this can be used to provide bandwidth guarantees over a reasonable period of time. In the next sections we investigate ways in which this service can be supported by both the RCS as well as the WFQ disciplines.

7.3.1 CR service with the RCS discipline

Consider Figure 7.3, that duplicates the schedulability check of Figure 7.2 for the G service flows at NE m. In addition, Figure 7.3 depicts a potential CR flow that can be added to the mix of G flows, assuming it is given a local deadline (at the EDF scheduler) of D^m_{CR} and has a traffic envelope, $\overline{A}_{\text{CR}} := \sigma_{\text{CR}} + \rho_{\text{CR}}\tau$, $\tau \geq 0$. This flow can be considered as representing the aggregate of all the CR service flows. It is clear from Figure 7.3 that as long as $\rho_{\text{CR}} + 55.8 \leq 155$, we can find some $D^m_{\text{CR}} \geq 0$ so that no deadlines of the G service flows will be violated at the EDF scheduler. In general, the parameters D^m_{CR}, σ_{CR}, and ρ_{CR} must be carefully determined to ensure that the CR service has a minimal impact on the G service flows carried by the NE.

The first step is to determine the local deadline D^m_{CR} that can be assigned to the CR service flows, so that they have absolutely no impact on the delay guarantees for G service flows at this NE. Consider a set of G service flows, numbered $1, \ldots, N$, that are multiplexed onto the output link under consideration. Let a flow n, have a *flowspec* of $(\sigma_n, \rho_n, c_n, L_n)$, and a reserved rate of R_n, $n = 1, \ldots, N$. From (7.6), we know that the envelope for flow n at this NE is given by

$$\overline{A}_n(\tau) = \min\{L_n + \xi_n\tau,\ \sigma_n + R_n\tau\}, \quad \tau \geq 0,\ n = 1, \ldots, N,$$

where $\xi_n := \min\{c_n, R_n\}$. The deadline associated with flow n at the EDF scheduler at node m is determined from Section 7.2.1 as $L_n/R_n + \widehat{L}^m/r^m =: D^m_n$, where r^m is the speed of the link. We assume that the deadlines for all the N Guaranteed Services flows are feasible at the EDF scheduler, which by (4.6) implies that the following constraint is satisfied:

$$\sum_{n=1}^{N} \min\left\{L_n + \xi_n(\tau - D^m_n)^+,\ \sigma_n + \rho_n(\tau - D^m_n)^+\right\} + \widehat{L}^m \leq r^m\tau, \quad \tau \geq 0, \tag{7.9}$$

where $(x)^+ \equiv \max\{x, 0\}$.

Now, assume that a CR service flow with an envelope of $\overline{A}_{\text{CR}}$, and a local scheduler deadline of D^m_{CR}, is multiplexed with the G service flows defined above. In order for a feasible schedule to exist, the following schedulability check has to be verified:

$$\sum_{n=1}^{N} \min\left\{L_n + \xi_n(\tau - D^m_n)^+,\ \sigma_n + \rho_n(\tau - D^m_n)^+\right\} \\ + \sigma_{\text{CR}} + \rho_{\text{CR}}(\tau - D^m_{\text{CR}})^+ + \widehat{L}^m \leq r^m\tau, \quad \tau \geq 0, \tag{7.10}$$

Letting $\tau \to \infty$ in (7.10), we get the following condition on ρ_{CR}:

$$\rho_{\text{CR}} \leq r^m - \sum_{n=1}^{N} \rho_n. \tag{7.11}$$

Given the total burst size, σ_{CR}, of the aggregate CR service flows, in order for a feasible schedule to exist, the local deadline assigned to the CR service flows must satisfy,

$$D_{\mathrm{CR}}^m \geq \min_{t \geq 0} \left\{ \min_{0 \leq \tau \leq t} \left\{ \tau : \sum_{n=1}^{N} \overline{A}_n(t - D_n^m) + \overline{A}_{\mathrm{CR}}(t - \tau) + \widehat{L}^m \leq r^m t \right\} \right\}$$

Assuming $\sigma_{\mathrm{CR}} = 100$ Kbytes, and $\rho_{\mathrm{CR}} = 99$ Mb/s, for the example considered in Section 7.2, we must have $D_{\mathrm{CR}}^m \geq 111$ msec. Figure 7.3 illustrates how the addition of this CR service flow with a deadline of around 111 msec to the schedulability check in Figure 7.2, still results in a feasible schedule.

