Quality of Service Modeling and Analysis for Carrier Ethernet

Richa Malhotra

CTIT Ph.D.-thesis Series No. 08-126
Centre for Telematics and Information Technology
University of Twente, P.O. Box 217, 7500 AE Enschede, Netherlands

ISBN 978-90-365-2711-8

Printed by DeltaHage BV, Delft, The Netherlands

Most of this research has been sponsored by the Netherlands Organisation for Scientific Research (NWO).

QUALITY OF SERVICE MODELING AND ANALYSIS FOR CARRIER ETHERNET

PROEFSCHRIFT

ter verkrijging van
de graad van doctor aan de Universiteit Twente,
op gezag van de rector magnificus,
prof. dr. W.H.M. Zijm,
volgens besluit van het College voor Promoties,
in het openbaar te verdedigen
op vrijdag 31 oktober 2008 om 13.15 uur

door

Richa Malhotra

geboren op 29 maart 1976
te Amritsar (Punjab), India

Dit proefschrift is goedgekeurd door de promotoren
prof. dr. J.L. van den Berg
prof. dr. M.R.H. Mandjes

Acknowledgements

The journey to completing this PhD has not always been an easy one especially since I did it in combination with my work and family life. I would like to thank all those people who assisted and supported me in this effort.

Firstly, I would like to thank my promotors for guiding me through the research and having faith in my determination to complete the thesis. Michel, you helped and supervised me in spite of your move from Twente to Amsterdam and also during your stay at Stanford. Hans, I am glad you guided me more frequently. Your multiple thorough revisions of the thesis were very useful.

Combining my PhD research with my job at Alcatel-Lucent would not have been possible without the support of my managers. I would like to thank Harold Teunissen, Paul Reinold and Michael Doubrava for supporting my ambition to complete this thesis.

I would like to express my gratitude to NWO for funding the project, which assisted me in finalizing my research and this dissertation. I am especially grateful to Nick den Hollander who was very helpful in finding solutions during difficult and uncertain times. Furthermore, I appreciate Boudewijn Haverkort's efforts for supporting this project at the University of Twente. I am also thankful to my fellow project members in the NOBEL, DAIDALOS and EQUANET projects.

My (ex) colleagues at Alcatel-Lucent provided the stimulating environment for my research. Ronald, we have worked together on many Ethernet related topics and wrote several papers together. Your support with the simulation environment was especially useful. Arjan, discussions with you on the practical and standards related issues for Ethernet were very helpful. Maarten, I benefited from your experience with your PhD, which you did while working at Alcatel-Lucent. I want to thank you for your advice and willingness to answer any questions I had. I would also like to thank fellow colleagues from Alcatel-Lucent sales and product units. Discussions and interactions with them greatly contributed and shaped my understanding of Ethernet networks in general. Especially Gert Manhoudt, Michiel van Everdingen, Jeff Towne and Stephan Roullot were always very open to the innovative ideas we proposed. I would also like to thank Sue Atkins, Dirk-Jaap Plas, Dennis Bijwaard, Ronald de Man and my Bell Labs Europe Hilversum colleagues.

For parts of my research I worked together with Werner Scheinhardt and Sindo

Núñez Queija. I want to thank them both for the fruitful discussions I had with them.

I have been a visiting researcher at the DACS group at the University of Twente for my research. I would like to thank all my colleagues there who made it a pleasant and productive environment: Pieter-Tjerk de Boer, Lucia Cloth, Desislava Dimitrova, Tiago Fioreze, Geert Heijenk, Assed Jehangir, Marijn Jongerden, Georgios Karagiannis, Fei Liu, Silvia Meijran, Giovane Moura, Aiko Pras, Anne Remke, Ramin Sadre, Anna Sperotto and Yimeng Yang.

Friendships and family ties I have built in the Netherlands are very important to me especially because I came here leaving my own family behind in India. I would first like to express my gratitude to Joke and Rolf Soetbrood. It was their support, which gave me courage to continue my work here in the Netherlands, in spite of some very difficult personal circumstances. I would also like to thank my in-laws and friends for the warm and pleasant gatherings, especially Ans, Pieter, Sylvia, Willie, Claudia, Anton, Jolanda, Wouter, Nelly, Pranab, Marja, Chetna, Amrish, Wietze, Lies, Jayanthi, Seshan, Isabelle and Nilesh.

I am grateful to my parents, Kanchan and Jeewan as well as Riti, Suman and Punit for their unconditional love and support. My father has a special connection to this thesis. If it were not for his encouragement, I would have never come to the Netherlands and completed this PhD. My husband Ronald deserves the most special acknowledgement. Not only have we worked and published papers together as colleagues, but he has also been extremely supportive on the personal front. I could always count on him not just for taking over the responsibilities at home but also for last minute help with reviews, Dutch translations and bug fixes. Finally, I am thankful to my daughters Deepshikha and Nisha for providing the much needed break from the frequent stresses resulting from my PhD work.

Contents

Chapter 1

Introduction

Until the early 1990s, networks were dominated by TDM switching and transmission. ATM and SONET were new, and the Internet was in its infancy. Ethernet was a Local Area Network (LAN) technology, and ATM was supposed to displace Ethernet all the way to the desktop. Today's networking landscape is quite different from that vision. Rather than ATM to the desktop, the reverse has happened. Ethernet which was predominantly a LAN technology has started to penetrate as a transport technology – first in the metropolitan area, then in access and core networks. The success of Ethernet is probably best demonstrated by its increasing revenues despite the recent downturn in the telecommunications market. Worldwide Ethernet equipment revenues have increased from \$2.5 billion in 2004 to \$13 billion in 2007 and are expected to reach \$16 billion by 2010 ([33]). ATM switch revenues on the other hand have declined from \$5 billion in 2000 to \$1.3 billion in 2006 ([34]).

Ethernet initially played an important role in the emergence of the metropolitan area networking market where its use gave it the name *Metro Ethernet*. It provided an easy and cheap way to interconnect for example multiple sites of the same enterprise by means of a Virtual LAN (VLAN) giving the end user the illusion of being on the same LAN. A similar VLAN could also be realized between a residential end-user and his Internet Service Provider (ISP) providing high speed Internet access as shown in Figure 1.1.

Today, Ethernet is moving into the mainstream evolving into a carrier grade technology. Termed as *Carrier Ethernet* it is expected to overcome most of the shortcomings of native Ethernet. It is envisioned to carry services end-to-end serving corporate data networking and broadband access demands as well as backhauling wireless traffic as shown in Figure 1.2.

As the penetration of Ethernet increases, the offered Quality of Service (QoS) will become increasingly important and a distinguishing factor between the different service providers. The challenge is to meet the QoS requirements of end applications such as response times, throughput, delay and jitter by managing the network resources at hand. Since Ethernet was not designed to operate in large public networks

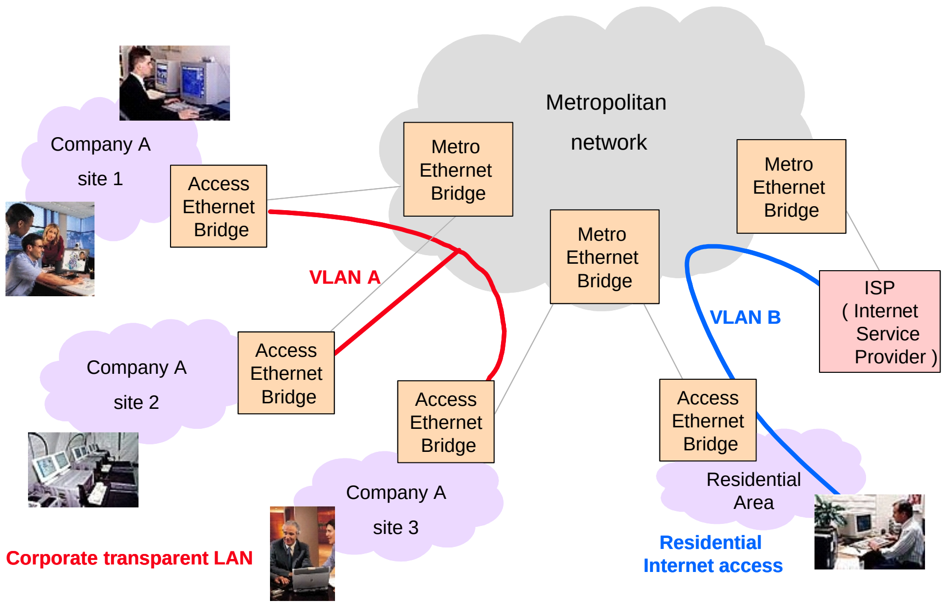


Figure 1.1: Metropolitan Ethernet Network.

it does not possess functionalities to address this issue. In this thesis we propose and analyze mechanisms which improve the QoS performance of Ethernet enabling it to meet the demands of the current and next generation services and applications.

In the rest of this thesis we use the terms *Carrier Ethernet* and *Metro Ethernet* interchangeably. This is because the research presented in this thesis, on one hand, improves Ethernet and helps it become *carrier-class* and on the other hand, its applicability is not restricted to the size or extent of the network (metro, access or core).

This introductory chapter further on presents the context of our research, posing the research questions to be addressed and outlining the objective, scope, structure and contributions of the thesis. The technological details of Carrier Ethernet are presented in Chapter 2.

1.1 QoS in Carrier Ethernet

The success of Carrier Ethernet depends greatly on its ability to live up to the QoS demands of the applications delivered over it. In this respect, the inherent variations in user traffic cause unpredictable congestion patterns and pose difficulties for QoS provisioning. Efforts are underway to address this issue for Carrier Ethernet. However, still many challenges remain, which have to be overcome ([19]). In this section we address the status of QoS features in Ethernet (in Section 1.1.1), identify what is still missing, and mention which of these missing elements will be studied

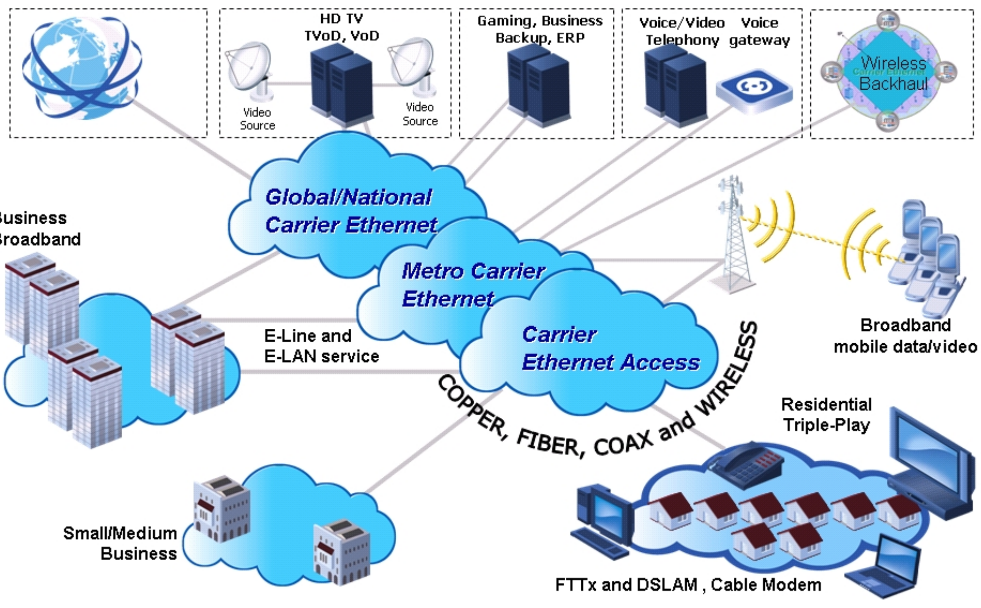


Figure 1.2: Metro Ethernet forum's vision for Carrier Ethernet (source [20]).

in this monograph (in Section 1.1.2).

1.1.1 Current state

In this section we present the QoS features currently available and enforced by standardization bodies for Carrier Ethernet. These QoS attributes focus on building customer confidence in Ethernet, which is extremely important at the current stage of Ethernet deployments. This is done primarily by enabling the formalization of strict performance agreements for different Ethernet services and making the service provider accountable for them. Following are the QoS features which should be available in current Carrier Ethernet products.

- *Class of Service*: Class of service refers to the classification of traffic into multiple classes or groups. For Carrier Ethernet this is possible with the *p-bits* in the Ethernet frame header as explained in [73]. Once traffic is classified into separate groups, it can then be treated differently depending on its QoS requirements.
- *Service Level Agreements* (*SLAs*): A SLA is a commercial agreement binding both the service provider and its customer to a specified level of service. In Carrier Ethernet it should currently be possible to define bandwidth profile attributes such as the traffic rates and maximum burst sizes per customer as part of its SLA. Furthermore, service performance attributes such as packet

delay, packet delay variation and packet loss ratio should also be supported (see MEF's certification rules, [21]).

- *Operation, Administration and Management* (*OAM*): OAM methods are monitoring functionalities which report on the performance achieved by customer traffic streams. These results can then be compared to the SLA to assess if the service provider has lived up to its promised targets. If not, this can be incorporated in the pricing and billing options for the customer. This OAM functionality has been standardized [65] and service providers claiming to use Carrier Ethernet should support it.

1.1.2 Missing QoS elements

The SLAs and the OAM methods are essential in building the customer's confidence in using Ethernet services, as they make the service provider liable for the delivered performance. However, they fail to answer a critical question:

> *If the OAM methods show that the performance targets agreed in the SLAs are not being met, what actions can the service provider take to fix this?*

In order to deal with this issue, the service provider needs tools and techniques to optimize and tune the operation of his network to ensure that the SLAs can be guaranteed. In this respect, the list of QoS features presented in Section 1.1.1 is incomplete.

In this section we assess which QoS features are still missing in Carrier Ethernet today and mention the ones which will be researched in this thesis. For this purpose, we review a general QoS framework for packet technologies in Figure 1.3 (from [35]). This lists the complete set of mechanisms required for QoS provisioning and is organized into three logical planes: control, management and data plane.

The *control plane* mechanisms deal with the pathways that carry the user data traffic. It includes admission control, QoS routing and resource reservation mechanisms. Admission control refers to the act of accepting or rejecting a user traffic connection based on a particular policy. QoS routing refers to finding a path for each traffic connection such that its quality requirements can be met. And resource reservation is the act of reserving network resources once a traffic connection has been accepted by admission control.

The *management plane* functionalities deal with operation, administration and management aspects of user traffic. It includes metering, policy, SLAs and service restoration mechanisms. SLA has already been defined in Section 1.1.1. Metering involves monitoring the traffic streams against the traffic profile that is usually specified in the SLA. Policy is a set of rules which helps decide on the admission of

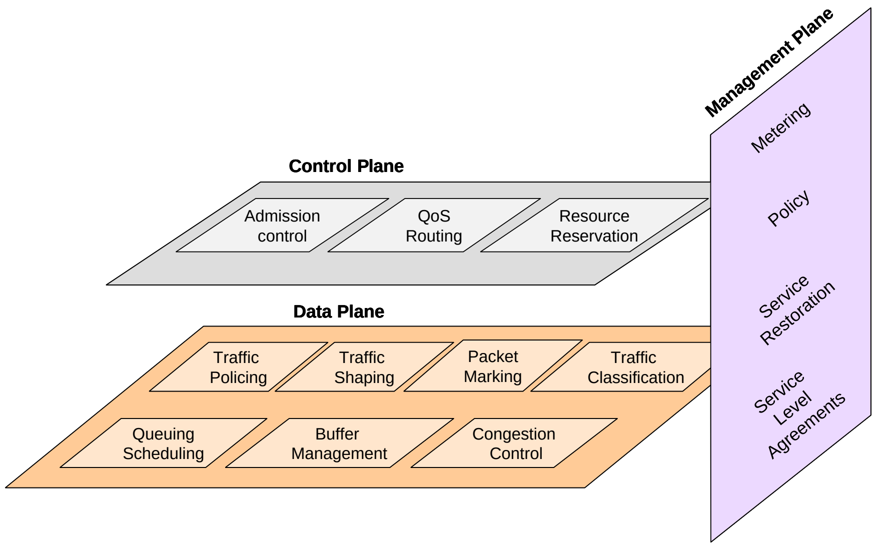


Figure 1.3: QoS building blocks.

new users or customers in the network. Service restoration relates to methods for recovery consequent to a failure in the network.

The *data plane* mechanisms deal directly with user traffic. Traffic classification relates to the ability to classify incoming traffic into multiple classes or groups (see also Section 1.1.1). Buffer management involves deciding on which of the packets, awaiting transmission, should be dropped or stored. queueing and scheduling deals with the selection and ordering of data packets for transmission on the outgoing link. In combination with traffic classification this leads to division of network bandwidth among the different traffic classes. Congestion avoidance controls the traffic load such that it remains below the network capacity. Packet marking involves marking data out of the SLA traffic profile. Traffic shaping regulates the rate of traffic leaving a node. Traffic policing involves monitoring and enforcing the traffic limit agreed upon in the SLA at the edge nodes of the network.

In this thesis we focus on the data plane mechanisms for a typical Metro Ethernet network (as shown in Figure 1.4). This is because a thorough understanding of the performance at the data plane is needed to develop provisioning methods and guidelines. These methods and guidelines can then be used for network planning by exploiting the management and control plane functionality. For example, insight into the influence of network and traffic parameters on performance can result in network provisioning tools or included as policy and admission control decisions.

It is important to note that an alternative approach exists for QoS provisioning, i.e., without using the mechanisms mentioned above. *Over-dimensioning* of resources ensures that enough bandwidth is available for all data transport all the time. The basic idea behind this approach is that if the available resources are

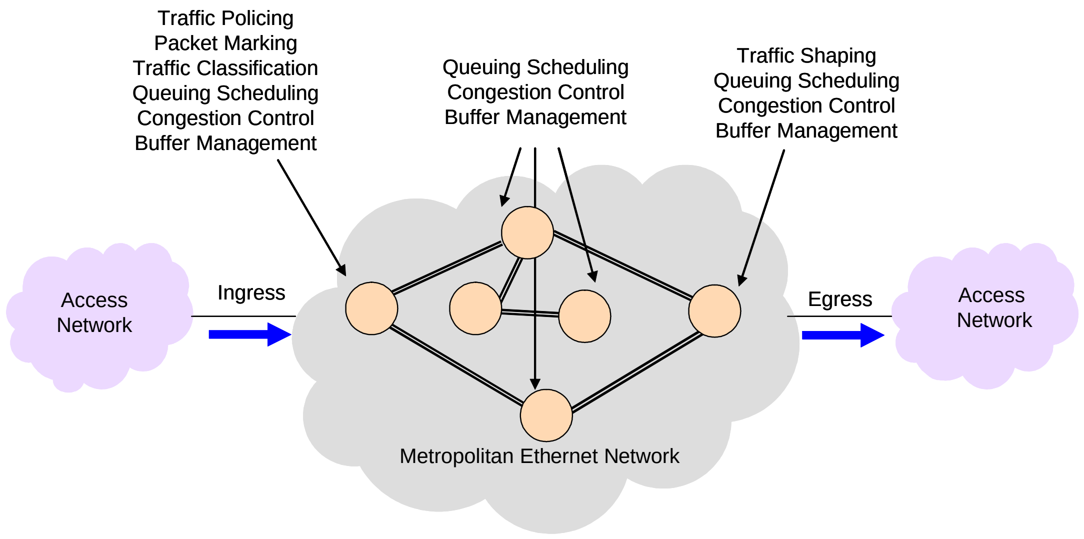


Figure 1.4: Mapping of data plane QoS mechanisms onto a Metro Ethernet Network.

abundant, then congestion will never occur and QoS will not be compromised. This method is simple and straightforward and works well in core networks, where large aggregate data streams are relatively smooth. However, in access and metro networks, traffic is more bursty causing frequent and unpredictable bottlenecks making over-dimensioning uneconomical as noted in [23].

1.2 Objective and scope of the thesis

Ethernet was not designed to be deployed as a transport technology, therefore, it is not surprising that the current QoS model for Ethernet is not appropriate to meet the demands of the next generation applications ([7]). The main research question in this respect is

> *How and to what extent should Ethernet technology evolve to meet the QoS requirements of current and future services and applications, while retaining its original benefits of being simple and inexpensive?*

The question above requires the resolution of the following issues:

- Which QoS mechanisms need to be enhanced to make this transition and how?
- Is it possible to reuse some existing functionality in standard Ethernet?
- Given a set of QoS mechanisms in a Metro Ethernet Network (MEN), what is the QoS performance which can be promised to various MEN customers?

- How do the various network and traffic parameters influence this performance? Which network parameters can be tuned (and how) to achieve a desired performance target?

In relation to the questions above, we formalize the following objective for this thesis:

> *To analyze existing QoS mechanisms, and develop new mechanisms where necessary, that improve the performance of Ethernet and higher (end-user) layer applications.*

We aim at analyzing the performance not just at the Ethernet layer but also at (higher) application-layer as this is useful in understanding what a SLA at Ethernet level means for an end-user. By applying or developing new modeling techniques, we aim at obtaining generic results and guidelines quantifying the influence of network and traffic parameters on QoS performance. This will assist the optimal deployment of Ethernet services in access, metro and core networks. Where possible we will try to base the design of new mechanisms on existing functionality in Ethernet. This will ensure that the QoS improvements do not come at high costs, thus retaining the original benefit of Ethernet.

Scope

In this thesis we address three key QoS mechanisms, which are essential in offering and meeting performance guarantees. These are:

- Traffic policing
- Congestion control
- Scheduling

The mechanisms which we decided not to study in detail in this thesis are traffic shaping, buffer management, packet marking and traffic classification (see Figure 1.4). We remark that traffic shaping and Random Early Detection (RED) based buffer management techniques have been extensively researched in literature in the context of other packet technologies. Packet marking and traffic classification in Ethernet are restricted by the 3 bits available in the packet header, providing limited possibilities. For example, if 1 bit is used for marking in and out of SLA profile packets, the remaining 2 bits would allow for only 4 traffic classes. Further information on the usage of these bits is provided in [75]. In view of these considerations, we have chosen to not include research on these issues in this monograph.

With respect to the QoS mechanisms chosen within the scope of this thesis, one might wonder what their relation is to existing solutions for other packet networking technologies (such as IP and ATM). It is important to note that every packet technology has its own distinct features and its QoS mechanisms are designed exploiting these features. For example, ATM is a connection-oriented technology and its QoS mechanisms make use of the possibility of control on each connection. IP, although connectionless in nature, has a rather intricate addressing and routing scheme associated with it. Its QoS mechanisms can use the knowledge of location provided by the source and destination address of a data packet. Ethernet on the other hand is a connectionless technology with a flat addressing and routing scheme. These aspects on one hand make it simple, plug-and-play and cheap. On the other hand, however, they pose challenges for introducing QoS capabilities. Therefore, the QoS mechanisms designed for packet technologies such as ATM and IP cannot be directly applied to Ethernet because it lacks their inherent features. Furthermore, not all QoS mechanisms for other packet technologies are in a stage to meet the challenge we have at hand.

Despite the above considerations we remark that some of the proposed methods and analysis in this thesis can also be applied to other packet technologies. This is especially true for methods presented in this thesis that do not rely on Ethernet specific hardware. Furthermore, a large part of this thesis focuses on analytical modeling of Ethernet QoS functionality. Although these models have been inspired by Ethernet, they can be broadly applied to similar mechanisms for other packet technologies. A more detailed discussion on this issue is provided in Section 1.3 and the relevant chapters of this thesis.

1.3 Organization and contributions

Having outlined the objective and the scope of our research, the organization of the rest of the thesis is as follows. In Chapter 2, we provide more technological details on Carrier Ethernet. We review some basic Ethernet switching concepts and specifically address the services and applications which are being deployed and offered with it and the need for QoS therein. The main technical contributions of the thesis are organized into three parts, each dedicated to one of the QoS mechanisms within the scope of the thesis. The thesis ends with some concluding remarks.

In the remainder of this section, we address the main parts of the thesis in more detail. For each of the three QoS mechanisms we present the research questions and then point out how this thesis contributes to resolving them. The research questions presented in this section focus on the specific issues for the considered QoS mechanisms and differ from the more high level and general questions raised in Section 1.2, which apply to all QoS mechanisms. We have tried to minimize the overlap between this section and other parts of the thesis. However, since our goal was to keep the chapters self-contained, some amount of repetition was unavoidable.

1.3.1 Part I - Traffic policing

Traffic policing is the method used to monitor and enforce the bandwidth profiles agreed in the SLA as explained in Section 1.1.2. A (bufferless) token bucket is widely used for this purpose as it is simple and inexpensive as compared to a leaky bucket due to absence of buffering. However, traffic using the higher layer Transport Control Protocol (TCP) is known to have serious performance problems with a token bucket policer ([90]). This is primarily due to the fact that TCP's flow control mechanism was designed to deal with dynamic congestion in networks. Its enduring yet futile attempts to grab more bandwidth than the fixed contractual traffic rate results in continuous collapse of its transmission window. This results in throughputs far below the contractual rate. In this respect the following questions arise.

Research questions

- How can an operator of a MEN ensure that the policing mechanism at the ingress of its network on one hand enforces the SLA while at the same time does not jeopardize higher application level performance?
- Is a cost-effective solution possible, which does not incorporate expensive buffer upgrades to shape traffic to the contractual traffic rate?

Contributions of the thesis

In the thesis we propose two policing methods to address the issues raised above and assess their impact on data traffic which uses TCP and real-time streaming traffic which uses UDP:

- In Chapter 3 (based on [47]), we present and analyze an Ethernet policer which provides feedback to the MEN customer network on SLA violation. The use of this mechanism results in buffering at the customer side equipment which resolves the TCP performance problem by itself. Furthermore, it also works well for prioritized UDP traffic.
- In Chapter 4 (based on [83]), we present and analyze a dynamic token bucket policing mechanism. This method adapts to the variations in customer traffic including those due to changes in TCP's transmission window. This results in TCP throughputs close to the contractual traffic rate. For constant rate UDP traffic the policer's bucket size, as to be expected, remains unchanged.

Both policing methods have been extensively analyzed using network simulations and experiments. The simulator not only models the capabilities of an Ethernet switching node but also details of the TCP stack.

Relation to other packet technologies

The policing method in Chapter 3 exploits Ethernet specific functionality, which is a novelty over previous literature. Applying this method to other packet technologies would require introduction of feedback messaging in the hardware. The mechanism in Chapter 4 does not use Ethernet specific features and can be directly applied to any packet technology. Although traffic policing has been widely studied for other packet technologies, the previous work ([40], [12]) has not managed to configure the token bucket parameters independently of the policed traffic profile as done in Chapter 4.

1.3.2 Part II - Congestion control

In the second part of the thesis we address the issue of congestion control for Ethernet networks. We do so by exploring the congestion control possibilities already provided in traditional Ethernet. The IEEE 802.3x standard ([76]) defines a pause mechanism or a backpressure signal to enable congestion notification messages. A congested node can send a backpressure/pause message to its upstream neighbors to signal the stop of all transmission towards it for a period of time. Alternatively, an ON/OFF pause message can be sent signaling the beginning and end of the *transmission pause* phase. Within an Ethernet network the use of this signal results in a hop-by-hop congestion control method. Most of the previous work on the analysis of this scheme has concentrated on the protocol and its implementation aspects. The relation between the QoS performance and the key parameters of the backpressure mechanism has not been established. In this respect the following questions still remain to be answered.

Research questions

- What is the effect of the backpressure parameter settings such as congestion detection thresholds and buffer sizes on throughput and delay performance?
- Can the congestion thresholds be used to optimize or achieve the desired trade-off between throughput and delay?
- How does this performance depend on different scenarios and traffic types?

Contributions of the thesis

In this thesis we present two stochastic models of the backpressure congestion control mechanism focusing on two different aspects of the scheme:

- In Chapter 5 (based on [48]) we model the interaction of TCP end-to-end congestion control with Ethernet hop-by-hop congestion control. We do so by introducing a stylized Markov model. The model specifically captures the hop-by-hop nature of backpressure congestion control with two queues in tandem. The solution of the proposed model is compared to the results obtained by simulations. The analysis provides useful insight into the influence of key parameters such as buffer sizes, congestion detection thresholds, round trip times and traffic burstiness on the performance as a result of the interaction between TCP and Ethernet.

- In Chapter 6 (based on [45]) we develop and solve a fluid queue model of the Ethernet congestion control mechanism. Fluid queues abstract from the details at packet level by approximating the flow of packets by fluid flowing at a constant rate. The fluid model we propose focuses on the feedback aspects of the backpressure mechanism rather than its hop-by-hop behavior. Our explicit solution of the model provides the relation between performance measures (such as throughput and delay), congestion detection thresholds and traffic (rate) parameters. This is especially useful for tuning network parameters to achieve desired QoS performance.

- In Chapter 7 (based on [44]) we present an extensive numerical study of the model analyzed in Chapter 6. In particular, we address an essential design issue for the backpressure mechanism by studying the effect of the congestion control thresholds on traffic performance measures such as file transfer time for data files as well as throughput and delay for real-time applications. Numerical experiments are performed to evaluate the main trade-offs such as the trade-off between the signaling overhead and the achieved throughput.

Relation to other packet technologies

The basic idea behind the IEEE 802.3x hop-by-hop congestion control mechanism exists in ATM as well as part of its Available Bit Rate (ABR) transport capability ([57], [39]). However, since ATM is a connection-oriented technology it allows for a lot more control. Unlike the ON/OFF mechanism in Ethernet, the hop-by-hop congestion control functionality in ATM is achieved through fine-grained information in the Resource Management (RM) cells. These cells can continuously increment or decrement the traffic rate of each connection in small steps. As a consequence, most of the research on congestion control for ATM builds on and optimizes the use of these RM cells ([59], [62]). Although this mechanism can be expected to provide better performance than the Ethernet ON/OFF traffic control, it is extremely complex to implement and sustain. Therefore, we have chosen to build on and investigate the functionality which is currently available in Ethernet. This not only enables the immediate applicability of our research in current Ethernet networks but also comes at low costs as it does not require new hardware.

Current congestion control work for IP networks concentrates mainly on Explicit

Congestion Notification (ECN) [25] and Random Early Detection (RED) [27] like mechanisms. The idea of hop-by-hop congestion control does not currently exist in IP but could be easily incorporated. Nevertheless, the fluid queue based analytical models presented and analyzed in this thesis could also be used to model ECN, RED and other enhancements and variations being proposed to Ethernet backpressure based congestion control ([29], [50]).

1.3.3 Part III - Scheduling

In the third part of the thesis we address the issue of dividing the outgoing link capacity over the multiple traffic classes supported in Ethernet. Priority queueing, weighted fair queueing and their combinations have been proposed in literature for this purpose and can be applied to Ethernet as well. No matter which variant is chosen, it seems inevitable to provide highest and strict priority to the time-sensitive traffic class in order to satisfy its strict delay requirements as shown in [38]. The danger with allocating strict priority to a particular traffic class is that it can starve the lower priority traffic classes. The network operator has to ensure that on one hand it meets the strict delay requirements for time-sensitive stream traffic while still satisfying the throughput guarantees agreed in the SLAs for the time-insensitive but loss-sensitive traffic. In this respect the following questions arise.

Research questions

- To what extent should the load of the strict high priority traffic be controlled so as to avoid starvation of low priority traffic?
- Given a particular load of high priority streaming traffic, what performance can be guaranteed to lower priority traffic?

Contributions of the thesis

In order to answer the questions raised above, we need to model the division of link bandwidth among multiple traffic-class flows. A special class of queueing systems, called Processor Sharing (PS) queues ([88]) are especially useful in modeling such cases. PS queues model systems in which the available capacity is divided equally among all active flows. Extensions proposed to traditional PS queues allowing for priority systems are difficult to analyze and no closed-form formulas exist. In Chapter 8 (based on [46]), we approximate a prioritized queueing system with mixed traffic types with an adapted PS model. We evaluate the accuracy of the proposed PS model to the prioritized model. The results show that our simple approximation works quite well for a wide range of parameter values.

Relation to other packet technologies

In Chapter 8, we address scheduling issues which are generic and can be applied to any packet technology. To the best of our knowledge, the simple yet effective approximations presented in this chapter are not available within the existing QoS literature for IP or ATM.

Chapter 2

Carrier Ethernet

In this chapter we describe Carrier Ethernet in more detail than in Chapter 1, paying special attention to the QoS issues therein. However, we begin the chapter with some essentials of Ethernet switching in Section 2.1. This section enables the reader to understand and identify the inherent features of Ethernet, which distinguish it from other packet technologies and therefore explain the context of this thesis. In Section 2.2 we discuss the reasons behind the popularity of Ethernet as a transport technology. This is followed by the drawbacks of native Ethernet explaining the need for Carrier Ethernet in Section 2.3. A brief overview of the basic characteristics of Carrier Ethernet is also provided. We then focus on the importance of QoS for Carrier Ethernet networks and the drivers behind it in Section 2.4. In particular, we discuss the kind of applications being deployed by Carrier Ethernet. We also discuss the role of QoS in improving the cost-effectiveness of Ethernet network deployments. We end the chapter with positioning the research presented in this thesis in relation to other work in literature primarily focused on improving QoS for Ethernet networks.

2.1 Ethernet switching preliminaries

In this section we explain how packet switching works in Ethernet on a high level. It is important to note that *Bridged* or *Switched Ethernet* is different from *Shared Ethernet*, where a collision domain exists. In this section we restrict ourselves to switched Ethernet, as this is particularly relevant for this thesis.

A switched Ethernet network is shown in Figure 2.1. The switches in this network are connecting end-stations directly as well as (shared) Ethernet LANs. The end-stations as part of a shared Ethernet LAN are interconnected via a hub. Each Ethernet switch and end-station has its own unique MAC (Medium Access Control) address which is used to route data to it. In order to understand how a packet is transmitted in such a network let us follow a packet sent from source S1 to

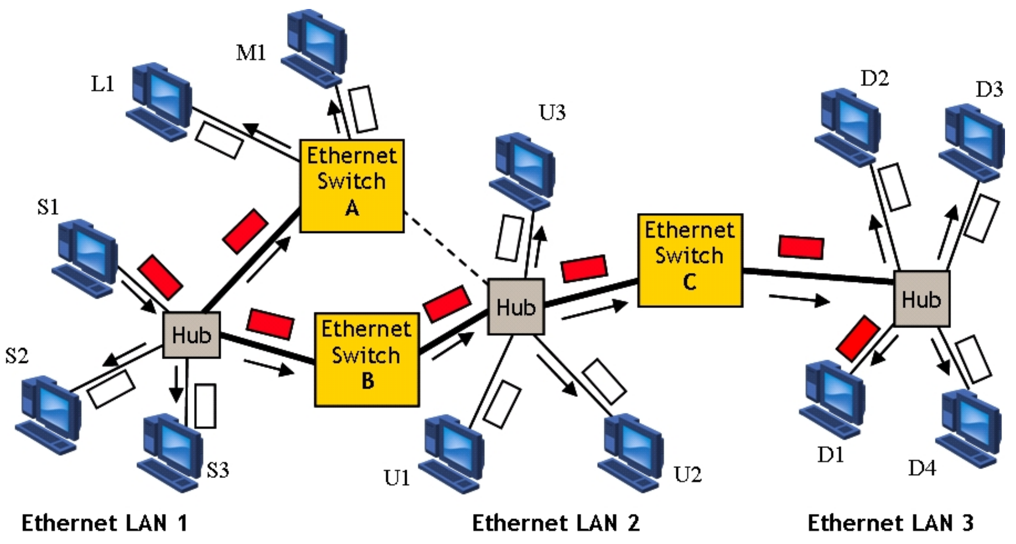


Figure 2.1: A switched Ethernet network.

destination D1. The arrows indicate the path followed by this packet. The packet from S1 will reach all the stations in its LAN as well as sent to switch A and B. Both switches A and B will learn through which one of their ports S1 can be reached. Since initially, neither does switch A nor B know where D1 resides, the packet will be sent everywhere and all stations in Figure 2.1 will receive the packet meant for D1. On receiving this packet, the stations will see that the destination address does not match their own address and they will discard the packet except for D1. The white (blank) packets in Figure 2.1 indicate that the receiving end-station drops the packet. The dark red packets indicate that the packet is not dropped by the station or switch. This type of packet transfer is called unknown unicast packet transfer, which results in the packet being broadcast to all stations connected to the network. If D1 sends a packet back to S1, the switches will learn the address of D1 and through which of their ports it can be reached. As a consequence further transmission between S1 and D1 will not reach all the end-stations but will be restricted to the path followed by the dark red packets. However, the learnt addresses are flushed from memory periodically. Therefore, if S1 and D1 do not communicate for a while then new transmissions between them would again lead to broadcast traffic. This explains the broadcasting based routing of packets in Ethernet as opposed to more formal path calculation per traffic stream in IP.

Another aspect associated with Ethernet's method of routing is the possibility of packet duplication and multiplication when loops are present in the network topology. For this reason all Ethernet networks need to create a logical tree, free of loops, on which traffic can be routed and flooded. The protocol used to create this tree connecting the entire network is called the Spanning Tree Protocol (STP).

Figure 2.1 shows the spanning tree in thick lines. The dotted line is a blocked link which is unused. If the link between Switch-A and Ethernet LAN-1 fails, the blocked link can be activated again by the STP, which recalculates the tree in the event of a failure. The network created by the spanning tree connecting multiple LANs is called a VLAN. All the end stations that are part of this VLAN experience themselves as being on the same LAN. Another interesting property of such a VLAN is that if a new station is included in one of the Ethernet LANs, it automatically joins the whole VLAN. This gives Ethernet its plug and play operation.

It is possible to create multiple VLANs for the network shown in Figure 2.1, for example one between L1 and Ethernet LAN-1. Each such VLAN has its own broadcasting domain. Ethernet switches do not route packets from one VLAN to another and traffic within different VLANs is kept separated. In this way one can create multiple and distinct broadcasting domains. This simple concept of a VLAN extends to creating virtual private networks (VPNs) for enterprises or high speed internet connections between end-users and their ISPs as shown in Figure 1.1.

2.2 Why Ethernet in public networks?

Among all the packet technologies available, why is Ethernet such a popular choice among service providers for data transport? In this section we address this question by explaining the reasons and some inherent features of Ethernet, which have triggered its selection.

- *Wide presence in LANs*: Ethernet today constitutes 97% of all LAN traffic. Introducing Ethernet based transport further on in the network helps avoid expensive and unnecessary protocol translations. It can be installed as a backbone network while retaining the existing investment in Ethernet hubs, switches and wiring plants.

- *Increasing data rates*: Over the past years, Ethernet has evolved to support greater speeds and distances. Ethernet data rates have climbed from 10 Mbps to 10 Gbps and continue to grow at a steady rate making it suitable for data transport in larger networks. Current developments are moving to 40 and 100 Gbps and this is expected to be standardized in 2010.

- *Plug and Play*: Since Ethernet was originally designed for LANs, it inherently possesses plug and play operation as explained in the previous section. A new device or station associated with a new user for instance, just needs to be added to a VLAN. Therefore, it does not require extensive provisioning as compared to IP. As a result configuring and provisioning Ethernet VPNs is simpler than IP VPNs.

- *Inherent broadcasting capabilities*: Unlike IP, Ethernet has inherent broadcasting capabilities as explained in Section 2.1. This is useful not only for creating VPNs but also for broadcasting applications such as IPTV.

- *Low costs*: The reasons behind the low costs of Ethernet are many. Ethernet interfaces are cheap. Using Ethernet in metropolitan area and beyond means that the equipment used in LANs does not have to be replaced and helps save on upgrade costs. It is also a simple technology that does not require extensive provisioning. Furthermore, since it is a packet technology it can multiplex multiple data streams on the existing circuit-infrastructure, helping provide services at low costs to the end-user.

2.3 Making Ethernet carrier grade

Ethernet technology was originally designed to work in small LANs. Therefore, it is not surprising that its extension into larger metro and wide area networks raises a number of concerns. For example, native Ethernet has a limit of 4096 VLANs. Since every customer gets his own VLAN, this limit imposes a restriction on the scalability of Ethernet. Network topology recalculation subsequent to a failure is done using the STP, which does not live up to the 50 ms restoration time that operators are used to with SDH/SONET. Furthermore, this protocol does not make efficient use of available bandwidth. Native Ethernet also lacks a QoS architecture to provide certain performance guarantees.

Carrier Ethernet overcomes most of the concerns of native Ethernet. It is defined as being ubiquitous, standardized carrier-class service with five distinguishing attributes: scalability, standardized services, service management, reliability and QoS. Although many of these attributes are receiving attention in standards, many hurdles still need to be overcome ([19]) to make Ethernet a true carrier class technology. Below we briefly address the current state for each of the five attributes of Carrier Ethernet.

- *Scalability*: The most important scalability issue due to limited number of VLANs is being addressed in the standards. Tunneling technologies such as MPLS and Provider Backbone Bridges ([71]) provide the possibility to aggregate Ethernet MAC addresses. These solutions are expected to provide carrier-class scaling of Ethernet networks and are further explained in the IEEE 802.1ah draft.

- *Standardized services*: Carrier Ethernet comes with all attributes closely supported by standardized services. The MEF, ITU and IEEE are striving to standardize different functionalities aimed at improving Ethernet. Among other aspects they are addressing how the various Ethernet service types should be deployed and what kind of performance guarantees they should fulfill.

- *Service Management*: The attribute of service management ([11], [43]) is directed at providing the possibility to identify and manage failures of links as well as monitor performance and connectivity aspects of services. This can help the service provider to check and show if the agreed upon SLAs are being met and to identify problems as explained in Section 1.1.1.

- *Reliability*: The traditional STP designed originally in 1993 for native Ethernet had several limitations with respect to the convergence time and its utilization of network bandwidth. Fortunately, multiple ([75]) and rapid ([74]) spanning tree protocols are already a considerable improvement to this protocol and overcome many of its drawbacks.