From the previous discussion, it is clear that the values of D_{CR}^m and σ_{CR} and ρ_{CR} are local decisions that have to be made at each NE, depending on the G service flows that it typically carries. In particular, for the example considered in Section 7.2.2, we computed the appropriate values for these parameters. So far, we have described the CR service and explained how it can be efficiently supported by the RCS discipline. The question remains, as to whether this service, can be supported by other service disciplines. In the next section, we demonstrate that with a minor extension, the many variants of GPS, which sometimes are generically called Weighted Fair Queueing (WFQ) disciplines, can also be made to support the CR service.

7.3.2 CR service with the WFQ service discipline

The CR service can also be offered if the WFQ scheduling discipline is used at the NE. One way in which it can be offered, is to have two priorities. One for the G service flows, and the other for the CR service flows, with the CR flows being served only if there are no packets from the G service, that are waiting to be served. Figure 7.4, gives a possible representation of the scheduler, with HIGH denoting the higher priority for the G service flows, and LOW denoting the lower priority queues for the CR service flows. When a packet from any of the G Service flows is present, the CR service flows (indicated by the shaded queues in Figure 7.4) are completely ignored in the calculation of the weights for the WFQ discipline. The CR flows are served only when there are absolutely no packets in the G service queues, at which time, the WFQ discipline only uses the weights that are in the non-empty CR service flows to schedule packet transmissions. For purposes of illustration, each flow is depicted as having a queue dedicated to it, but the buffers can be shared among the different flows. However, it is advisable to separate the buffers used for the CR and the G service, so as to ensure that the CR service has a minimal impact on the G service.

Considering a single link m at a network element, we assume that there are N, G Service flows and N', CR service flows multiplexed onto an output link operating at the rate r^m. Let $R_1, R_2, \ldots R_N$, denote the rates reserved for the G service flows and $\beta_1, \beta_2, \ldots \beta_{N'}$, denote the rates committed to

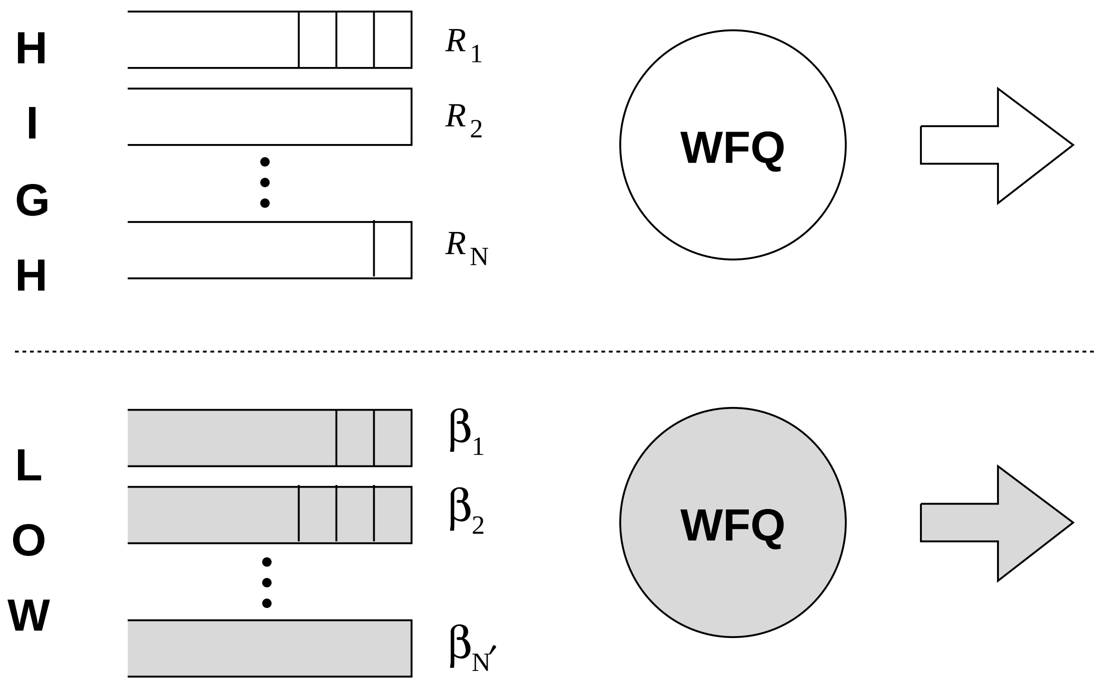

Figure 7.4: Prioritized, WFQ implementation of the CR service

the CR service flows. The schedulability check for the G service is simply

$$\sum_{n=1}^{N} R_n \leq r^m. \tag{7.12}$$

For the CR service it is necessary to check that there is sufficient capacity for both the flows, namely

$$\sum_{n=1}^{N} \rho_n + \sum_{n=1}^{N'} \beta_n \leq r^m. \tag{7.13}$$

Since the WFQ implementation for the CR service flows comes into play only when there are no G service packets, it is clear that there is almost no impact on the G service flows that are currently being carried by the NE. At most a G service packet has to endure an extra packet delay while the server is off serving the CR service flows. But this extra packet delay is already accounted for in the delay bounds that are known for WFQ and so there is really no difference in terms of the end-to-end delay guarantee [36, 26].

On the other hand, the worst case delay encountered by the CR service flows are affected by the G service traffic, and this can be analyzed using the service curve framework [15]. Clearly, the service curves for each of the CR service flows, strongly depends on the traffic characteristics of the G service flows. Let us assume that the G service flows can be characterized by the envelopes $\overline{A}_n$, $n =$

$1, \ldots, N$. Note that, these envelopes can in general, be different from the flow characteristics at the network ingress, since they have to account for the possible increase in burstiness caused by all the NEs on the path. If the flows are being reshaped at each hop, then they can be characterized by the envelopes of the reshapers themselves.

Given the traffic envelopes of the G service flows, we know that if the CR service queues are back-logged in the interval $[t, t+\tau]$, then they collectively receive a service of at least

$$\overline{S}_{\mathrm{CR}}^{m^*}(\tau) = \left[r\tau - \sum_{n=1}^{N} \overline{A}_n(\tau) \right]^+, \quad \tau \geq 0.$$

Assuming that the aggregate CR service traffic is characterized by the envelope $\overline{A}_{\mathrm{CR}}$, then the delay encountered by any CR service packet is upper bounded by

$$D_{\mathrm{CR}}^{m^*} = \max_{t \geq 0} \left\{ \min_{\tau \geq 0} \left\{ \tau : \overline{A}_{\mathrm{CR}}(t) \leq \overline{S}_{\mathrm{CR}}^{m^*}(t+\tau) \right\} \right\} \tag{7.14}$$

Equation (7.14) is simply the horizontal distance between the traffic envelope, $\overline{A}_{\mathrm{CR}}$, and the service curve $\overline{S}_{\mathrm{CR}}^{m^*}$ (see Figure 7.1 for a graphical illustration of the delay computation). For the example considered in Section 7.2, we can readily compute the upper bound on the delay experienced by the CR flows to be $D_{\mathrm{CR}}^{m^*} = 111$ msec, as before, and this is illustrated in Figure 7.5. The value of $D_{\mathrm{CR}}^{m^*}$ (or D_{CR}^{m} for the RCS discipline), gives an indication of the buffer requirements that are needed to ensure that packets from the CR service flows are not lost. Regardless of which scheduling discipline is used, i.e., RCS or WFQ, it is necessary to have sufficient buffers to ensure fairly low packet loss for the CR service flows. Since the value of $D_{\mathrm{CR}}^{m^*}$ (or D_{CR}^{m}) can be quite large, it may no longer be possible to engineer the buffer sizes based on a worst case analysis of the flows. In light of this the CR service should not be used by applications that require very stringent loss guarantees to be provided by the network. In general, if tight delays are not a requirement, it is much more efficient to have an end-to-end recovery mechanism for the lost packets, a good example being TCP/IP.