- *Quality of Service*: The QoS aspects of Carrier grade Ethernet have already been addressed in Chapter 1. Section 1.1.1 presents an overview of the QoS possibilities in Carrier Ethernet today, whereas Section 1.1.2 points out what is still missing. This thesis focuses on some of these missing elements, thereby addressing some essential QoS challenges faced by Carrier Ethernet.

2.4 QoS drivers

In this section we further motivate the need for QoS in Carrier Ethernet. This demand is driven by two challenges faced by Carrier Ethernet. Firstly, it should be able to satisfy application requirements and user perception. Secondly, Carrier Ethernet should retain and improve cost-effectiveness of current and future network deployments. In this section we discuss in more detail the applications for which Ethernet networks are being used and the demands they impose. We also discuss typical Ethernet deployments, their relative costs due to the extent of multiplexing and complexity with respect to QoS issues.

2.4.1 Application demands

A variety of applications are being supported by Carrier Ethernet networks. Each of them imposes their own requirements which we discuss here.

- *Enterprise networks*: The enterprise market needs to connect its worldwide workforce while reducing operating costs and simplifying management and administration. While Ethernet fits very well in this market due to its inherent broadcasting capabilities, low costs and familiarity, it has to live up to the demand for guaranteed bandwidth performance. Enterprises which pay for a particular bandwidth to create their VPNs, want to be assured that they ‘get what they pay for’.

- *Residential triple play*: A triple play service is the combined delivery of high-speed Internet access, television and telephone over a single broadband connection. Delivering residential triple play services using Ethernet networks requires Ethernet to support not only high peak bandwidth but also priority voice, high definition and on demand video services. Delay, jitter and throughput requirements should be met for both voice and video traffic. Satisfying the requirements of such services per customer undoubtedly requires proper QoS support in Ethernet.
- *Wireless backhaul traffic*: Mobile broadband services and applications are being widely adopted worldwide. This is expected to lay immense pressure on the transport capacity between base stations and core networks. Carrier Ethernet should provide a cost-effective way to transport this increasing traffic volume.
- *Service convergence*: The data communications industry is entering an era of service convergence. Services will be managed and offered over any access network. This requires Ethernet networks to be compatible and integrate with a generic control plane. In this respect, Ethernet should at least support techniques for estimation of its available QoS resources and admission control of new services.

2.4.2 Improving cost-effectiveness

In this subsection we explain the impact of the various Ethernet service connectivities on the cost-effectiveness of the technology and its QoS issues. Ethernet connectivity comes in different flavors. 'Virtual' or 'Private' refers to the extent of sharing of networks resources. 'Line' or 'LAN', is the choice between point-to-point and multipoint-to-multipoint connectivity. The more shared the connectivity the more the gain from statistical multiplexing and therefore more cost-effective the offered Ethernet service. However, the more shared the service, the greater is the need for QoS mechanisms to ensure the performance guarantees for each user.

- *Virtual vs Private*: 'Virtual' refers to shared and 'Private' refers to dedicated and reserved bandwidth. When specific bandwidth is reserved for a customer whether he uses it or not is called *private*. When bandwidth is shared among multiple customers the connectivity is called *virtual private*. Traffic of each customer is kept separate by configuring a VLAN per customer, however the different VLANs do share the underlying network capacity (see Figure 2.2). This capacity could be a SDH/SONET circuit, WDM channel or an MPLS path/pseudowire etc. It is obvious that the private approach is expensive because the service provider cannot use this bandwidth for other purposes. Virtual service on the other hand multiplexes traffic from multiple customers onto the same link bandwidth. Therefore, the same resource can be shared

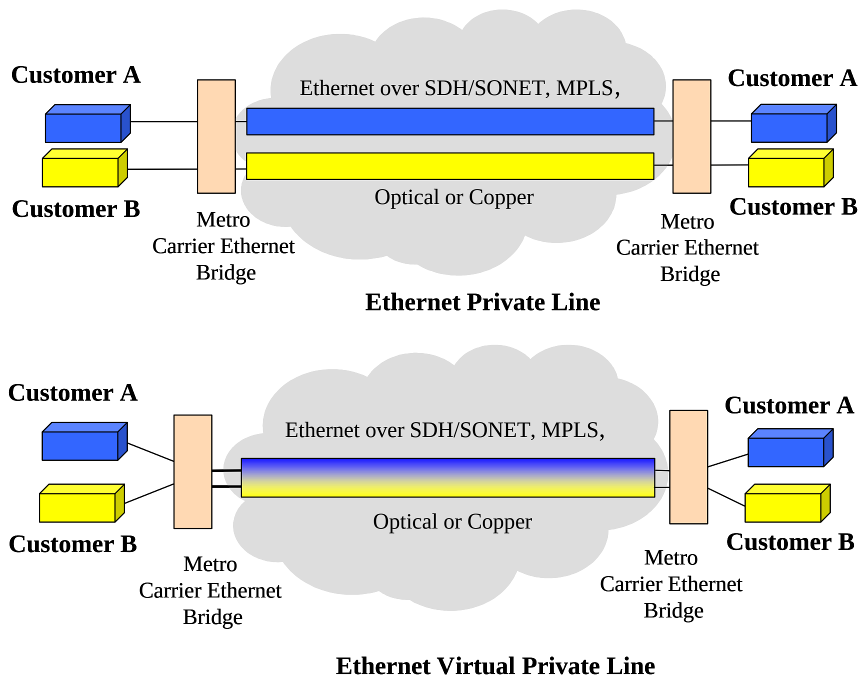

Customer A
Customer B
Ethernet over SDH/SONET, MPLS,
Optical or Copper
Metro Carrier Ethernet Bridge
Customer A
Customer B
Metro Carrier Ethernet Bridge
Ethernet Private Line
Customer A
Customer B
Ethernet over SDH/SONET, MPLS,
Optical or Copper
Metro Carrier Ethernet Bridge
Customer A
Customer B
Metro Carrier Ethernet Bridge
Ethernet Virtual Private Line

Figure 2.2: Ethernet private line and virtual private line connectivity.

among different users and as a result, services can be offered at lower costs. With respect to providing QoS guarantees however, the opposite is true. It is easier to control and monitor QoS for traffic streams for which bandwidth is dedicated than when they share the network resources. Furthermore, temporary congestion moments can hamper performance of different customers in an unpredictable way. At these instances, QoS mechanisms are required to ensure that performance guarantees are still met. It is important to note, that one should not infer that the private or dedicated connectivity is devoid of all QoS issues. Most service providers believe that installing a traffic policer conforming to a contract suffices. Unfortunately, they do not pay attention to the interaction of the policers with end application characteristics which could result in undesirable performance and user perceived quality as shown in Chapters 3 and 4 .

- *Line vs LAN*: Ethernet line connectivity refers to point-to-point connectivity whereas LAN refers to multipoint-to-multipoint connectivity. Both Line as well as LAN can be configured as virtual or private. Figure 2.3 shows a LAN service configured as a combination of multiple point-to-point connections or private lines where the underlying bandwidth is dedicated.

  Figure 2.4 is a true LAN providing any-to-any connectivity by sharing the underlying bandwidth. The complication with LAN connectivity, whether

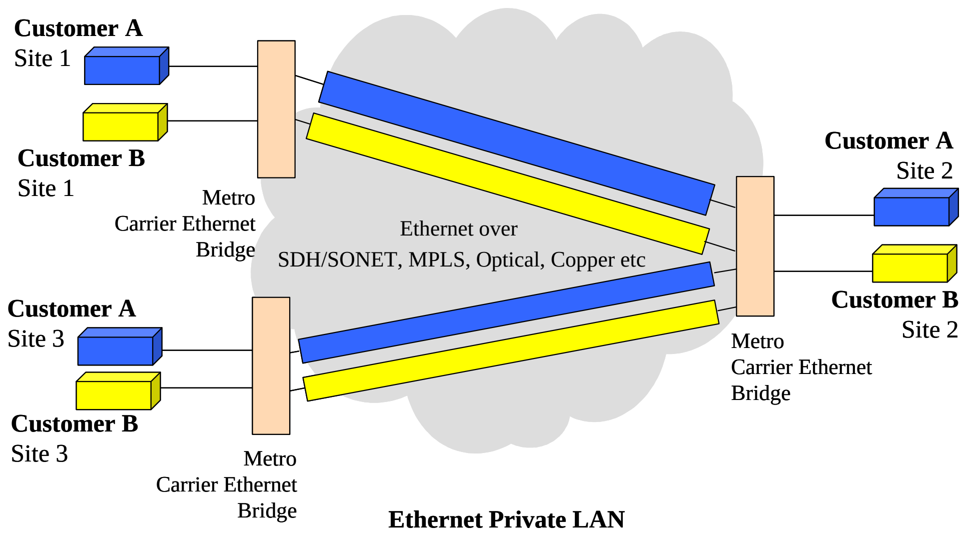


Figure 2.3: Ethernet private LAN connectivity.

virtual or private, is that it is difficult to predict the amount of traffic flowing between the multiple end points. This traffic flow is also typically expected to change over time. This, however, is a traffic engineering or a routing issue. An elegant and innovative way to solve this problem is presented in [84].

2.5 Remarks on Ethernet QoS research

In this section we briefly discuss QoS mechanisms proposed in standards and literature for Ethernet networks and its relation to the work we present in this thesis. Since Ethernet's move into the metro and wide area networks is a relatively new development, it is not surprising that the QoS literature in this context is rather limited. In fact most of the QoS research for Ethernet is focused on congestion control and generic QoS frameworks ([63]). The congestion control work for Ethernet mainly focuses on protocol and implementation modifications of the feedback functionality provided in IEEE 802.3x. For example, [29] proposes that the backpressure/pause functionality should not be applied to the time-sensitive traffic class and references [8] and [50] propose that the backpressure signal should be sent directly to the ingress points of the network instead of hop-by-hop and whose stability is analyzed in [36]. Recently, a forward congestion notification mechanism has also been proposed ([37]). However, all these protocol enhancements and modifications to the backpressure/pause functionality still rely on proper configuration of the congestion detection thresholds to optimize the network performance. This particular issue has not been addressed by previous work in literature. The work on conges-

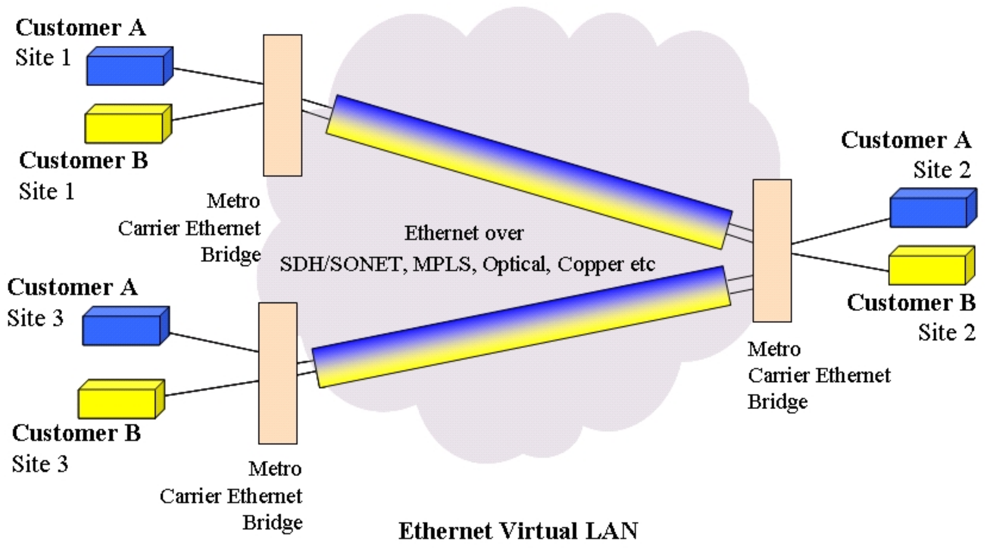


Figure 2.4: Ethernet virtual LAN connectivity.

tion control presented in this thesis is of an advanced nature, in the sense that we provide not just detailed network simulations but also extensive analytical modeling and analysis of the backpressure mechanism. This enables proper parameter selection and tuning the achieved performance with the scheme. Furthermore, we also address other key QoS mechanisms such as traffic policing and scheduling.

Part I

Traffic policing

Introduction to Part I

Traffic policing involves monitoring and enforcing the traffic limit agreed upon in the SLA as explained earlier in Chapter 1. The traditional method of policing with a bufferless token bucket is simple and inexpensive. However, it imposes a bursty drop pattern which adversely interacts with TCP's congestion control mechanism resulting in throughputs far below the (SLA) contractual traffic rate (see [90]). This is an extremely undesired situation for both the service provider who provisions his network according to this traffic rate and its customer who pays for it.

In this part of the thesis we propose two new bufferless policing methods and analyze their impact on higher layer application performance. We show that the mechanisms we propose are a considerable improvement over the traditional token bucket policer in terms of TCP throughput and also work well for UDP. These mechanisms are presented in Chapters 3 and 4.

- In Chapter 3, we propose and analyze a bufferless token bucket policer which exploits the IEEE 802.3x backpressure method available for Ethernet networks. In particular it warns the customer by sending a *transmission-pause* message if he is about to send traffic above the contractual traffic rate. A thorough analysis of this scheme shows that this mechanism has the consequence that TCP bursts are smoothened by packets being buffered at the customer equipment. This is achieved without introducing a dedicated shaper for this purpose. This feedback policing method results in improved TCP performance with throughputs close to the contractual traffic rate.
- In Chapter 4, we propose and analyze a bufferless token bucket with a dynamic bucket size. The bucket size adapts to the bursty nature of the incoming traffic without any knowledge of the traffic profile. This is particularly useful for TCP traffic generating varying bursts due to fluctuations in its transmission window. If the traffic is constant rate, then the bucket size remains constant, which is suitable for UDP traffic. Since the mechanism does not rely on Ethernet specific hardware, it can be applied to any packet networking technology.

Chapter 3

A backpressure based policer

In this chapter, we present and analyze a novel bufferless token bucket policer, which interacts well with TCP's flow control. The policing method exploits the Ethernet backpressure mechanism described in the IEEE 802.3x standard ([76]), which is primarily used for avoiding congestion ([55], [66], [86], [22]). While enforcing an SLA with a policer it is normal and logical to drop all customer traffic which exceeds the maximum traffic rate and/or packet burst size agreed in the SLA contract. This is because the service provider provisions his network with this limit in mind. The policing mechanism presented in this chapter, however, does not simply drop all packets that exceed the maximum traffic rate. Instead it adds an element of feedback with the backpressure mechanism to the sender (customer), if it is approaching this traffic limit (as described in [81]). Transmission of the Ethernet backpressure message results in temporary pause of data transmission and queueing of packets at the customer egress queues. This temporary buffering of packets has the effect that traffic sent by the customer is automatically smoothened requiring minimal effort from both the service provider and its customer.

The rest of the chapter is organized as follows. In Section 3.1, we present the Ethernet backpressure based policing method. In Section 3.2, we present the experimental setup used to analyze the performance of the policing method introduced in Section 3.1. Section 3.3 provides the performance results achieved by our backpressure based policing mechanism relative to the traditional practice of dropping packets that exceed the peak traffic rate. We focus on both TCP as well as UDP traffic performance in terms of throughput, delay and jitter. We also look at fairness in throughput results for TCP traffic and briefly at the influence of threshold values for the backpressure mechanism. Finally in Section 3.4, we present the conclusions of our study.

3.1 A traffic policing mechanism based on backpressure

In this chapter, we study the use of Ethernet backpressure in a traffic policing mechanism for metropolitan area networks. The new policing mechanism is realized by coupling the backpressure to a token bucket rate controller, rather than to a queue for congestion control, which is the approach used in the literature.

Before we can explain our proposed traffic-policing mechanism, we must first discuss the concept of backpressure. Backpressure is intended to provide flow-control on a hop-by-hop basis, by allowing ports to turn off their upstream link partners for a period of time. In the case of a half-duplex link, the link partner or end station is turned off by sending a jamming signal. The signal causes the end-station to perceive the medium as busy; accordingly, it stops transmitting, and backs off. In the case of a full-duplex link, the upstream link partner is turned off using a medium access control (MAC) layer flow-control mechanism defined in the IEEE 802.3 standard (see [76]). This mechanism is based on a special frame (called a *pause frame*) in which a period of time (called a *pause time*) is specified. When an end station or router receives the pause frame, it reads the pause time and does not attempt to transmit until the pause time has passed.

In metropolitan or other public networks, bandwidth is usually sold by specifying a committed information rate (CIR), a peak information rate (PIR), or both. The sender is allowed to send more than the CIR, but excess packets are marked and may later be dropped from the network (see [32]). In contrast, when the sender exceeds the PIR, packets are dropped immediately. To enforce the PIR, the incoming traffic rate on a port is monitored, using the token bucket mechanism shown in Figure 3.1. Tokens are added to the token bucket at a rate equal to the PIR, until the peak bucket size (PBS) is reached. When a frame is sent, the number of tokens in the bucket is decreased by the number of bytes in the frame. Packets that arrive on a link are forwarded, as long as there are tokens in the bucket. If there are insufficient tokens in the bucket when a packet arrives, the packet is dropped.

We propose to trigger backpressure on an incoming link if the number of tokens falls below a pre-defined threshold, which indicates that the PIR is about to be exceeded. Then, as soon as the number of tokens in the bucket rises above another pre-defined threshold, the backpressure can be released. The backpressure-based traffic-policing mechanism we propose will monitor the input traffic rates at the ingress ports of the MAN and, if the input rate at any port starts to exceed the PIR, the mechanism will send a backpressure signal on that port. In this way, backpressure will be used to notify the sender and to prevent excess packets from being sent to the MAN, thereby avoiding packet drops at the metro bridge.

Unlike a traditional congestion-based backpressure mechanism, the traffic-policing mechanism we propose is not triggered by queues that have built up in the network,

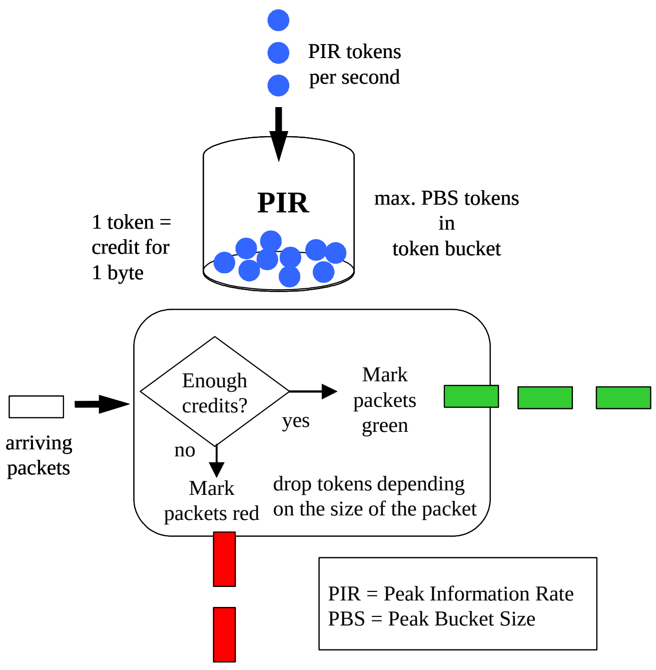


Figure 3.1: A token bucket rate controller.

so delay differs greatly from what it is in a congestion-based mechanism. For this reason, the performance of a traditional backpressure mechanism is not comparable to the performance of our mechanism.

3.2 Experimental Setup

In order to analyze the performance of our proposed backpressure policing mechanism, we ran tests on a live network. Because backpressure can affect the end-station directly, test results can vary greatly, depending on the implementation of the protocol stacks and buffering and queueing specifications. For these reasons, simulation often fails to reflect reality; by using a live network, we can examine and understand real behavior.

In the scenarios considered, one or more servers are connected to a MAN, either directly or through a router. Access to the MAN is supplied by a metro bridge that is part of the MAN and that uses a token filter to perform traffic policing. Figure 3.2 illustrates this scenario. The servers, the router, and the bridge are interconnected by 100 Mb/s Ethernet links. Because our focus was on the effect of backpressure on the end-station, it was not necessary to use an elaborate MAN with many bridges. In our set-up, the MAN is simulated by adding a configurable delay to all packets in the bridge, and multiple clients are simulated by opening multiple connections

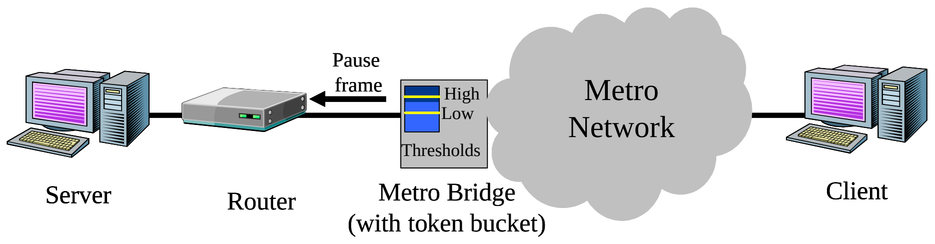


Figure 3.2: Setup used in the experiments.

from a single client. The bridge and token-bucket functionality were implemented on a personal computer (PC) running Linux. The server and the clients used the Microsoft Windows 2000[1] TCP stack.

It is important to note that we do not consider a congestion situation. This means that packets are only dropped when there is a violation of the traffic contract (i.e., when the offered traffic rate exceeds the PIR). By removing other factors that could affect the results, this approach allows us to concentrate on and to study the effect of the backpressure and token bucket combination.

3.3 Experimental Results and Analysis

The results discussed in this section focus on two types of traffic; TCP file transfers, and UDP multimedia streams. Both the TCP and UDP traffic streams were generated by an application developed for this purpose. We also consider the performance of a real application (i.e., NetMeeting).

In the test runs, we have considered two configurations of the router. In the first configuration, the router does not make a distinction between UDP and TCP traffic. In the second configuration, the router gives strict priority to (time-sensitive) UDP traffic, meaning that UDP packets are always forwarded before queued TCP packets. However, this distinction does not affect the normal policing method without backpressure, because in that method each incoming packet is immediately forwarded.

3.3.1 TCP File Transfers

In the experiments, TCP traffic was generated by a file transfer session. The number of simultaneous TCP connections was varied from 1 to 9, but the total amount of data transferred was kept constant at 24 MB. The simultaneous TCP connections were policed as an aggregate with a PIR of 400 KB/s and a bucket size of 80 KB; this means that the token bucket could fill up in 0.2 seconds. Low and high thresholds

[1]Microsoft and Windows are registered trademarks of the Microsoft corporation.

were set to 60% and 80% of the bucket size, respectively. The scenario used for the tests in this section is shown in Figure 3.2. However, the results would be the same if the end-stations were directly connected to a metro bridge (this configuration can be pictured by removing the router from Figure 3.2). In that case, packets would be buffered in the end-stations instead of in the router.

Throughput

Figure 3.3 and Figure 3.4 show the aggregate throughput results for different numbers of simultaneous connections without and with backpressure, respectively. From the figures we can see that backpressure improves TCP performance, irrespective of network delay and the number of active TCP connections. Indeed, with backpressure, TCP performance is close to optimal, because the 400 KB/s rate counts the number of bytes in raw Ethernet frames. The explanation for this near-optimal behavior is that backpressure prevents frame drops, so no retransmissions are needed. Therefore, all transmitted frames contribute to the effective throughput.

Without backpressure, we observed the following:

- Reasonable delay values improve TCP throughput performance;
- With multiple connections, the total throughput is also good with reasonably small delay values (e.g., 5 ms); and
- TCP performs poorly when the network has very low delay and there are only a few TCP connections.

To understand the rather poor performance of a single TCP connection with low delay, we consider how TCP's fast retransmit algorithm works (see [77]). This algorithm relies on the receiver sending duplicate acknowledgements (ACKs) when it receives out-of-order segments. Suppose that, after receiving a number of duplicate ACKs, the sender decides to re-send a supposedly lost packet, without waiting for the retransmission timer to expire. Now, when the network delay is sufficiently high and the token bucket starts dropping packets, there will usually be a number of ACKs in transit from the receiving PC to the sending PC. There will also be a number of data packets in transit from the bridge to the receiving PC, which will also generate ACKs going back to the sending PC. When these ACKs are received, the sliding-window algorithm causes the sending PC to send more data packets. Since these packets are sent some time after the token bucket started to drop, it is likely that the token bucket will contain enough tokens to let some of the packets pass. These packets will appear to the receiving PC to be out-of-order, so they will generate duplicate ACKs that will trigger the fast retransmit algorithm.

We can now explain why the fast retransmit algorithm does not work as well when the network has very low delay, because in that case only very few data packets and

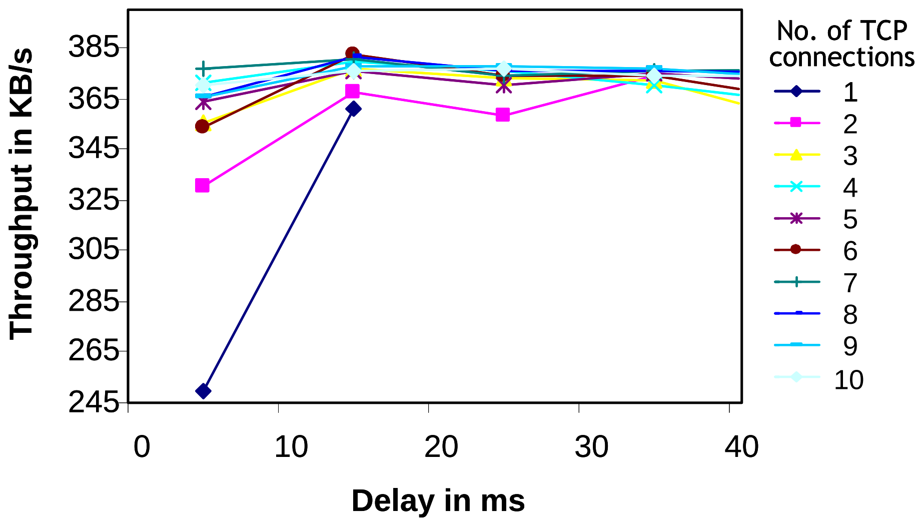


Figure 3.3: Throughput without backpressure.

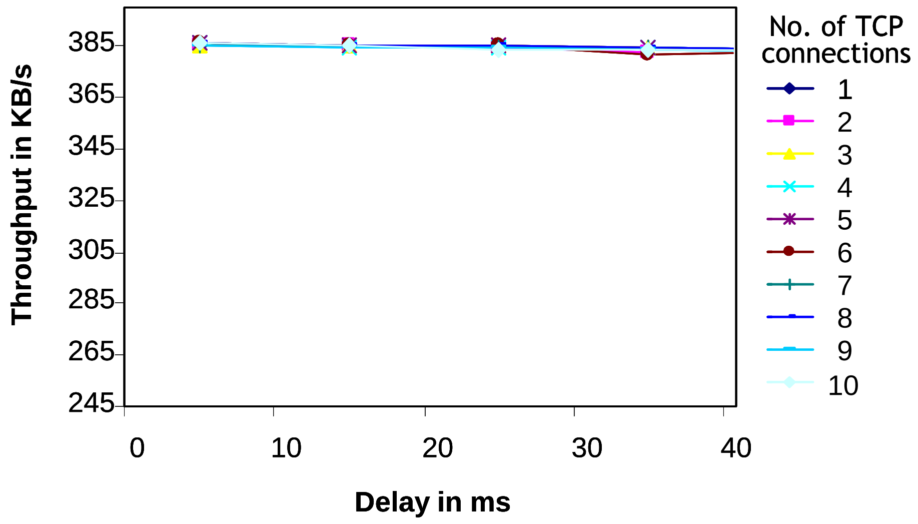


Figure 3.4: Throughput with backpressure

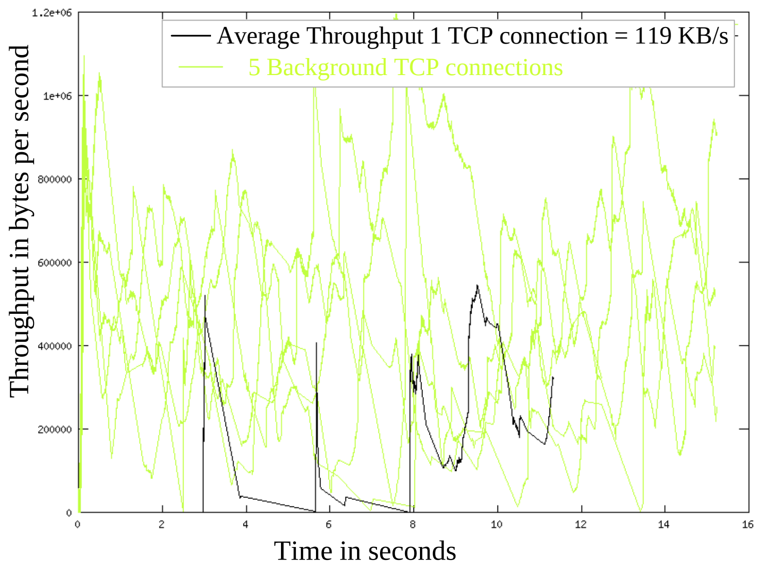


Figure 3.5: Fairness scenario result without backpressure.

ACKs will be in transit when the token bucket drops. This means that the sender will not receive many ACKs and will not send many more new packets. Thus, the receiver will not send duplicate ACKs either, and a single TCP connection with low delay will spend a considerable amount of time waiting for the retransmission timer to expire.

With a large number of connections and low delay, the fast retransmit algorithm is still unlikely to work, but in this case there is a good chance that, while one connection is waiting for a retransmission timer to expire, other connections can use the available tokens.

Fairness

Another aspect explored in the tests was fairness in the treatment of multiple TCP streams. Here we focused on short-term fairness for small file transfers, which is important for such things as Web browsing. In order to analyze fairness, multiple TCP streams (the light green lines in Figures 3.5, 3.6, and 3.7) were kept running. After some time, one additional TCP connection (the dark line in Figures 3.5, 3.6, and 3.7) was initiated. The throughput of this connection was then monitored for multiple test runs. It is important to note that the PIR values used to evaluate fairness were 2MB/s and all the TCP connections shown in Figures 3.5 to 3.7 were policed as an aggregate.

Figures 3.5 and 3.6 show two runs of the same scenario without backpressure, with a network delay of 15 ms. In the graphs it can be seen that the light lines of the

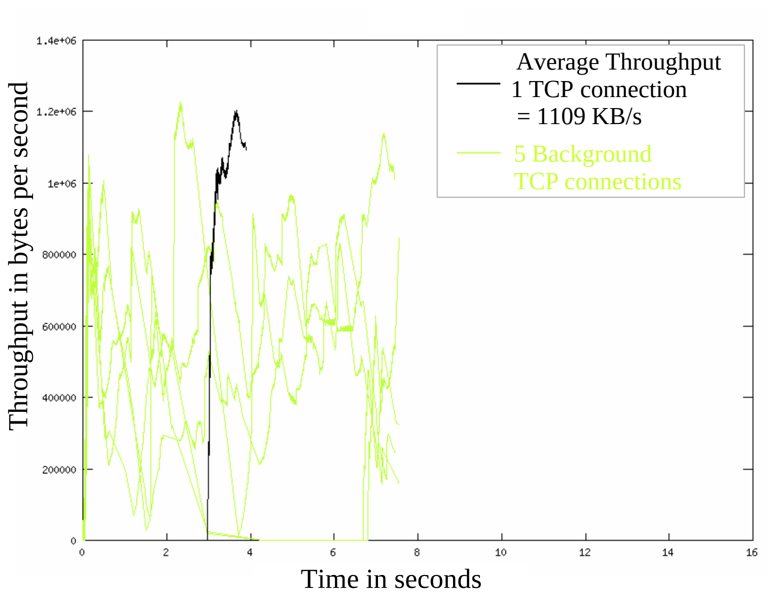


Figure 3.6: Fairness scenario result without backpressure.

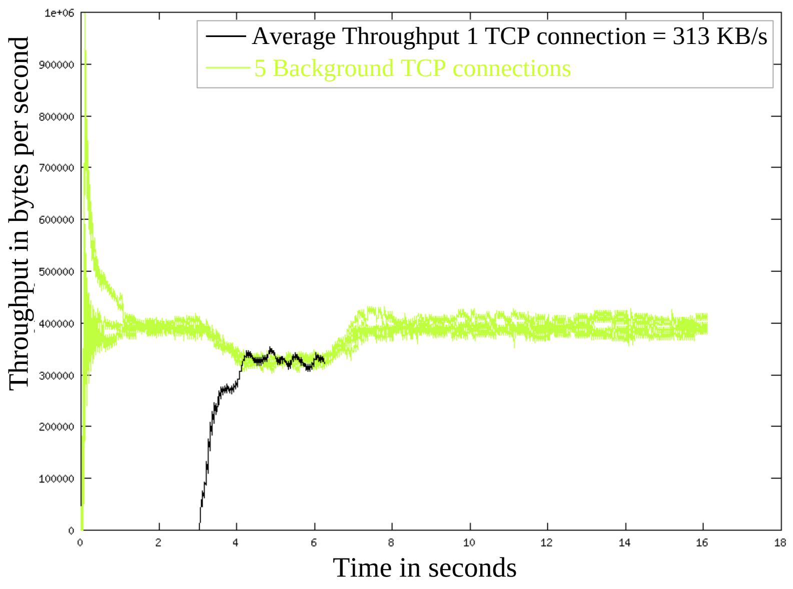


Figure 3.7: Fairness scenario result with backpressure.

background connections behave very erratically. In the first graph, the new TCP connection is hindered by the other connections, and it takes quite some time for the file transfer to complete. In the second graph, the new TCP connection suffers less from packet drops, and it can send the file in a very short time (i.e., one-tenth the time required by the new TCP connection in the first graph). Further runs of the same scenario demonstrate that the difference in performance observed in these two runs is typical for this scenario; clearly, short-term behavior is unfair.

Figure 3.7 shows a typical run with backpressure. It is clear that bandwidth is distributed equally to all active connections. Neither adding more background TCP connections nor changing the network delay alters this equal distribution. The delay introduced by backpressure due to buffering of packets smoothens TCP's traffic generation thereby improving performance. A next step might be to see how giving different delays to different TCP connections would influence fairness.

We conclude that backpressure improves the fairness of bandwidth distribution and provides increased throughput.

3.3.2 UDP Streams Mixed with TCP File Transfers

To study the effect of the backpressure mechanism on multimedia applications, we tested the performance of constant-rate UDP flows. In the absence of other traffic, such multimedia flows perform well if their cumulative rate stays below the PIR; however, once the cumulative rate exceeds the PIR, packets will be dropped, regardless of the policing method used. Therefore, we looked at a mix of UDP and TCP traffic, concentrating on UDP packet loss and jitter. We varied the number of TCP connections from 1 to 7, and used UDP packet sizes of 1000 bytes and a constant traffic rate of 200 KB/s, which is half the available bandwidth. In each run, a total of 8000 UDP packets were sent. The token bucket parameters were set to the same values that they had in the TCP tests. We measured jitter by calculating the standard deviation of packet delay and the difference between maximum delay and minimum delay.

Since most multimedia applications are severely affected by packet loss and jitter, we also considered a scenario in which a router is set up to provide quality of service (QoS) for multimedia traffic. A QoS policy for multimedia traffic usually prescribes that certain traffic classes be given priority over other traffic classes, and sets a limit on the amount of traffic in each traffic class. As long as these limits are not exceeded, packets in the multimedia traffic class are given priority over other packets. For our experiment, this means that UDP packets are given priority over TCP packets.

Figure 3.8 shows packet loss in different scenarios. The small diamonds indicate a delay of 15 ms, while the large squares indicate the results with a delay of 50 ms. The light blue lines show UDP drops when backpressure is not enabled. These packets are dropped in the metro bridge, because the PIR is exceeded. The maroon

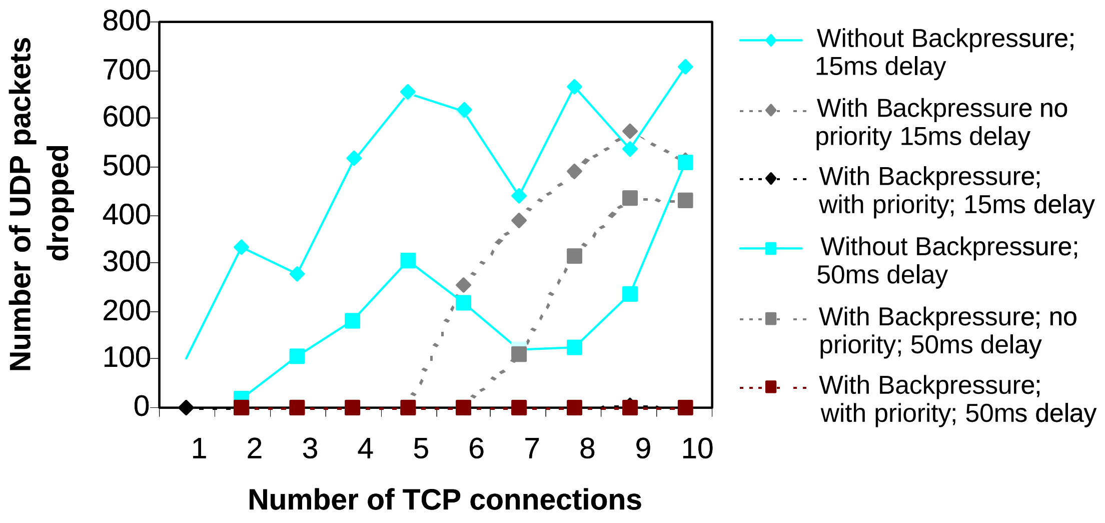


Figure 3.8: Comparison of packet loss for various scenarios.

and black lines show UDP drops when backpressure is enabled, but no priority is given to UDP traffic. The packets in this scenario are dropped in the router, because the queues overflow. Finally, the black lines show UDP drops when backpressure is enabled and priority is given to UDP traffic. (As noted earlier, giving priority to UDP traffic has no effect when backpressure is not enabled, because no queueing occurs.)

The graph in Figure 3.8 shows that the number of packet drops without backpressure is large. Results are very unstable without backpressure, because of the randomness of the packet drops with respect to the different connections. Indeed, repeated runs of the scenario give different results each time. The fluctuation of the light (blue) lines is explained by the fact that the points in the graphs represent only one run. The backpressure results are more stable and consistent. Backpressure reduces the number of packet drops. With priority queueing and backpressure enabled, there is no UDP packet loss at all.

Figure 3.9 shows the standard deviation of delay for the same scenarios that were used for the results in Figure 3.8. Jitter is very low without backpressure; it is somewhat higher when backpressure is enabled and priority is given to UDP traffic. The worst results occur when backpressure is enabled and no priority is given to UDP traffic. With a low number of TCP connections, it does not matter much if the router has priority queueing enabled or not, but with more TCP connections, jitter increases if the router does not have priority queueing. When the router does give priority to UDP traffic, the jitter stabilizes, even if the number of TCP connections increases. The reason for this is that without priority queueing and multiple connections, TCP packets can delay UDP packets, while when priority queueing is enabled, UDP packets get priority and have no delay other than waiting

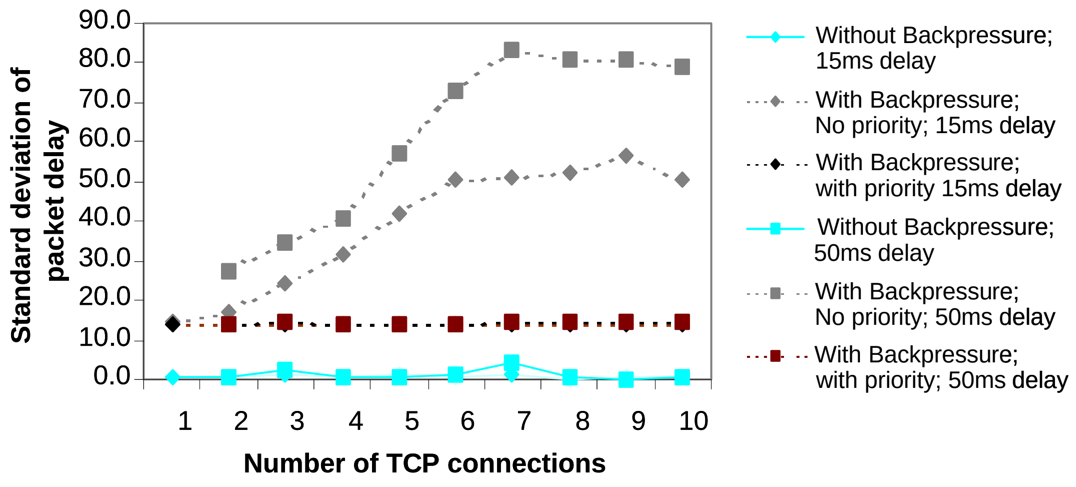


Figure 3.9: Standard deviation of delay for various scenarios.

for the backpressure signal.

When we look at the maximum end-to-end delay, the same pattern emerges. Without backpressure, maximum end-to-end delay is negligible (i.e., 0 to 10 ms); with backpressure, and with no priority given to UDP traffic, it can be approximately 510 to 520 ms. However, with priority given to UDP traffic, the use of backpressure reduces maximum delay to 50 to 60 ms. (Note that the backpressure pause time in our tests was 40 ms.) In the last section, we explain these delay values.

3.3.3 NetMeeting

This section presents qualitative results of tests using NetMeeting audio and video and TCP file transfers. The results of these tests are hard to quantify; to some extent, audio and video performance is subjective.

All tests in this section involved 7 TCP connections running simultaneously with the NetMeeting session. The delay was fixed at 15 ms and the token bucket rate at 400 KB/s. Separate tests were run for video and audio. With a still image, a video stream generates about 9 KB/s of traffic; when there is a lot of movement, it generates about 15.5 KB/s. NetMeeting generates an audio stream of up to 4.5 KB/s.