7.4 Summary

In this chapter, we examined how delay guarantees can be provided to the end-user in the context of the Internet. The delay requirements of the user have to somehow be translated at the network level so that individual network elements can reserve sufficient resources to provide a satisfactory level of service to the end-user. We considered the Guaranteed Services specification[49], which is soon to become a proposed standard for the Internet and demonstrated how an RCS discipline enables the network element to efficiently support this service. In addition, through the help of an example we demonstrated how the Guaranteed Services specification only allows a limited utilization of the link. We proposed that this unutilized bandwidth be used to support another type of service called

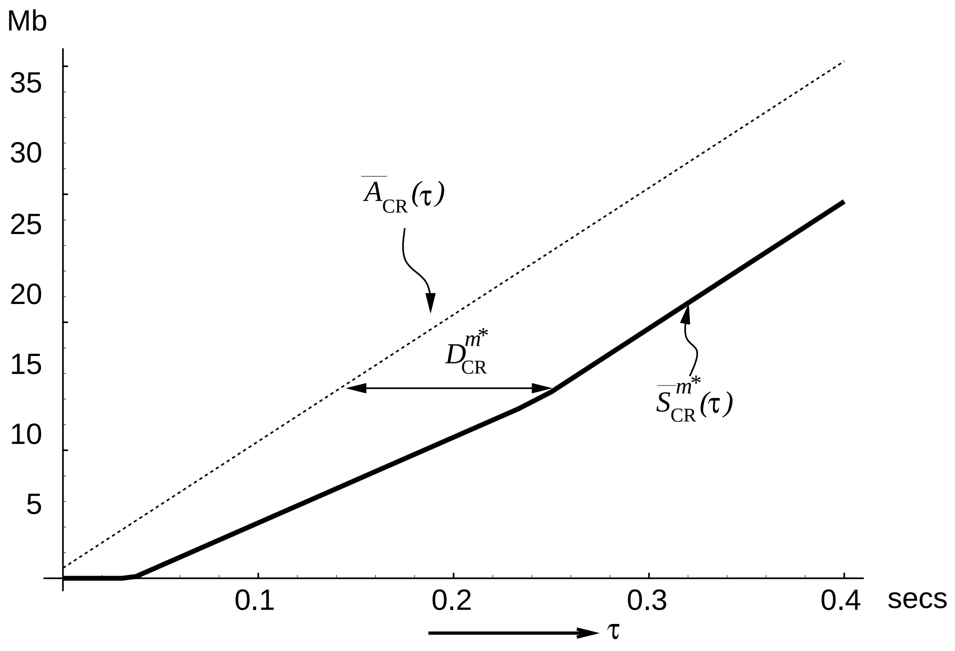

Figure 7.5: Illustration of the computation of $D^{m^*}_{\mathrm{CR}}$ for the WFQ discipline

the Committed Rate service[22]. We outline how the Committed Rate Service can be be provided by both the RCS disciplines as well as the GPS-based service disciplines.

Chapter 8

Conclusion and Future Work

One of the primary goals of this thesis was to establish a framework in which the network can efficiently provide end-to-end delay guarantees to individual connections. Towards this end we established that Rate Controlled Service (RCS) disciplines offer a powerful solution. Specifically, we established that with the right choice of scheduler and shaper parameters, the RCS discipline provides better end-to-end delay guarantees than any other scheduling discipline that is known today.

An RCS discipline has two main components which are the per-connection traffic shapers and a link scheduler. We examined each of these with respect to their impact on the end-to-end delay guarantees. Intuitively, it might seem that the per-hop traffic shapers add to the end-to-end delay guarantees that can be obtained by summing up the scheduler delay bounds along the path of the connection. However, by deriving some general properties of traffic shapers we showed that if the connection encountered the same traffic shapers at all nodes, then it is only the first one that adds to the end-to-end delay bound. Furthermore, we established that for the same connection it does not pay to have different traffic shapers at each of the nodes.

A Traffic Shaper is characterized by a shaper envelope. Given the shaper envelope as well as an envelope on the input traffic we obtained a bound on the delay experienced in the shaper. In addition we tackled the inverse problem of finding the "minimal" shaper given a certain maximum delay that could be tolerated by the connection.

We identified the Earliest Deadline First (EDF) scheduler as being the scheduler of choice for the RCS discipline. For an RCS discipline employed an EDF scheduler, we found that if there are a sufficient number of hops along the path of a connection, it is advantageous to smooth the traffic to its average rate at the first shaper itself. We also derived shaper envelopes that can be used to provide better end-to-end delay guarantees compared to GPS-based service disciplines which, until now, provided the best known end-to-end delay guarantees.

The delay guarantees that we have computed assume sufficient buffering in the network to hold all the packets that are delayed. So, in addition to providing a delay guarantee, the network element has to ensure that it has sufficient buffers to accommodate the flow. On a per-connection basis, we computed the buffers required at each network element to ensure that packets are not dropped. Also, we demonstrated that for a constant bit rate stream the total jitter that could be introduced by the RCS disciplines was bounded above by the end-to-end delay guarantee.

RCS disciplines are basically non-work-conserving. We outlined a modification to the RCS discipline that makes it work conserving without sacrificing the end-to-end delay guarantees that can be provided. Additionally, if the link scheduler uses an EDF policy we developed a relatively simple service discipline that also provides the same end-to-end delay guarantees. We also analyzed the buffer requirements for each of these disciplines, which, as expected, were found to be larger than those required for RCS disciplines.

We concluded this thesis by examining how RCS disciplines can be used in the Internet, which is by far the most ubiquitous network today. We observed that the RCS disciplines can be used to support the Guaranteed (G) services specification which is currently a proposed standard for the Internet. We determined the parameters that need to be exported by a network element that employs the RCS discipline. By considering a typical example we observed that if the network only supports the G services flows, its links are likely to be significantly underutilized. We proposed a Committed Rate (CR) service that utilizes the bandwidth left over by the G service flows. We demonstrated how the CR service could be supported by both RCS and WFQ disciplines, although RCS disciplines were found to offer the benefit of an implementation synergetic with the support for G service.

8.1 Future Work

In this thesis we developed a framework in which efficient end-to-end delay guarantees can be provided. However, several applications may be satisfied with an end-to-end delay quantile, as opposed to firm delay guarantees. A useful continuation of this work would be to extend the framework developed here to provide statistical delay guarantees on a per-connection basis.

One of the basic assumptions in this thesis was the decoupling between routing and Quality of Service (QoS) support. This assumption is also made in the specification for Integrated Services support for the Internet. Ideally, we would like the best possible route to be chosen given the QoS required by the connection. Within the framework developed here, it would be useful to identify the parameters that the RCS discipline would need to export to the routing layer so that QoS can be taken into account during the route selection process.

In this thesis we examined the tradeoff between the shaper envelopes and the end-to-end delay guarantees for some specific types of traffic envelopes. It would be useful to know what the optimal shaper envelopes are so that the best possible delay bounds can be guaranteed for a given set of connections and paths.

With the rapid escalation in the speed of links we are already at the stage where switch technology can no longer keep up. If switches are not significantly faster than the input links one cannot afford to only have queueing at the output of the switch. With input queueing we need to consider the well known head-of-the-line blocking problem. One possible solution to this problem is to maintain several queues at the switch input, one corresponding to each switch output. However, we then need a switch scheduler to arbitrate between the several queues at each of the inputs. We are currently investigating scheduling policies that can provide delay and throughput guarantees through the switch fabric in cases where there is both input and output queueing.

Bibliography

[1] Contributions on the Int-Serv Mailing list.

[2] *ATM UNI Specification, Version 3.1,* ATM Forum, September 1994.

[3] *Draft of UNI Signalling 4.0,* ATM Forum 94-180R5, April 1995.

[4] CSM-622-2E and CSM-622-2SE data sheet. Xylan Corp, Calabassas, California 91302., 1997.

[5] F. Baker, R. Guérin, and D. Kandlur. Specification of committed rate quality of service. Internet Draft, `draft-ietf-intserv-commit-rate-svc-00.txt`, Integrated Services WG, IETF, June 1996.

[6] A. Banerjea, D. Ferrari, B. A. Mah, M. Moran, D. C. Verma, and H. Zhang. The tenet real-time protocol suite: Design, implementation, and experiences. *IEEE/ACM Transactions on Networking*, 4(1):1–10, February 1996.

[7] V. E. Beneš. *General Stochastic Processes in the Theorey of Queues.* Addison-Wesley series in Statistics. Addison-Wesley, 1963.

[8] J.C.R. Bennett and Hui Zhang. WF^{2}Q: Worst-case fair weighted fair queueing. In *to appear in Proceedings of the IEEE INFOCOM '96*, San Francisco, CA, March 1996.

[9] R. Braden, et. al. Resource ReSerVation Protocol (RSVP) - version 1, functional specification. Internet Draft, `draft-ietf-rsvp-spec-15.txt`, May 1997.

[10] C.-S. Chang. Stability, queue length and delay of deterministic and stochastic queueing networks. *IEEE Transactions on Automatic Control*, 39(5):913–931, May 1994.