Without backpressure

In this scenario, audio quality is not bad, but, sporadically, a small part of the audio stream is lost. The reason for this is that as soon as packet drops start to occur, TCP lowers its rate. However, because audio only needs a small part of the bandwidth, almost no audio frames are dropped. When more background traffic is introduced, more video packets are dropped, which leads to even stronger periods of freezing of the video image and more corruption of the image.

With backpressure without priority for UDP

In this scenario, there is some delay in the audio stream, but no packets are dropped, and performance is optimal. There is more delay (as much as 2 seconds) in the video stream. The larger the video stream, the greater the delay. If the video image does not change much, the delay decreases. Many changes to the video image over an extended period of time lead to delay and, ultimately, to some dropping, which occurs when the queue is full.

With backpressure with priority for UDP

In this scenario, audio performance is good, because there are no drops, and video performance is optimal. Unlike the case in the previous scenario, there are no noticeable delays. Furthermore, because there are no drops, image freezing does not occur either.

In conclusion, our tests indicate that NetMeeting performs best when backpressure is employed and the router gives priority to UDP traffic.

3.3.4 Role of Thresholds

Two of the most important parameters in the backpressure policing mechanism are the low and high thresholds. In order to set them appropriately, it is essential to study their influence on results. In this section, we analyze the performance of the backpressure mechanism with different threshold values.

Position of the low and the high threshold

In our tests, we fixed the low threshold at 60% and the high threshold at 80% of the size of the token bucket.

The low threshold should not be set too low, because it can take some time for the sender to react to the backpressure signal, and in that time packets can still be received. Indeed, if the low threshold is set too low, there will not be enough tokens to forward these extra packets, and they will be dropped, which is exactly what the backpressure mechanism is designed to prevent.

The low threshold setting can also affect the maximum burst size, as follows. Without backpressure, the maximum burst size is the token bucket size, but if backpressure is enabled, the maximum burst size is the difference between the token

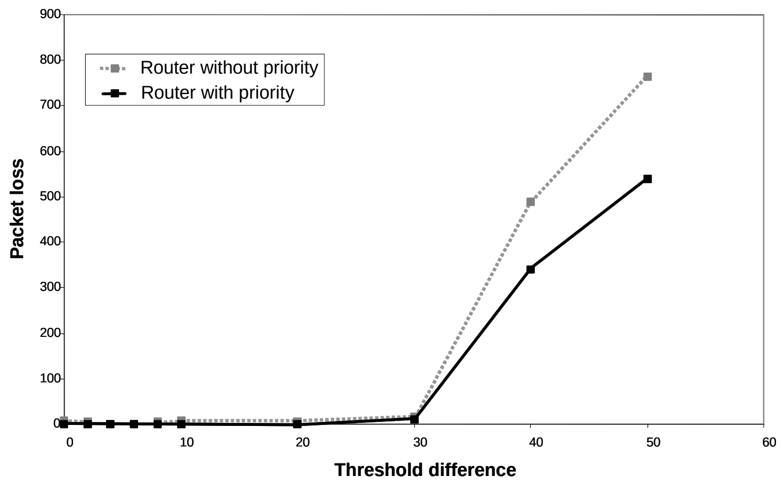


Figure 3.10: Packet loss for UDP traffic with different threshold differences.

bucket size and the low threshold, plus one or two packets transmitted before the sender can react to the backpressure signal.

Difference between the low and the high threshold

The difference between the low and the high threshold has a significant impact on the results, because it directly affects the pause time for the sender. When the difference between the two thresholds increases, the sender has to wait longer before the high threshold is reached; on the other hand, fewer pause frames have to be sent out.

To test the impact of the difference between the thresholds on test results, we performed several tests with varying threshold differences. For these tests, the token bucket size was set to 80 KB and the rate to 400 KB/s. Six TCP connections were set up simultaneously with a UDP connection with the same token bucket settings that were used in the previous experiments. Figure 3.10 shows the packet loss in these tests, with and without UDP priority queueing. It can be observed that in both cases a large difference between the thresholds leads to packet loss. Drops occur somewhat sooner without UDP priority queueing. The reason for this is that, when the thresholds are far apart, the router has to pause for a long time, which causes its queues to be filled up. Then, when the queues are full, packets are dropped.

Figure 3.11 shows the maximum jitter with and without UDP priority queueing. Without priority queueing, the maximum jitter is almost constant at 450 to 500 ms. With priority queueing, the maximum jitter starts low and increases when the threshold difference increases. The reason for this is that, with priority queueing, jitter is caused primarily by the duration of the backpressure signal.

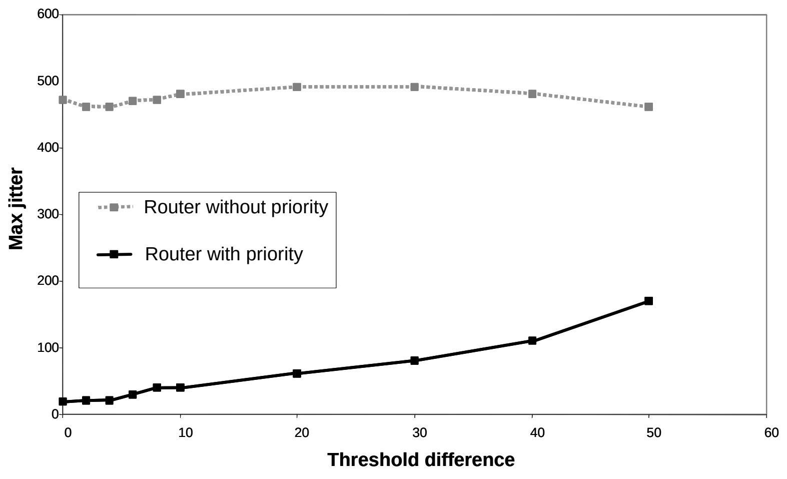


Figure 3.11: Maximum jitter with different threshold differences.

3.3.5 Maximum Jitter

Maximum jitter denotes the maximum difference in delay between two packets. One contributor to jitter is the pause duration or pause time. As we have noted earlier, whenever the PIR is exceeded, a pause frame is sent, and the following packet has to wait at least as long as the pause time. The pause time is the time needed to accumulate enough tokens in the bucket. More specifically it is the difference between the high and the low thresholds divided by the PIR. For all tests in this chapter, except for the threshold tests, the pause time was (64 KB - 48 KB) divided by 400 KB/s, which equals 40 ms.

Queueing delay is more difficult to calculate, because it varies, depending on scenario and queue type. A queue is typically limited either by the number of packets or by the number of bytes it can contain. We assume a queue that has a limit on the number of packets, but the reasoning below can easily be applied to a queue that has a limit on the number of bytes. To determine the maximum jitter, we assume that the queue is completely filled. We also assume that the following parameter values are known:

UDP_psize	=	average UDP packet size,
non_UDP_psize	=	average non-UDP packet size (usually 1,514 bytes),
$fraction\_UDP\_in\_queue$	=	fraction of the queue occupied by UDP packets,
$queue\_size$	=	queue size in packets.

Note that UDP packet sizes can vary significantly by application. Also, the percentage of the queue consisting of UDP packets can be hard to estimate, because it can vary over time.

Given the values of the parameters above, we can calculate the following

$$\begin{aligned} queue\_UDP &= \text{number of UDP bytes in the queue,} \\ queue\_other &= \text{number of other (i.e., non-UDP) bytes in the queue,} \end{aligned}$$

as follows:

$$\begin{aligned} queue\_UDP &= fraction\_UDP\_in\_queue \times queue\_size \times UDP\_psize, \\ queue\_other &= (1 - fraction\_UDP\_in\_queue) \\ &\quad \times queue\_size \times non\_UDP\_psize. \end{aligned}$$

Finally, this allows us to calculate the queueing delay:

$$queueing\_delay = (queue\_UDP + queue\_other)/PIR.$$

In our tests, the size of all UDP packets was 1000 bytes, and the size of almost all TCP packets was 1514 bytes. Since the UDP rate is half the PIR, we estimate the queue to contain 50% UDP and 50% TCP traffic. The total queue size of the router we used in our tests is 164 packets. Without priority queueing, packets are dropped at the router, so we know that most of the time the router queue is almost completely filled. The number of UDP bytes in the queue is approximately $0.5 \times 164 \times 1000 = 82$ KB. The number of TCP bytes is approximately $0.5 \times 164 \times 1514 = 124$ KB. So the total queue size will be about 206KB. The queue empties at the rate of the PIR, which is set at 400KB/s. So it will take approximately 515 ms for the queue to be emptied. This is consistent with the value we observed in our UDP test with backpressure enabled and priority queueing. With priority queueing, almost no UDP queues are built up, and the maximum jitter should be equal to or slightly higher than the pause time of 40 ms. This is consistent with the 50 to 60 ms maximum jitter we observed in our tests.

3.4 Conclusions

In this chapter, we have proposed and analyzed a traffic-policing mechanism based on backpressure. The backpressure mechanism sends a backpressure signal to the customer, if the agreement not to send traffic at greater speed than the specified peak rate is about to be violated. Thresholds, combined with a policing method based on a token bucket, are used to create this mechanism. When the number of tokens in the token bucket falls below a low threshold level, a backpressure signal is sent; it is released when the number of tokens rises above a high threshold level. This mechanism is compared to the simple and widely used approach of simply dropping traffic when the contract is violated.

Experimental results indicate that the backpressure mechanism is extremely effective for TCP traffic; it optimizes both throughput performance and fairness. However, for UDP-based multimedia applications, the effect is mixed. In the absence of QoS support, the use of backpressure can interfere with UDP performance, because considerable jitter can occur. In this case, the operator can choose to disable the backpressure mechanism. But when priority is given to UDP traffic, which should be the case since it transports time-sensitive traffic, backpressure performs well. For example, it reduces packet drops, which is extremely important for real-time applications that are also drop sensitive (such as NetMeeting video). The delay with backpressure and with UDP priority is larger than it is without backpressure, but it is still within typical end-to-end delay requirements. For example, the use of backpressure causes an end-to-end delay of 50 to 60 ms for voice (as compared to 10 to 20 ms without backpressure), but this is still far below the upper bound of 100 ms for reasonable voice performance. In fact, 40 ms of the 50 to 60 ms delay is due to the pause time of the backpressure. Setting the thresholds closer, further reduces this delay. Thus, we can conclude that backpressure performs well for both TCP and UDP, and is a better choice than the simple “drop above PIR” practice.

Chapter 4

A dynamic token bucket policer

In Chapter 3, we presented a policing method which improves performance of TCP traffic using the IEEE 802.3x capability. In this chapter we propose an alternative approach which does not require Ethernet specific hardware functionality. We aim at achieving TCP throughputs close to the (SLA) contractual traffic rate. Ideally this TCP performance should not depend on the specifics of the aggregate traffic profile which is being policed. In other words, the TCP goodput should be close to the policed rate even if the target (policed) flows behave unexpectedly due to large values of round trip times (RTTs), or varying level of aggregation (number of TCP connections policed as an aggregate), difference between link speeds and the policed rate etc. We try to achieve this by introducing a dynamic bucket size for the token bucket policer. Our mechanism monitors TCP's reaction to packet drops and dynamically determines the appropriate bucket size. With rigorous simulations we show that our scheme converges to the optimal bucket size in most cases and this is achieved without any knowledge of parameters such as round trip time or level of aggregation of the traffic being policed. Although TCP's interaction with a token bucket policer has been widely studied ([70], [12], [40], [90]), the previous work has not achieved the configuration of token bucket parameters independent of the traffic profile.

The rest of the chapter is organized as follows. In Section 4.1 we first explain the traditional token bucket policing mechanism. TCP performance with such a token bucket policer is analyzed in Section 4.2 with special attention to the effect of increasing the bucket size. This analysis not only sheds light on the adverse interaction of the token bucket and TCP but also motivates and rationalizes the design of our dynamic token bucket solution. The results of Section 4.2 indicate that one possible solution is to use a very large static bucket size for the policer. In Section 4.3 we discuss the disadvantages of using such a large static bucket size. Our dynamic bucket size solution is presented in Section 4.4 and its performance analysis in section 4.5. We conclude the chapter in Section 4.6.

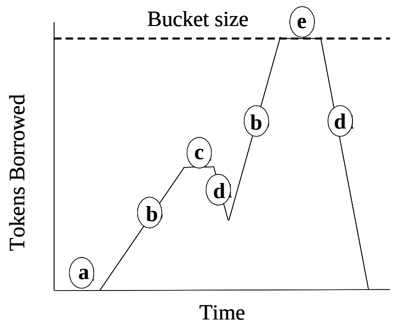


Figure 4.1: Schematic overview of tokens borrowed.

4.1 Token bucket policer

In this section we describe the token bucket policer that is widely used in packet networks to police traffic according to SLAs. The explanation provided here is different from that in Section 3.1 and is aimed at setting the path to the introduction of our dynamic token bucket policer later in this chapter.

A token bucket policer [51] uses tokens to monitor the incoming traffic rate and drops packets if they do not conform to the desired policed rate or peak information rate (PIR). When a customer exceeds its PIR, packets are not immediately dropped. Instead the initial burst exceeding the rate is allowed to pass through by borrowing tokens from the token bucket. Figure 4.1 shows the possible situations of the tokens being borrowed from a token bucket, where a token represents one byte. The solid line shows the amount of tokens borrowed and the dotted line shows the bucket size. Initially, there are no tokens borrowed ((a) in Figure 4.1). If the traffic rate is equal to or smaller than the PIR, i.e. the customer is not exceeding the agreed rate, no tokens have to be borrowed. However, when the traffic rate exceeds the PIR, tokens need to be borrowed for the amount of traffic exceeding the PIR (b). If the customer then starts sending exactly as much as the PIR, the amount of borrowed tokens remains stable (c). If the traffic rate drops below the PIR again, the part of the rate that is not used determines the amount of tokens that are returned (d). To limit the total amount of tokens that can be borrowed, a bucket size is determined. If the amount of tokens borrowed reaches that limit (e), the policer stops lending new tokens until some of the old ones are returned. Hence, if the customer continues to exceed its PIR, the excess part of the packets will be dropped.

4.2 TCP Performance with token bucket policing

In the previous section we have explained the functioning of a token bucket policer. In this section we analyze the performance of TCP with such a token bucket policer

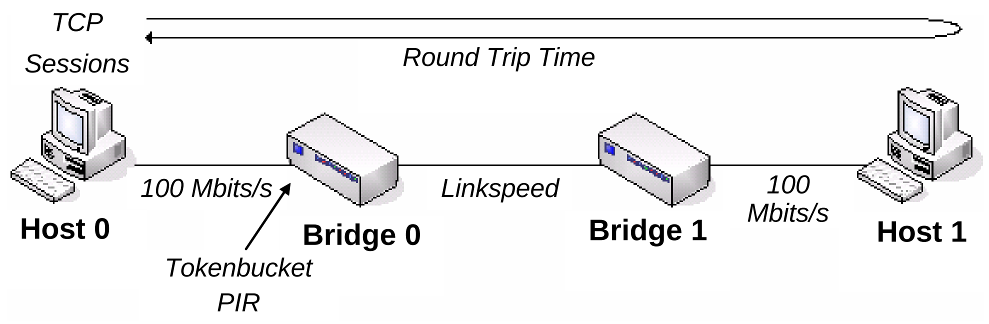


Figure 4.2: Simulated network scenario.

and identify potential problems. In order to carry out this analysis we have used a simulation environment consisting of two discrete event simulators: OMNeT++ [2] and Network Simulator (NS2, [1]). OMNeT++ was used to simulate the network and NS2 to generate the traffic flows. The support of different kinds of traffic is more mature in NS2 compared to OMNeT++, whereas the user-interface and the modularity are strong points of OMNeT++ over NS2. We have used the Libsynk library [61] to couple NS2 and OMNeT++. This library facilitates the synchronization of the simulation time and the exchange of events between the simulators.

The network topology used for all simulations is shown in Figure 4.2. The token bucket policing functionality was built into the OMNeT++ nodes, which are Ethernet bridges in Figure 4.2. In the simulations we considered a wide variety of scenarios with different traffic and networks parameters such as the round trip time (RTT), number of parallel TCP connections, WAN linkspeeds and the PIR. We have used RTT values of 10 ms, 50 ms and 100 ms. The WAN linkspeeds were 2, 10 and 50 Mbits/s and the PIR values were 0.5, 1, 2, 5 and 10 Mbits/s. The traffic scenario consisted of parallel persistent TCP connections policed as an aggregate. The number of such TCP connections was varied as 1, 2, 5 and 10 connections. Since we want to improve the performance of flows sharing a single policer, the considered topology and parameters are sufficient to address most relevant parameters and situations. In this chapter our goal is to focus on the direct interaction of TCP with the token bucket. Therefore we assume that policing is the only source of packet drops and consider congestion free scenarios. Consequently, we did not choose PIR values that exceed the link speed. We also did not consider the cases in which the goodput is limited by the TCP window size instead of the PIR. This happens for instance, in the case of a 100 ms RTT, packet sizes of 1500 bytes, and a maximum TCP window size of 42.67, the maximum achievable goodput is $1/0.1 \times 8 \times 1500 \times 42.67 = 5.1$ Mbits/s. So even with a PIR of 10 Mbits/s, the goodput is limited to 51% of the PIR. We discuss the impact of the various parameters below.

Token bucket size: Extensive simulations were carried out with all possible combinations of the parameter values introduced previously in this section. In the

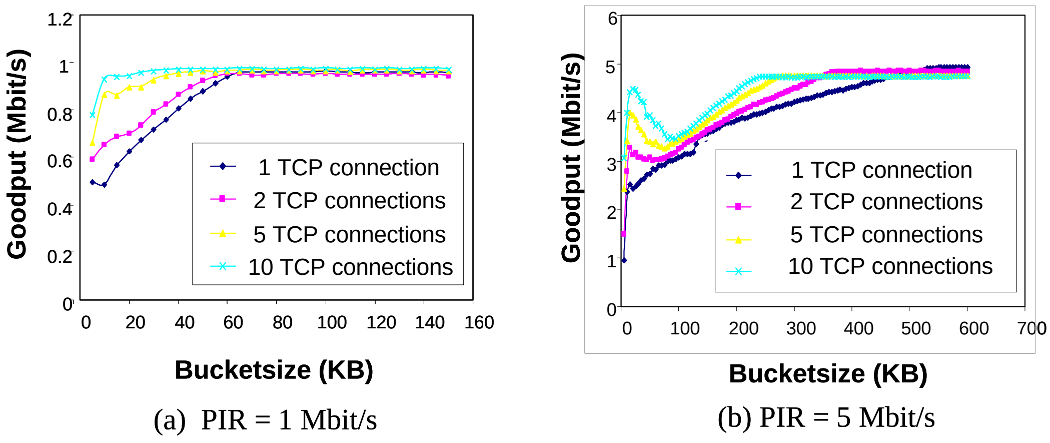


Figure 4.3: Bucket size vs TCP goodput (linkspeed=10Mbit/s, RTT=50ms).

majority of the cases the trends observed regarding the effect of the token bucket size were as shown in Figure 4.3a. However, some exceptions were also observed. These are shown in Figure 4.3b.

We first address the results shown in Figure 4.3a. In order to explain the reason behind the trends in Figure 4.3a, we will use Figure 4.4. Figure 4.4 presents a schematic overview of the effect of the bucket size on TCP performance. Figure 4.4a shows the tokens borrowed over time in case of a small bucket size. In the slow start phase of a TCP file-transfer, TCP increases its rate quickly. The PIR is quickly exceeded and many tokens are borrowed from the bucket. At some point the maximum bucket size is reached, after which no more tokens can be borrowed and packets are dropped. Due to consecutive packet drops, TCP reduces its window multiple times. This results in a traffic rate significantly lower than the PIR, causing tokens to be returned. Since the TCP sending rate drops below the PIR, the token bucket recovers and tokens are slowly returned, until no tokens are borrowed anymore (start of T2). After a certain period, TCP recovers and starts increasing its sending rate such that tokens have to be borrowed again (end of T2). Only during the T2 period goodput is 'lost', since the sending rate is lower than the PIR, while TCP has enough data to send. This implies that the length of the period T2 has a considerable impact on the goodput.

Figure 4.4b presents the results from the same scenario as Figure 4.4a, but now with a larger bucket size. Since more tokens can be borrowed with a larger bucket, it also takes longer before all tokens are returned again. Therefore, the resulting T2 in this case is a lot smaller, which increases throughput compared to the previous example. The time B denotes the time between the last packet drop and the first time new tokens are borrowed. Note that this time is equal for both figures, which is determined by TCP's congestion control mechanism and not by the

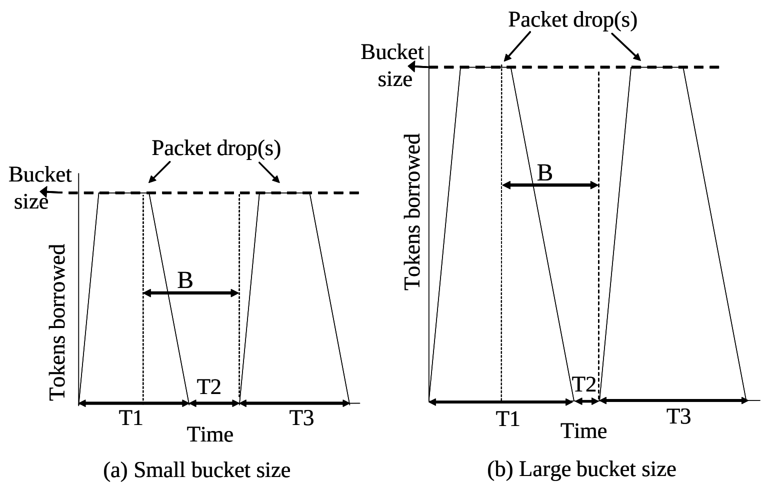


Figure 4.4: Influence of bucket size on TCP packet drops.

token bucket. Whereas the rate of the tokens being returned in this case is solely determined by the PIR, since TCP does not send anything during this period.

The phenomenon that bigger bucket sizes lead to better TCP performance was observed in almost all scenarios. However, one of the few exceptions in trends is shown in Figure 4.3b. At a token bucket size of 20 KB, a peak in goodput can be seen. For 5 and 10 simultaneous TCP connections, this peak at 20kB is higher than the goodput observed with bucket sizes of up to 150 KB. We explain this behavior by looking at the TCP window sizes of the 20 KB and 100 KB bucket size cases with 10 simultaneous TCP connections in Figure 4.5. Note that the higher the window size of a TCP connection becomes, the higher the offered traffic rate and burst will be. Figure 4.5b shows the results with bucket size 100 KB, in which case synchronization of all TCP flows occurs. All TCP connections try to increase the window size simultaneously until the token bucket has loaned out all its tokens and packets are dropped. Since all connections experience packet drops at the same time, they all reduce their window size to 0 also at the same time. Therefore, no traffic is sent at all for a short time. For a bucket size of 20 KB in Figure 4.5a, this synchronization does not occur. Packet loss affects only a few of the 10 connections and only these connections reduce their window size to zero. The other connections can take over that bandwidth and increase their window sizes. This asynchronization in the TCP flows leads to a higher goodput with a bucket size of 20 KB as compared to a bucket size of 100 KB. However, the optimal goodput is still achieved at bucket sizes greater than 250 KB.

Number of TCP flows: Figure 4.3a shows that a single TCP connection can suffer significantly with a token bucket policer. In this case if very small bucket sizes (<10 KB) are used the goodput achieved is only half the PIR. Increasing the

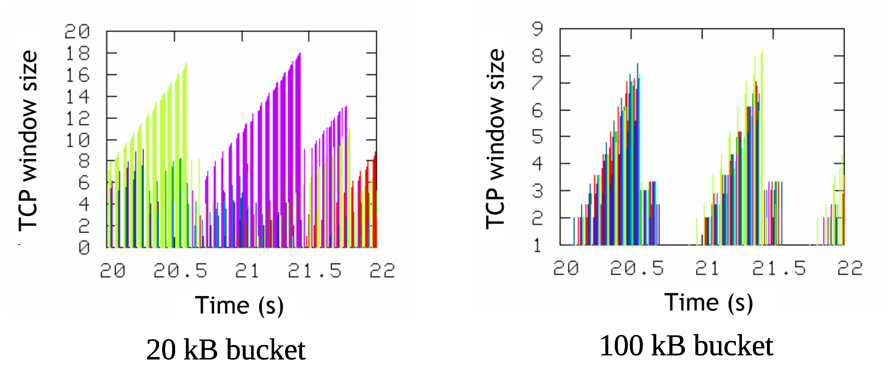


Figure 4.5: TCP window size dynamics with linkspeed =10 Mbits/s, RTT=50 ms, PIR=5 Mbits/s and 10 TCP connections.

number of TCP connections sharing the token bucket reduces the bucket size needed to reach optimum goodput in all cases. The reason behind this is that multiple TCP connections are likely to be asynchronously affected by packet loss. Thus the connections that are not affected by loss can take advantage of the situation, improving the aggregate goodput. On the other hand a single TCP connection will be more affected by packet loss. Thus the aggregate goodput will be hampered deterministically.

4.3 Drawbacks of a large static bucket size

From the results of the previous section we can conclude that choosing a sufficiently large value for the bucket size will result in optimal goodput. However, what is sufficiently large for one scenario might not be large enough for another. This implies that in order to obtain optimal goodput for all possible cases, the size of the bucket would have to be set to the maximum of all required bucket sizes. Using such a large bucket size would allow large data bursts into the network, which has the following potential drawbacks:

A. Unpredictable traffic: Larger bursts imply unpredictable traffic, which makes it more difficult to engineer and provision network capacity, requiring more overdimensioning to reduce congestion.

B. Temporary congestion: Allowing larger bursts induces more congestion in the network. Although congestion would only be temporary, large buffers would be required to contain these bursts.

C. Increased packet drops/latency: If the buffers in the network are not sufficiently large to contain the packets bursts allowed by the policer, packet and throughput loss will occur. This would imply that we would be shifting the problem

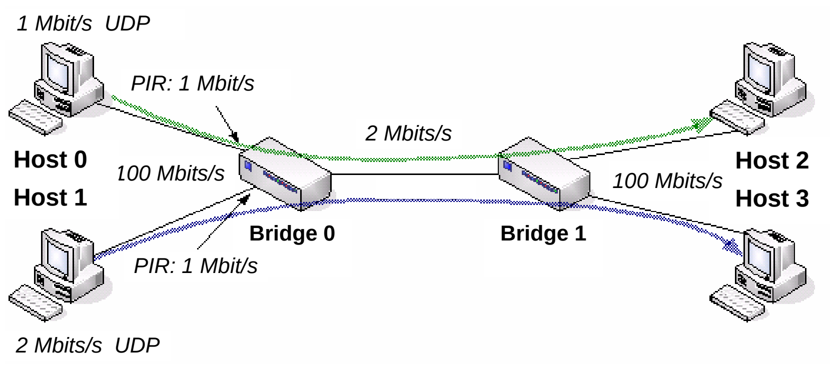


Figure 4.6: Large bucket size scenario.

from the policer to the buffers. On the other hand, one should bear in mind that large buffers potentially lead to higher latency and jitter, which is undesirable for time-sensitive flows. We illustrate this phenomenon with a specific scenario in our context. This scenario is outlined in Figure 4.6.

In Figure 4.6 host 0 and host 1 are sending Poisson traffic to host 2 and 3 respectively using UDP. Packets are 1500 bytes in size. Host 0 sends 1 Mbit/s and is policed at 1 Mbit/s. Host 1 sends 2Mbit/s but is allowed only 1 Mbit/s by its traffic contract. About half of the packets from host 1 are dropped, because it is exceeding its PIR. Figure 4.7 shows the average delay of the traffic from host 0 for different bucket sizes. It is clear that for larger bucket sizes, the flow from host 0 experiences considerable delay (up to 350 ms), while complying perfectly with its traffic contract. This is a side effect of Host 1 temporarily being allowed to burst more traffic than 1 Mbit/s into the network. This burst fills the buffers of bridge 0 and since the traffic of host 2 is using the same buffers, latency is introduced. Note that at bucket sizes of 1100 KB and larger, the average delay does not increase anymore. This is because in these cases the buffer limit of 56 packets is reached in bridge 0 and further incoming packets are dropped. If bigger buffers are used, the delay will increase to a larger value. Since in practice the same policer could also be used for non-TCP flows, it can considerably hamper their performance by increasing the jitter and latency. Similar behavior will also occur with TCP flows that only need a small bucket size. Therefore, we conclude that it is impractical to design a policer with an excessively large static bucket size optimized only for TCP traffic.

4.4 A dynamic bucket size policer

The previous section shows that on one hand TCP goodput benefits from a large bucket size, on the other hand this large bucket size has drawbacks, such as increased

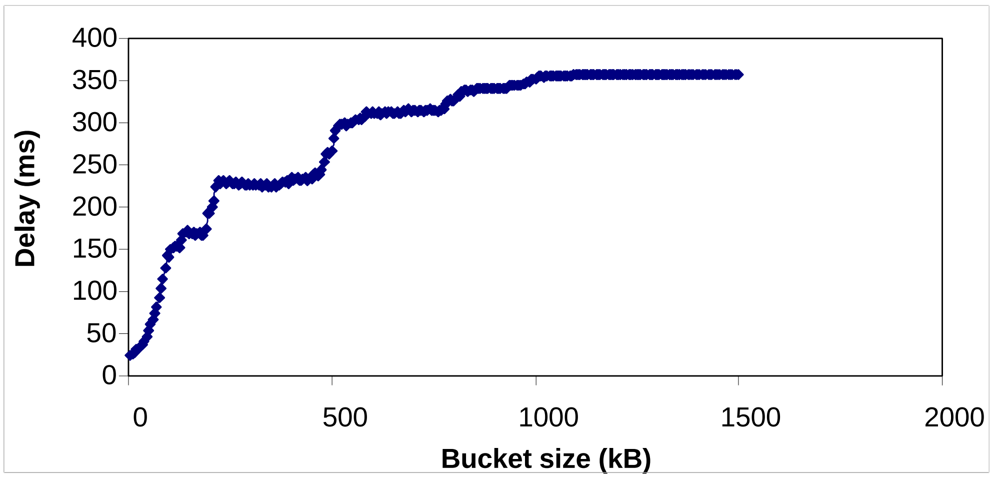


Figure 4.7: Average packet delay of traffic from Host 0 on the large bucket size scenario depicted in Figure 4.6.

congestion and delay in the network. In this section we present our solution, which aims at automatically adjusting the bucket size to find an attractive trade-off between high goodput and non-excessive burst size. Our mechanism monitors the response of the policed traffic to packet drops and uses this information to decide whether or not to modify the bucket size. This simple principle works elegantly for both loss-sensitive TCP and loss-insensitive real-time flows. For TCP traffic the policer detects that consequent to packet loss, TCP stops sending for too long causing the available policing bandwidth to be under-utilized. In this case, it increases the bucket size. For loss-insensitive real-time flows, the policer observes that the policed traffic does not react to packet loss and the available policing bandwidth continues being utilized as before. In this case, the policer leaves the bucket size unaltered, avoiding situations illustrated in Section 4.3.C.

Figure 4.8 shows a graphical overview of our solution. The bucket size is initially set to a preset minimum. As explained in Section 4.2, period T2 represents time during which goodput is 'lost', since then the sending rate is lower than the PIR. The dynamic token bucket policer monitors the activity in every cycle with the goal to increase the bucket size in the next cycle such that the period T2, would be exactly zero, leading to optimal goodput. In order to achieve this, our scheme identifies two relevant time periods. Period A starts at the last packet drop and ends at the time when all tokens are depleted. Period B also starts at the last packet drop but ends when tokens are being borrowed again. During A we deplete all tokens from a bucket of size, let's say, $Bucket\_size$. Therefore, in period B we could possibly deplete all tokens from a bucket of size $(B/A)\times Bucket\_size$. Hence, the bucket size should be a factor B/A larger for T2 to become zero.

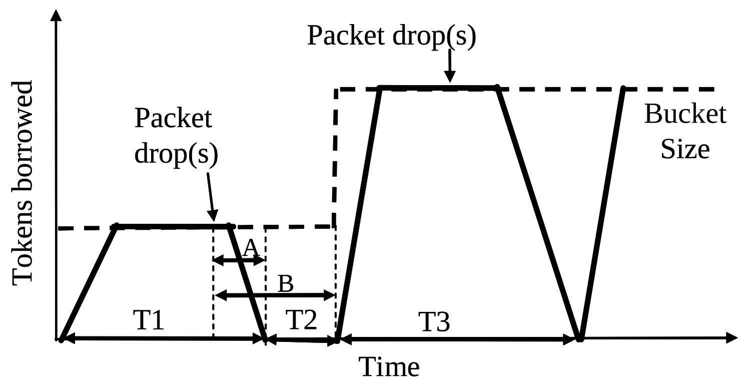


Figure 4.8: Behaviour of the dynamic bucket size policer.

In the performance assessment of this scheme we have chosen to increase the bucket size in every consecutive cycle by only 80% of the calculated (B/A) value. We increase the bucket size by 5% if the bucket size is already above this 80%. This way, the optimal bucket size is obtained in a few steps, but the size is not increased too much when sudden bursts in the traffic occur.

The monitoring time of our scheme should also be limited in order to avoid misinterpreting inactive periods as consequences of packet loss. We chose a value of 0.6 seconds for this monitoring time. If within this time all tokens are returned and borrowed multiple times, we take the first occurrence of the tokens being returned and the last occurrence of tokens being borrowed again. Furthermore, the dynamic bucket size can only vary between two values, the minimum and the maximum bucket size. We chose a minimum bucket size value of 10 KB and disabled the maximum bucket size. In practice, the maximum bucket size should be set to the absolute limit of the burst that is allowed to enter the network.

In order to prevent too large bucket sizes, the bucket size is decreased by a percentage every fixed time period in which the bucket is not increased. We have chosen a decrease of 5% every second.

The choice of parameters values for our scheme such as the monitoring time (0.6 s), bucket size increase (0.8 B/A) and decrease factor (5%), were estimated by initial simulations.

4.5 Simulation results

In this section our goal is to systematically assess the performance of the dynamic bucket size policer introduced in the previous section. We have used both real-time

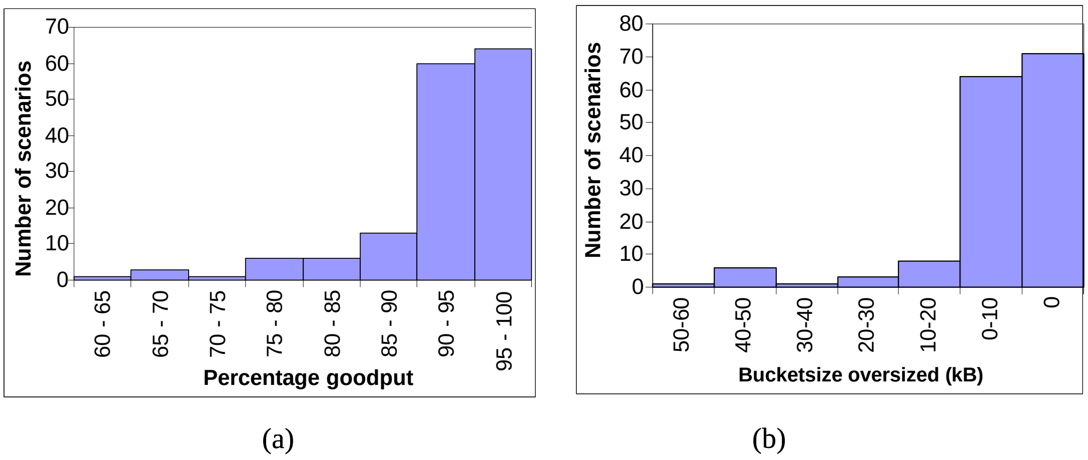


Figure 4.9: Achieved bucket size and goodput results with the dynamic token bucket.

UDP Poisson traffic and TCP data traffic. With respect to the UDP traffic our goal was to overcome the drawbacks explained in Section 4.3.C. In case of a Poisson UDP stream, our solution starts with the minimum bucket size. If the UDP rate is larger than the PIR, the token bucket will slowly borrow tokens up to the bucket size at which point packets are dropped. UDP will not adjust its sending rate because of the drops and will continue exceeding the PIR. In this case, the bucket never becomes empty and the bucket size is never increased, keeping the burstiness minimal and the goodput optimal.

In order to assess the performance of TCP flows we have used the same scenarios as in Section 4.2. We focus on two aspects to evaluate our solution. First, the goodput should be close to the configured PIR. Second, this goodput should not have been achievable with a smaller bucket size than that chosen by the dynamic token bucket policer. Below we discuss these two performance aspects.

4.5.1 Goodput

In this sub-section we discuss the performance of the dynamic bucket size policer in terms of goodput. The goodput is expressed as percentage of PIR. A total of 156 simulation scenarios were executed. The average goodput over all tests is 92%. Figure 4.9a groups the results of these scenarios into different goodput ranges. We discuss these different cases below.

90-100% PIR: Almost 80% of the scenarios (124 out of 156) considered provide goodput in the range of 90-100% of PIR. Figure 4.10a shows the behavior in these cases. This figure is the same as Figure 4.3a, but now the dots show the goodput with our scheme. The goodput is always above 94%, while the bucket size is kept to a minimum.

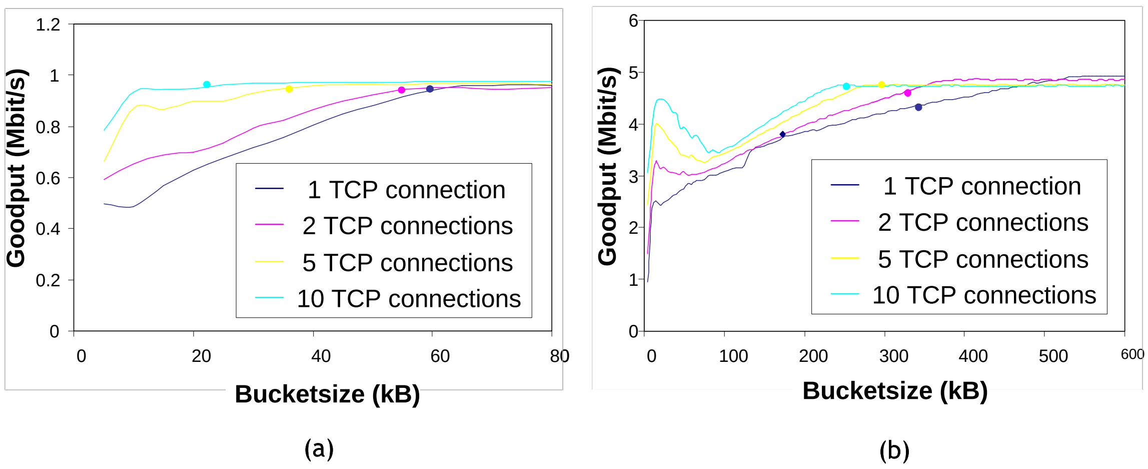


Figure 4.10: Bucket sizes and goodput achieved with the dynamic token bucket policer in comparison to a fixed bucket size policer.

60%-90% PIR: 16% of the 156 simulations executed resulted in goodput in the range 75-90% of PIR, whereas only 2% perform in the range of 60-75% of PIR. The results and performance trends in these cases are illustrated in Figure 4.10b. The figure shows results of our proposed policing scheme in addition to the results of Figure 4.4. It can be observed that our scheme does not get confused with the early goodput peaks at 10 KB bucket size, since the goodput is not optimal there. For 5 and 10 TCP connections, our scheme finds the optimal average bucket size. For 1 and 2 TCP connections, the average bucket size is a bit too low, causing a lower goodput. To find out the reason for this we have to look into the details of the algorithm used to determine the bucket size.

Figure 4.11 tracks the algorithm used to calculate the bucket size during the course of a simulation run with 1 TCP connection. The figure shows the dynamic (calculated) bucket size, the tokens borrowed and the small vertical lines at time 151.5 s denote packet drops, where each line represents one drop. At time 151.5 s, all the tokens in the bucket have been borrowed. This causes multiple packet drops, as indicated by the blue lines. It will take up to time 153 s for all tokens to be returned and for new tokens to be borrowed again. This implies that our dynamic bucket policing scheme should use a monitoring time of 1.5 s to determine the new bucket size. However, we have set this monitoring time to 0.6 seconds. This results in the bucket size being increased at 152.1 s instead of at 153 s. The mechanism assumes that this time is the end of T2 and calculates how large the bucket size should be to make T2 zero. This results in a new bucket size which is insufficient for the burst in the next cycle starting at 153 s. This effect repeats every cycle and results in loss of goodput. We have noticed this problem mostly in scenarios with a relatively large RTT (50, 100ms) and not with small RTTs (10 ms). The reason

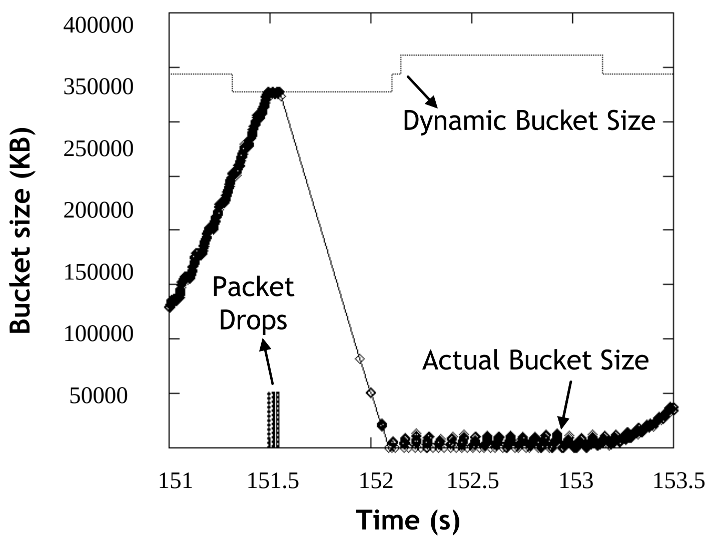


Figure 4.11: Dynamic bucket size (top line), actual bucket size (bottom line) and packet drops (vertical lines). Linkspeed: 10 Mbit/s, RTT: 50 ms, PIR 5 Mbit/s, TCP connections: 1.

being that a monitoring time of 0.6s is relatively short for large RTTs but not so for small RTTs. It is unclear if a larger monitoring time would be enough to solve this issue or a stronger coupling is required between the monitoring time and the RTT. This is a subject for future research.