[11] C-S. Chang. A filtering theory for deterministic traffic regulation. In *Proceedings of the IEEE INFOCOM*, Kobe, Japan, 1997.

[12] D. Clark, S. Shenker, and L. Zhang. Supporting real-time applications in an integrated packet network: Architecture and mechanisms. In *Proceedings of ACM SIGCOMM'92*, Baltimore, MD, August 1992.

[13] R. L. Cruz. A calculus for network delay, Part I: Network elements in isolation. *IEEE Transactions on Information Theory*, 37(1):114–131, Jan 1991.

[14] R. L. Cruz. A calculus for network delay, Part II: Network analysis. *IEEE Transactions on Information Theory*, 37(1):132–141, January 1991.

[15] R. L. Cruz. Quality of service guarantees in virtual circuit switched networks. *IEEE Journal on Selected Areas in Communication*, 13(6):1048–1056, August 1995.

[16] R. L. Cruz and C. M. Okino. Service guarantees for window flow control. In *Thirty-Fourth Annual Allerton Conference on Communication, Control, and Computing*, Monticello, Illinois, October 1996.

[17] G. Delp, J. Byrn, M. Branstad, K. Plotz, P. Leichty, A. Slane, G. McClannahan, and M. Carnevale. *ATM function specifications: CHARM introduction - version 3.* IBM AS/400 Division, LAN Development. Technical Report ROCH431-Charm Intro. IBM Confidential.

[18] A. Demers, S. Keshav, and S. Shenker. Analysis and simulation of a fair queueing algorithm. *Journal of Internetworking: Research and Experience*, 1:3–26, January 1990.

[19] M. L. Dertouzos. Control robotics: The procedural control of physical processes. In *Proceedings of the IFIP Congress*, pages 807–813. North-Holland publishing company, 1974.

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

[21] L. Georgiadis, R. Guérin, and A. Parekh. Optimal multiplexing on a single link: Delay and buffer requirements. In *Proceedings of the IEEE INFOCOM '94*, 1994.

[22] L. Georgiadis, R. Guérin, V. Peris, and R. Rajan. Efficient support of delay and rate guarantees in an internet. In *Proceedings of the ACM SIGCOMM'96 Conference*, pages 106–116, Stanford University, CA, 1996.

[23] L. Georgiadis, R. Guérin, V. Peris, and K. N. Sivarajan. Efficient network QoS provisioning based on per node traffic shaping. *IEEE/ACM Transactions on Networking*, 4(4):482–501, 1996.

[24] S. J. Golestani. A stop-and-go queueing framework for congestion management. In *Proceedings of ACM SIGCOMM'90*, pages 8–18, Philadelphia, PA, Sept. 1990.

[25] S. J. Golestani. A self-clocked fair queueing scheme for broadband applications. In *Proceedings of IEEE INFOCOM'94*, pages 636–646, Toronto, Canada, June 1994.

[26] P. Goyal and H. Vin. Generalized guaranteed rate scheduling algorithms: A framework. Technical Report TR-95-30, Department of Computer Sciences, University of Texas at Austin, 1995.

[27] P. Goyal, H. Vin, and H. Chen. Start-time fair queueing: A scheduling algorithm for integrated services packet switching networks. In *Proceedings of the ACM SIGCOMM'96 Conference*, pages 157–168, Stanford University, CA, 1996.

[28] J. Y. Hui and E. Arthurs. A broadband packet switch for integrated transport. *IEEE Journal of Selected Areas in Communication*, 5(8):1264–1273, October 1987.

[29] C. Kalmanek, H. Kanakia, and S. Keshav. Rate controlled servers for very high-speed networks. In *Proceedings of the IEEE GLOBECOM*, pages 300.3.1–300.3.9, San Diego, CA, Dec. 1990.

[30] D. D. Kandlur, K. G. Shin, and D. Ferrari. Real-time communication in multi-hop networks. In *Proceedings of IEEE INFOCOM '91*, pages 300–307, 1991.

[31] D. Knuth. *The Art of Computer Programming, Vol 3: Sorting and Searching.* Addison-Wesley, 1975.

[32] J. Kurose. On computing per-session performance bounds in high-speed multi-hop computer networks. *Performance Evaluation Review*, 20(1), June 1992.

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

[34] P. Pancha and M. El Zarki. Leaky bucket access control for VBR MPEG video. In *Proceedings of the IEEE INFOCOM'95*, Boston, April 1995.