4.5.2 Average bucket size

In this sub-section we address the efficacy of our policing scheme in terms of minimizing the allowed burst into the network. The allowed burst corresponds to the bucket size determined by the dynamic bucket size mechanism during the course of a simulation run. Figure 4.9b quantifies and classifies the cases in which our policing scheme calculates an oversized bucket size. An oversize of 10 KB, means that the goodput achieved by the dynamic bucket size policer for this case could have been achieved with a 10 KB lower static bucket size. It shows that in most scenarios (64%), there is no smaller bucket size with the same goodput (0 KB bucket oversized). In about 25% of the cases the bucket size falls within 10 KB oversize. A handful of scenarios show a largely oversized bucket. These correspond to the cases where the configured PIR is close or equal to the linkspeed. In fact, in these cases we do not really need any policing function. The dynamic bucket policer observes small bursts in TCP traffic and interprets this as a signal to increase the bucket size, which is completely unnecessary. This leads to an overestimation of the required bucket size. Since this scenario can be easily identified, a different strategy can be used to resolve the problem, e.g. by completely disabling the policing function.

4.6 Conclusions

In this chapter we addressed the performance and direct interaction of TCP and a bufferless token bucket policer. We first analyzed the influence of the token bucket policer on TCP goodput for a wide variety of scenarios. The results pointed towards using a very large bucket size (sufficiently large to cover all possible scenarios) as a possible solution for improving TCP goodput. However, a large static bucket size allows large data bursts into the network which we showed to be of major concern. To overcome this we introduced a dynamic bucket size policer with monitoring and estimation functions. The monitoring function continuously monitors TCP's reaction to packet loss. The estimation function estimates the required bucket size to minimize collapse of TCP sending rate and therefore minimizes under-utilization of the configured bandwidth.

Performance assessment of the dynamic bucket size policer was conducted for different RTTs, link capacities, policing rates and number of persistent TCP flows. Results indicate that in most (80%) cases the dynamic bucket size policer converges to the optimal token bucket size required specific to the scenario considered. The TCP goodput achieved was 92% of the policed rate on average.

Situations under which the dynamic policer could improve its performance were also identified. These were the ones where the policed rate is very close to the access link capacity and the case of large RTTs. In the former case, the dynamic policer estimates a larger bucket size than required. It misjudges small bursts as indication of a large bucket size requirement. The large bucket size, although not utilized, could cause problems if TCP and UDP traffic are policed as an aggregate. However, it is simple to resolve this issue. Network provisioning can easily identify parts of the network where the policed rate is close to the access link capacity and completely disable the policing function at these points. The other situation where the dynamic bucket size policer performs non-optimally occurs when the RTT is large (100 ms). In this case, the monitoring time used was sometimes too small to adequately estimate TCP's reaction to packet drops. Optimizing the monitoring time as well as the interaction of the dynamic token bucket policer with network congestion are subjects for future research.

Part II

Congestion control

Introduction to Part II

Congestion control in Ethernet has become synonymous with the feedback flow control possibility provided by the IEEE 802.3x pause frames or backpressure. Most previous work in literature provides simulation based analysis of this mechanism and has concentrated on the extensions to the protocol and its implementation issues ([86], [29], [50]). In this part of the thesis we develop analytical models providing insight into the influence of the backpressure congestion control parameters on network and traffic performance. These models and their analysis, focusing on different aspects of the backpressure scheme are presented in the following chapters:

- Chapter 5 questions the use of the backpressure mechanism in Ethernet when the widely used higher layer protocol TCP already has its own method to deal with congestion. To address this issue we develop a Markov model with two nodes in tandem capturing the essentials of Ethernet congestion control. Furthermore, packet drops are coupled to reduction in input traffic rate and successful transmissions result in an increase of the input traffic rate, capturing the Additive Increase and Multiplicative Decrease (AIMD) aspect of TCP's congestion window. The analysis presented in the chapter provides useful insight into the effect of buffer sizes, congestion detection thresholds and burstiness in the TCP input traffic stream on the performance.

- Chapter 6 focuses in more detail on the effects of the Ethernet congestion-control parameter-values. The Ethernet backpressure mechanism uses two thresholds, one to signal the onset of congestion and the other to signal its end. These two threshold positions, on one hand, can prove very powerful in avoiding congestion by regulating the incoming traffic rate. On the other hand, their positions greatly influence the achieved throughput, delay and the signaling overhead. The proper tuning of these parameters is critical to the success of the backpressure congestion control method. Chapter 6, designs and analyzes a fluid flow queueing model of the Ethernet congestion control mechanism in a single node. This model is analytically solved by setting up Kolmogorov equations, solving them using spectral expansion and finally finding sufficient constraints to solve for the unknowns in the solution.

- Chapter 7 presents a numerical performance study of the backpressure mechanism using the model developed in Chapter 6. In particular, the influence of the congestion control thresholds on network and traffic performance is demonstrated, thereby, addressing an essential design criterion for the backpressure scheme. Furthermore, the chapter also demonstrates how the required trade-off between for example throughput, delay and signaling overhead can be made by a network operator.

Chapter 5

Interaction of Ethernet and TCP congestion control

An interesting and essential aspect in studying the performance of the Ethernet congestion control mechanism is to understand its influence and interaction on higher layer application traffic, especially since the most widely used transport protocol, TCP, has its own congestion control mechanism. Considerable work has been done and is ongoing that aims at understanding and predicting the performance of TCP under various situations and different network parameters. Use of an additional underlying hop-by-hop congestion control scheme adds another complication to this study. On one hand, the extra buffering of packets introduced by hop-by-hop control can avoid unnecessary TCP rate fluctuations. On the other hand, if the congestion cannot be solved by hop-by-hop control it might unnecessarily delay the reaction of TCP. Another aspect that needs to be studied before the hop-by-hop mechanism can be implemented is the influence of the various network and traffic parameters on the performance. The throughput of a TCP source is known to be inversely proportional to the round trip time and the square root of the packet loss probability. This relation which is widely known as the 'root p' law (see [58]) might not be valid for the combination of TCP and hop-by-hop flow control.

The Ethernet hop-by-hop congestion control mechanism has been evaluated with simulations in the literature. However, all of this previous work ([55], [85], [66], [22]) lacks guidelines for configuring the congestion control parameters of the scheme. Another class of (limited) literature concentrates on modifications of the protocol. For example, [29] proposes to distinguish the interpretation of the backpressure/pause message for different traffic classes. There is also a significant amount of work done on ATM (Asynchronous Transfer Mode) based hop-by-hop flow control as in [60] and [52]. However, most of this work is based on negotiating transmission rates between the congested node and its upstream neighbors. The proposed mechanisms use resource management cells which do not exist in Ethernet.

In this chapter we develop a Markovian model that captures the interaction of Ethernet hop-by-hop congestion control with TCP end-to-end congestion control. This model applies to TCP versions which use packet loss as a measure of congestion in the network. We assess the validity of the model by comparing it to extensive simulations. The results demonstrate the influence of various parameters on the performance of the schemes and provide guidelines for the choice of parameters such as buffer thresholds for congestion detection.

The rest of the chapter is organized as follows. In Section 5.1, we introduce the two congestion control mechanisms. The Markov model describing the integration of the hop-by-hop and the end-to-end congestion control mechanisms is presented in Section 5.2. The simulation model and parameter values used to obtain the results are described in Section 5.3. Section 5.4, shows the performance results both with the Markov model as well as the simulations, which demonstrate the influence of various network and traffic parameters. The parameters include buffer thresholds for congestion detection, the network round trip time (RTT) and the traffic burstiness. The conclusions are presented in Section 5.5.

5.1 IEEE 802.3x hop-by-hop and TCP end-to-end flow control

The main goal of a hop-by-hop congestion control mechanism is to contain temporary bursts and congestion by utilizing network resources (such as buffer space). The hop-by-hop nature of the congestion control scheme should prevent that congestion escalates to higher layers, for example TCP. In order to model and assess whether the interaction of hop-by-hop with end-to-end congestion control will result in improved network performance, we first need to understand the details of each of these mechanisms separately. These two different schemes as well as some initial modeling concepts are described in the subsections below.

5.1.1 IEEE 802.3x backpressure based hop-by-hop flow control

The backpressure scheme defined in IEEE 802.3x (see [76]), is intended to provide flow control on a hop-by-hop basis by allowing ports to ‘turn off’ their upstream link neighbors for a period of time. In the case of a half-duplex link this is realized by sending a jamming signal. The end-station perceives the medium as busy, stops transmitting and backs-off. For the case of a full-duplex connection, the IEEE 802.3x standard defines a MAC layer flow control mechanism. This mechanism is based on a special frame called ‘pause frame’ in which the pause period is specified. The end station or router receiving the pause frame looks at the pause period and does not transmit or attempt transmission for that amount of time. Alternatively

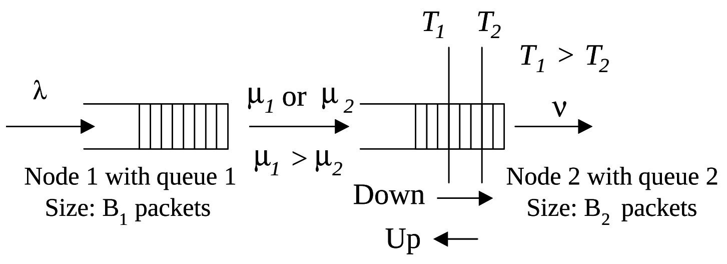


Figure 5.1: Hop-by-hop flow control.

a ON/OFF pause messages can be sent signaling the beginning and end of the 'transmission pause' phase.

This mechanism is illustrated by Figure 5.1. When the occupancy of queue 2 exceeds T_1 it can signal its upstream node 1 to stop all data transmission. When the queue occupancy of the congested node 2 drops below a low threshold T_2 the upstream neighbor can start transmission again at usual rate μ_1. The Markov model used to describe this mechanism in this chapter assumes that on receiving the congestion signal, the upstream neighbor lowers its rate to a generic rate μ_2 instead of strictly stopping all transmissions. A positive value for μ_2 may mimic the delay in reception of congestion messages by the upstream nodes by allowing possible incoming packets into a congested queue even after a congestion message has been sent.

It is clear from Figure 5.1 and the explanation above that when queue-2 occupancy is above T_1, the service rate of queue 1 is μ_2. Similarly, when the number of packets in queue 2 is below T_2, the service rate of queue 1 is μ_1. In the region between T_1 and T_2 it can be either μ_1 or μ_2, depending on the last threshold that was crossed. For example, when the queue-2 occupancy drops from above T_1 to below T_1, queue 1 continues to transmit at μ_2. Similarly, a rise in queue-2 occupancy from below T_2 to above T_2 implies that queue 1 continues transmission at rate μ_1. We will denote the state of queue 2 corresponding to a high transmission rate (μ_1) of queue 1 as the `up` or '1' state and that corresponding to μ_2 as the `down` or '2' state.

5.1.2 TCP end-to-end flow control

The widely used transmission control protocol (TCP), works on the principle that end systems should react to congestion anywhere in an end-to-end data path by adjusting their transmission rates to avoid total collapse. The TCP congestion control mechanism is a well studied subject and is often modeled as an Additive Increase and Multiplicative Decrease (AIMD) scheme (see [87], [18] and [6]). The TCP sending rate is controlled using a window. The window is halved in the event of a packet loss and increased by one packet upon acknowledgment of reception of all packets belonging to the current window. In this chapter, we assume that the packets sent in one window follow a Poisson process with the rate corresponding to the window size.

A single TCP source being an unrealistic scenario, we aim at modeling multiple TCP streams as input sources to a system with hop-by-hop flow control. An approach to modeling multiple TCP streams using Markov chains is to model each stream separately with its own AIMD characteristics. This approach has clear disadvantages with respect to the scalability of the Markov chain transition state space as observed in, e.g., [82]. An alternative to this approach is to model the traffic aggregate of TCP streams with a more generic AIMD mechanism. It is worth noting, though, that this option has the drawback that it is difficult to predict the exact sending rates, among which, the aggregate TCP streams will switch, when detecting packet losses. This complication arises from the fact that packet losses need not affect all TCP streams as observed in [82] and [14].

The arrival process of the aggregate input traffic stream is considered to be a Markov-modulated Poisson process (see [24]) with arrival rates varying between N different values λ_1 to λ_N, where, $\lambda_N < \lambda_{N-1} < \ldots < \lambda_1$. The state of the input traffic stream is represented by a, so that the corresponding traffic rate is λ_a where a varies from 1 to N. Every time there is a packet drop, the input traffic state increases to twice its original value, causing a significant drop in the traffic rate (unless $2a > N$, in which case the input rate drops to λ_N). The input traffic state decreases by one step at a rate equal to 1/RTT. This corresponds to an increase in traffic rate, the increase being dependent on the values of the λ_as. This simple approach incorporates the basic AIMD nature in the input traffic.

The different λ_as represent the possible sending rates of an aggregate of multiple TCP streams. It is not straightforward to specify the value of N, i.e., the total number of possible traffic rates, nor the value of the λ_as. This is so, because the aggregation of multiple TCP streams, each with its own AIMD characteristics can result in very unpredictable traffic characteristics. This unpredictability is caused by the varying number of packet drops and the unknown proportion in which they affect the multiple TCP streams. One extreme case could occur when the packet drops affect all streams equally, leading to the phenomenon called 'global synchronization' [26]. In this case the AIMD nature of a single flow is preserved. The other extreme could be that only one TCP stream is affected. Our approach to study the impact of such traffic will be to vary the value of N, thus increasing the granularity of the input process and observe the trends in the results.

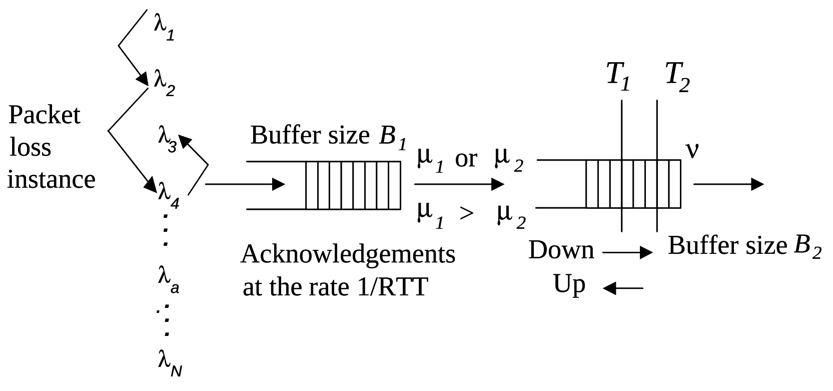


Figure 5.2: Integrated hop-by-hop and end-to-end flow control model.

5.2 Integrated model of hop-by-hop and end-to-end flow control

In this section, our goal is to integrate the models for hop-by-hop flow control and end-to-end TCP flow control. The backpressure based hop-by-hop congestion control mechanism, works by tuning the buffer occupancy per hop to avoid packet drops as much as possible. The TCP flow control on the other hand, reduces the rate of traffic into the network when packet drops may occur anywhere in an end-to-end data path. In Section 2, we have explained these two mechanisms in detail. The integrated model used for TCP input traffic along with backpressure hop-by-hop flow control is shown in Figure 5.2. It consists of augmenting the model for hop-by-hop flow control depicted in Figure 5.1 with a reactive Markov-modulated arrival process that captures the AIMD behavior of multiple TCP streams. If a packet is lost at either queue 1 or queue 2, the state of the arrival process is adjusted to twice its current value. For example, if this occurs while the TCP rate is λ_3, the new rate is λ_6. On the other hand, the state of the arrival process is decreased by one level on average every 1/RTT time units.

5.2.1 Transition equations

Below we define the states of the system shown in Figure 5.2. Clearly, the resulting process is also (continuous time) Markovian. The state of the system is represented by (i, j, k, a), which implies i packets in queue 1, j packets in queue 2, state k – `up` (1) or `down` (2) – for the second queue and state a for the TCP traffic rate (varying from 1 to N). The maximum number of TCP sending rate levels is N. For clearness of presentation we enumerate the different types of transitions.

1. $(i, j, k, a) \xrightarrow{\lambda_a} (i+1, j, k, a)$;
2. $(B_1, j, k, a) \xrightarrow{\lambda_a} (B_1, j, k, 2a)$, if $2a \leq N$;
3. $(B_1, j, k, a) \xrightarrow{\lambda_a} (B_1, j, k, N)$, if $2a > N$;
4. $(i, j, k, a) \xrightarrow{\nu} (i, j-1, k, a)$, if $j \geq 1$, $j \neq T_2$;
5. $(i, T_2, k, a) \xrightarrow{\nu} (i, T_2 - 1, 1, a)$, if $1 \leq T_2 \leq B_2$;
6. $(i, j, 1, a) \xrightarrow{\mu_1} (i-1, j+1, 1, a)$, if $j \neq B_2$, $i \geq 1$;
7. $(i, j, 2, a) \xrightarrow{\mu_2} (i-1, j+1, 2, a)$, if $j \neq B_2$, $i \geq 1$;
8. $(i, T_1, 1, a) \xrightarrow{\mu_1} (i-1, T_1 + 1, 2, a)$, if $T_1 < B_2$, $i \geq 1$;
9. $(i, B_2, 1, a) \xrightarrow{\mu_1} (i-1, B_2, 1, 2a)$, if $2a \leq N$, $i \geq 1$;
10. $(i, B_2, 1, a) \xrightarrow{\mu_1} (i-1, B_2, 1, N)$, if $2a > N$, $i \geq 1$;
11. $(i, B_2, 2, a) \xrightarrow{\mu_2} (i-1, B_2, 2, 2a)$, if $2a \leq N$, $i \geq 1$;
12. $(i, B_2, 2, a) \xrightarrow{\mu_2} (i-1, B_2, 2, N)$, if $2a > N$, $i \geq 1$;
13. $(i, j, 1, a) \xrightarrow{1/\mathrm{RTT}} (i, j, 1, a-1)$, $a > 1$.

Transition equation 1 implies that arrivals into the first queue occur at a rate λ_a. Equations 2 and 3 correspond to losses when queue 1 is full, while simultaneously increasing the input process state by a factor of two. Equation 3 in addition implies that if the calculated increase in the state of the input traffic is more than N then the state will drop to state N. Equation 4 implies departures from the second queue occur at rate ν as long as the second queue is non empty. Equation 5 ensures that when a departure from the second queue coincides with the queue occupancy dropping below the low threshold T_2, then the state of queue 2 changes into '1' (`up`). Equations 6 to 8 are about departures from queue 1 and simultaneous arrivals into queue 2. This happens at rate μ_1 if the state of queue 2 is '1' or at rate μ_2 if the state of queue 2 is '2'. However, when a departure from queue 1 and simultaneous arrival into queue 2 coincides with an increase of the queue occupancy of queue 2 from T_1 to $T_1 + 1$ then the state of queue 2 simultaneously changes to '2' (`down`). It is important to note that if T_1 is set equal to B_2 then effectively hop-by-hop flow control is not activated. Transition equations from 9 up to 12 are similar to 6 through 8, but with the additional condition that queue 2 is full when departures from queue 1 arrive into queue 2. This implies that packets will be lost and this will simultaneously affect the state of the input process and trigger a lowering of the traffic rate (increase by a factor of 2 of the state descriptor a). Equation 13 deals with an increase of the TCP traffic rate. The RTT determines the rate at which acknowledgements arrive and corresponds to the increase in TCP window size, therefore increasing the rate of the input process.

5.2.2 Performance measures

By $\pi(i,j,k,a)$ we denote the stationary probabilities obtained after solving the system $\pi \cdot Q = 0$ and $\pi \cdot e = 1$, where Q is the transition matrix of the above described Markov model and e is a column vector with all entries equal to 1. Then, the expected number of packets in queue 1 is

$$\mathbb{E}[\text{Queue 1}] = \sum_i i \sum_{j,k,a} \pi(i,j,k,a) \ ,$$

and the expected number of packets in queue 2 is

$$\mathbb{E}[\text{Queue 2}] = \sum_j j \sum_{i,k,a} \pi(i,j,k,a) \ .$$

From Little's law, the expected waiting time (that is, the time spent in the queue, including transmission) of packets in queue 1 is

$$\mathbb{E}[W_1] = \frac{\mathbb{E}[\text{Queue 1}]}{\lambda_{\text{effec},1}} \ ,$$

where, $\lambda_{\text{effec},1}$ is the effective arrival rate at queue 1 and is given by (using the PASTA property),

$$\lambda_{\text{effec},1} = \sum_{a=1}^{N} \lambda_a \sum_{\substack{i,j,k \\ i<B_1}} \pi(i,j,k,a) \ .$$

In addition, let us denote the effective loss rate at queue 1 by $\lambda_{\text{loss},1}$. This can be computed by

$$\lambda_{\text{loss},1} = \sum_{a=1}^{N} \lambda_a \sum_{j,k} \pi(B_1,j,k,a) \ .$$

The loss probability of queue 1 is

$$P_1 = \frac{\lambda_{\text{loss},1}}{\lambda_{\text{effec},1} + \lambda_{\text{loss},1}} \ .$$

The expected waiting time of customers in queue 2 is

$$\mathbb{E}[W_2] = \frac{\mathbb{E}[\text{Queue 2}]}{\lambda_{\text{effec},2}} \ ,$$

where $\lambda_{\text{effec},2}$ is the effective arrival rate at queue 2 and by using conditional PASTA is given by

$$\lambda_{\text{effec},2} = \mu_1 \sum_{\substack{i,j,a:i>0 \\ j<B_2}} \pi(i,j,1,a) + \mu_2 \sum_{\substack{i,j,a:i>0 \\ j<B_2}} \pi(i,j,2,a).$$

Let us denote the loss rate at queue 2 by $\lambda_{loss,2}$:

$$\lambda_{\text{loss},2} = \mu_1 \sum_{\substack{i,a \\ i>0}} \pi(i, B_2, 1, a) + \mu_2 \sum_{\substack{i,a \\ i>0}} \pi(i, B_2, 2, a).$$

The loss probability of queue 2 is

$$P_2 = \frac{\lambda_{\text{loss},2}}{\lambda_{\text{effec},2} + \lambda_{\text{loss},2}}.$$

Finally, the net throughput of the entire system can be expressed in different ways:

$$\begin{aligned} \text{Throughput} &= \lambda_{\text{effec},2} \\ &= \lambda_{\text{effec},1} \times (1 - P_2) \\ &= \nu \times \sum_{\substack{j>0 \\ i,j,k,a}} \pi(i, j, k, a). \end{aligned}$$

5.3 Simulation model and mapping parameters

In the previous sections, we have presented the Markov model of the interaction of the Ethernet hop-by-hop flow control and the TCP end-to-end flow control mechanisms. In many ways this model is a simplification of reality. For TCP, only the main characteristics were included in order to reduce the model complexity. To assess how close the model comes to reality, a simulator containing a network model with 2 nodes was used. Figure 5.3 shows the simulation model. The OMNeT++ simulator (see [2]) was used to model the Ethernet Bridge, which includes the standard Ethernet mechanisms such as MAC address learning and forwarding, and the hop-by-hop backpressure signal. In the simulations, TCP traffic was generated using the NS-2 TCP stack ([1]). The two different simulation environments were combined with LibSynk ([61]) as shown in Figure 5.3.

It is worthwhile to mention the differences between the simulation model and the Markov model.

- The Markov model assumes exponential service times whereas in the simulation these are fixed and deterministic. The time it takes to transmit a packet is computed using the packet size and the link speed. We have used link speeds, which would lead to the same number of packets being processed, as the average service times in the Markov model.
- Another aspect which is not explicitly taken into account by the Markov model is the dependency of the RTT on the maximum (possible) burst size. The RTT governs the maximum window size and thus the maximum rate. However, this

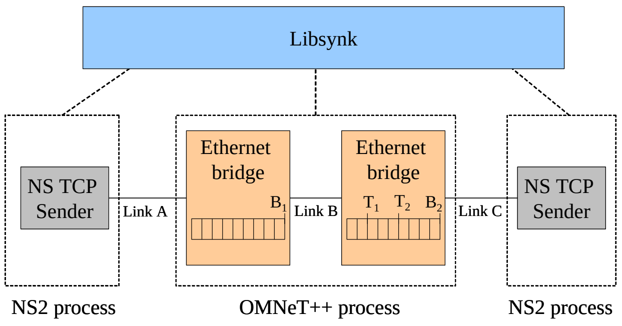


Figure 5.3: Simulation set-up.

is expected to become noticeable only when there are few TCP flows active, which is not the scenario we aim for. When multiple TCP flows are active, this phenomenon is less relevant.

- The Markov model does not model the slow start behavior of TCP flows.
- The input process in the Markov model is a Markov-modulated Poisson process (see [24]). TCP packets sent within a window do not resemble a Poisson process. Still the Markov model captures high and low sending rates and their adaptation to network congestion.

In order to execute and compare the tests with the Markov model and the simulation model we first need to map the parameters from one onto the other. For both models we have used $B_1 = 10$ and $B_2 = 20$ packets. For the relation between the input traffic sending rates i.e., the λ_as and the simulated TCP traffic scenario we fixed the link speed of Link A to λ_1. Note that for the Markov model arrivals occur according to a stochastic point process, so that the actual inflow of packets may exceed the specified rates for some periods of time. We have executed simulations with different scenarios for file downloads and compared the results against the Markov model.

5.4 Results

In this section we present the results from the Markov model and the simulation model. The goal is to obtain a better understanding of the effect of various network and traffic related parameters on the interaction of the hop-by-hop and end-to-end congestion control schemes. This comparison also helps in assessing the validity of the Markov models. We have investigated the influence of three main aspects. The first aspect being the different steps in the input process, which determine the granularity of bursts of the input traffic stream. The second aspect is the RTT, which influences the rate at which the input process recovers from loss; the third aspect is the choice of thresholds. The performance measure used is TCP-level throughput, relating to successfully transmitted packets, often also referred to as goodput. The graphs plot the expected increase in TCP throughput with the use of hop-by-hop flow control as compared to TCP without any additional form of flow control. For all these different aspects we compare the trends observed in the Markov model with simulations for different traffic download scenarios. Having incorporated most of the relevant aspects into the Markov model, the results aim at showing consistent trends, both with the Markov model and the simulations.

5.4.1 Scenarios

Since the main goal of hop-by-hop flow control is to contain temporary congestion, we have looked at scenarios with bursty traffic input. It is important to note that in our model and the simulations we assume finite buffer capacity i.e., $B_1 = 10$ and $B_2 = 20$ packets. For the Markov model we have used either $N = 2$ or $N = 3$. In all cases $\lambda_N = 1$ and different values for λ_1 and, if $N = 3$ for λ_2, are considered. The other parameters are as follows $\mu_1 = 100$, $\mu_2 = 2$ and $\nu = 15$ packets per second. The corresponding simulation scenario considered is with (see Figure 5.3) link speeds of links B and C equal to 100 and 15 packets per second respectively, which translates into 1200 kbps and 180 kbps. The capacity of link A is varied along with λ_1. The TCP Reno version is used for the simulations with packet length as well as MTU equal to 1526 bytes.

5.4.2 Round trip time

The round trip time has an important influence on the performance of TCP. Since the RTT dictates the rate at which acknowledgements are received, it influences the way in which the window and thus the TCP sending rate is increased. Thus, the RTT provides the key to recovery of TCP streams from losses. In this section our goal is to study its influence in conjunction with hop-by-hop flow control. This subsection shows the results for varying RTT. For the Markov model we use $N = 2$, $\lambda_2 = 1$ and varying values for λ_1. Similarly, for the simulation model we vary the capacity of link A in Figure 5.3. The traffic scenario used for the simulations is 8 flows where each flow is sending an 80 KB file with 1 second intervals. The RTT is increased by introducing additional link delay. Figure 5.4 plots the percentage increase in throughput on using hop-by-hop flow control in combination with TCP as compared to TCP alone.

It can be observed from Figure 5.4 that as the RTT increases, the throughput benefit with the use of hop-by-hop flow control also increases. This general trend was also observed with various other simulation scenarios with smaller numbers of TCP flows and varying file sizes. In order to explain this, it is important to note that

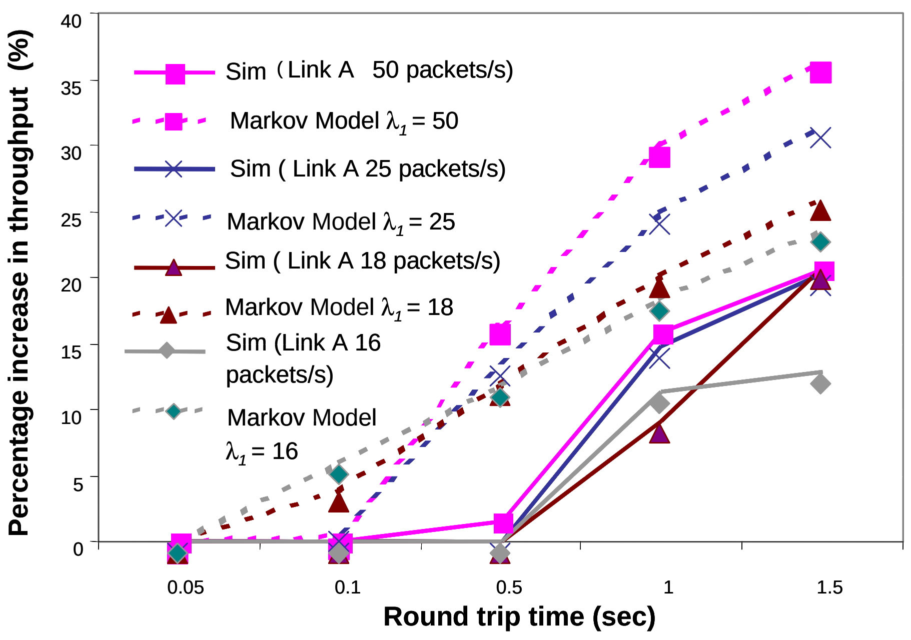


Figure 5.4: Increased throughput with the hop-by-hop flow control and its dependence on round trip time.

with hop-by-hop flow control we use more buffer space, so that there will in general be fewer packets lost. When the RTT is short, TCP recovers quickly from loss and switches back to a faster rate much sooner than when the RTT is long. Thus with a short RTT the congestion level is constantly rather high, ruling out any further optimization. When the RTT is long, the impact of packet loss is greater, since it takes longer to recover from losses. At loss instances, the fact that hop-by-hop flow control causes less packet losses, provides a visible and significant benefit. As there is less loss with the use of hop-by-hop flow control, the TCP sending rate is better adjusted to network resources. With hop-by-hop flow control TCP lowers its sending rate only when the resources (buffer space) are completely exhausted, thus avoiding unnecessary fluctuations in its sending rate.

The Markov model captures the general trend of increase of throughput with increase of RTT. However it seems that the Markov model is more accurate with the trend for higher load λ_1/ν. This can be explained as follows. Since the number of TCP flows and the traffic scenario is kept constant in the simulations, low values of link speeds (λ_1) increase the chance that the link capacity is always utilized to its maximum due to excessive load. For greater link speed values, the same traffic scenario might not provide enough load to constantly send at a rate equal to the link speed irrespective of the amount of packet drops. Variations in the traffic sent on link A with simulations brings it closer to the Poisson assumption in the Markov model, while constant and complete utilization of link A, makes it more deterministic.

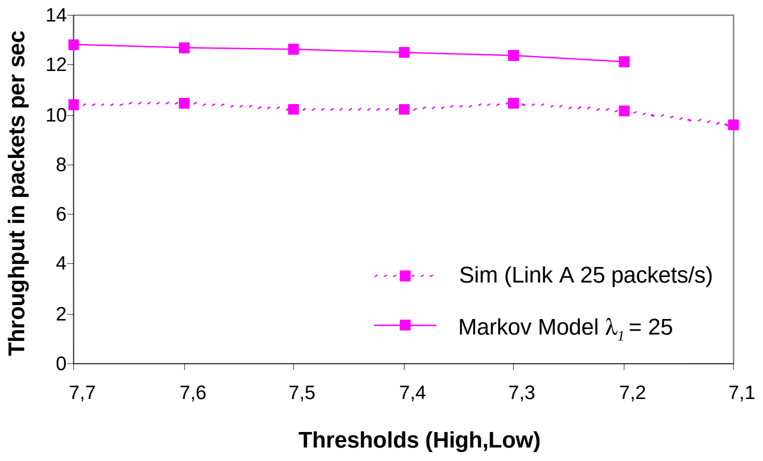


Figure 5.5: Results with varying low threshold values while the high threshold is kept constant.

5.4.3 Thresholds

In this section we study the influence of the thresholds T_1 and T_2. The scenario considered is the same as in the previous section but now with $\lambda_1 = 25$. In Figure 5.5 the high threshold T_1 is kept constant and the lower threshold T_2 is varied. In Figure 5.6 the placement of the threshold is varied while keeping $T_1 = T_2$.

From Figure 5.5 we observe the general trend that the throughput tends to decrease as the low threshold T_2 is placed further away from T_1. At the same time it is important to note that this difference is very small. If one were to consider the overhead of the hop-by-hop flow control messages sent every-time the thresholds were crossed, it would be preferred to have the thresholds placed far apart. Figure 5.6 varies the threshold values from 2 to B_2 keeping $T_1 = T_2$. The results indicate that the thresholds should not be placed too close to B_2 as around this value, the performance degrades significantly. The trends shown in Figures 5.5 and 5.6 were also observed with other values of λ_1. The optimum in our experiments was always about 70-80% of buffer size. However, it should be kept in mind that all the additional tests were done with same values of μ_2 and the same delay on link A.

5.4.4 Granularity of bursts

It was explained in the previous sections that for a given TCP traffic scenario it is difficult to estimate the value of N and the various λ_as for the Markov model. This complication arises from the fact that the aggregate traffic behavior of multiple TCP streams is highly complex, since there might be no consistency in the fluctuation of traffic rates of the various TCP streams. Our goal in this section is to study the

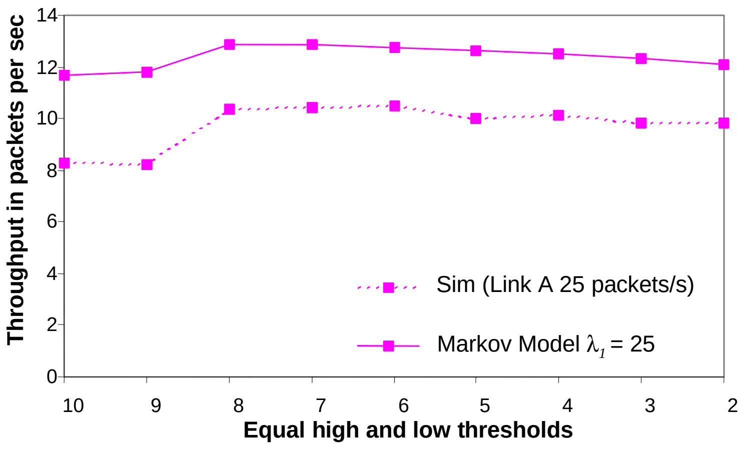


Figure 5.6: Results showing varying choice of thresholds with high threshold kept equal to the low threshold.

influence of the granularity of the variations in TCP sending rate on the performance of hop-by-hop flow control. For the Markov model this implies that we increase N and introduce corresponding intermediate values for λ_a. We have compared the results of the Markov model with different TCP simulation traffic scenarios. We plot the results with $N = 2$ and $N = 3$. It is important to mention that the values and trends observed with higher values of N did not show extreme differences from $N = 3$. We observed that only $N = 2$ provides significantly higher throughput values as compared to other values of N. For the results shown in Figure 5.7, we have always used $\lambda_1 = 50$ and $\lambda_N = 1$. For $N = 3$, we have used $\lambda_2 = 25$. We have observed in additional tests that λ_a being linear or non-linear in a does not have a major impact on the results. The comparison presented with different simulation scenarios also helps understand the reason behind the difference in performance of the traffic scenarios. The simulation scenarios include the previous results with 8 TCP flows sending file sizes of 80 KB every second and an alternative scenario where 8 flows are used to send infinite file sizes, i.e., there is always data to send.

It can be observed from Figure 5.7 that the simulations with idle times show far greater benefit with the use of hop-by-hop flow control as compared to the simulations where there is always infinite data to be sent. The reason behind this difference is not only the level of congestion caused by the two scenarios but also the different burstiness of their traffic characteristics. The scenario with idle times provides hop-by-hop flow control the opportunity to buffer packets from the high rate preceding the idle time, instead of dropping them. During the idle time, the buffered packets can be sent directly rather than waiting for retransmissions. So hop-by-hop flow control manages to smoothen the TCP sending rate. However, with files of infinite sizes there is always data to send, also the level of congestion

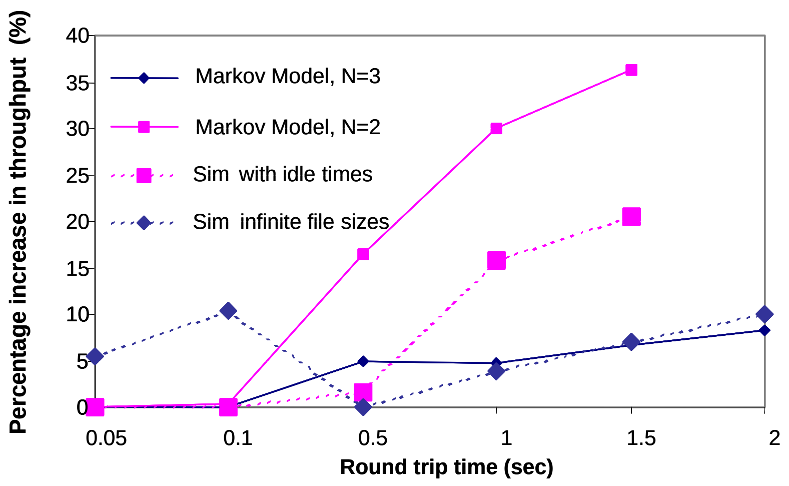


Figure 5.7: Effect of input (TCP) traffic sending rate granularity.

is far too high to be bridged by extra buffering. There will still be drops and it will result in fluctuations in TCP sending rate. Apparently due to the multiple streams the drop in the aggregate sending rate is not that great. This phenomenon can also be better understood by looking at the results from the Markov models. The Markov model with 2 input steps, follows the trends of the simulation scenario with idle times whereas the more granular input traffic steps in the Markov model follows trends of the simulations without any idle times. From these results it is evident that when the TCP streams react in granular steps to loss instead of extreme rates, the extra buffering has limited impact. In other words, when TCP itself smoothens out traffic due to congestion, hop-by-hop flow control has nothing more to add. The simulations and the Markov model do however show some very different results with large file sizes for low RTTs. We have observed higher packet loss in the simulations for RTT=0.1 sec. The reason for this could be that many packets are lost affecting all streams at the same time, providing bursts in the TCP input traffic which the hop-by-hop flow control manages to compensate for with extra buffering.

5.5 Conclusions

In this chapter we have modeled the interaction of the IEEE 802.3x hop-by-hop congestion control mechanism with TCP end-to-end congestion control. We have introduced a Markov model and compared it with simulations of a real TCP stack. The Markov model aims at capturing the interaction of hop-by-hop with TCP congestion control for multiple TCP streams and their aggregate traffic behavior. It does not aim at modeling details of a single TCP flow. The results indicate that the model manages to capture the qualitative performance trends. Only in some specific cases where backpressure is not used, certain TCP effects and packet drop patterns cause an unexpected degradation in throughput performance. With simulation, these cases show an unexpected increase in the benefit of using hop-by-hop flow control.

The model also provides useful insight in the effect of various network and congestion control mechanism parameters on the results. Studying the influence of thresholds for the hop-by-hop congestion control scheme, we have observed that for the scenarios considered, setting the high and low thresholds close to each other is most optimal. The choice of the threshold should be at 70-80% of the buffer size. This percentage should be adjusted if there is significant transmission delay on the link to which the congestion messages are sent. Setting the thresholds far apart from each other did not show significant degradation in performance. Considering the fact that we did not model the overhead in sending the congestion message on the downstream traffic, it is probably advisable to set the thresholds further from each other (with the additional constraint that the low threshold does not coincide with empty buffer).

The influence of the RTT on the performance was also studied. It can be concluded from the results that increasing the RTT provides greater benefit with the use of hop-by-hop congestion control along with TCP. Since the RTT determines the increase in TCP sending rate after a packet loss, longer RTTs will delay TCP's recovery from loss. In these cases hop-by-hop flow control buffers avoid unnecessary packet loss thus reducing unnecessary long reductions in TCP sending rate. We also observed that the greater the load or congestion level, the less the influence of the RTT on the results as well as that of hop-by-hop congestion control. Short RTTs allow TCP to recover very quickly from packet loss and almost always keep sending at a high traffic rate thus causing extreme congestion. It is difficult to solve extreme congestion if the input traffic rate is not adapted. It is also important to note that since the RTT controls the maximum window size, a very large value of RTT will force TCP flows to send at such a low rate that there will be no congestion at all. In these cases again hop-by-hop flow control will not provide any real benefits.

Burstiness and the level of congestion definitely play a role in the performance improvement of any congestion mechanism. We have studied this aspect by looking at results from the Markov model with extreme differences in input traffic rate and with more granular functions and comparing them with 2 distinct simulation scenarios. TCP traffic scenarios with idle times between files show the largest benefit with the use of hop-by-hop congestion control. The Markov model with extreme bursty changes in traffic rate follows closely these trends in the simulations. When the traffic has idle times, the hop-by-hop congestion control helps in smoothing out traffic and avoids drops and unnecessary retransmissions. However, when there are no idle times and there is always traffic to send, TCP end-to-end congestion mostly adapts well to the bottleneck, thus use of hop-by-hop control does not provide significant benefit. There are some exceptions when certain combinations of the

RTT and TCP parameters cause significant drops that can possibly be avoided by using hop-by-hop control. These are subjects for future research.

Chapter 6

A fluid queue model of Ethernet congestion control

In Chapter 5 we modeled the interaction of Ethernet congestion control with TCP using a tandem queue Markov model. In this Chapter we focus in more detail on the effects of the parameter values of the Ethernet congestion control scheme by designing and analysing a fluid queue model. Fluid queues are a frequently used modeling framework in performance analysis of communication systems. These models approximate packet streams by flows of fluid. *Feedback* fluid queues are of special significance when modeling congestion control mechanisms, in particular closed-loop controls in which the input process is affected by the current value of the buffer content. Practical examples are TCP congestion control, random early detection [27], explicit congestion notification [25] and Ethernet congestion control mechanisms (see Chapter 5).

Previous work on feedback fluid queues [49] predominantly focused on queues in which a *single* buffer threshold signals both the onset and end of congestion. In these models, depending on whether the buffer occupancy is below or above the threshold, the traffic source is allowed to transmit at a high peak rate or slowed down to a (lower) guaranteed traffic rate, respectively. The special case at which the threshold lies at 0 was also addressed in, e.g., [15, 64].

These 'single-threshold systems' have serious drawbacks. In the first place, in case the threshold is crossed often, many feedback signals have to be sent to the traffic source, and these may consume a significant part of the available bandwidth. In the second place, it can be argued that it is not optimal that a single threshold performs the function of signaling the beginning and end of congestion: to maximize throughput a full buffer is preferred (thus delaying the signal for the onset of congestion as much as possible), whereas to minimize the buffer delay requires minimizing the buffer occupancy (thus delaying the signal for the end of the congestion phase as much as possible). Therefore, to find a good trade-off between throughput and buffer delay, there is a need for mechanisms with *two* separate thresholds: one to signal the beginning of the congestion phase, and another to signal the end. In this chapter we propose and analyze such a mechanism.

The model addressed in this chapter belongs to the class of (*Markovian*) *fluid models*. The 'classical' fluid model [5, 41, 28] is characterized by a generator matrix Q governing a Markovian background process and a diagonal matrix $R = \mathrm{diag}\{r_1, ..., r_d\}$: if the background process is in state i, traffic is generated at a rate $r_i \geq 0$. It was shown that the steady-state buffer content distribution obeys a system of linear differential equations, and after imposing the proper additional constraints these can be solved. From a methodological standpoint, an important contribution was due to Rogers [69], who succeeded to express the steady-state buffer content distribution in terms of the fundamental Wiener-Hopf factorization. Another key paper is by Ahn and Ramaswami [4], who explicitly exploit relations with quasi-birth-death processes. An alternative method for exact analysis of fluid models using stochastic petri nets is provided in [30]. We also mention that a recent literature overview on Markov fluid queues is given in, e.g., [16].

In the second half of the 1990s models emerged in which the source behavior was influenced by the buffer content; see for instance [3, 15, 49, 64]; in [31, 72], Q and R depend *continuously* on the buffer level. Importantly, in all these feedback fluid models the buffer content uniquely defines the probabilistic properties of the source. The model analyzed in the present chapter departs from this property. In fact the input process has two 'modes'. The first mode applies as long as the upper threshold B_1 has not been reached from below. As soon as that happens, we switch to the second mode, until the lower threshold B_2, smaller than B_1, is hit from above, i.e., the buffer occupancy falls below B_2. In this way the threshold B_1 is used to signal the onset of congestion, and B_2 to signal the end of congestion. Important novelty of our model as compared to earlier feedback models is that the input process may behave in two different ways between the two thresholds, depending on the past.

In this chapter our goal is to find the steady-state buffer content distribution and associated performance measures of the model with two thresholds, as was described above. The methodology used involves the derivation of Kolmogorov equations and balance equations, which result in a solution in terms of a spectral expansion; then additional conditions are identified that solve for the unknowns in the solution. Although this method is in principle similar to the approach in, e.g., [49], our model poses new challenges, due to its specific features (i.e., the two thresholds, the two modes of the input process). In the analysis particularly the behavior at the thresholds should be handled with care. As a result, the derivation of the conditions to solve for the unknowns turns out to be substantially more difficult than in the model of [49].

We mentioned above that the model we investigate can be useful in detecting congestion and can be generically applied to any congestion control mechanism for packet networks such as [27], [25] and [48]. Here, we explain how this kind of feedback control has special significance with respect to congestion control in

Ethernet metropolitan networks. The backpressure scheme defined in IEEE 802.3x [76], is intended to provide flow control on a hop-by-hop basis by allowing ports to *turn off* their upstream link neighbors for a period of time. For a full-duplex connection, this mechanism is based on a special frame called *pause frame* in which the pause period is specified. The end-station (or router) receiving the pause frame looks at the pause period, and does not transmit or attempt transmission for that amount of time. Alternatively, an ON/OFF pause message can be sent signaling the beginning and end of the transmission pause phase. This congestion control method is usually implemented by using a high and a low threshold in the (congested) queue. When the queue occupancy exceeds the high threshold the *PauseOn* message is sent and when the queue occupancy drops below the low threshold the *PauseOff* message is sent and consequently transmission is resumed. Previous works [48], [47] and [56] on Ethernet congestion control have concentrated on the throughput gain which can be achieved by using the scheme. We are not aware of any literature with an analytically tractable model of the mechanism. The model presented in this chapter can be used to optimally configure the high and low thresholds and decide on when to initiate the transmission pause phase and when to end it, consequently, addressing an essential design criterion for the Ethernet congestion avoidance scheme.

The rest of the chapter is organized as follows. In Section 6.1 we describe the model with two thresholds in detail. In Section 6.2 we analyze the model and derive the equilibrium distribution of the buffer content. This section consists of three parts. Section 6.2.1 gives the balance equations for the buffer occupancy. Section 6.2.2 uses the spectral expansion method to provide the solution to the buffer occupancy in a compact form, which involves several unknown coefficients. In Section 6.2.3 we derive as many constraints as there are unknowns, so that these coefficients can be identified. In Section 6.3 we demonstrate our analysis by considering a numerical example and graphically present the buffer content distribution; we remark that a full numerical assessment of the backpressure mechanism, relying on the methods developed in this chapter, is found in [44]. We do include here, however, numerical evidence for the claim that the two-threshold mechanism leads to a reduction of the signaling overhead. Finally in Section 6.4 we conclude the chapter with a summary of our results, and a discussion on future work.

6.1 Model

In this section we provide the formal definition of the model. We consider a fluid queue with an infinite buffer and constant output rate c. Let $W(t)$ be the content of the queue at time t, which is a stochastic process due to the probabilistic way in which fluid enters the queue. A popular model for such an input process is a so-called *Markov fluid source.* This model prescribes that the rate at which fluid enters the queue per unit time depends on the current state of some background irreducible

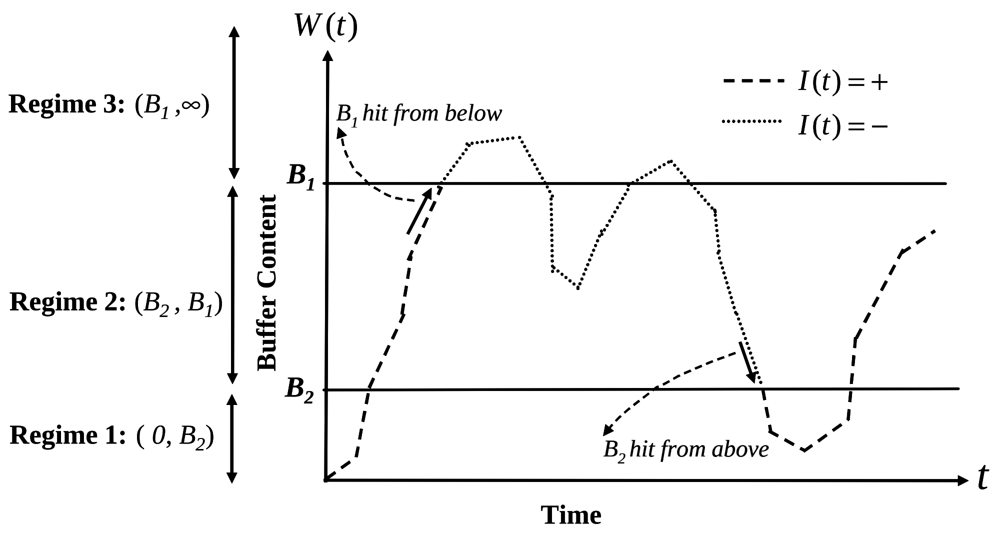


Figure 6.1: Different regimes for the buffer content $W(t)$.

continuous-time Markov chain $X(t)$, defined on a finite state-space $\{1, \ldots, d\}$, for $d \in \mathbb{N}$. At times when $X(t) = i$, the current input rate is $r_i \geq 0$. When we let the corresponding generator matrix be $Q \equiv (q_{ij})_{i,j=1}^{d}$, with $\sum_{j=1}^{d} q_{ij} = 0$ and $q_{ij} \geq 0$ for $i \neq j$, and define the d-dimensional traffic rate vector $\boldsymbol{r} \equiv (r_1, \ldots, r_d)^{\mathrm{T}}$, we call this input process a Markov-fluid source with parameters Q and $\boldsymbol{r}$.

In our feedback fluid model, the input stream alternates between two modes. In one mode the input process behaves like a Markov fluid source with generator Q^{+} (dimension $d \times d$) and traffic rate vector $\boldsymbol{r}^{+}$ (dimension d). Similarly, in the other mode it behaves like a Markov fluid source with generator Q^{-} (also dimension $d \times d$) and traffic rate vector $\boldsymbol{r}^{-}$ (also dimension d).

We introduce the indicator variable process $I(\cdot)$, taking values in $\{`+', `-'\}$, which gives the current mode of operation of the input source. It is important to note that whenever $I(t)$ switches from one mode to another, the background process $X(t)$ stays in the same state; only its dynamics will from that time onwards behave according to the other generator matrix. However, the rate at which the fluid buffer receives fluid *does* change instantaneously from r_i^{+} to r_i^{-} (or vice versa), when the background process $X(t)$ is in state i at the switching instant. Which of the two modes is currently valid at some time t depends on the behavior of the content process $W(t)$ relative to two thresholds, an upper threshold B_1 and a lower threshold B_2. The first mode ('+') applies as long as $W(t)$ has not reached the upper threshold B_1 from below. As soon as that happens, $I(t)$ switches to the other mode ('−'), until $W(t)$ hits the lower threshold B_2 from above, etc.

Let us describe this in some more detail, see also Figure 6.1. Suppose $W(0) = 0$, i.e., the process starts with an empty buffer, and let the indicator process $I(t)$ start in '+', where it will stay as long as the process $W(t)$ has not reached B_1 from below. During this period the input process behaves as a Markov-fluid source with d-dimensional generator Q^+ and traffic rate vector $\boldsymbol{r}^+$, and the buffer is drained at a rate c. At some point the buffer content $W(t)$ reaches the upper threshold B_1. Suppose that $X(t)$ is then in state j. Then $I(t)$ switches to '$-$', while the background process $X(t)$ stays in state j. From then on the input process behaves as a Markov-fluid source with generator Q^- and traffic rate vector $\boldsymbol{r}^-$, while the buffer is still drained at rate c. In particular, the current flow rate will change from r_j^+, which is larger than c, to r_j^-, which may or may not be larger than c. Further on at some moment the buffer content $W(t)$ drops to the lower threshold B_2. Suppose that $X(t)$ is in state k at this moment, then $I(t)$ switches to '+' while the background process stays in state k, and the input rate changes from $r_k^- < c$ to r_k^+.

Thus, the process continues, and will converge to an equilibrium distribution, assuming the queue is stable. With π^- denoting the equilibrium distribution corresponding to Q^-, i.e., $\pi^- Q^- = 0$ and $\sum_{i=1}^d \pi_i^- = 1$, the equilibrium condition is

$$\sum_{i=1}^{d} \pi_i^- r_i^- < c;$$

throughout this chapter we assume that this condition is satisfied. For technical reasons, we will also assume that for all states $i = 1, \dots, d$ we have $r_i^+ \neq c$ and $r_i^- \neq c$, so that the content of the queue is never constant over time (unless it is zero). Let W be the steady-state buffer content, i.e., a random variable distributed as $\lim_{t\to\infty} W(t)$ and define I and X similarly. Define also, for $i = 1, \dots, d$ and $x \geq 0$,

$$F_i^-(x) := \mathbb{P}(I = -, X = i, W \leq x), \quad F_i^+(x) := \mathbb{P}(I = +, X = i, W \leq x).$$

Our goal in this chapter is to identify $F_i^-(x)$ and $F_i^+(x)$. Having solved for these distribution functions, we can calculate two important performance measures for the system, namely the throughput ϑ and the distribution of the delay D, as follows:

$$\vartheta = \sum_{i=1}^{d} \left(r_i^+ F_i^+(\infty) + r_i^- F_i^-(\infty) \right), \tag{6.1}$$

and for $d \geq 0$,

$$\mathbb{P}(D \leq d) = \left(\sum_{i=1}^{d} \left(r_i^+ F_i^+(dc) + r_i^- F_i^-(dc) \right) \right) \Bigg/ \left(\sum_{i=1}^{d} \left(r_i^+ F_i^+(\infty) + r_i^- F_i^-(\infty) \right) \right).$$

We define some additional notation which will be helpful in considering the various cases while solving for $F_i^-(x)$ and $F_i^+(x)$. We define the sets of 'up-states' and 'down-states' for both modes, and their cardinalities, as follows:

$$\begin{aligned} S_{\rm D}^- &:= \{i : r_i^- < c\} \quad \text{and} \quad d_{\rm D}^- := \#\{S_{\rm D}^-\};\\ S_{\rm U}^- &:= \{i : r_i^- > c\} \quad \text{and} \quad d_{\rm U}^- := \#\{S_{\rm U}^-\};\\ S_{\rm D}^+ &:= \{i : r_i^+ < c\} \quad \text{and} \quad d_{\rm D}^+ := \#\{S_{\rm D}^+\};\\ S_{\rm U}^+ &:= \{i : r_i^+ > c\} \quad \text{and} \quad d_{\rm U}^+ := \#\{S_{\rm U}^+\}. \end{aligned}$$

The subscript D is a mnemonic for 'Down', referring to the buffer being drained, while U stands for 'Up', referring to the buffer filling up. Evidently, $d_{\rm D}^- + d_{\rm U}^- = d_{\rm D}^+ + d_{\rm U}^+ = d$.

6.2 Analysis

In this section we analyze the buffer content distribution, by presenting a procedure to compute $F_i^-(x)$ and $F_i^+(x)$, for $i = 1, \ldots, d$. From the model description in the previous section we know that $F_i^-(x)$ and $F_i^+(x)$ have different characteristics in different intervals of the buffer content. Therefore, we define Regimes 1, 2, and 3, as shown in Figure 6.1. For each of these regimes we analyze $F_i^-(x)$ and $F_i^+(x)$. Two cases are rather straightforward, and therefore we start with these.

3^+: $F_i^+(x)$ *in Regime* 3. When the buffer occupancy reaches B_1 the indicator switches to the '$-$' state (if it was not in '$-$' already). The indicator changes to '$+$', only after the buffer occupancy would drop below B_2, where $B_2 < B_1$. Therefore, $F_i^+(x)$ is constant in the interval $[B_1, \infty)$. For $i = 1, ..., d$ and $x \geq B_1$,

$$F_i^+(x) = F_i^+(B_1). \tag{6.2}$$

1^-: $F_i^-(x)$ *in Regime* 1. If the buffer occupancy drops below B_2, then the indicator switches to '$+$'. Therefore, the fluid source with the '$-$' indicator can never be active below B_2 but only in the interval $[B_2, \infty)$. We have, for all $i = 1, ..., d$ and $x \leq B_2$,

$$F_i^-(x) = 0. \tag{6.3}$$

Nevertheless, it is important to note that even though $F_i^-(B_2) = 0$ the density at B_2, i.e.,

$$f_i{}^-(B_2) = \left.\frac{{\rm d}F_i^-(x)}{{\rm d}x}\right|_{x \downarrow B_2},$$

might not be equal to zero.

The other cases 1^+, 2^+, 2^- and 3^- are analyzed in the following subsections. We follow an approach similar to that in [5, 49] to find the complete solution to $F_i^-(x)$ and $F_i^+(x)$. We derive the balance equations for both $F_i^-(x)$ and $F_i^+(x)$ in Section

6.2.1, by first considering the Kolmogorov forward equations. What makes the Kolmogorov equations especially complicated in our case are the transitions around the thresholds B_1 and B_2. Assuming that a transition takes place somewhere in a small time interval, the exact time of the transition is unknown, and as a consequence so is the indicator of the generator matrix in that interval. Section 6.2.1 deals with these issues. In Section 6.2.2 the spectral expansion method is used to find the solution to the differential equations. These can be written down in a rather simple form, but involve an extensive set of unknown coefficients. In Section 6.2.3 we find as many conditions as the number of unknowns, so that the stationary distribution of the buffer content can be determined.

6.2.1 Derivation of the balance equations

We found above simple solutions for $F_i^+(x)$ in Regime 3 and $F_i^-(x)$ in Regime 1, given by Eqns. (6.2) and (6.3). Our goal in this subsection is to derive the Kolmogorov forward equations for the other cases, from which we then easily obtain the corresponding balance equations. We slightly abuse notation by also using F_i^- and F_i^+ for the time-dependent distribution functions, i.e., we define $F_i^-(t,x) := \mathbb{P}(I(t) = -, X(t) = i, W(t) \le x)$ and $F_i^+(t,x)$ analogously.

1^+: *$F_i^+(x)$ in Regime* 1. Regime 1 refers to $0 < x < B_2$, where we have for small h

$$F_i^+(t+h,x) = \left(1 - h\sum_{j\neq i} q_{i,j}^+\right) F_i^+\left(t, x - h(r_i^+ - c)\right) + h\sum_{j\neq i} q_{j,i}^+ F_j^+(t,x) + o(h).$$

Rearranging, dividing by h, and using the fact that the rows of the Q^+ matrix add up to zero, we obtain

$$\begin{aligned}\frac{F_i^+(t+h,x) - F_i^+(t, x - h(r_i^+ - c))}{h} &= q_{i,i}^+ F_i^+(t, x - h(r_i^+ - c)) \\ &\quad + \sum_{j\neq i} q_{j,i}^+ F_j^+(t,x) + \frac{o(h)}{h}.\end{aligned}$$

By taking $h \downarrow 0$, we find

$$\frac{\partial}{\partial t} F_i^+(t,x) + (r_i^+ - c)\frac{\partial}{\partial x} F_i^+(t,x) = \sum_j q_{j,i}^+ F_j^+(t,x).$$

(Remark that, formally, these partial derivatives are not necessarily well-defined. As we are interested in the stationary behavior of the queue, this fact does not play a role – in fact we can assume the queue content has a proper density at time 0.) Assuming stationarity we set $F_i^+(t,x) = F_i^+(x)$

and in addition we set all derivatives with respect to t equal to 0. We thus obtain

$$(r_i^+ - c)\frac{\mathrm{d}}{\mathrm{d}x}F_i^+(x) = \sum_j q_{j,i}^+ F_j^+(x). \tag{6.4}$$

2^+: *$F_i^+(x)$ in Regime* 2. We now consider the interval $B_2 < x < B_1$. For i in S_U^-, we simply have the same equations as in Regime 1, leading to Eqn. (6.4). However, when i in S_D^-, we have to include the possibility that a transition can occur from the '$-$' state into the '$+$' state. This will happen when the buffer content at time t is between B_2 and $B_2 - h(r_i^- - c)$ (which is just above B_2 due to $i \in S_\mathrm{D}^-$), and the background process does not change its state during $(t, t+h]$. A complication here is that before the transition, Q^- is active and after the transition Q^+ is active. However, we do know that the probability that $X(t)$ remains in i during during $(t, t+h]$ is $1 + o(1)$, no matter what the precise form is, and this knowledge is sufficient. We thus find, for i in S_D^-,

$$\begin{aligned} F_i^+(t+h, x) = &\left(1 - h\sum_{j\neq i} q_{i,j}^+\right) F_i^+\left(t, x - h(r_i^+ - c)\right) \\ &+h\sum_{j\neq i} q_{j,i}^+ F_j^+(t,x) \\ &+ (1+o(1))\left(F_i^-(t, B_2 - h(r_i^- - c)) - F_i^-(t, B_2)\right) + o(h). \end{aligned} \tag{6.5}$$

By rearranging and dividing by h on both sides we obtain

$$\begin{aligned} \frac{F_i^+(t+h,x) - F_i^+(t, x - h(r_i^+ - c))}{h} &= q_{i,i}^+ F_i^+(t, x - h(r_i^+ - c)) \\ + \sum_{j\neq i} q_{j,i}^+ F_j^+(t,x) + (1+o(1))\, &\frac{(F_i^-(t, B_2 - h(r_i^- - c)) - F_i^-(t, B_2)}{h} + \frac{o(h)}{h}. \end{aligned}$$

Then taking $h \downarrow 0$, and assuming stationarity we find

$$(r_i^+ - c)\frac{\mathrm{d}}{\mathrm{d}x}F_i^+(x) = \sum_j q_{j,i}^+ F_j^+(x) - (r_i^- - c)\left.\frac{\mathrm{d}}{\mathrm{d}x}F_i^-(x)\right|_{x\downarrow B_2}. \tag{6.6}$$

Since for i in S_U^-, we already found the same equations as in Regime 1, leading to Eqn. (6.4), we can combine the two cases to find, for $i = 1, \ldots, d$,

$$(r_i^+ - c)\frac{\mathrm{d}}{\mathrm{d}x}F_i^+(x) = \sum_j q_{j,i}^+ F_j^+(x) - A_i^-, \tag{6.7}$$

where

$$A_i^- = \begin{cases} (r_i^- - c)\left.\dfrac{\mathrm{d}}{\mathrm{d}x}F_i^-(x)\right|_{x\downarrow B_2} & \text{for } i \text{ in } S_\mathrm{D}^-; \\ 0 & \text{for } i \text{ in } S_\mathrm{U}^-. \end{cases} \tag{6.8}$$

2^-: *$F_i^-(x)$ in Regime* 2. In Regime 2, i.e., $B_2 < x < B_1$, for i in S_U^-, we have the simple case

$$F_i^-(t+h,x) = \left(1 - h\sum_{j\neq i} q_{i,j}^-\right) F_i^-(t, x-h(r_i^- - c)) + h\sum_{j\neq i} q_{j,i}^- F_j^-(t,x) + o(h).$$

By rearranging, dividing by h, taking $h \downarrow 0$ and assuming stationarity we obtain

$$(r_i^- - c)\frac{\mathrm{d}}{\mathrm{d}x}F_i^-(x) = \sum_j q_{j,i}^- F_j^-(x). \tag{6.9}$$

For $i \in S_D^-$, the equation is more complicated. If we consider a time interval of h time units, then in this interval the buffer occupancy will drop by $|h(r_i^- - c)|$. We have to make sure that it does not drop to or below B_2. If this occurs then the indicator switches to the '+' state, which is the probability we want to subtract from the equations (as it was already taken into account in (6.6)). Therefore, we include the term $-F_i^-(t, B_2 - h(r_i^- - c))$ which ensures that after h time units the buffer occupancy cannot drop to or below B_2. We thus obtain

$$\begin{aligned} F_i^-(t+h,x) = \left(1 - h\sum_{j\neq i} q_{i,j}^-\right) & (F_i^-(t, x-h(r_i^- - c)) \\ & -F_i^-(t, B_2 - h(r_i^- - c))) \\ & +h\sum_{j\neq i} q_{j,i}^- F_j^-(t,x) + o(h). \end{aligned}$$

By rearranging, taking $h \downarrow 0$ and assuming stationarity we get

$$(r_i^- - c)\frac{\mathrm{d}}{\mathrm{d}x}F_i^-(t,x) = \sum_j q_{j,i}^- F_j^-(t,x) + (r_i^- - c)\left.\frac{\mathrm{d}}{\mathrm{d}x}F_i^-(t,x)\right|_{x\downarrow B_2}. \tag{6.10}$$

We can now combine Eqns. (6.9) and (6.10) into one equation for i as

$$(r_i^- - c)\frac{\mathrm{d}}{\mathrm{d}x}F_i^-(x) = \sum_j q_{j,i}^- F_j^-(x) + A_i^- \tag{6.11}$$

where A_i^- is given by Eqn. (6.8).

3^-: *$F_i^-(x)$ in Regime* 3. The equations for $B_1 < x < \infty$, are the most complicated. This is because we have to take into account two aspects. Firstly, we have to exclude the possibility of a transition from the '$-$' into the '+' state when $X(t)$ is in S_D^- and the buffer content is just above B_2 at time t. It is clear from the explanation for $F_i^-(x)$ in Regime 2 that this is done by including a term $-F_i^-(t, B_2 - h(r_i^- - c))$. Secondly, we have to include the possibility of a transition from the '+' state into the '−' state. If at time t, the buffer content is somewhere in the interval $[B_1, B_1 - h(r_i^+ - c)]$, and the background state is in S_{U}^+, then in another h time units, the buffer content will increase and will definitely reach B_1 and jump into the '−' state. We therefore add a term $(1+o(1))\big(F_i^+(t,B_1) - F_i^+(t,B_1 - h(r_i^+ - c))\big)$, similar to the term $(1+o(1))\big(F_i^-(t,B_2 - h(r_i^- - c)) - F_i^-(t,B_2)\big)$ we added in the equation for F_i^+ in Regime 2. We find, for i in $S_{\mathrm{D}}^- \cap S_{\mathrm{U}}^+$,

$$\begin{aligned} F_i^-(t+h,x) = &\left(1 - h\sum_{j\neq i} q_{i,j}^-\right)\left(F_i^-(t, x - h(r_i^- - c))\right.\\ &\left.-F_i^-(t, B_2 - h(r_i^- - c))\right)\\ &+ (1+o(1))\big(F_i^+(t,B_1) - F_i^+(t,B_1 - h(r_i^+ - c))\big) \\ &+ h\sum_{j\neq i} q_{j,i}^- F_j^-(t,x) + o(h). \end{aligned} \tag{6.12}$$

By rearranging, dividing by h, taking $h \downarrow 0$, and assuming stationarity, we obtain

$$\begin{aligned} (r_i^- - c)\frac{\mathrm{d}}{\mathrm{d}x}F_i^-(x) = \sum_j q_{j,i}^- F_j^-(x) + (r_i^- - c)\left.\frac{\mathrm{d}}{\mathrm{d}x}F_i^-(x)\right|_{x\downarrow B_2} \\ +(r_i^+ - c)\left.\frac{\mathrm{d}}{\mathrm{d}x}F_i^+(x)\right|_{x\uparrow B_1} \end{aligned} \tag{6.13}$$

In order to derive the equations for the other values of i we should note that in Eqn. (6.12) the term $F_i^-(t, B_2 - h(r_i^- - c))$ appears for all $i \in S_{\mathrm{D}}^-$ and the term $(1+o(1))\big(F_i^+(t,B_1) - F_i^+(t,B_1 - h(r_i^+ - c))\big)$ for $i \in S_{\mathrm{U}}^+$. Further on in (6.13) the term $F_i^-(t, B_2 - h(r_i^- - c))$ results in

$$(r_i^- - c)\left.\frac{\mathrm{d}}{\mathrm{d}x}F_i^-(x)\right|_{x\downarrow B_2},$$

whereas $(1+o(1))\big(F_i^+(t,B_1) - F_i^+(t,B_1 - h(r_i^+ - c))\big)$ leads to

$$(r_i^+ - c)\left.\frac{\mathrm{d}}{\mathrm{d}x}F_i^+(x)\right|_{x\uparrow B_1}.$$

Therefore, we obtain, for any $i = 1, \ldots, d$,

$$(r_i^- - c)\frac{\mathrm{d}}{\mathrm{d}x}F_i^-(x) = \sum_j q_{j,i}^- F_j^-(x) + A_i^- + A_i^+, \tag{6.14}$$

where A_i^- is given by Eqn. (6.8) and A_i^+ is

$$A_i^+ := \begin{cases} (r_i^+ - c)\left.\dfrac{\mathrm{d}}{\mathrm{d}x}F_i^+(x)\right|_{x\uparrow B_1} & \text{for } : i \in S_{\mathrm{U}}^+; \\ 0 & \text{for } : i \in S_{\mathrm{D}}^+. \end{cases} \tag{6.15}$$

Table 6.1: *Overview of balance equations for* $F^+(x)$

Regime	Interval	$\boldsymbol{F}^+(x)$
1	$(0, B_2)$	$\frac{\mathrm{d}}{\mathrm{d}x}\boldsymbol{F}^+(x)(R^+ - C) = \boldsymbol{F}^+(x)Q^+$
2	(B_2, B_1)	$\frac{\mathrm{d}}{\mathrm{d}x}\boldsymbol{F}^+(x)(R^+ - C) = \boldsymbol{F}^+(x)Q^+ - \boldsymbol{A}^-$
3	(B_1, ∞)	$\boldsymbol{F}^+(B_1)$

Table 6.2: *Overview of balance equations for* $F^-(x)$

Regime	Interval	$\boldsymbol{F}^-(x)$
1	$(0, B_2)$	0
2	(B_2, B_1)	$\frac{\mathrm{d}}{\mathrm{d}x}\boldsymbol{F}^-(x)(R^- - C) = \boldsymbol{F}^-(x)Q^- + \boldsymbol{A}^-$
3	(B_1, ∞)	$\frac{\mathrm{d}}{\mathrm{d}x}\boldsymbol{F}^-(x)(R^- - C) = \boldsymbol{F}^-(x)Q^- + \boldsymbol{A}^- + \boldsymbol{A}^+$

In order to get an overview of the balance equations in the different regimes we have summarized the results so far in matrix form in Tables 6.1 and 6.2, where $\boldsymbol{F}^+(x) \equiv (F_1^+(x), ..., F_d^+(x))$; the row vectors $\boldsymbol{F}^-(x)$, $\boldsymbol{A}^-$ and $\boldsymbol{A}^+$ are defined similarly. R^+ is the diagonal matrix $\text{diag}\{r_1^+, ..., r_d^+\}$ and R^- the diagonal matrix $\text{diag}\{r_1^-, ..., r_d^-\}$. We also introduce $C := cI_d$, where I_d is the identity matrix of dimension d.

6.2.2 Solution to the balance equations

In the previous subsection we have derived the balance equations for both $\boldsymbol{F}^-(x)$ and $\boldsymbol{F}^+(x)$. In this subsection we provide the solutions to these equations, using the spectral expansion method used in [5] and [49]. The solution can then be presented in a simple form, but it involves a number of unknown coefficients.

1^+ : The balance equation for $\boldsymbol{F}^+(x)$ in Regime 1 is $\mathrm{d}\boldsymbol{F}^+(x)/\mathrm{d}x \cdot (R^+ - C) = \boldsymbol{F}^+(x)Q^+$. The spectral expansion of the solution to this equation is given by

$$\boldsymbol{F}^+(x) = \sum_{j=1}^{d} a_j^{1+} \boldsymbol{v}_j^+ \exp[z_j^+ x]$$

where $(z_j^+, \boldsymbol{v}_j^+)$ is an eigenvalue-eigenvector pair satisfying $z_j^+ \boldsymbol{v}_j^+ (R^+ - C) = \boldsymbol{v}_j^+ Q^+$, and the a_j^{1+} are coefficients.

In the above solution we tacitly assumed that the matrix $Q^+(R^+ - C)^{-1}$ has full eigenspace, in that all eigenvalues are simple (i.e., have multiplicity 1). Two remarks are in place here. (A) In the first place, we mention that it is known that if the Q^+ matrix has a specific structure, the eigenvalues are indeed simple (and real); most notably, as shown in [80], if Q^+ corresponds to a birth-death Markov process this property indeed applies. (B) Secondly, eigenvalues with multiplicity k larger than 1 do not lead to any conceptual problems. Standard theory on linear differential equations entails that then the density of the stationary queue content has terms proportional to $x^j \exp(-z_j^+ x)$, with $j = 0, \dots, k-1$. We decided to assume in our analysis that the eigenvalues of $Q^+(R^+ - C)^{-1}$ (and later also those of $Q^-(R^- - C)^{-1}$) are simple as the corresponding formulas for the 'non-simple case' do not add much extra insight, and are notationally cumbersome.

2^+ : We now consider $\boldsymbol{F}^+(x)$ in the interval $B_2 < x < B_1$ for which the balance equation is

$$\frac{\mathrm{d}}{\mathrm{d}x}\boldsymbol{F}^+(x)(R^+ - C) = \boldsymbol{F}^+(x)Q^+ - \boldsymbol{A}^-.$$

This equation has an inhomogeneous term because of which we cannot write the solution as for $\boldsymbol{F}^+(x)$ in Regime 1. Since A_i^- is a constant (see Eqn. (6.8)), differentiation of the above equation with respect to x gives us a homogeneous equation in $\boldsymbol{f}^+(x) \equiv \mathrm{d}\boldsymbol{F}^+(x)/\mathrm{d}x$ as below

$$\frac{\mathrm{d}}{\mathrm{d}x}\boldsymbol{f}^+(x)(R^+ - C) = \boldsymbol{f}^+(x)Q^+.$$

Now that we have a homogeneous equation its solution is given as

$$\boldsymbol{f}^+(x) = \sum_{j=1}^{d} \tilde{a}_j^{2+} \boldsymbol{v}_j^+ \exp[z_j^+ x]$$

where $(z_j^+, \boldsymbol{v}_j^+)$ is the same eigenvalue-eigenvector pair as before and $\tilde{a}_j^{2+}$ are coefficients. As Q^+ is the generator, it has eigenvalue 0, and hence one of the eigenvalues z_j^+ is zero, say $z_{j^*}^+ = 0$. With this in mind, integration immediately yields that

$$\boldsymbol{F}^+(x) = \sum_{j=1, j\neq j^*}^{d} a_j^{2+} \boldsymbol{v}_j^+ \exp[z_j^+ x] + a_{j^*}^{2+} \boldsymbol{v}_{j^*}^+ x + \boldsymbol{w}^{2+}$$

where $a_j^{2+} = \tilde{a}_j^{2+}/z_j^+$ for $j \neq j^*, a_{j^*}^{2+} = \tilde{a}_{j^*}^{2+}$, and the components w_i^{2+} of $\boldsymbol{w}^{2+}$ are integration constants.

2^- : For $B_2 < x < B_1$, the balance equation for $\boldsymbol{F}^-(x)$ is $\mathrm{d}\boldsymbol{F}^-(x)/\mathrm{d}x \cdot (R^- - C) = \boldsymbol{F}^-(x)Q^- + \boldsymbol{A}^-$. This is again a inhomogeneous equation and the spectral method cannot be used directly. Therefore, we follow the same procedure as for $\boldsymbol{F}^+(x)$ in Regime 2. We differentiate the equation on both sides to get a homogeneous equation in $\boldsymbol{f}^-(x)$, which we integrate to obtain

$$\boldsymbol{F}^-(x) = \sum_{j=1, j\neq j^*}^{d} a_j^{2-} \boldsymbol{v}_j^- \exp[z_j^- x] + a_{j^*}^{2-} \boldsymbol{v}_{j^*}^- x + \boldsymbol{w}^{2-}$$

Table 6.3: *Overview of solution for $F^+(x)$*

Regime	Interval	$\boldsymbol{F}^+(x)$
1	$(0, B_2)$	$\sum_{j=1}^{d} a_j^{1+} \boldsymbol{v}_j^+ \exp[z_j^+ x]$
2	(B_2, B_1)	$\sum_{j=1, j\neq j^\star}^{d} a_j^{2+} \boldsymbol{v}_j^+ \exp[z_j^+ x] + a_{j^\star}^{2+} \boldsymbol{v}_{j^\star}^+ x + \boldsymbol{w}^{2+}$
3	(B_1, ∞)	$F^+(B_1)$

where $(z_j^-, \boldsymbol{v}_j^-)$ is an eigenvalue-eigenvector pair satisfying $z_j^- \boldsymbol{v}_j^- (R^- - C) = \boldsymbol{v}_j^- Q^-$, and the a_j^{2-} are coefficients. The components w_i^{2-} of $\boldsymbol{w}^{2-}$, are integration constants and the coefficient $a_{j^\star}^{2-}$ corresponds to the eigenvalue $z_{j^\star}^- = 0$ of Q^-.

3^- : The balance equation for $\boldsymbol{F}^-(x)$ in Regime 3 is $\mathrm{d}\boldsymbol{F}^-(x)/\mathrm{d}x \cdot (R^- - C) = \boldsymbol{F}^-(x)Q^- + \boldsymbol{A}^- + \boldsymbol{A}^+$. In this equation we have two inhomogeneous terms as opposed to one in the previous cases. Nevertheless, we can still apply the same method as for $\boldsymbol{F}^+(x)$ in Regime 2 and $\boldsymbol{F}^-(x)$ in Regime 2. This is because both the inhomogeneous terms in the equation above consist of constant elements $\lim_{x\downarrow B_2} \mathrm{d}F_i^-(x)/\mathrm{d}x$ and $\lim_{x\uparrow B_1} \mathrm{d}F_i^+(x)/\mathrm{d}x$ or zero (see Eqns. (6.8) and (6.15)) which disappear after differentiating with respect to x. Therefore, after differentiation and then integration we get the solution for $\boldsymbol{F}^-(x)$ in Regime 3 as

$$\boldsymbol{F}^-(x) = \sum_{j=1, j\neq j^\star}^{d} a_j^{3-} \boldsymbol{v}_j^- \exp[z_j^- x] + a_{j^\star}^{3-} \boldsymbol{v}_{j^\star}^- x + \boldsymbol{w}^{3-}$$

where $(z_j^-, \boldsymbol{v}_j^-)$ is again the eigenvalue-eigenvector pair that satisfies $z_j^- \boldsymbol{v}_j^- (R^- - C) = \boldsymbol{v}_j^- Q^-$, and the a_j^{3-} are coefficients. The components w_i^{3-} of $\boldsymbol{w}^{3-}$ are integration constants and the coefficient $a_{j^\star}^{3-}$ corresponds to $z_{j^\star}^- = 0$.

We summarize the solutions found in the various intervals in Tables 6.3 and 6.4.

6.2.3 Derivation of conditions for finding the unknowns in the solution

In Section 6.2.2, we provided the solution to $\boldsymbol{F}^-(x)$ and $\boldsymbol{F}^+(x)$ using the spectral expansion method. However, the solution includes many unknowns which need to be found with additional conditions. Tables 6.3 and 6.4 presents an overview of the solution where the vectors $\boldsymbol{a}^{1+}$, $\boldsymbol{a}^{2+}$, $\boldsymbol{a}^{2-}$, $\boldsymbol{a}^{3-}$, $\boldsymbol{w}^{2+}$, $\boldsymbol{w}^{2-}$ and $\boldsymbol{w}^{3-}$ are unknown.

Table 6.4: *Overview of solution for $F^{-}(x)$*

Regime	Interval	$\boldsymbol{F}^{-}(x)$
1	$(0, B_2)$	0
2	(B_2, B_1)	$\sum_{j=1, j\neq j^\star}^{d} a_j^{2-} \boldsymbol{v}_j^{-} \exp[z_j^{-} x] + a_{j^\star}^{2-} \boldsymbol{v}_{j^\star}^{-} x + \boldsymbol{w}^{2-}$
3	(B_1, ∞)	$\sum_{j=1, j\neq j^\star}^{d} a_j^{3-} \boldsymbol{v}_j^{-} \exp[z_j^{-} x] + a_{j^\star}^{3-} \boldsymbol{v}_{j^\star}^{-} x + \boldsymbol{w}^{3-}$

Table 6.5: *Overview of the unknowns in the different regimes*

Regime	Interval	$\boldsymbol{F}^{+}(x)$	$\boldsymbol{F}^{-}(x)$
1	$(0, B_2)$	$a_j^{1+} \quad j = 1, ..., d$	0
2	(B_2, B_1)	$a_j^{2+}, w_j^{2+} \quad j = 1, ..., d$	$a_j^{2-}, w_j^{2-} \quad j = 1, ..., d$
3	(B_1, ∞)	$F_j^{+}(B_1)$, for $j = 1, .., d$	$a_j^{3-}, w_j^{3-} \quad j = 1, ..., d$
Total number of unknowns: $8d$			

Table 6.5 enumerates all unknowns giving a total of $8d$. In this section our goal is to find $8d$ conditions so as to solve the system.

A. *Boundary conditions* at $x = 0$ and $x = \infty$ are as in [5] and [41]:

- $F_i^{+}(0) = 0$, for i in S_{U}^{+}. This is because it cannot be that simultaneously the buffer is empty and the background process is in an up-state. This gives us d_{U}^{+} conditions.
- For $x \to \infty$, $F_i^{-}(x)$ should remain bounded, and therefore for all z_j^{-} with a non-negative real part, the corresponding a_j^{3-} has to be zero. Notice that this also entails that the equilibrium distribution of $W^{-}(t)$ is given by $\boldsymbol{w}^{3-}$. This gives us d_{D}^{-} conditions.