[35] A. K. Parekh and R. G. Gallager. A generalized processor sharing approach to flow control in integrated services networks: The single-node case. *IEEE/ACM Transactions on Networking*, 1(3):344–357, June 1993.

[36] A. K. Parekh and R. G. Gallager. A generalized processor sharing approach to flow control in integrated services networks: The multiple node case. *IEEE/ACM Transactions on Networking*, 2(2):137–150, April 1994.

[37] A. K. J. Parekh. *A Generalized Processor Sharing Approach to Flow Control in Integrated Services Networks.* PhD thesis, Laboratory for Information and Decision Systems, Massachusetts Institute of Technology, Cambridge, MA 02139, February 1992. No. LIDS-TH-2089.

[38] D. Pollard. *Convergence of Stochastic Processes.* Springer-Verlag, 1984.

[39] H. Sariowan. *A Service-Curve Approach to Performance Guarantees in Integrated Service Networks*. PhD thesis, University of California, San Diego, 1996.

[40] S. Shenker and L. Breslau. Two issues in reservation establishment. In *Proceedings of the ACM SIGCOMM '95*, Cambridge, MA, August 1995.

[41] S. Shenker, C. Partridge, B. Davie, and L. Breslau. Specification of predictive quality of service. Internet Draft, `draft-ietf-intserv-predictive-svc-01.txt`, Integrated Services WG, IETF, 1995.

[42] S. Shenker, C. Partridge, and R. Guérin. Specification of guaranteed quality of service. Internet Draft, `draft-ietf-intserv-guaranteed-svc-07.txt`, Integrated Services WG, IETF, February 1997.

[43] S. Shenker, C. Partridge, and J. Wroklawski. Specification of controlled delay quality of service. Internet Draft, `draft-ietf-intserv-control-del-svc-02.txt`, Integrated Services WG, IETF, November 1995.

[44] S. Shenker and J. Wroklawski. Network element service specification template. Internet Draft, `draft-ietf-intserv-svc-template-03.txt`, Integrated Services WG, IETF, November 1996.

[45] J. A. Stankovic, M. Spuri, M. Di Natale, and G. Buttazzo. Implications of classical scheduling results for real-time systems. *IEEE Computer*, May 1995.

[46] D. Stiliadis and A. Varma. Design and analysis of frame-based fair queueing: a new traffic scheduling algorithm for packet-switched networks. In *Proceedings of ACM SIGMETRICS*, pages 104–115, May 1996.

[47] J. S. Turner. New directions in communications (or which way to the information age?). *IEEE Communications Magazine*, 24(10):8–15, October 1986.

[48] D. C. Verma, H. Zhang, and D. Ferrari. Guaranteeing delay and jitter bounds in packet switching networks. In *Proceedings of Tricomm '91*, pages 35–46, 1991.

[49] J. Wroclawski. Specification of the controlled-load network element service. Internet Draft, `draft-ietf-intserv-ctrl-load-svc-04.txt`, Integrated Services WG, IETF, November 1996.

[50] G. G. Xie and S. S. Lam. Delay guarantee of virtual clock server. *IEEE/ACM Transactions on Networking*, 3(6):683–689, Dec 1995.

[51] O. Yaron and M. Sidi. Performance and stability of communication networks via robust exponential bounds. *IEEE/ACM Transactions on Networking*, 1(3), June 1993.

[52] H. Zhang. *Service Disciplines for Packet-Switching Integrated-Services Networks.* PhD thesis, Computer Science, University of California at Berkeley, 1993.

[53] H. Zhang. Service disciplines for guaranteed performance service in packet-switching networks. *Proceedings of the IEEE*, 83(10):1374–1396, Oct 1995.

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

[55] H. Zhang and D. Ferrari. Rate-controlled service disciplines. *Journal of High Speed Networks*, 3(4):389–412, 1994.

[56] L. Zhang. Virtual clock: A new traffic control algorithm for packet switching networks. In *Proceedings of the ACM SIGCOMM'90*, pages 19–29, Philadelphia, PA, Sept. 1990.

[57] Z.-L. Zhang, D. Towsley, and J. Kurose. Statistical analysis of the generalized processor sharing discipline. *IEEE Journal of Selected Areas in Communication*, 13(6):1071–1080, Aug. 1995.