B. *Continuity conditions.* $F_i^{+}(x)$ and $F_i^{-}(x)$ are both continuous at the thresholds B_1 and B_2. This gives us the following $4d$ equations:

- $\lim_{x\uparrow B_2} F_i^{+}(x) = \lim_{x\downarrow B_2} F_i^{+}(x)$.
- $\lim_{x\uparrow B_1} F_i^{+}(x) = F_i^{+}(B_1)$.
- $\lim_{x\downarrow B_2} F_i^{-}(x) = 0$.
- $\lim_{x\downarrow B_1} F_i^{-}(x) = \lim_{x\downarrow B_1} F_i^{-}(x)$.

C. *Substitution conditions.*
As in [49] we need to substitute the solutions given in Section 6.2.2 into the

inhomogeneous balance equations of Section 6.2.1. We have 3 inhomogeneous systems. Potentially each of these can lead to d conditions. Therefore, in total we would get $3d$ equations from the substitution.

Boundary conditions, continuity conditions and substitution conditions together were sufficient to solve the model in [49], but as we have a more complicated model in which the fluid source alternates between two modes, this is not the case here. Therefore, we introduce and prove the following:

D. *Additional conditions.*
In the first place, suppose that the buffer is filling up in the '+' state. At some point it will reach B_1 and then switch to the '−' process. Since there is no density beyond B_1 (and the phase being '+') it is highly unlikely that the buffer level is just below B_1 while the background process is in a down-state. Similarly, when the buffer is filling up ('−' phase) it is unlikely that the buffer content is just above B_2 while the background process is in an up-state.

The lemma below states these additional conditions more precisely, and is proved by deriving the balance equations at $x = B_1$ and $x = B_2$. Note that these were not addressed in Section 6.2.1.

Lemma 1 (i) For all $i \in S_{\mathrm{D}}^{+}$,

$$\frac{\mathrm{d}}{\mathrm{d}x} F_i^{+}(x)\bigg|_{x \uparrow B_1} = 0.$$

(ii) For all $i \in S_{\mathrm{U}}^{-}$

$$\frac{\mathrm{d}}{\mathrm{d}x} F_i^{-}(x)\bigg|_{x \downarrow B_2} = 0.$$

Proof: (i) We first consider the case for i in $S_{\mathrm{D}}^{-} \cap S_{\mathrm{U}}^{+}$. In this case we have

$$\begin{aligned} F_i^{+}(t+h, B_1) \;=\;& (1 - h\sum_{j\neq i} q_{i,j}^{+}) F_i^{+}(t, B_1 - h(r_i^{+} - c)) + h\sum_{j\neq i} q_{j,i}^{+} F_j^{+}(t, B_1) \\ & + (1 + o(1))\big(F_i^{-}(t, B_2 - h(r_i^{-} - c)) - F_i^{-}(t, B_2)\big) + o(h). \end{aligned} \tag{6.16}$$

By rearranging, dividing by h, taking $h \downarrow 0$ and assuming stationarity we obtain for i in $S_{\mathrm{D}}^{-} \cap S_{\mathrm{U}}^{+}$,

$$(r_i^{+} - c)\,\frac{\mathrm{d}}{\mathrm{d}x} F_i^{+}(x)\bigg|_{x\uparrow B_1} = \sum_j q_{j,i}^{+}(F_j^{+}(B_1)) - (r_i^{-} - c)\,\frac{\mathrm{d}}{\mathrm{d}x} F_i^{-}(x)\bigg|_{x\downarrow B_2}.$$

Comparison with (6.8) and (6.15) shows that

$$\sum_j q^+_{j,i} F^+_j(B_1) = A^+_i + A^-_i, \tag{6.17}$$

and it is in fact easy to see that this holds for any i. Let us compare this to Eqn. (6.7) for $F^+_j(x)$ in the interval (B_2, B_1), letting $x \uparrow B_1$, which gives

$$\sum_j q^+_{j,i} F^+_j(B_1) = (r^+_i - c) \frac{\mathrm{d}}{\mathrm{d}x} F^+_i(x) \bigg|_{x \uparrow B_1} + A^-_i. \tag{6.18}$$

From this comparison we conclude that for i in S^+_{D}

$$(r^+_i - c) \frac{\mathrm{d}}{\mathrm{d}x} F^+_i(x) \bigg|_{x \uparrow B_1} = A^+_i = 0. \tag{6.19}$$

from which the first claim immediately follows.

(ii) The proof of this part is similar, comparing the two equations for $F^-_i(x)$ at B_2, which are $F^-_i(B_2) = 0$ and

$$(r^-_i - c) \frac{\mathrm{d}}{\mathrm{d}x} F^-_i(x) \bigg|_{x \downarrow B_2} = \sum_j q^+_{j,i} F^-_j(B_2) + A^-_i. \qquad \square$$

The following lemma entails that the number of substitution conditions is really only $2d$ (not $3d$).

Lemma 2 The equations from the substitution condition for $\boldsymbol{F}^-(x)$ in the interval (B_2, B_1) are implied by the continuity condition for $\boldsymbol{F}^-(x)$ at B_2.

Proof: We start with the substitution for $F^-_j(x)$ in the interval (B_2, B_1). The balance equation in this case is, see (6.11),

$$(r^-_i - c) \frac{\mathrm{d}}{\mathrm{d}x} F^-_i(x) = \sum_k q^-_{k,i} F^-_k(x) + A^-_i.$$

On substituting the solution back into the equation and using the fact that $z^-_{j^\star} = 0$ and $\sum_k q^-_{k,i} v^-_{j,k} = z^-_j v^-_{j,i}(r^-_i - c)$, where $v^-_{j,i}$ refers to the ith component of vector $\boldsymbol{v}^-_j$, we find

$$(r^-_i - c)(a^{2-}_{j^\star} v^-_{j^\star,i}) = \sum_j q^-_{k,i} w^{2-}_k + A^-_i. \tag{6.20}$$

By the definition of A^-_i and using the first part of Lemma 1 we can simply substitute

$$A^-_i = (r^-_i - c) \frac{\mathrm{d}}{\mathrm{d}x} F^-_i(x) \bigg|_{x \downarrow B_2} = (r^-_i - c) \left(\sum_{j=1, j \neq j^\star}^{d} a^{2-}_j v^-_{j,i} z^-_j \exp[z^-_j B_2] + a^{2-}_{j^\star} v^-_{j^\star,i} \right)$$

for all $i = 1, ..., d$ in the equation above to give us

$$\sum_k q^-_{k,i} w^{2-}_k + (r^-_i - c)\left(\sum_{j=1, j\neq j^\star}^{d} a^{2-}_j v^-_{j,i} z^-_j \exp[z^-_j B_2]\right) = 0, \qquad \text{or,}$$

$$\sum_k q^-_{k,i} w^{2-}_k + \sum_k q^-_{k,i}\left(\sum_{j=1, j\neq j^\star}^{d} a^{2-}_j v^-_{j,k} \exp[z^-_j B_2]\right) = 0$$

as the substitution conditions should hold. However, since we may freely add $\sum_k q^-_{k,i} a^{2-}_{j^\star} v^-_{j^\star,k} B_2$ to the left-hand side of the above (since it is zero), and then combine the sums into one, the conditions turn out to be equivalent to $\sum_k q^-_{k,i} F^-_k(B_2) = 0$, Hence they are indeed implied by the continuity equations at B_2, which say that $F^-_k(B_2) = 0$ for all k. □

Let us now explain the intuition behind Lemma 2. If we look at Tables 6.1 and 6.2, we could suspect that an overlap could arise for $F^-_i(x)$ in Regime 2. This is because this is the only inhomogeneous equation which involves a single indicator state (being the '$-$' state). The continuity equations would also have the same characteristics. As for the other inhomogeneous equations, these involve terms with both the '$+$' and the '$-$' states, whereas the continuity equations still involve terms with a single indicator state. Hence, for these cases there is a clear difference between the characteristics of the substitution and the continuity equations, and the substitution equations do give additional information.

Let us now count the number of conditions we have at our disposal. The boundary conditions give us d^+_{U} and d^-_{D} equations and the four continuity conditions give us another $4d$ conditions. The substitution conditions could have potentially given us another $3d$ conditions. Adding these to the $d^+_{\mathrm{D}} + d^-_{\mathrm{U}}$ conditions from Lemma 1, we would together have $9d$ conditions, d more than we need. With Lemma 2 we proved that an overlap of d conditions exists between the continuity and the substitution conditions, eventually adding up to exactly $8d$ conditions equal to the total number of unknowns in the solution (see Table 6.5), which we need to solve the system.

However, it is important to note that all the $8d$ equations are linear in the different unknowns enlisted in Table 6.5, with a rank of $8d - 1$. This is easy to see since the linear system can be solved upto a multiplicative constant. The redundancy in the equations can be removed by replacing any one of the equations in the linear system by the normalization equation

$$\sum_{i=1}^{d} \left(F^+_i(\infty) + F^-_i(\infty)\right) = 1. \tag{6.21}$$

Thus, we arrive at the following result.

Proposition 3 We have $8d$ conditions on the coefficients, matching the number of unknowns.

In the next section we illustrate, by means of a numerical example, how one can identify the $8d$ unknowns.

6.3 Numerical example

In this section we provide numerical results aimed at demonstrating the computation of the stationary distribution of the buffer content and the other performance measures. We consider a two state numerical example, i.e., with $d = 2$. We consider the following generator matrices and rate vectors,

$$Q^+ = \begin{pmatrix} -1 & 1 \\ 2 & -2 \end{pmatrix}, \quad Q^- = \begin{pmatrix} -0.8 & 0.8 \\ 5 & -5 \end{pmatrix}, \quad \boldsymbol{r}^- = \begin{pmatrix} 16 \\ 0 \end{pmatrix}, \quad \boldsymbol{r}^+ = \begin{pmatrix} 25 \\ 0 \end{pmatrix}.$$

The other parameters are $c = 15$, $B_1 = 15$ and $B_2 = 10$. The diagonal matrices R^-, R^+, and C then equal $\mathrm{diag}\{16, 0\}$, $\mathrm{diag}\{25, 0\}$, and $\mathrm{diag}\{15, 15\}$, respectively. After the numerical determination of the eigensystems of the matrices $Q^+(R^+ - C)^{-1}$ and $Q^-(R^- - C)^{-1}$ we apply the conditions as listed in Section 6.2.3. We then solve the resulting linear system of equations for the $8d = 16$ unknowns. This gives us the complete and unique solution for the stationary distribution of the buffer content, $\boldsymbol{F}^+(x)$ and $\boldsymbol{F}^-(x)$. The graphical representation of $F_i^+(x)$ and $F_i^-(x)$ for $i = 1, 2$ is shown in Figure 6.2.

The total throughput of the system can be calculated from Eqn. (6.1) as

$$\vartheta = r_1^+ F_1^+(\infty) + r_1^- F_1^-(\infty) = 14.1924.$$

We can compute the expected buffer content by first computing $F(x) = \sum_{i,j} F_i^j(x)$, then computing the combined probability density as $f(x) = \mathrm{d}F(x)/\mathrm{d}x$. The expected buffer content is then given by

$$\mathbb{E}D = \int_0^\infty x f(x) \mathrm{d}x = 12.2040.$$

In a second experiment we consider the effect of having two thresholds on the signaling overhead. The expected number of phase-transitions per unit time equals

$$\sum_{i \in S_{\mathrm{U}}^+} f_i^+(B_1)(r_i^+ - c) + \sum_{i \in S_{\mathrm{D}}^-} f_i^-(B_2)(c - r_i^-);$$

this (plausible) formula is derived in [44]. The effect of varying B_2, for a given value of B_1, is shown in Figure 6.3. The other parameters are as above. We observe that indeed the signaling frequency is reduced by choosing B_2 much smaller than B_1.

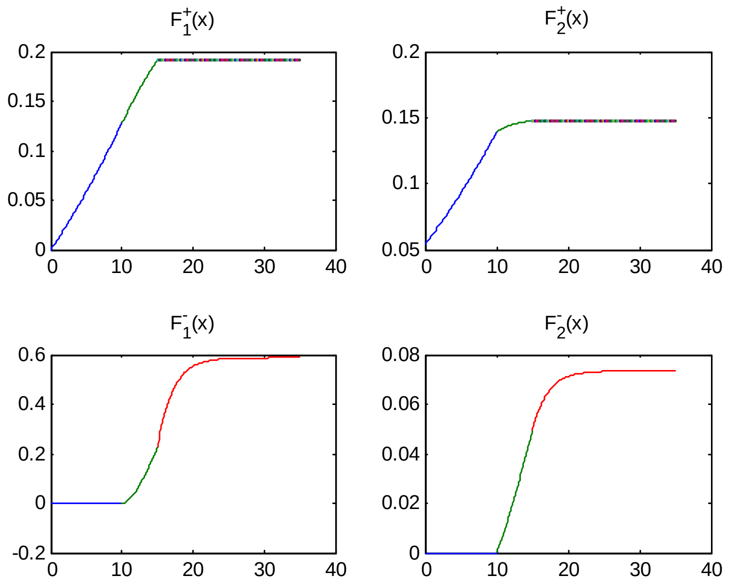


Figure 6.2: Probability distribution functions of the buffer content with $d = 2$, $B_1 = 15$, $B_2 = 10$.

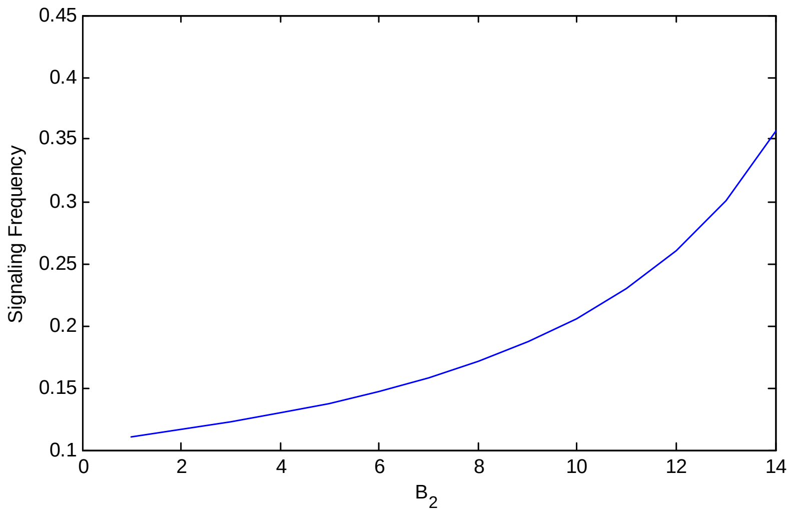


Figure 6.3: Signaling frequency as B_2 grows to $B_1 = 15$.

6.4 Concluding remarks

We have analyzed a feedback fluid queueing system in which the traffic source rates adapt to congestion. The system has two thresholds: the higher threshold B_1 aims at signaling the beginning of a congestion period, whereas the lower threshold B_2 serves to signal the end of the congestion period. This idea is modeled by letting the input process alternate between two Markov fluid processes: the first applies as long as the upper threshold B_1 has not been hit from below. As soon as that happens, the traffic source switches to the other process, until B_2 is hit from above.

The resulting model falls in the class of Markovian feedback fluid queues. The numerical complexity of the methodology used to solve for the buffer content distribution and throughput boils down to solving two d-dimensional eigensystems, as well as a (fairly standard) linear system of $8d$ equations; here d denotes the cardinality of the state space on which the two Markov processes are defined.

A central design problem which can be investigated using our model, and is addressed in [44], is the optimization of the threshold positions while considering the trade-off between throughput and delay. Challenging extensions to the present analysis are the systematic assessment of the signaling frequency in the model (that is, what is the rate at which the input process alternates between the two Markov fluid processes), as a function of the model parameters; this is a relevant issue, as the signaling overhead needs to be controlled. Yet another important direction for future research is to incorporate delay in the reception of feedback signals and in the adaptation of the source traffic rate.

Chapter 7

Design issues of Ethernet congestion control

In the previous chapter we presented a fluid queue model for Ethernet backpressure based congestion control and provided its solution. We mentioned that the model presented (and solved) there could be relied on when configuring the high and low thresholds, thus addressing a pivotal design criterion for the Ethernet congestion avoidance scheme. We also remarked in Chapter 6 that the backpressure scheme has the attractive property that the signaling overhead (in terms of the number of pause messages sent per unit time) is lower than when using just one threshold (that detects both start and end of congestion periods), but we did not systematically quantify this effect. Also, the reduction of signaling overhead may be at the expense of a loss in throughput, or degraded performance in terms of delay.

The primary goal of the current chapter is to demonstrate the effect of the thresholds, and to obtain insight into the trade-offs mentioned above. In order to do so, we also derive analytic formulas for the performance metrics of interest in terms of the model parameters, as well as the parameters agreed upon in the service level agreement. Numerical experiments are performed to evaluate the main trade-offs of this model (for instance the trade-off between the signaling frequency and the throughput). These can be used to generate design guidelines. We also provide an elementary, yet powerful, Markovian model that can be used as an approximate model in situations of large traffic aggregates feeding into the system; the trade-offs and guidelines identified for the feedback fluid model turn out to carry over to this more stylized model.

The organization of this chapter is as follows. Section 7.1 describes our fluid model, specializing to the situation of just one source feeding into the queue. It also recapitulates the main results from Chapter 6. Then Section 7.2 presents derivations of the main performance metrics considered in this chapter: the throughput, the mean packet delay, signaling frequency, and the mean transmission time of a burst of packets. Here we note that packet delays are of crucial interest for *streaming* applications; these generate traffic with an 'intrinsic duration and rate (which is generally variable) whose time integrity must be preserved by the network' [67] — think of telephony, streaming video and audio. On the other hand, the transmission time, to be thought of as the time it takes for bursts of packets ('jobs') to go through a node, is a main performance metric for *elastic* applications, such as E-mail, file transfer, but also pictures or video sequences transferred for local storage before viewing. Section 7.3 presents the numerical experiments that demonstrate how to evaluate the trade-offs mentioned above, and presents a number of general guidelines. We also include in Section 7.5 a model and corresponding numerical experiments that indicate that the main findings carry over to the situation in which there are a substantial number of concurrent users. Section 7.6 concludes this chapter.

7.1 Model and preliminaries

In this section we describe the model of which we analyze a number of key performance metrics in Section 7.2, and which we numerically assess in Section 7.3. To this end, we first define generator matrices Q^+ and Q^- on the state space $\{1,2\}$:

$$Q^+ = \begin{pmatrix} -p_1 & p_1 \\ p_2 & -p_2 \end{pmatrix}; \qquad Q^- = \begin{pmatrix} -m_1 & m_1 \\ m_2 & -m_2 \end{pmatrix}.$$

Also, we introduce traffic rate vectors $\boldsymbol{r}^+ = (r_p, 0)^{\mathrm{T}}$ and $\boldsymbol{r}^- = (r_m, 0)^{\mathrm{T}}$, with $r_p > c$ and $r_m > c$; these should be thought of as rates at which traffic is generated, in that traffic flows into the system at rate r_i^ℓ if a background process $X^\ell(\cdot)$, governed by generator matrix Q^ℓ, is in state i (with $\ell \in \{+,-\}$, and $i \in \{1,2\}$). In other words, we identify the on-state with state 1 ('burst'), and the off-state with state 2 ('silence'). The capacity of the buffer is assumed to be infinite (a similar analysis can be done for the finite-buffer case, though).

In this chapter we consider the model of Chapter 6, featuring the special case that the dimension of the underlying sources is 2. In this feedback fluid model, the input stream alternates between two 'modes' (also referred to as 'phases'). In one mode the input process behaves like a Markov fluid source with generator Q^+ and traffic rate vector $\boldsymbol{r}^+$: when the background process is in state $i \in \{1,2\}$ at time t, traffic is generated at a constant rate r_i^+, whereas the queue is drained at a constant rate c. Similarly, in the other mode it behaves like a Markov fluid source with generator Q^- and traffic rate vector $\boldsymbol{r}^-$.

The queueing process alternates between the two above-mentioned modes as follows. We first introduce the indicator variable process $I(\cdot)$, taking values in $\{+,-\}$, which gives the current mode of operation of the input source. It is important to note that whenever $I(t)$ switches from one mode to another, the background process $X(t)$ stays in the same state; only its dynamics will from that time onwards behave according to the other generator matrix. However, the rate at which the fluid buffer receives fluid *does* change instantaneously from r_i^+ to r_i^- (or vice versa), when the background process $X(t)$ is in state i at the switching instant. Which of the two modes is currently valid at some time t depends on the behavior of the content process $W(t)$ relative to two thresholds, an upper threshold B_1 and a lower threshold B_2. The first mode ('+') applies as long as $W(t)$ has not reached the upper threshold B_1 from below. As soon as that happens, $I(t)$ switches to the other mode ('$-$'), until $W(t)$ hits the lower threshold B_2 from above, etc. The queueing dynamics are illustrated by Fig. 6.1.

It is not hard to verify that the equilibrium condition of this model is

$$\frac{m_2}{m_1+m_2}\cdot r_m < c,$$

i.e., in the '$-$'-phase there should be a negative drift. We let

$$F_i^\ell(x) := \mathbb{P}(I=\ell, X=i, W\leq x),$$

with $x\geq 0$, $i\in\{1,2\}$, and $\ell\in\{+,-\}$, be the steady-state distribution of the workload W, jointly with the state of the background process $X\in\{1,2\}$, and the phase $I\in\{+,-\}$. In Chapter 6 we presented an algorithm to compute $F_i^\ell(\cdot)$, as follows. Let z^ℓ ($\ell\in\{+,-\}$) be the non-zero eigenvalue of the matrix $Q^\ell(R^\ell-cI)^{-1}$. It is easily verified that

$$z^+ = \frac{p_2}{c}-\frac{p_1}{r_p-c}, \qquad \text{and} \qquad z^- = \frac{m_2}{c}-\frac{m_1}{r_m-c}.$$

Notice that $z^-<0$ because of the stability condition. Then the analysis in Chapter 6 entails that 3 regimes should be distinguished, cf. Fig. 6.1. More precisely, there are constants $\gamma_{j,i}^\ell$, $\delta_{j,i}^\ell$, $\varepsilon_{j,i}^\ell$ (with regime $j\in\{1,2,3\}$, state $i\in\{1,2\}$, and mode $\ell\in\{-,+\}$), such that

$$\begin{aligned}
F_i^-(x) &= 0, \qquad x\leq B_2;\\
F_i^-(x) &= \gamma_{2,i}^- + \delta_{2,i}^- e^{z^-x} + \varepsilon_{2,i}^- x, \qquad B_2 < x \leq B_1;\\
F_i^-(x) &= \gamma_{3,i}^- + \delta_{3,i}^- e^{z^-x}, \qquad x > B_1;
\end{aligned}$$

also

$$\begin{aligned}
F_i^+(x) &= \gamma_{1,i}^+ + \delta_{1,i}^+ e^{z^+x}, \qquad x\leq B_2;\\
F_i^+(x) &= \gamma_{2,i}^+ + \delta_{2,i}^+ e^{z^+x} + \varepsilon_{2,i}^+ x, \qquad B_2 < x \leq B_1;\\
F_i^+(x) &= F_i^+(B_1), \qquad x > B_1.
\end{aligned}$$

In Chapter 6 a procedure is detailed that enables us to compute these 10 constants, by introducing 10 linear constraints to be imposed on the parameters.

7.2 Performance metrics

In this section we derive (or recall) formulas for a number of performance metrics.

Throughput. We have the evident formula, already given in Chapter 6,

$$\vartheta = r_p \cdot F_1^+(\infty) + r_m \cdot F_1^-(\infty).$$

Alternatively, it is clear that the throughput can be written as (realize that $F_1^+(0) = 0$)

$$\vartheta = c \cdot \mathbb{P}(W > 0) = c\left(1 - F_2^+(0)\right). \tag{7.1}$$

Packet delays. The delay D is defined as the delay experienced by an arbitrary packet (in our model an infinitesimally small 'fluid particle'), and is hence a 'traffic-average'. This performance metric is particularly relevant for streaming traffic, as argued in the introduction, due to its inherent time-integrity requirements. The distribution of D was given in Chapter 6:

$$\mathbb{P}(D \le t) = \frac{r_p F_1^+(tc) + r_m F_1^-(tc)}{r_p F_1^+(\infty) + r_m F_1^-(\infty)};$$

note that the denominator can be interpreted as the average amount of fluid that arrives per unit of time, whereas the numerator is the fraction thereof that corresponds to a delay smaller than t. The mean delay can be computed as

$$\mathbb{E}D = \int_0^\infty \mathbb{P}(D > t)\mathrm{d}t = \int_0^\infty \left(1 - \frac{r_p F_1^+(tc) + r_m F_1^-(tc)}{r_p F_1^+(\infty) + r_m F_1^-(\infty)}\right)\mathrm{d}t.$$

Signaling frequency. The signaling frequency is defined as the expected number of phase transitions per unit time, and is a measure for the signaling overhead. With $f_i^\ell(x) := \mathrm{d}F_i^\ell(x)/\mathrm{d}x$, we first observe that the expected number of upcrossings per unit time through level x is, reasoning as in, e.g., [10, 72],

$$f_1^+(x) \cdot (r_p - c) + f_1^-(x) \cdot (r_m - c); \tag{7.2}$$

here the first (second) term reflects the number of upcrossings while in the '+'-phase ('−'-phase). Likewise the expected number of downcrossings per unit time is given by

$$f_2^+(x+) \cdot c + f_2^-(x+) \cdot c. \tag{7.3}$$

As an aside we mention that, as argued in [10, 72], expressions (7.2) and (7.3) should match, since for any level the mean number of upcrossings per unit time equals the mean number of downcrossings per unit time.

Relying on the above reasoning it is now directly seen that the expected number of phase-transitions per unit time equals

$$\varphi := f_1^+(B_1)\cdot(r_p-c)+f_2^-(B_2)\cdot c = 2f_1^+(B_1)\cdot(r_p-c) = 2f_2^-(B_2)\cdot c;$$

here the $f_1^+(B_1)\cdot(r_p-c)$ term corresponds to the number of upcrossings per unit of time through B_1 while in (to be understood as 'coming from') the '+'-phase, and the $f_2^-(B_2)\cdot c$ term to the number of downcrossings per unit of time through B_2 while in (i.e., coming from) the '$-$'-phase. It is further noted the last two equalities are due to the fact that the number of upcrossings per unit time through B_1 while in the '+'-mode should match the number of downcrossings per unit time through B_2 while in the '$-$'-mode.

Transmission and sojourn time. The next performance metric, T, is the transmission time of a burst, i.e., the time it takes to put the entire burst into the buffer. Let $f_T(\cdot)$ be the density of T. Consider the event $\{T = x\}$. We list three useful properties:

- A first observation is that if $x > B_1/(r_p-c)$, the system must have been in the '$-$'-phase during at least part of the transmission time (as the buffer content grows at rate $r_p - c$ while in the '+'-phase).
- A second observation is the following. Suppose the elastic job enters the system when there is y in the buffer. If x is larger than $(B_1-y)/(r_p-c)$ and the phase is '+', then the phase shifts from '+' to '$-$' during the transmission time.
- A third observation is that if the phase is '$-$' upon arrival, the phase remains '$-$' during the entire transmission time.

It leads to the following expression, with $f^\ell(\cdot)$ the density of the buffer content *seen by an arriving job*, intersected with being in the ℓ-phase, $\ell\in\{+,-\}$:

$$\begin{aligned} f_T(x) = &\int_0^{\max\{B_1-(r_p-c)x,0\}} p_1 e^{-p_1 x} f^+(y)\mathrm{d}y \\ &+\int_{\max\{B_1-(r_p-c)x,0\}}^{B_1} \exp\left(-p_1\cdot\frac{B_1-y}{r_p-c}\right)\cdot m_1\exp\left(-m_1\left(x-\frac{B_1-y}{r_p-c}\right)\right)f^+(y)\mathrm{d}y \\ &+\int_{B_2}^{\infty} m_1 e^{-m_1 x} f^-(y)\mathrm{d}y; \end{aligned} \tag{7.4}$$

the first term corresponds to the situation in which the queue was in the '+'-phase at the arrival epoch of the burst, and remains in the '+'-phase during the transmission time, whereas in the second term the queue makes a transition to the '$-$'-phase during the transmission time; in the third term the queue was in the '$-$'-phase at the arrival epoch of the burst, and remains (automatically) in the '$-$'-phase during the transmission time. From the density, the mean transmission time $\mathbb{E}T$ can be computed.

It now remains to identify $f^+(y)$ and $f^-(y)$. As a burst enters while the source is in the off-state, i.e., $X = 2$, and taking into account the different rates at which the source can transmit when switching on,

$$f^+(y) = \frac{f_2^+(y)p_2}{\int_0^\infty (f_2^+(x)p_2 + f_2^-(x)m_2)\mathrm{d}x}; \qquad f^-(y) = \frac{f_2^-(y)m_2}{\int_0^\infty (f_2^+(x)p_2 + f_2^-(x)m_2)\mathrm{d}x}.$$

The numerator of $f^+(y)$ is to be interpreted as the rate at which the source turns on while the phase is '+' and the buffer is y, whereas the denominator is the rate at which the source turns on, irrespective of the phase and buffer content; the expression for $f^-(y)$ can be interpreted likewise.

We now see how the formulas change when we do not consider the time it takes before the burst is stored in the buffer, but instead the time before the entire burst has left the queue, which we will refer to as the sojourn time S. This random variable is most easily expressed in terms of its Laplace transform. We have to distinguish between the same three cases as in (7.4). Regarding the first term, observe that if the initial buffer level is y and the on-time is x, the entire burst has left the queue after

$$x + \frac{y + (r_p - c)x}{c} = \frac{y}{c} + \frac{r_p x}{c}$$

units of time. Regarding the second term, the amount of traffic in the buffer at the end of the transmission time is

$$B_1 + (r_m - c)\left(x - \frac{B_1 - y}{r_p - c}\right),$$

and hence the sojourn time is

$$x + \frac{1}{c}\left(B_1 + (r_m - c)\left(x - \frac{B_1 - y}{r_p - c}\right)\right) = \frac{r_m x}{c} + \frac{r_p - r_m}{r_p - c}\frac{B_1}{c} + \frac{r_m - c}{r_p - c}\frac{y}{c}.$$

Regarding the third term, then the sojourn time is

$$x + \frac{y + (r_m - c)x}{c} = \frac{y}{c} + \frac{r_m x}{c}.$$

We thus obtain

$$\begin{aligned}
\mathbb{E}^{-\alpha S} &= \int_0^\infty \int_0^{\max\{B_1 - (r_p - c)x, 0\}} p_1 e^{-p_1 x} f^+(y) \exp\left(-\alpha\left(\frac{y}{c} + \frac{r_p x}{c}\right)\right) \mathrm{d}y\mathrm{d}x \\
&+ \int_0^\infty \int_{\max\{B_1 - (r_p - c)x, 0\}}^{B_1} \exp\left(-p_1 \cdot \frac{B_1 - y}{r_p - c}\right) \cdot m_1 \exp\left(-m_1\left(x - \frac{B_1 - y}{r_p - c}\right)\right) f^+(y) \\
&\qquad \exp\left(-\alpha\left(\frac{r_m x}{c} + \frac{r_p - r_m}{r_p - c}\frac{B_1}{c} + \frac{r_m - c}{r_p - c}\frac{y}{c}\right)\right) \mathrm{d}y\mathrm{d}x \\
&+ \int_0^\infty \int_{B_2}^\infty m_1 e^{-m_1 x} f^-(y) \exp\left(-\alpha\left(\frac{y}{c} + \frac{r_m x}{c}\right)\right) \mathrm{d}y\mathrm{d}x.
\end{aligned}$$

By differentiating, inserting $\alpha := 0$, and multiplying with -1, we obtain $\mathbb{E}S$. The formulas do not provide much additional insight, and we have decided to omit them here.

The transmission time and sojourn time are specifically meaningful in the case of elastic traffic. Then we let the size of the elastic job (in, say, bits) be exponentially distributed with mean μ^{-1}, and choose $p_1 = \mu r_p$ and $m_1 = \mu r_m$. In this situation, the amount of traffic to be sent has a fixed distribution (viz. exponentially with mean μ^{-1}). The mean sojourn time reads

$$\mathbb{E}S = \frac{1}{c}\int_0^\infty y(f^+(y) + f^-(y))\mathrm{d}y + \frac{1}{\mu c},$$

where the first term represents the mean amount of time needed to serve all traffic the tagged job sees in the queue upon arrival, and the second term the time needed to serve the tagged job itself.

Multi-dimensional sources. The above results can be extended to sources with dimension higher than 2 (and hence also to the situation of multiple sources), as the model of Chapter 6 presents the steady-state distribution for any dimension of the underlying Markov fluid source; in fact, the formulas for the throughput and the (packet-)delay distribution were already given in Chapter 6. The formula for the signaling frequency follows along the same lines as sketched above, by an upcrossings/downcrossings argument, where all states should be taken into account in which B_1 can be reached from below while being in the '+'-phase, as well as all states in which B_2 can be reached from above while being in the '$-$'-phase.

7.3 Numerical experiments

In this section we describe a number of experiments, that assess the impact of the model parameters on the performance. Four key metrics are considered, viz. (i) throughput, (ii) signaling frequency, (iii) expected (packet) delay (streaming traffic), (iv) expected transmission time (elastic traffic). We then indicate how our model can be used in the design of the backpressure system, or, more specifically, when selecting suitable values for the thresholds. The last part of the section addresses an alternative model that can be used in case of larger aggregates feeding into the queue.

7.3.1 Experiments

Experiment I: Effect of the thresholds – streaming traffic. In this first experiment we study the effect of the thresholds on the performance in case of streaming traffic. In Chapter 5 we found (for a considerably more stylized model) that, for a given value of the upper threshold B_1, the throughput was maximized by choosing the lower threshold B_2 as closely as possible to B_1. What we did not address in [48] is to what extent this affects the signaling frequency, packet delays, and transmission times.

In this example we chose the following parameters, with $c = 10$:

$$Q^+ = Q^- = \begin{pmatrix} -1 & 1 \\ 1 & -1 \end{pmatrix}; \quad \boldsymbol{r}^+ = \begin{pmatrix} 25 \\ 0 \end{pmatrix}; \quad \boldsymbol{r}^- = \begin{pmatrix} 15 \\ 0 \end{pmatrix}.$$

Remark that this situation is typical for a streaming user: when there is low (high, respectively) congestion, it is allowed to transmit at a high (low) rate, but the generator matrices, i.e., Q^+ and Q^-, are *not* affected by the level of congestion. In other words: a sample-path of the process consists of a sequence of on- and off-times. The results are presented in Fig. 7.1. It is noted that the mean buffer content and the mean packet delay can be easily translated in one another, noticing that (due to Little's formula) the mean buffer content equals the product of the throughput and the mean packet delay. This motivates why we have chosen to show just the throughput and the mean packet delay, and to leave out the mean buffer content; the reader can compute the mean buffer content easily. We mention that in all our experiments the mean buffer content showed the same qualitative behavior as the mean packet delay.

Consider the situation of a fixed value of B_1, and compare the situations of (A) $B_2 < B_1$ and (B) $B_2 = B_1$. From the graphs we will see that, compared to situation (B), under (A) the throughput, signaling frequency, and mean packet delay are lower. In other words: there is a trade-off between throughput on one hand, and signaling frequency and mean packet delay on the other hand.

These trends can be explained as follows. First observe that epochs at which the buffer content is B_1 and the phase jumps from '+' to '−' are regeneration epochs, in that the process *probabilistically* starts all over. Let time 0 be such a regeneration epoch, and let $W_{\mathrm{A}}(t)$ be the workload process in situation (A), and $W_{\mathrm{B}}(t)$ the workload in situation (B). Then it is seen that $W_{\mathrm{A}}(t) \leq W_{\mathrm{B}}(t)$ sample-path-wise, and hence $\mathbb{P}(W_{\mathrm{A}} = 0) \geq \mathbb{P}(W_{\mathrm{B}} = 0)$, and hence, according to (7.1), the throughput is indeed lower under (A) than under (B). Likewise, it can be argued that regeneration cycles last shorter under (B), and as there are two signals per regeneration period, the signaling frequency under (A) is lower than under (B). With a similar argumentation, it also follows that the mean packet delay is lower under (A) than under (B).

Experiment II: Effect of the transmission rate – streaming traffic. In this experiment we study the effect of the peak rate r_p on the performance. In the service level agreement, typically the r_p will be specified. The effect of having a higher r_p is the following. Observe that regeneration periods become shorter when r_p increases, and hence the signaling rate increases. Also (on a sample-path basis) the workload process increases in r_p, leading to a higher throughput and mean packet

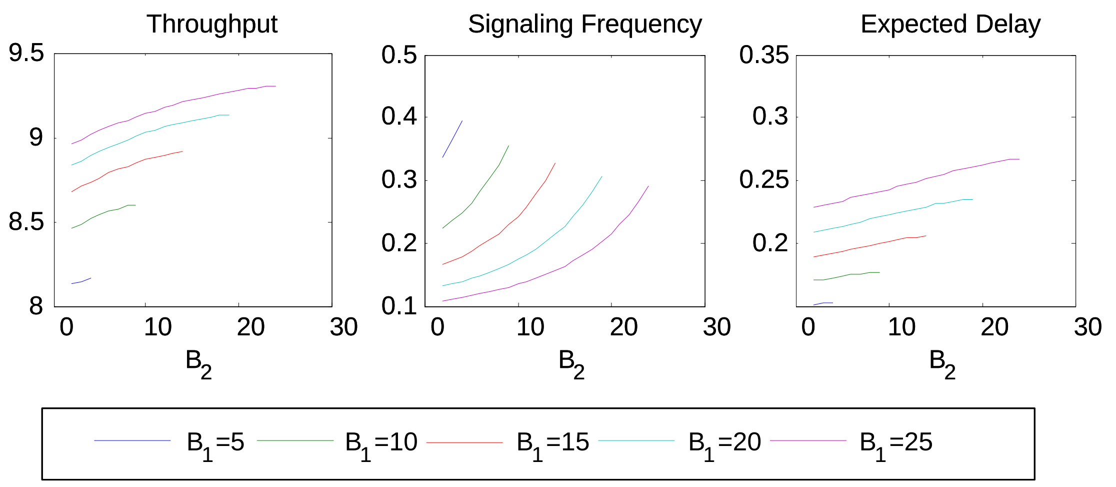


Figure 7.1: Effect of thresholds on streaming traffic.

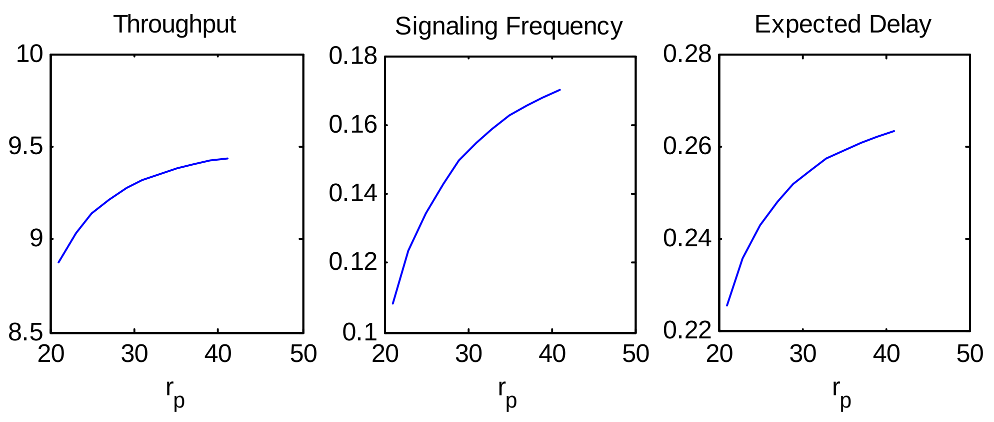


Figure 7.2: Effect of transmission rate r_p on streaming traffic

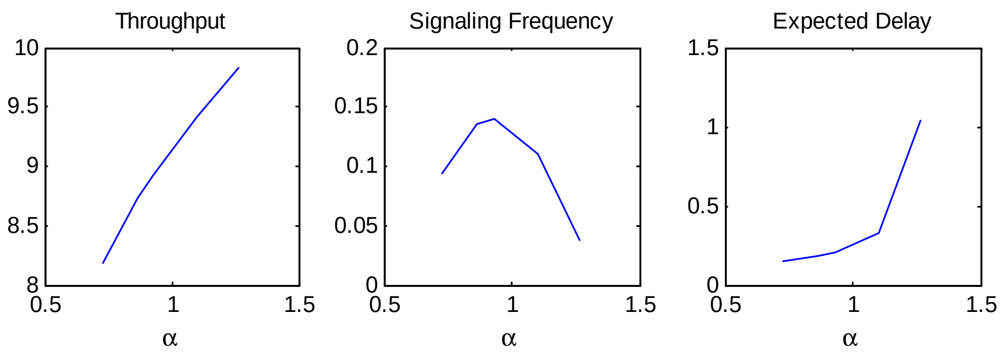


Figure 7.3: Effect of multiplying r_p and r_m by factor α (α is such that the stability condition is satisfied).

delay. Hence, we see a similar effect as in Experiment I. Doubling the peak rate r_p, though, does clearly not lead to doubling the throughput. Remark that it may, at first glance, be slightly counterintuitive that the performance in term of packet delay *degrades* when increasing r_p, but this effect is due to the fact that the buffer content increases. In the numerical experiment, we use the parameters of Experiment I (except that we vary the value of r_p). We chose $B_1 = 25$ and $B_2 = 10$; the graphs are shown in Fig. 7.2.

We also include a related experiment here, where both r_p and r_m are multiplied by α (but α is such that the stability condition remains fulfilled). Now the '+'-phase lasts shorter, while the '$-$'-phase lasts longer. Hence it can be argued that both the delay and throughput increase when r_p and r_m grow, but it is not *a priori* clear what happens with the signaling frequency. The results are presented in Fig. 7.3; it is seen that the signaling frequency shows non-monotone behavior in α.

Notice that the above insights are of interest for the user. The r_p is the fastest rate he can transmit at, whereas the r_m can be regarded as some minimally guaranteed transmission rate. These are rates that are agreed upon in the service level agreement. Clearly, the higher the transmission rates, the more the customer will be charged. The figures may guide the user in choosing his r_p and r_m, taking into account this trade-off.

Experiment III: Effect of the thresholds – elastic traffic. In this third experiment we study the effect of the thresholds on the performance for the case of elastic traffic. We wonder if, in order to maximize the throughput, just as in the case of streaming traffic, it is again optimal to choose $B_2 = B_1$; we are also interested in the impact of the choice of the thresholds on the other performance metrics.

In this example we chose the following parameters, with $c = 10$:

$$Q^+ = \begin{pmatrix} -1 & 1 \\ 1 & -1 \end{pmatrix}; \quad Q^- = \begin{pmatrix} -\frac{3}{5} & \frac{3}{5} \\ 1 & -1 \end{pmatrix}; \quad \boldsymbol{r}^+ = \begin{pmatrix} 25 \\ 0 \end{pmatrix}; \quad \boldsymbol{r}^- = \begin{pmatrix} 15 \\ 0 \end{pmatrix}.$$

Remark that this situation is typical for an elastic user: when there is low (high, respectively) congestion, it is allowed to transmit at a high (low) rate, but the generator matrices, i.e., Q^+ and Q^-, are now adapted too in order to reflect the fact that the burst lasts longer when the transmission rate is reduced. A sample-path of the process is now a sequence of job sizes (i.e., measured in *volume*, in, say, bits — hence *not* time) and off-times (measured in *time*, to be interpreted as 'read-times'). In this example the job sizes have an exponential distribution with mean $1/\mu = r_m/m_1 = r_p/p_1 = 25$; and the read-times have an exponential distribution with mean 1. The numerical outcome is presented in Fig. 7.4.

Consider again the situation of a fixed value of B_1, and compare the situations of (A) $B_2 < B_1$ and (B) $B_2 = B_1$. Under (A) regeneration cycles are longer than under (B), and hence the signaling frequency is lower. We have not found, however, a sound argumentation that reveals in which situation the throughput and packet

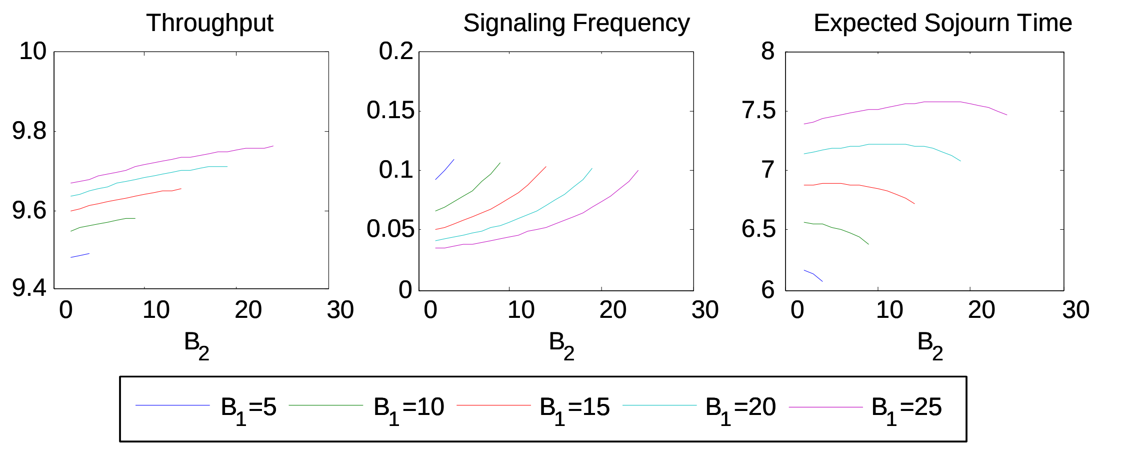


Figure 7.4: Effect of thresholds on elastic traffic.

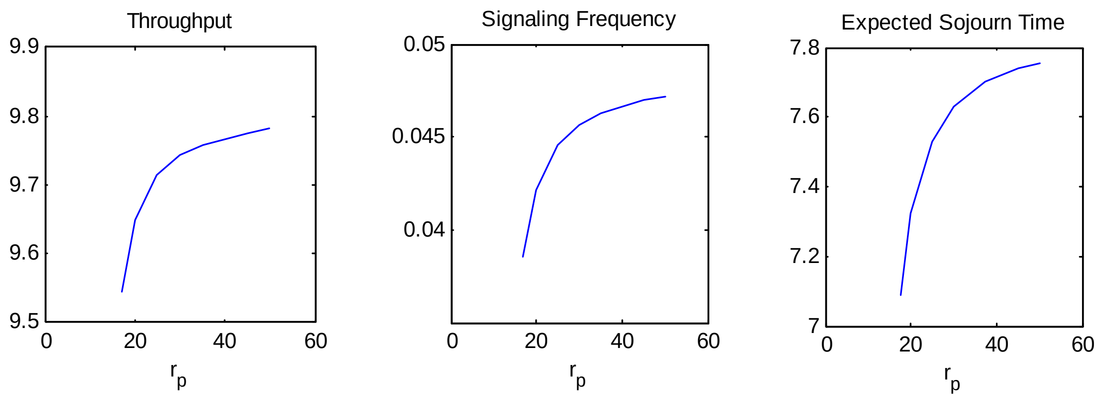


Figure 7.5: Effect of varying transmission rate r_p and p_1 while their ratio $r_p/\ p_1 = 25$

delay are higher. Intuitively one would think that under (B) throughput is higher, which is confirmed by the graphs. The expected sojourn time $\mathbb{E}S$ turns out to have non-monotone behavior in this parameter setting; varying B_2 has clearly impact on the buffer content seen by an arriving job, but in a rather unpredictable way.

We mention that in this experiment the parameters are chosen such that the 'mean drift' while being in the '+'-phase is positive, which implies that the upper threshold B_1 will be reached in a relatively short time (roughly equal to $B_1 - B_2$ divided by this mean drift). The case of a negative 'mean drift' while being in the '+'-phase is less interesting, as it can then be argued that the process will be in the '+'-phase most of the time, and the queue roughly behaves as a non-feedback queue with generator Q^+ and traffic rates $\boldsymbol{r}^+$. In other words: in this case the value of B_1 has hardly any impact on the throughput.

Experiment IV: Effect of the transmission rate – elastic traffic. When r_p in-

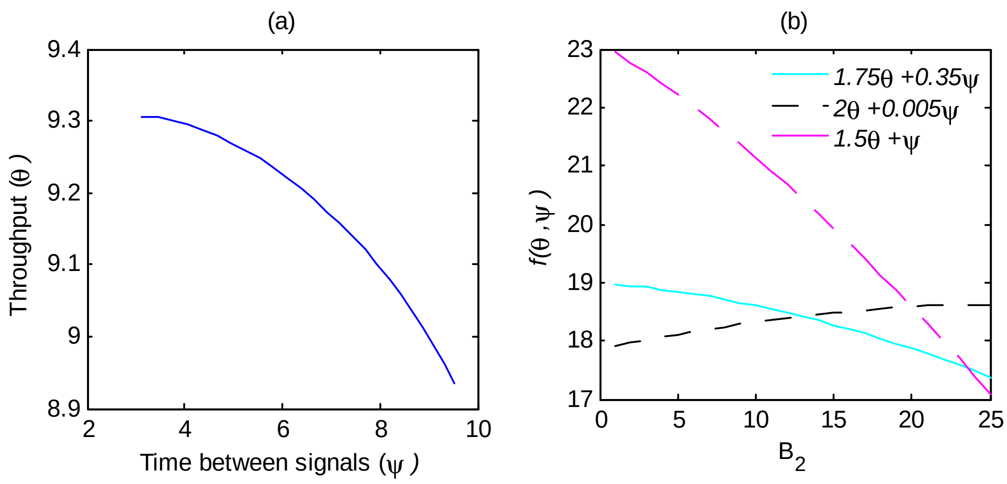


Figure 7.6: Trade-off between throughput ϑ and time between signals ψ (streaming traffic).

creases (with μ held constant, i.e., p_1 increases as well), regeneration cycles become shorter, and hence the signaling frequency increases. As could be intuitively expected also the throughput and expected sojourn time increase, but again we lack a solid argumentation; see Fig. 7.5.

7.4 Design issues

Above we saw that there is a trade off between the signaling frequency and the throughput, and it is the network provider's task to balance these, according to his (subjective) preference. We here sketch how such a decision is facilitated by our model. Figs. 7.6 and 7.7 depict the trade-off between the throughput ϑ and the time between two subsequent signals $\psi := 1/\varphi$, for a given B_1 by varying $B_2 \in [0, B_1]$; it provides us with a (decreasing) function $\vartheta = g(\psi)$ (see the left panels in Figs. 7.6 and 7.7). The provider having objective function $f(\vartheta, \psi)$, increasing in both ϑ and ψ, is faced with the following optimization problem:

$$\max_{\vartheta,\psi} f(\vartheta, \psi) \quad \text{under} \quad \vartheta = g(\psi).$$

Having identified the optimally achievable pair $(\vartheta^\star, \psi^\star)$, we can now reconstruct what the corresponding value $B_2^\star$ was. Clearly, a similar procedure can be set up with both B_1 and B_2 being decision variables.

In Figs. 7.6-7.7 we graphically illustrate how to identify the optimum for the objective function $f(\vartheta, \psi) = \xi_1\vartheta + \xi_2\psi$. Fig. 7.6 uses the parameters of Experiment I, whereas Fig. 7.7 uses the parameters of Experiment III; B_1 is chosen equal to

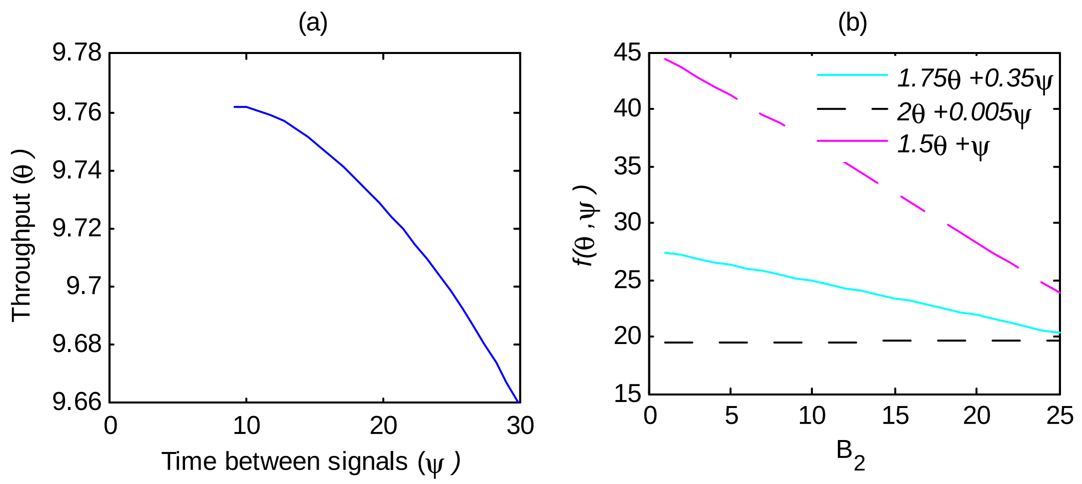


Figure 7.7: Trade-off between throughput ϑ and time between signals ψ (elastic traffic).

25. The left panels show the trade-off between ϑ and ψ, whereas the right panels show the value of the objective function (for several choices of ξ_1 and ξ_2) as a function of B_2. In some of these examples it turns out that the objective function is maximized by choosing B_2 as small as possible. Evidently, this result is specific for the performance measures (ϑ and ψ) and the objective function chosen; the right panel of Fig. 7.6 shows that other choices may lead to a structurally different outcome (there B_2 should be chosen close to 25).

7.5 A model for higher aggregation levels

The experiments in the previous section involved a single source, but the main findings carry over to the situation with multiple sources. This can be validated in detail by redoing the numerical computations, but we here take an alternative approach. This approach is simpler, and somewhat less precise, but still capable of capturing the main trends.

Instead of having both a fluid content process (recorded by $W(t)$) and one or multiple sources (recorded by $X(t)$), we model the buffer content (resulting from the *ensemble* of all sources) by a birth-death-like process: during the '+'-phase the buffer content behaves as an M/M/1 queue with arrival rate λ^+ and departure rate μ^+, whereas during the '$-$'-phase it behaves as an M/M/1 queue with arrival rate λ^- (of traffic quanta of size, say, 1) and departure rate μ^-. Thus, the rate of change of the buffer is no longer determined by vectors $\boldsymbol{r}^-$ and $\boldsymbol{r}^+$ and generator matrices Q^- and Q^+ as before, but simply by birth-and-death parameters λ^+, μ^+, λ^-, μ^-. What remains the same as before, is that these depend on the current mode (that is, '+' or '−'), just as $\boldsymbol{r}$ and Q did before. To analyze a given situation, we can tune the λ^+, μ^+, λ^-, μ^- (satisfying the equilibrium condition $\lambda^- < \mu^-$), so that they roughly match the first and second order characteristics of the buffer dynamics. For this model we verify whether the trends, as observed in Section 7.3.1, still apply.

Let τ^+ be the duration of the '+'-phase, and τ^- the duration of the '−'-phase. It is immediate (for instance from Wald's theorem) that

$$\mathbb{E}\tau^- = \frac{B_1 - B_2}{\mu^- - \lambda^-}.$$

The computation of $\mathbb{E}\tau^+$ is standard, but a bit more tedious. With a_i denote the mean time until B_1 is reached, starting in $i \in \{0, \ldots, B_1 - 1\}$, it is evident that

$$(\lambda^+ + \mu^+)a_i = \lambda^+ a_{i+1} + \mu^+ a_{i-1} + 1, \tag{7.5}$$

for $i = 1, \ldots, B_1 - 1$; also $\lambda^+ a_0 = \lambda^+ a_1 + 1$ and $a_{B_1} = 0$. With $b_i = a_{i+1} - a_i$, Eqn. (7.5) can be rewritten as $\lambda^+ b_i = \mu^+ b_{i-1} - 1$, where $b_0 = -1/\lambda^+$. It is then easy to verify that

$$b_i = -\frac{(\mu^+)^i}{(\lambda^+)^{i+1}} - \left(1 - \left(\frac{\mu^+}{\lambda^+}\right)^i\right) \Big/ \left(1 - \frac{\mu^+}{\lambda^+}\right) = \frac{1}{\lambda^+ - \mu^+}\left(\left(\frac{1}{\varrho^+}\right)^{i+1} - 1\right),$$

with $\varrho^+ := \lambda^+/\mu^+$, and realizing that $-a_i = b_i + \cdots + b_{B_1-1}$ (use $a_{B_1} = 0$),

$$\mathbb{E}\tau^+ = a_{B_2} = -\sum_{j=B_2}^{B_1-1} b_j = \frac{B_1 - B_2}{\lambda^+ - \mu^+} - \frac{1}{\lambda^+}\frac{1}{1-\varrho^+}\frac{(\varrho^+)^{B_2-B_1} - 1}{(\varrho^+)^{B_2} - (\varrho^+)^{B_2-1}};$$

if $\lambda^+ > \mu^+$, then this may be (roughly) approximated by $(B_1 - B_2)/(\lambda^+ - \mu^+)$ (as could be expected), whereas if $\lambda^+ < \mu^+$, then it roughly equals

$$\frac{1}{\mu^+}\frac{1}{(1-\varrho^+)^2}\left(\frac{1}{\varrho^+}\right)^{B_1}.$$

The signaling frequency equals $\varphi = 2/(\mathbb{E}\tau^+ + \mathbb{E}\tau^-)$, by virtue of 'renewal reward'. As is easily verified, the mean time per cycle spent in state 0 is

$$\mathbb{E}\tau_0^+ = \frac{(\varrho^+)^{B_1-B_2} - 1}{(\varrho^+)^{B_1} - (\varrho^+)^{B_1-1}},$$

so that the throughput is given by

$$\vartheta = \frac{\mathbb{E}\tau^+ - \mathbb{E}\tau_0^+}{\mathbb{E}\tau^+ + \mathbb{E}\tau^-} \cdot \frac{1}{\mu^+} + \frac{\mathbb{E}\tau^-}{\mathbb{E}\tau^+ + \mathbb{E}\tau^-} \cdot \frac{1}{\mu^-}.$$

The thresholds B_1 and B_2 can be optimally selected by following a scheme similar to the one sketched in Section 7.4. In Fig. 7.8 we consider an example that again focuses on the trade-off between the throughput ϑ and the time between two consecutive signals ψ. We see the same type of behavior as in the single-source case. The input parameters are $\lambda^+ = 7$, $\mu^+ = 5$, $\lambda^- = 4$, $\mu^- = 5$, so that there is a positive drift during the '+'-phase. The thresholds are $B_1 = 25$ and $B_2 = 10$.

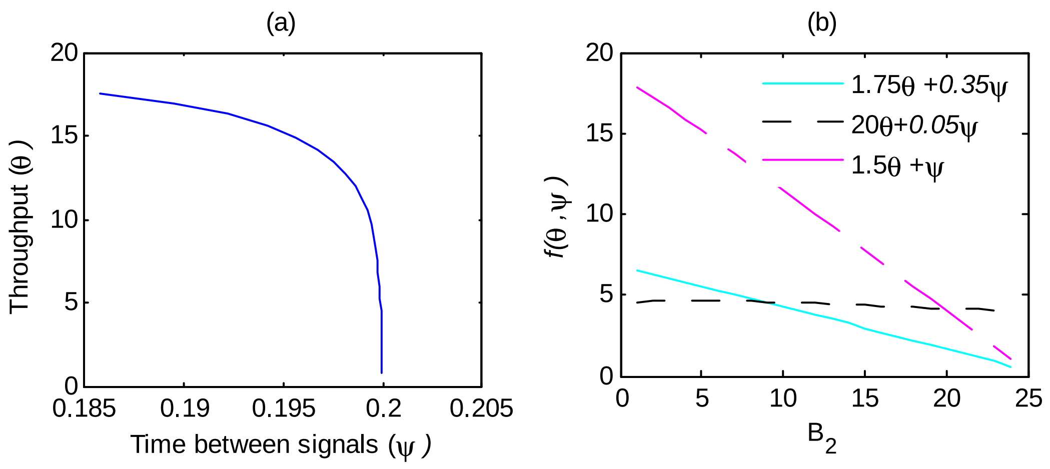


Figure 7.8: Trade-off between throughput ϑ and time between signals ψ for higher aggregation model.

7.6 Concluding remarks

This chapter addressed a methodology for resolving design issues in backpressure-based control mechanisms. Relying on a feedback fluid model Chapter 6, we derived closed form expressions (in terms of the solution of certain eigensystems, and additionally a system of linear equations) for a number of key performance metrics. It enables us to investigate in detail the trade-offs involved – for instance the trade-off between throughput and the signaling overhead – and thus facilitates a proper selection of the protocol's design parameters (such as the values of the thresholds). It also sheds light on the effect of changing the transmission rates. We also presented a more stylized model that is particularly useful when the input consists of a substantially larger aggregate of users.

Part III

Scheduling

Introduction to Part III

Carrier Ethernet currently has the ability to categorize incoming traffic into multiple traffic classes as explained in Chapter 1. The next step is to divide the network resources over these traffic classes in such a way that their requirements can be met. Scheduling mechanisms fulfill this purpose and are widely studied in literature in the context of other packet technologies. These mechanisms can be broadly classified into two basic types: priority queueing (PQ) and weighted fair queueing (WFQ). PQ provides absolute priority to the selected traffic class over the lower priority ones. Being in the high priority class is particularly suitable for time-sensitive traffic as their delay guarantees can be maintained. The disadvantage of PQ is that in times of traffic overload combined with absence of proper admission control, it can lead to starvation of lower priority queues. WFQ and its variants aim at providing fair bandwidth division among the various traffic classes by guaranteeing each class a certain minimum bandwidth. However, it is difficult to maintain delay bounds with this scheduling discipline. A combination of PQ and WFQ provides the possibility to minimize delay and jitter guarantees for time-sensitive stream traffic while allocating bandwidth fairly among elastic flows. Nevertheless, it remains essential to control the traffic load on the strict priority queue in order to avoid starvation of lower priority traffic. In absence of such control not only are the delay guarantees of the high priority stream traffic in jeopardy, but complete and consistent starvation of lower priority elastic traffic can lead to collapse of the TCP/elastic streams.

In this part of the thesis we do not intend to develop yet another scheduling mechanism. Instead we aim at capturing the influence of high priority (stream) traffic load on the performance of the low priority (elastic) traffic. This can assist in the admission control of both stream and elastic traffic into the network taking their interaction into account.

In Chapter 8, we propose a simple yet effective approximation of the performance of elastic traffic integrated with prioritized stream traffic. The approximation is based on modifications to processor sharing type of queueing models for elastic traffic performance accounting for the presence of prioritized stream traffic. The proposed prioritized model includes parameters for both elastic and stream traffic load We validate the accuracy of our model by comparison to simulations. We show that the approximation works well for a wide range of parameter values.

Chapter 8

Integrating elastic traffic with prioritized stream traffic

Current and emerging communications services and applications are extremely diverse in nature as they serve a wide range of end-user needs. However, the kind of traffic being generated by these applications and services can still be broadly classified into two basic categories: stream and elastic. Stream traffic is generated by applications such as VoIP, streaming video etc. These applications have strict bandwidth, end-to-end packet delay and jitter requirements for reliable operation. Elastic traffic on the other hand is generated by applications such as file-transfer, web-browsing, etc. For these applications only the total amount of time required to download the file or web-page is of importance. The end-to-end delay of individual packets and jitter are not relevant. In order to satisfy the strict delay requirements of stream traffic it is given priority over elastic traffic at each hop in a network. However, it is not the intention that this preferential treatment to stream traffic results in performance targets for elastic traffic not being met.

The flow level performance of elastic traffic integrated with prioritized stream traffic has not received much attention in literature. References [79] and [9] consider a priority system consisting of elastic streams only. Some articles such as [53] address this issue by modeling service interruptions for a processor sharing system for elastic traffic. References [17] and [42] consider the integration of stream and elastic traffic and provide approximations. The drawback of the approach in [42] is that it studies elastic flow transfer times under large system asymptotics and does not consider non-exponential service distributions. Reference [17] on the other hand poses no limit on the server capacity or bandwidth which can be used by elastic flows. In practice, elastic flows are often limited by policing rate, access rates or maximum TCP window limitation. This fact needs to be accommodated in the model especially for metropolitan and core networks. A Markov chain analysis of this problem is available in [54]. This requires numerically solving the steady state solution which turns out to be very computation intensive. This approach does not provide the possibility to capture the behavior in simple analytical expressions. Approximation techniques are an alternative to Markov chain analysis. Existing methods such as quasi-stationarity can provide simple approximations but are sometimes far too abstract to capture the influence of all relevant parameters. In this chapter we bridge the gap between these two approaches and propose a simple yet detailed approximation, which captures the behavior and performance of both stream and elastic traffic in a priority system with traffic control. We compare our approach on one hand to simulations and on the other hand to existing approximation methods. We assess the accuracy of our approximations for a wide range of parameters. Numerical results show that our proposed approximation outperforms other approximation methods.

8.1 Model

In this section, we describe a model aimed at understanding the relation between the performance of stream and elastic traffic in a priority system and the influence of admission control. We consider a flow level model. Stream flows are allocated a fixed bandwidth when admitted into the system and treated with priority over elastic traffic. A simple control strategy is used for the high priority stream traffic flows. When all the capacity is exhausted, further incoming stream calls are blocked. The low priority traffic consists of elastic flows which adapt to server capacity available after serving stream flows. No form of admission control is exercised on the elastic flows. Both stream and elastic flows are never served at more than a peak rate.

This model is motivated by the fact that the streaming flows have strict bandwidth and delay requirements which can be met if the requested capacity is allocated to them completely. Therefore, it is preferred to block flows whose requirements cannot be met rather than allow them into the system and jeopardize the performance of every flow. This also implies that once the bandwidth/capacity is completely saturated further stream flows should not be admitted into the system. Since elastic flows adapt to server capacity it is not necessary to block them but should be served with low priority so as not to jeopardize the delay requirements of the stream flows. Our model also uses a peak rate for both stream and elastic traffic. This limitation can be justified by the limit imposed by policers, access rates and TCP window.

The model we consider is a $M/G/C$ system serving two priority classes, as shown in Figure 8.1. The high priority traffic consists of stream flows. We renormalize the bandwidth such that their peak rate is 1 and this capacity is allotted to each stream flow admitted in the system. Each flow occupies this unit server capacity for a certain period of time called the service time requirement of the flow. High priority stream flows are admitted into the system as long as there is capacity available. This implies that if there are already C high priority flows in the systems (each occupying unit server capacity), further incoming stream flows will be blocked and dropped.

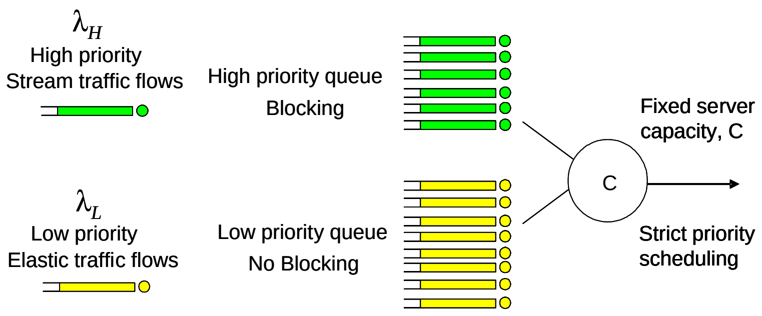


Figure 8.1: A model for stream and elastic traffic in a priority system with admission control.

We assume that high priority stream calls arrive into the system according to a Poisson process with rate λ_H. The service time requirements of the high priority traffic are generally distributed with cumulative distribution $B_H(\cdot)$ and finite first two moments β_H and $\beta_H^{(2)}$.

The lower priority class is a processor sharing system. This means that the capacity left over from serving stream flows is equally divided among the elastic traffic flows. Processor sharing queues have been shown to be effective in modeling elastic streams that use the transport control protocol [68], [67]. The capacity left over for the elastic flows varies in time and is reduced to zero if there are C stream flows in the system, where C is an integer. The low priority traffic flows are served with processor sharing with a peak rate of 1. This implies that when the number of low priority traffic flows in the system is less than $C - N_H$ each flow receives unit server capacity and when the number of flows are greater than $C - N_H$ each flow receives $(C - N_H)/N_L$ server capacity, where, N_H and N_L are the number of high and low priority flows respectively and vary over time. We assume that low priority traffic flows arrive according to a Poisson process with rate λ_L. The service time requirements of the low priority customers are generally distributed with cumulative distribution $B_L(\cdot)$ and finite first two moments β_L and $\beta_L^{(2)}$.

It might seem a limitation of the model that it considers equal peak rate for both stream and elastic traffic. In this regard it is interesting to note that since stream traffic can potentially occupy the complete server capacity, it is evident that in moderate to high load situations the effective rate (peak rate) received by the lower priority elastic traffic will always be less than that of the stream traffic.

8.2 Analysis of high priority traffic

For high priority stream traffic each flow is served at unit rate. Thus each stream occupies one server (of unit capacity) out of the C servers. If there are already C high priority flows in the system then any new flow arrivals are lost. Hence this reduces to an Erlang loss queue i.e., a $M/G/C/C$ system with exponential interarrival times, general service time distribution, C servers and no waiting room. The probability of blocking or rather loss for the high priority traffic flows can be directly obtained from the well-known Erlang Loss Formula

$$P_{loss} = \frac{\frac{(\lambda_H \beta_H)^C}{C!}}{\sum_{n=0}^{C} \frac{(\lambda_H \beta_H)^n}{n!}}, \tag{8.1}$$

and the probability of n high priority flows in the system is given by

$$\mathbb{P}[N_H = n] = \frac{\frac{(\lambda_H \beta_H)^n}{n!}}{\sum_{k=0}^{C} \frac{(\lambda_H \beta_H)^k}{k!}}, \text{ where } n = 1, ..., C. \tag{8.2}$$

The effective arrival rate of the high priority flows for this system can be computed as

$$\lambda_{H,\text{effective}} = \lambda_H \times (1 - P_{loss}). \tag{8.3}$$

8.3 Analysis of low priority traffic

The analysis of the sojourn time for the low priority flows is far more complicated than the high priority flows. The complication is due to the fact that the capacity available for the low priority flows fluctuates over time. In addition, if the number of high priority traffic flows is equal to C, the low priority elastic traffic experiences starvation, i.e., its service rate equals zero for that period of time. The complete system is a mix of a $M/G/C/C$ Erlang loss queue and a processor sharing system. A closed form formula for the sojourn time in such a system is not available in literature so far. In this section our goal is to provide some approximations for the sojourn time of such a system. First, we review some results from processor sharing modeling theory which plays an important role in modeling the performance of the elastic flows and in the approximations we propose. We then provide three approximations for the average file transfer time which is the same as the average sojourn time ($\mathbb{E}[S_L]$) of the file in the system.

8.3.1 Processor Sharing theory

In this subsection we will review some standard results from processor sharing models. Processor sharing (PS) models are well-suited for capturing the elastic behavior of TCP traffic that adapts its transmission rate to the available network capacity. In the standard PS model, the server capacity is divided equally among all traffic flows in the system. The reader is referred to [89] and [88] for an overview of results on standard PS models. An interesting generalization of the PS model is the so-called generalized processor sharing (GPS) model which allows the service rate for each traffic flow to be state dependent [13]. Consequently, if there are n traffic flows in the system, each traffic flow receives a fraction $f(n)$ of the service speed. The joint stationary distribution of the number of traffic flows N and their residual service time requirements $\underline{T} := (T(1), ..., T(N))$ can be given by (cf., e.g.[13])

$$\mathbb{P}[N = n, \underline{T} = \underline{\tau}] = \frac{(\rho^n/n!)\phi(n)}{\sum_{k=0}^{C}(\rho^k/k!)\phi(k)} \prod_{i=1}^{n} \frac{1 - B(\tau(i))}{\beta}, \text{ for } n = 0, 1, ..., \tau(i) \geqslant 0, \quad (8.4)$$

where $\phi(0) = 1$, $\phi(n) = \left(\prod_{i=1}^{n} f(i)\right)^{-1}, \rho = \lambda\beta$, λ is the average Poisson arrival rate of flows into the system, $B(\cdot)$ is the distribution function of the service time requirements and β is the average service time requirement. Further in this chapter we will consider a special case of a GPS system where the rate at which each traffic flow is served whenever there are i traffic flows in the system is given by $f(i) = 1$ if $0 \leq i \leq C$, and $f(i) = C/i$ if $i > C$. The mean number of traffic flows $\mathbb{E}[N_{GPS}(\rho, C)]$, in this GPS model for $0 \leq \rho < C$ is given by

$$\mathbb{E}[N_{GPS}(\rho, C)] = \frac{C^C}{C!A(C,\rho)} \left(\frac{C(\rho/C)^{C+1}}{1 - \rho/C} + \frac{(\rho/C)^{C+1}}{(1 - \rho/C)^2} \right) + \frac{1}{A(C,\rho)} \sum_{n=1}^{C} \frac{\rho}{(n-1)!}, \quad (8.5)$$

with

$$A(C, \rho) = \frac{C^C}{C!} \frac{(\rho/C)^{C+1}}{1 - \rho/C} + \sum_{n=0}^{C} \frac{\rho^n}{n!}.$$

8.3.2 Simple and known approximation techniques

In this subsection we will introduce straightforward approximations for the performance of low priority elastic flows. The first technique is based on quasi-stationarity and the second technique is based on an estimation of the left over capacity for elastic flows.

Approximation 1, based on quasi-stationarity:

Here we present an approximation based on quasi-stationarity. If we assume that there are i high priority traffic streams in the system, then the left over capacity for the low priority flows is $(C - i)$. Therefore, the average sojourn time (file transfer time) for low priority flows can be computed using Equation (8.5) with capacity $(C - i)$ instead of C. If this value is multiplied by the probability that there are i traffic flows in the high priority queue ($\mathbb{P}[N_H = i]$) and summed over all possible values of i we obtain $Approx_1$, the estimate of the average sojourn time for elastic traffic flows.

$$Approx_1 = \sum_{i=0}^{C} \frac{\mathbb{E}[N_{GPS}(\rho_L, C - i)]}{\lambda_L} \mathbb{P}[N_H = i]. \tag{8.6}$$

It is important to note that for values of N_H sufficiently close to C, there is negligible capacity left over for the low priority traffic and as a result the GPS model and the above approximation becomes unstable. Therefore, while analysing the performance of this approximation we chose to leave out the sojourn time results where $\mathbb{P}[N_H = C]$ was not negligible.

Approximation 2, based on estimation of available capacity for elastic traffic:

Here we propose a simple approximation based on the estimation of left over capacity for elastic flows. We first calculate the expected number of high priority traffic flows $\mathbb{E}[N_H]$. This value is then rounded off to an integer value RND($\mathbb{E}[N_H]$). The low priority system can then be assumed to be a GPS system with capacity $C-$RND($\mathbb{E}[N_H]$).

$$Approx_2 = \frac{\mathbb{E}[N_{GPS}(\rho_L, C - \text{RND}(\mathbb{E}[N_H]))]}{\lambda_L}, \tag{8.7}$$

where, $0 \le \rho_L < \{C-\text{RND}(\mathbb{E}[N_H])\}$. There are several evident drawbacks of this approximation. Firstly it depends on high priority traffic only through $\mathbb{E}[N_H]$, thus fluctuations in N_H are not taken into account. Since $\mathbb{E}[N_H]$ is a function of ρ_H, $Approx_2$ is also a function of ρ_H and is insensitive to the individual values of λ_H and β_H. Finally, the accuracy of this approximation depends greatly on the efficiency of the rounding off. Greater the difference between the original $\mathbb{E}[N_H]$ and the rounded off value, larger the error with $Approx_2$.

8.3.3 Approximation 3, New enhanced sojourn time approximation

The approximations $Approx_1$ and $Approx_2$ proposed in the previous section are straightforward but on the other hand have evident drawbacks. $Approx_1$ and $Approx_2$ are completely insensitive to the higher moments of the service time requirement distributions of both stream and elastic traffic. Here we present an approximation for which this insensitivity property does not hold. The basic idea of the approximation is to focus on the *workload*. Let us denote by W_H the total workload of the high priority traffic flows, by W_L the total workload of the low priority traffic flows and by W_{H+L} the total workload of the complete system. For $C \geqslant 1$

$$\mathbb{E}[W_H] + \mathbb{E}[W_L] = \mathbb{E}[W_{H+L}] \approx \mathbb{E}[W_{mix}], \tag{8.8}$$

where W_{mix} stands for the total amount of unfinished work in the corresponding GPS system without priorities (referred to as the mixed system), i.e. the system with

$$\lambda_{mix} \approx \lambda_{H,\text{effective}} + \lambda_L, \tag{8.9}$$

$$\begin{aligned} \beta_{mix} &\approx \frac{\lambda_{H,\text{effective}}\beta_H}{\lambda_{H,\text{effective}} + \lambda_L} + \frac{\lambda_L\beta_L}{\lambda_{H,\text{effective}} + \lambda_L}, \\ \beta^{(2)}_{mix} &\approx \frac{\lambda_{H,\text{effective}}\beta^{(2)}_H}{\lambda_{H,\text{effective}} + \lambda_L} + \frac{\lambda_L\beta^{(2)}_L}{\lambda_{H,\text{effective}} + \lambda_L}. \end{aligned} \tag{8.10}$$

The original system contains a non-GPS part, i.e., the high priority part and a GPS low priority part. However, we approximate the original system as a lossless GPS system with total load $\rho_{H,\text{effective}+L} = \rho_{H,\text{effective}} + \rho_L = \lambda_{H,\text{effective}}\beta_H + \lambda_L\beta_L$. Since the approximate system is lossless we use $\lambda_{H,\text{effective}}$ and not λ_H. The mean number of traffic flows $\mathbb{E}[N_{GPS}(\rho_{H,\text{effective}+L})]$ in this GPS system for $0 \leq \rho_{H,\text{effective}+L} < C$ can be computed by using Equation (8.5). Hence, the total workload of the mixed system can be expressed as

$$\mathbb{E}[W_{H+L}] \approx \mathbb{E}[W_{mix}] = \frac{\beta^{(2)}_{mix}}{2\beta_{mix}}\mathbb{E}[N_{GPS}(\rho_{H,\text{effective}+L}, C)]. \tag{8.11}$$

For the high priority traffic flows we have

$$\mathbb{E}[W_H] = \frac{\beta^{(2)}_H}{2\beta_H}\mathbb{E}[N_H], \tag{8.12}$$

where $\mathbb{E}[N_H]$ is obtained using Equation (8.2).

Now we can approximate the mean workload of the low priority traffic flows as follows using Equation (8.2),

$$\mathbb{E}[W_L] \approx \mathbb{E}[W_{mix}] - \mathbb{E}[W_H],$$

and hence,

$$\mathbb{E}[W_L] \approx \frac{\beta^{(2)}_{mix}}{2\beta_{mix}}\mathbb{E}[N_{GPS}(\rho_{H,\text{effective}+L}, C)] - \frac{\beta^{(2)}_H}{2\beta_H}\mathbb{E}[N_H]. \tag{8.13}$$

For high priority traffic flows the amount of unfinished work is the independent sum of the remaining service times of all high-priority traffic flows in the system. We adopt this idea to the low priority traffic flows to obtain the following approximate relation between the number of low priority traffic flows $\mathbb{E}[N_L]$ and corresponding mean amount of unfinished work $\mathbb{E}[W_L]$:

$$\mathbb{E}[N_L] \approx \frac{\mathbb{E}[W_L]}{\beta_L^{(2)}/2\beta_L}. \tag{8.14}$$

Note that this relation is generally not exact, unless the service times for low priority traffic flows are exponentially distributed. Combining Equations (8.13), (8.14) and Little's law we have the third approximation for the file transfer or sojourn time as

$$Approx_3 = \frac{2\beta_L}{\lambda_L \beta_L^{(2)}} \frac{\beta_{mix}^{(2)}}{2\beta_{mix}} \mathbb{E}[N_{GPS}(\rho_{H,\text{effective}+L}, C)] - \frac{2\beta_L}{\lambda_L \beta_L^{(2)}} \frac{\beta_H^{(2)}}{2\beta_H} \mathbb{E}[N_H]). \tag{8.15}$$

The accuracy of the proposed approximation and the comparison among the different approximations is assessed in the next section.

8.4 Results

In this section our goal is to assess the accuracy of the approximations proposed in Sections 8.3.2 and 8.3.3. The accuracy is assessed by comparing the approximations to accurate simulations of the model. Two types of service time requirement distributions are considered: exponential and hyper-exponential. The exponential distribution is the relatively simple case, as it does not involve many parameters and for which Equation (8.14) is exact. For the hyper-exponential distribution however, this is not the case. The hyper-exponential distribution contains many parameters which can be varied and thus helps in understanding a different aspect of the service time distribution. Since none of the approximated steps are exact for this distribution, it is also a good check of the accuracy. The results presented in this section are with the load per server of the low priority traffic kept constant at 0.5 whereas the load of the high priority is increased in gradual steps from 0.1 to 0.45. Various server capacity i.e., C values are considered along with the two cases $\beta_H < \beta_L$ and $\beta_H > \beta_L$. The results for $\beta_H = \beta_L$ were similar to $\beta_H < \beta_L$, so we do not present them in the chapter. We define the percentage error between the exact simulations values and the approximated values as

$$\Delta_i = \frac{Exact - Approx_i}{Exact} \times 100. \tag{8.16}$$

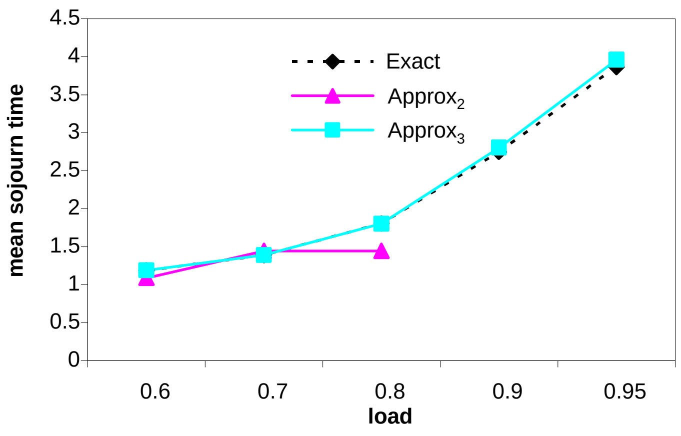


Figure 8.2: Exact and approximated mean sojourn times for low priority traffic for different values of load per server for $\beta_H/\beta_L = 1/4$ and $C = 4$.

8.4.1 Exponential service time distribution

In this section we consider an exponential service time requirement distribution for both high and low priority traffic. We keep the load per server of the low priority traffic fixed to $\rho_L/C = 0.5$ but vary the load of the high priority traffic i.e., $\rho_H/C = 0.1, 0.2, 0.3, 0.4, 0.45$ and observe the influence on the results and accuracy of the approximation.

Figures 8.2, 8.3 and 8.4 show results for $C = 4, 10$ and 20 respectively for the case $\beta_H = 1/4$ and $\beta_L = 1$. Figure 8.2 shows the mean sojourn times for low priority traffic as a function of the load per server values obtained from simulations (exact), and for $Approx_2$ and $Approx_3$. $Approx_1$ does not yield meaningful results in this case. This is due to the fact that there is a reasonable chance that N_H is sufficiently close to C, not leaving enough capacity for the low priority system, making it unstable (see Equation (8.7)). $Approx_2$ on the other hand greatly underestimates the sojourn time. This is because the rounding off of $\mathbb{E}[N_H]$ to an integer overestimates the original value of $\mathbb{E}[N_H]$. For high load values, $Approx_2$ fails to provide results. This is because the rounding off error leaves no capacity for low priority elastic traffic making the system unstable. $Approx_3$ performs very well in this case and the errors are below 4% even for large load and loss probabilities.

Figure 8.3 and 8.4 show similar trends as Figure 8.2. $Approx_1$ continues to greatly overestimate the sojourn times. Only for larger $C = 20$ and very low load it performs reasonable. The performance of $Approx_2$ improves considerably for large C values. The best results are with $C = 20$ where it follows the trends quite well. The error is of the order of 13%. However, $Approx_3$ clearly outperforms $Approx_1$ and $Approx_2$ with errors below 3%.

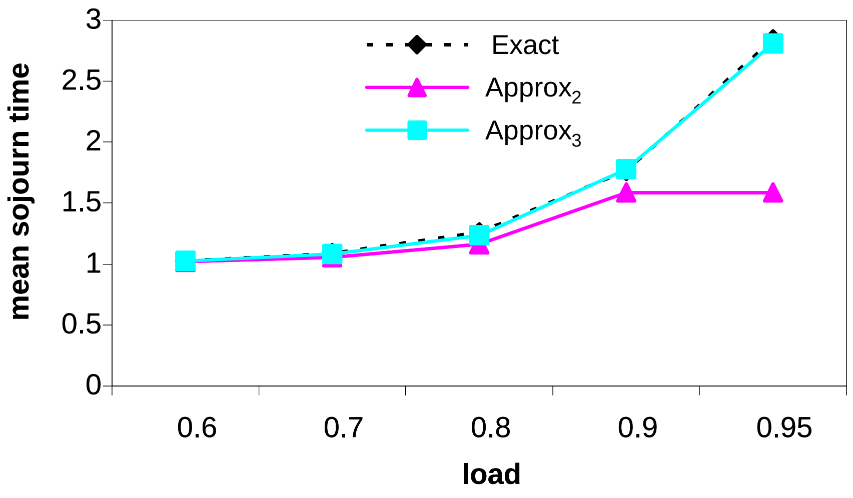


Figure 8.3: Exact and approximated mean sojourn times for low priority traffic for different values of load per server for $\beta_H/\beta_L = 1/4$ and $C = 10$.

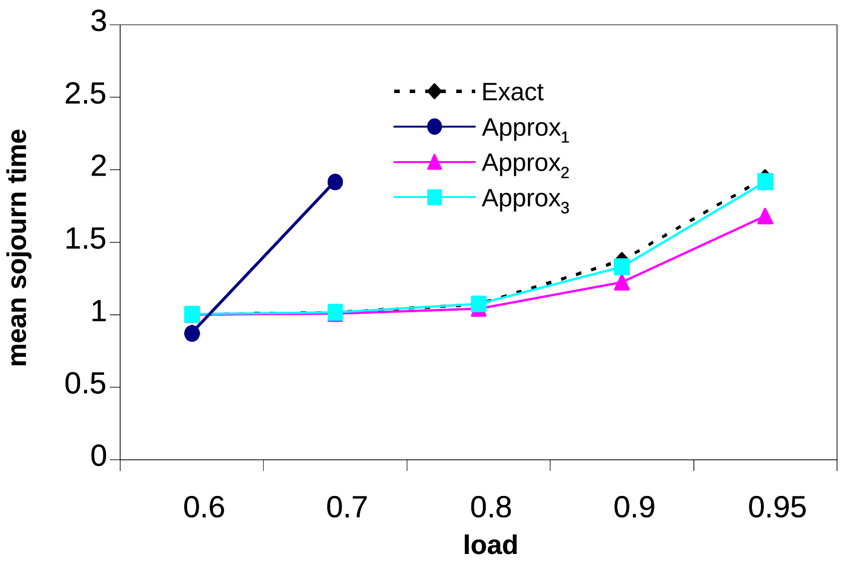


Figure 8.4: Exact and approximated mean sojourn times for low priority traffic for different values of load per server for $\beta_H/\beta_L = 1/4$ and $C = 20$.

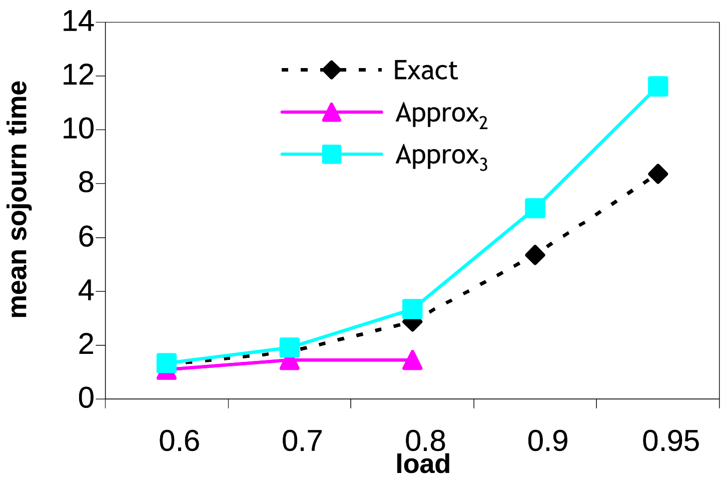


Figure 8.5: Exact and approximated mean sojourn times for low priority traffic for different values of load per server for $\beta_H/\beta_L = 4$ and $C = 4$.

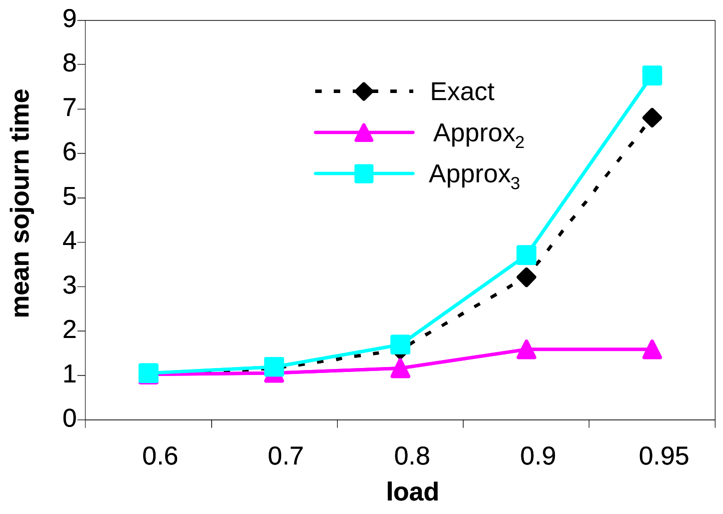


Figure 8.6: Exact and approximated mean sojourn times for low priority traffic for different values of load per server for $\beta_H/\beta_L = 4$ and $C = 10$.

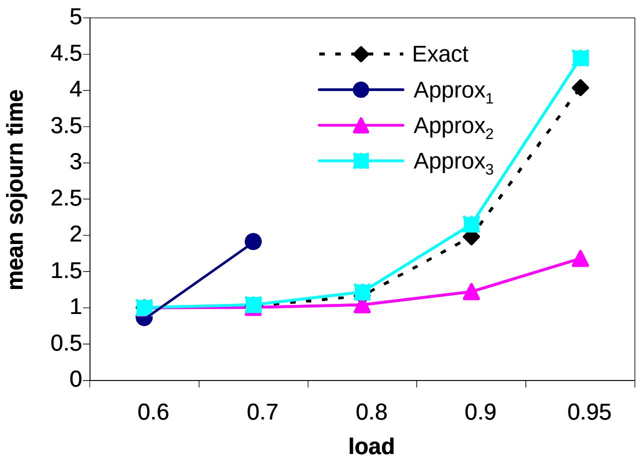


Figure 8.7: Exact and approximated mean sojourn times for low priority traffic for different values of load per server for $\beta_H/\beta_L = 4$ and $C = 20$.

Figures 8.5, 8.6 and 8.7 show results for the case $\beta_H > \beta_L$ with $\beta_H = 4$ and $\beta_L = 1$ for $C = 4, 10$ and 20 respectively. Similar trends can be observed as for the case with $\beta_H < \beta_L$, however the absolute errors with *Approx*$_3$ are distinctly larger especially for smaller C values. For $C = 4$ the maximum error is of the range of 38% whereas with $C = 20$ the maximum error is about 10% for load of 0.95. Lower load values, show a much smaller error. It is important to understand why the accuracy of *Approx*$_3$ is so much better for $\beta_H < \beta_L$ than for $\beta_H > \beta_L$. We approximate a priority system with a GPS system without priorities. The approximate GPS system treats all flows equally thus favoring flows with smaller service time requirement over larger ones. This implies that for $\beta_H < \beta_L$, both the original and the approximate system favor the flows with smaller service time requirement. Thus the errors in this case are very small. For the situation that $\beta_H > \beta_L$, the approximate system continues to favor smaller service time requiring flows whereas the original system favors or prioritizes stream calls having a larger service time requirement. Therefore, in this case the difference between the approximate and the original system is larger leading to larger errors.

In spite of reduced accuracy with *Approx*$_3$, we observe that it still outperforms *Approx*$_1$ and *Approx*$_2$. Nevertheless, we do conclude that *Approx*$_3$ is not suitable for access networks where link capacities are likely to be small.

8.4.2 Hyper-Exponential service time distribution

In this subsection we assess the accuracy of the approximations for a non-exponential distribution namely the hyper-exponential distribution. In contrast to the exponential distribution, the hyper-exponential distribution provides insight into the influ-

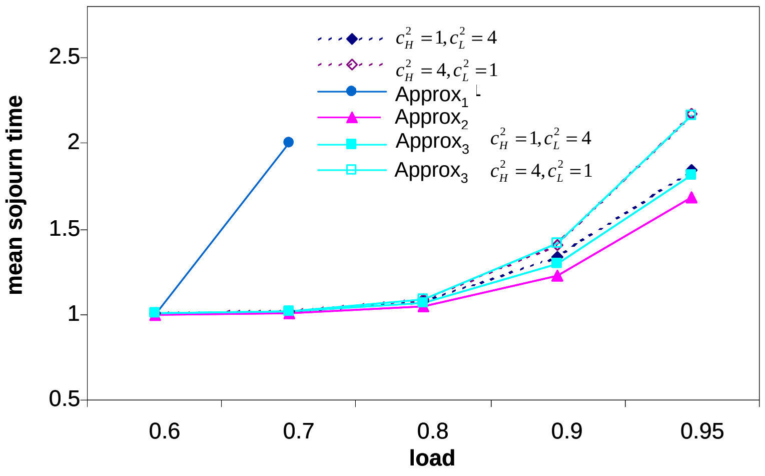


Figure 8.8: Exact and approximated values of the average sojourn times for $C = 20$, $\beta_H = 0.25, \beta_L = 1$ of various values of load per server.

ence of the variance of the service time distribution. The sensitivity and accuracy of the different approximations to the variation is also assessed. We consider hyper-exponential distribution with balanced means [78]. The coefficient of variation is used as a measure of variation and the following cases are considered: $c_H^2 > c_L^2$, $c_H^2 < c_L^2$ and $c_H^2 = c_L^2$. The average service time requirement considered includes both possibilities, $\beta_H < \beta_L$ as well as $\beta_H > \beta_L$.

Figures 8.8 and 8.9 demonstrate the sensitivity and accuracy of $Approx_1$, $Approx_2$ and $Approx_3$ for $C = 20$. It is clear from both the figures that $Approx_3$ outperforms $Approx_1$ and $Approx_2$. In fact $Approx_1$ and $Approx_2$ are completely insensitive to the choice of the coefficient of variation. The accuracy of $Approx_3$ is greatly influenced by the value of the coefficient of variation and as before the ratio of β_H to β_L. The results in Figure 8.8 for $\beta_H/\beta_L = 1/4$ show excellent accuracy of $Approx_3$ with errors below 3%. However, the results of Figure 8.9 for $\beta_H/\beta_L = 4$ show that the ratio of the coefficients of variation affects the performance of $Approx_3$. For $c_H^2 < c_L^2$, $Approx_3$ continues to perform very well with errors below 4%, but for $c_H^2 > c_L^2$ the error can rise to 25% for large load per server values. Therefore, a combination of $\beta_H > \beta_L$ and $c_H^2 > c_L^2$ is clearly a weakness of $Approx_3$.

Figures 8.10 and 8.11 show the exact and approximated mean sojourn times for low priority traffic for $C = 4$ and 20 respectively with different load per server, coefficients of variations with $c_H^2 = c_L^2$, $\beta_H = 1/4$ and $\beta_L = 1$. It is important to note that when $c_H^2 = c_L^2$, $Approx_3$ is also insensitive to the exact values of the coefficients of variations. The conclusions which can be drawn from these cases are similar to the exponential distribution case. $Approx_1$ and $Approx_2$ do not perform well. $Approx_3$ on the contrary, performs very well with maximum errors of about 5% under high load situations.

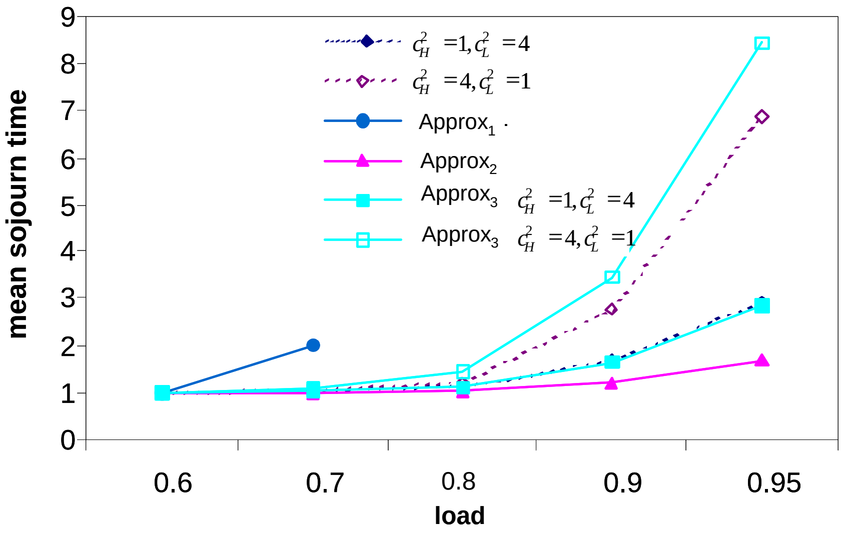


Figure 8.9: Exact and approximated values of the average sojourn times for $C = 20$, $\beta_H = 4, \beta_L = 1$ of various values of load per server.

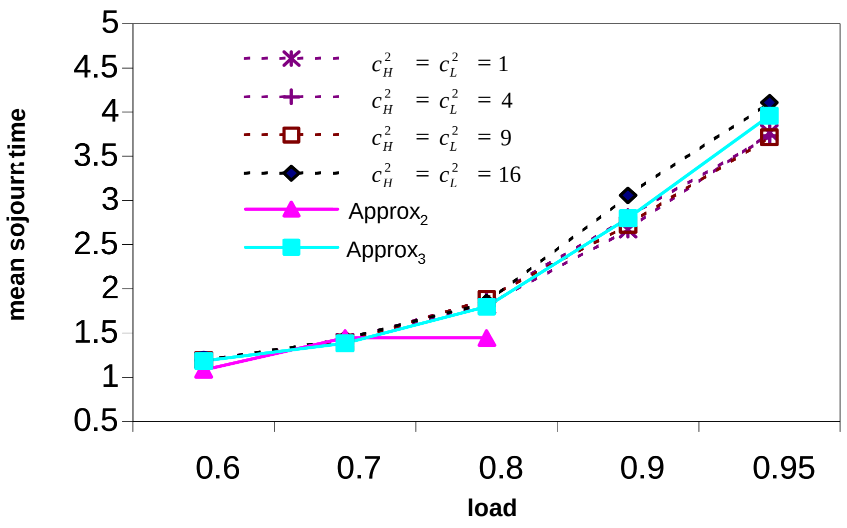


Figure 8.10: Exact and approximated mean sojourn times for low priority traffic for different values of load per server, coefficient of variations, with $\beta_H/\beta_L = 1/4$ and $C = 4$.

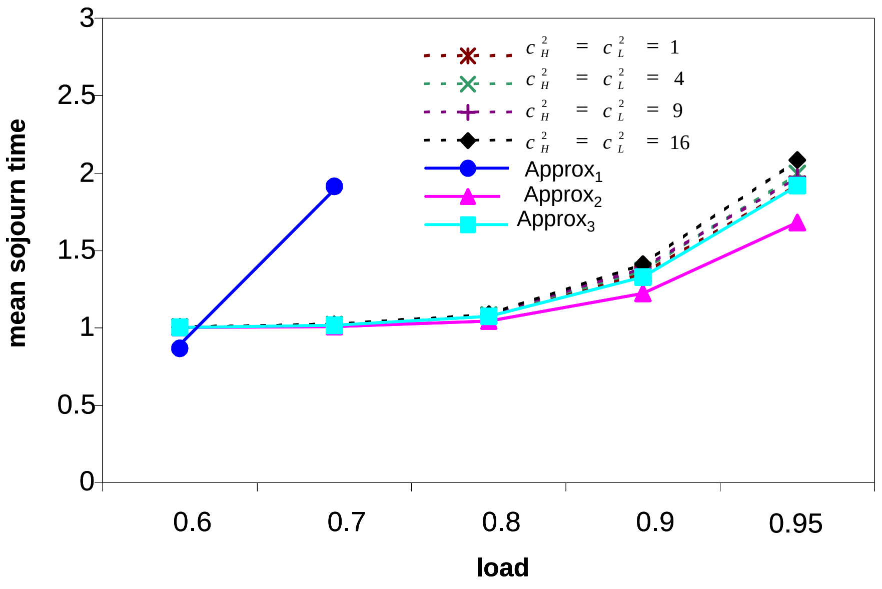


Figure 8.11: Exact and approximated mean sojourn times for low priority traffic for different values of load per server, coefficient of variations, with $\beta_H/\beta_L = 1/4$ and $C = 20$.

In Figures 8.12 and 8.13 we consider various coefficients of variation with $c_H^2 = c_L^2$, $\beta_H > \beta_L$ for $C = 4$, and 20 respectively. The performances of $Approx_1$ and $Approx_2$ are as before. $Approx_2$, however performs better in the case $\beta_H/\beta_L = 1/4$ with $C = 20$ than with $\beta_H/\beta_L = 4$. $Approx_3$ still performs the best but its performance is worse as compared to the case with $\beta_H/\beta_L = 1/4$. For $C = 4$, the errors are very high, but with $C = 10$ the errors are at a maximum of 11%. For $C = 20$ the performance is almost as same as with $\beta_H/\beta_L = 1/4$ with maximum errors of only 6%.

8.5 Conclusions

In this chapter we have proposed a model and approximations to quantify the trade-off between control and performance of elastic traffic when integrated with prioritized stream traffic. We defined a simple model with two priority classes, where the stream traffic class is served with higher priority over the elastic traffic class. It is assumed that both stream and elastic traffic have the same peak rate. Stream traffic which is admitted is served at its peak rate whereas the lower priority elastic traffic is served with processor sharing. We used a simple control strategy for the stream traffic, i.e., the streams are admitted and served at their peak rate as long as there is capacity in the network; else they are blocked and dropped. The elastic traffic takes advantage of the fluctuations in the number of high priority streams. The arrival of both traffic

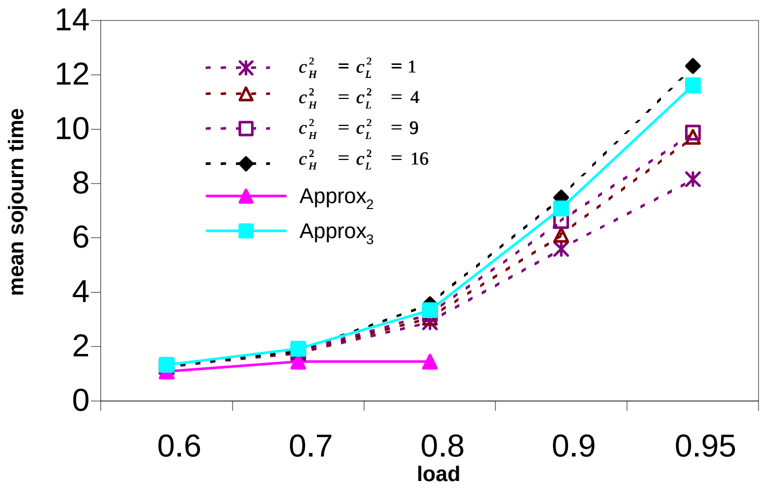


Figure 8.12: Exact and approximated mean sojourn times for low priority traffic for different values of load per server, coefficients of variations with $\beta_H/\beta_L = 4$ and $C = 4$.

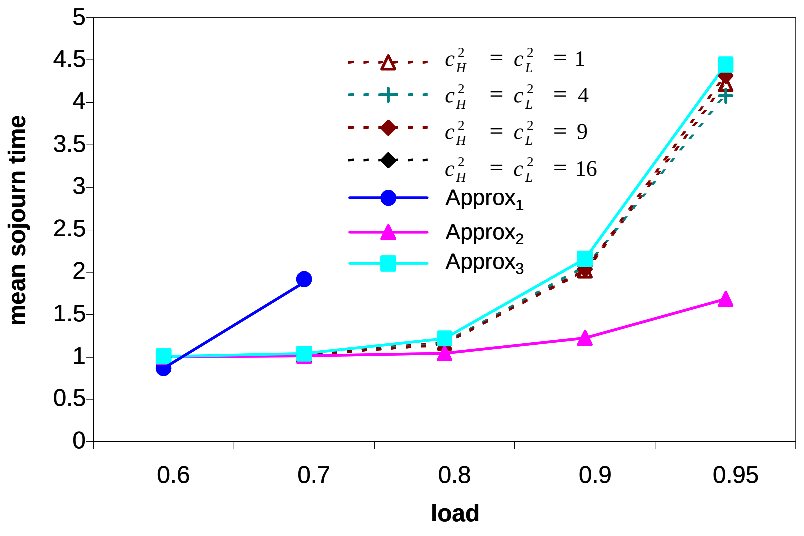


Figure 8.13: Exact and approximated mean sojourn times for different load per server values and coefficients of variation with $c_H^2 = c_L^2$, $C = 20$ and $\beta_H/\beta_L = 4$.

types was assumed to be Poisson, but their service time requirement distribution could be any generic distribution.

The model presented cannot be solved explicitly. To overcome this problem, we proposed an approximation which on one hand is simple and on the other hand is detailed enough to capture the influence of a wide range of parameters. The accuracy of the approximation was compared to simulations of the model as well as other simple approximation techniques. One of the alternative approximations was based on quasi-stationarity and the other simply on estimation of average left over capacity for the low priority elastic traffic.

The performance results support the conclusion that for all the cases and parameters considered our approximation works best and is a considerable improvement over the other known ways of performance approximations. However, our proposed approximation also has its weak points. It performs worst for the combination of several factors: small values of total capacity C, high load per server, service time requirement of stream traffic a lot larger than that of the lower priority elastic traffic ($\beta_H > \beta_L$) and high variation in high priority traffic service requirement distribution especially with $c_H^2 > c_L^2$.

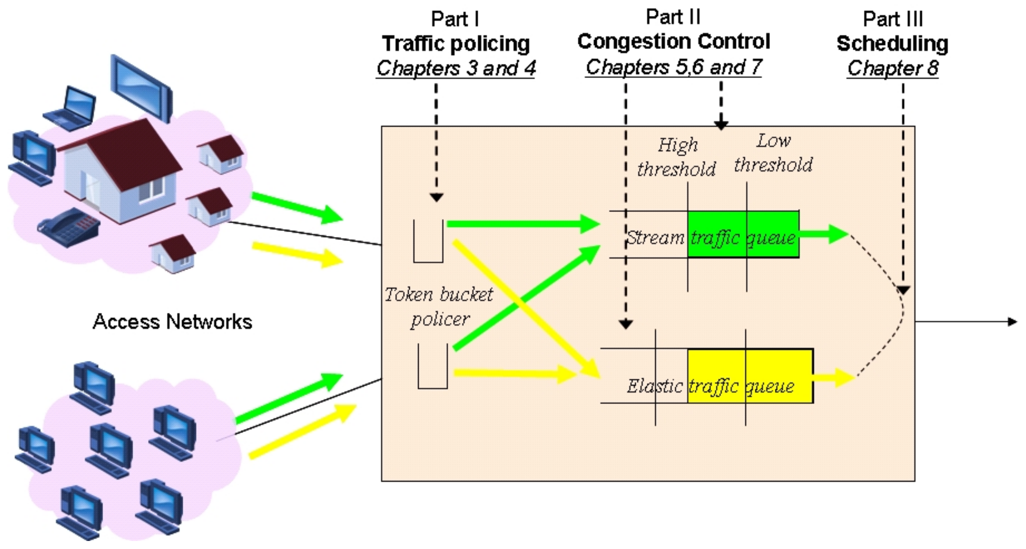
It might seem that in our proposed approximation, the assumption of equal peak rate for both stream and elastic traffic is very restrictive and not always true in practice. However, it is important to note that the case with high load combined with the fact that all available capacity can be occupied by high priority traffic implies that the effective peak rate achieved by low priority elastic traffic will never be the same as for stream traffic. This fact indicates that the error with different peak rates for high and low priority traffic should be comparable to the results presented in this chapter.

Concluding remarks

In this thesis, we have addressed three key QoS mechanisms for Carrier Ethernet networks: traffic policing, congestion control and scheduling.

Traffic policing is the first hurdle that traffic entering an Ethernet public network faces. It is not sufficient to monitor and penalize the incoming traffic to conform to a strict traffic contract. It is equally important to understand the effect of this mechanism on different traffic types and their performance. We have shown that if policing is done without keeping into account the characteristics of the higher layer applications the resulting throughput can be significantly lower than the contractual rate. In Part I of this thesis we have presented and analyzed two policing methods, the first based on an Ethernet backpressure warning signal and the second based on a dynamic token bucket. The warning signal to the customer results in improved throughput performance for TCP traffic through buffering and smoothening of packet transmissions. For UDP traffic this works well if this traffic type is given priority or by using the extensions proposed in [29]. The dynamic token bucket policer improves TCP performance by adapting to different traffic profiles caused by differences in round trip times, link capacities, policing rates and number of TCP flows simultaneously in progress. Results indicate that in most cases the dynamic bucket converges to the optimal token bucket size required specific to the scenario considered.

Having eliminated the non-optimal interactions of higher layer traffic with a policer, we addressed the congestion problems which could arise in nodes further in the network. This congestion could be caused by the inherent unpredictable nature of user traffic as well as by the aggregation and multiplexing of the different policed flows. In Part II of the thesis we have analyzed the use of feedback flow control as a way to manage this congestion. This possibility is provided by the standard IEEE 802.3x backpressure method or possibly one of its variants proposed in literature. The strength of our analysis lies in the fact that we provide insightful analytical relations between the congestion detection threshold settings and performance of higher layer, stream and elastic application traffic. For example, the transfer time of a data file and expected packet delay for real-time streaming applications can be computed for a given choice of thresholds. With an extensive numerical study of the proposed model we have demonstrated how the thresholds can be configured to



achieve the desired trade-offs between signaling frequency, throughput and delay. We have also shown that the benefit of using the backpressure method is especially evident for TCP data traffic with large round trip times, extreme burstiness or low to moderate traffic loads.

Having addressed the policing and congestion control issues for TCP elastic and UDP stream traffic separately in Parts I and II, we addressed the integration of these two traffic types in Part III of the thesis. There we designed a model where elastic traffic is handled with lower priority than stream traffic. Furthermore, traffic control is exercised on the stream traffic. The approximation proposed is as desired, simple, yet quite accurate in predicting the performance of elastic traffic flows in relation to the traffic load and blocking probability of the prioritized stream traffic. Accuracy of the approximation was assessed for different values of the total available capacity, system load and service requirement of stream and elastic traffic. For all the cases and parameters considered we have shown that our approximation is a considerable improvement over the other known ways of performance approximations such as a quasi-stationarity based method.

The figure above summarizes our contributions and their applicability to a typical Carrier Ethernet ingress node operating in a metropolitan area network. In an egress or core node, the traffic policing function is absent and only the results from Part II and Part III apply.

Future Research

In this section, we suggest some possibilities to continue the research presented in this thesis. The topics for future work specific to each model and/or mechanism analyzed in this thesis have already been discussed in the respective chapters. In this section we suggest some overall directions for future research spanning all the chapters of this thesis.

- *Interaction of QoS mechanisms*: The traffic policing, congestion control and scheduling mechanisms have been designed and analyzed separately in this thesis. A logical next step to this research is to study the interaction of these mechanisms in a single framework and include other QoS mechanisms as well. It would be worthwhile to assess if the analysis and guidelines we have proposed are also valid when these mechanisms coexist.

- *Multiple node scenarios*: In this thesis we have analyzed the proposed mechanisms for a single node or two nodes in tandem. The scalability of our models to multiple nodes in a network should be assessed with further work. This study could be considerably complicated by the presence of multiple congestion points in the network.

- *Differentiation in performance*: Another very important area for future research is the differentiation in performance achieved by multiple customers. In practice, different customers will have different bandwidth demands and hence, a different SLA. The QoS mechanisms should not only ensure that the guaranteed traffic rates are met, but also that the left-over capacity is divided fairly among all customers. This should ideally be done in proportion to their contractual traffic rates or in accordance with the billing or pricing strategies of the service provider.

We would like to conclude with some final remarks on the possibility to exploit the results of this thesis in practice. Although a topic more for development rather than research, it is useful that our results be incorporated in a network design tool to enable the operator to better plan his network. Some issues would have to be resolved before this can be realized, for example, the estimation of certain input parameter values in the proposed models and approximations. Furthermore, the research items previously mentioned in this section, should first have been investigated. The envisioned design tool could then be integrated with the management plane or used stand-alone.

Samenvatting

Ethernet, dat was ontworpen voor LANs, evolueert momenteel naar een carrier-grade technologie. Van dit zogenaamde "Carrier Ethernet" wordt verwacht dat het aan de voornaamste tekortkomingen van het conventionele Ethernet in publieke netwerken tegemoet kan komen. De visie is dat uiteindelijk Ethernet de services van een netwerk op een end-to-end-basis verzorgt. Hierbij denkt men in het bijzonder aan de ondersteuning van zakelijke datanetwerken, breedbandige access en het "backhaulen" van draadloos verkeer.

Met de opmars van Ethernet in het publieke domein speelt de kwaliteit van de aangeboden diensten (de Quality of Service, of kortweg QoS) een steeds grotere rol, in die zin dat het een onderscheidende factor tussen de verschillende dienstverleners wordt. De uitdaging is te voldoen aan de QoS-eisen van de applicaties (die doorgaans zijn geformuleerd in termen van responstijden, throughput, vertraging en jitter) door de netwerkresources adequaat in te zetten. Ethernet is in principe niet ontworpen voor grote publieke netwerken, en het beschikt niet over direct toepasbare functionaliteit om QoS te bieden. In dit verband stellen wij in dit proefschrift een aantal QoS-mechanismen voor en analyseren deze in kwantitatieve termen. Het primaire doel hierbij is te komen tot mechanismen die een dienstverlener (netwerk operator) in staat stellen te voldoen aan de kwaliteitseisen van huidige en toekomstige diensten.

In dit proefschrift hebben wij drie belangrijke QoS-mechanismen bekeken: policing, congestion control en scheduling van netwerkverkeer.

Policing is de eerste 'horde' die het verkeer moet nemen wanneer het het op Ethernet gebaseerde netwerk binnen komt. De taak van het policing mechanisme is het inkomende verkeer te monitoren en te limiteren, zodat het voldoet aan de afspraken die zijn vastgelegd in een verkeerscontract. Het is echter ook van groot belang om het effect van de policing methode op verschillende verkeerstypen te begrijpen. Wij hebben aangetoond dat als policing wordt uitgevoerd zonder rekening te houden met de kenmerken van de hogere-laag applicaties, de uiteindelijke throughput beduidend lager kan zijn dan de in het contract overeengekomen 'gegarandeerde' throughput. In Deel I van dit proefschrift stellen wij twee policing-methodes voor en analyseren deze; de eerste methode is gebaseerd op een waarschuwingssignaal dat het Ethernet backpressure mechanisme uitzendt, en de tweede op een dynamische token bucket.

Het waarschuwingssignaal naar de klant resulteert in verbeterde throughput voor TCP-verkeer door het bufferen en het 'gladstrijken' van het verkeer tot een gelijkmatige stroom. Dit werkt goed voor UDP-verkeer als dit verkeerstype met prioriteit wordt afgehandeld. De dynamische token bucket policer verbetert de performance van TCP-verkeer door zich aan te passen aan verschillende verkeersprofielen. De resultaten geven aan dat in de meeste gevallen de dynamische bucket convergeert naar de optimale token bucket, die specifiek is voor het scenario dat beschouwd wordt.

Na het elimineren van de niet optimale interacties van het hogere laag verkeer met een policer, hebben wij ons geconcentreerd op de congestieproblemen die kunnen ontstaan in nodes elders in het netwerk. Deze congestie kan veroorzaakt worden door de inherent onvoorspelbare aard van het gebruikersverkeer, evenals het samenvoegen van de verschillende verkeersstromen. In Deel II van het proefschrift hebben wij het gebruik van feedback voor congestion control geanalyseerd, als manier om deze congestie te beheersen. Een dergelijke feedback wordt gefaciliteerd door het standaard IEEE 802.3x backpressure mechanisme of één van de alternatieven die in de literatuur worden voorgesteld. De kracht van onze analyse ligt in het feit dat wij inzichtelijke, analytische relaties bieden tussen de parameter instellingen van het congestie detectie mechanisme aan de ene kant, en de performance van de hogere laag applicaties aan de andere kant. De transfer tijd van een gegevensbestand en de verwachte pakketvertraging voor real-time streaming applicaties kan bijvoorbeeld, voor gegeven parameterwaarden, berekend worden. Door middel van een uitvoerige en gedetailleerde numerieke studie van dit model, hebben wij procedures ontwikkeld waarmee de parameters geconfigureerd kunnen worden, met als doelstelling het gewenste compromis tussen de signalerings-frequentie, throughput en vertraging te bereiken.

De kwesties op het gebied van policing en congestion control voor elastisch TCP verkeer en streaming UDP verkeer zijn afzonderlijk behandeld in Deel I en II; de integratie van deze twee verkeerstypen staat centraal in Deel III van dit proefschrift. Daar beschouwen we een situatie waarbij elastisch verkeer een lagere prioriteit heeft dan streaming verkeer en er op het streaming verkeer admission control wordt uitgeoefend. De voorgestelde modellerings en analyse aanpak leidt, zoals gewenst, tot relatief eenvoudige benaderingsformules voor het voorspellen van de performance zoals die ervaren wordt door het elastische verkeer. De nauwkeurigheid van de benadering wordt onderzocht voor verschillende waarden van de beschikbare capaciteit, systeembelasting en de QoS-eisen van het streaming en elastisch verkeer. Voor alle beschouwde scenario's hebben wij aangetoond dat onze benadering een aanzienlijke verbetering te zien geeft ten opzichte van de andere bekende benaderingen, zoals een op quasi-stationariteit gebaseerde methode.

De onderzoeksresultaten uit dit proefschrift vormen een basis voor de ontwikkeling van een tool voor netwerkontwerp, waarmee de operator zijn netwerk efficiënt kan plannen, en uitspraken kan doen over de geboden QoS. Een aantal kwesties vragen dan eerst nog wel een oplossing, zoals de schatting van de waarden van de input-parameters in de voorgestelde modellen en benaderingen. Een dergelijk tool voor netwerkontwerp kan daarna in de management plane worden geïntegreerd, of stand-alone worden gebruikt.

Acronyms

ABR	Available Bit Rate
ACK	Acknowledgement
AIMD	Additive Increase Multiplicative Decrease
ATM	Asynchronous Transfer Mode
CIR	Committed Information Rate
CSMA/CD	Carrier Sense Multiple Access with Collision Detection
ECN	Explicit Congestion Notification
GPS	Generalized Processor Sharing
IEEE	Institute of Electrical and Electronics Engineers
IP	Internet Protocol
ISP	Internet Service Provider
ITU	International Telecommunications Union
LAN	Local Area Network
MAC	Medium Access Control
MAN	Metropolitan Area Network
MEF	Metro Ethernet Forum
MEN	Metropolitan Ethernet Network
MPLS	Multiprotocol Label Switching
MTU	Maximum Transfer Unit
OAM	Operation Administration and Management

PASTA Poisson Arrivals See Time Averages

PBS Peak Burst Size

PC Personal Computer

PIR Peak Information Rate

PQ Priority Queueing

PS Processor Sharing

QoS Quality of Service

RED Random Early Detection

RFC Request For Comments

RM Resource Management

RTT Round Trip Time

SDH Synchronous Digital Hierarchy

SLA Service Level Agreement

SONET Synchronous Optical Networking

STP Spanning Tree Protocol

TCP Transport Control Protocol

TDM Time Division Multiplexing

UDP User Datagram Protocol

VLAN Virtual Local Area Network

VoIP Voice over IP

VPN Virtual Private Network

WDM Wavelength Division Multiplexing

WFQ Weighted Fair Queueing

Bibliography

[1] The network simulator ns-2. `http://www.isi.edu/nsnam/ns/`.

[2] OMNeT++. `http://www.omnetpp.org/`.

[3] I. Adan, E. van Doorn, J. Resing, and W. Scheinhardt. Analysis of a single server queue interacting with a fluid reservoir. *Queueing Systems*, 29:313–336, 1998.

[4] S. Ahn and V. Ramaswami. Steady state analysis of finite fluid flow models using finite QBDs. *Queueing Systems*, 49:223–259, 2005.

[5] D. Anick, D. Mitra, and M. Sondhi. Stochastic theory of a data-handling system with multiple sources. *Bell System Technical Journal*, 61:1871–1894, 1982.

[6] F. Baccelli and D. Hong. The AIMD model for TCP sessions sharing a common router. In *Proceedings of the 39th Annual Allerton Conference on Communication, Control and Computing*, 2001.

[7] R. Baumann and U. Fiedler. Why QoS will be needed in Metro Ethernets. In *Proceedings of the International workshop on Quality of Service 2005 (IWQoS)*, pages 379–381, 2005.

[8] D. Bergamasco and R. Pan. Backward congestion notification version 2.0. *IEEE 802.1 Meeting*, September 2005.

[9] T. Bonald and J. Roberts. Performance of bandwidth sharing mechanisms for service differentiation in the Internet. In *Proceedings of the 13th ITC Specialist Seminar on IP Traffic Measurement, Modeling and Management*, pages 22–1–22–10, 2000.

[10] O. Boxma, H. Kaspi, O. Kella, and D. Perry. On/Off storage systems with state dependent input, output and switching rates. *Probability in the Engineering and Informational Sciences*, 19:1–12, 2005.

[11] R. Cavendish. Operation, administration, and maintenance of Ethernet services in wide area networks. *IEEE Communication Magazine*, 42(3):72–79, 2004.

[12] Y-C. Chen and X. Xu. An adaptive buffer allocation mechanism for token bucket flow control. In *Proceedings of the IEEE 60th Vehicular Technology Conference (VTC)*, volume 4, pages 3020–3024, 2004.

[13] J.W. Cohen. The multiple phase service network with generalized processor sharing. *Acta Informatica*, 12:245–284, 1979.

[14] J. Crowcroft and P. Oechslin. Differentiated end-to-end Internet services using a weighted proportional fair sharing TCP. *Computer Communication Review*, 28:53–69, 1998.

[15] A. da Silva Soares and G. Latouche. A matrix-analytic approach to fluid queues with feedback control. *International Journal of Simulation Systems Science and Technology*, 6:4–12, 2005.

[16] A. da Silva Soares and G. Latouche. Matrix-analytic methods for fluid queues with finite buffers. *Performance Evaluation*, 63:295–314, 2006.

[17] F. Delcoigne, A. Proutière, and G. Régnié. Modelling integration of streaming and data traffic. *Performance Evaluation*, 55:185–209, 2004.

[18] V. Dumas, F. Guillemin, and P. Robert. A Markovian analysis of Additive-Increase Multiplicative-Decrease (AIMD) algorithms. *Advances in Applied Probability*, 34(1):85–111, 2002.

[19] S. Elby, H. Chamas, W. Bjorkman, and V. Alesi. Carrier Ethernet: A reality check. In *Optical Fiber Communication and the National Fiber Optic Engineers Conference, OFC/NFOEC*, pages 1–6, 2007.

[20] Metro Ethernet Forum (MEF). A global industry alliance. `http://metroethernetforum.org/`.

[21] Metro Ethernet Forum (MEF). Certification program. Available at, `http://metroethernetforum.org/Certification`.

[22] O. Feuser and A. Wenzel. On effects of the IEEE 802.3x flow control in full-duplex Ethernet LANs. In *Proceedings of the 24th Conference on Local Computer Networks*, page 160, 1999.

[23] V. Fineberg. QoS support in MPLS networks. *MPLS/Frame Relay alliance White Paper*, 2003.

[24] W. Fischer and K. Meier-Hellstern. The Markov-modulated Poisson process (MMPP) cookbook. *Performance Evaluation*, 18:149–171, 1993.

[25] S. Floyd. TCP and explicit congestion notification. *ACM Computer Communication Review*, 24:10–23, 1994.

[26] S. Floyd. A report on recent developments in TCP congestion control. *IEEE Communications Magazine*, 39(4):84–90, 2001.

[27] S. Floyd and V. Jacobson. Congestion gateways for packet networks. *IEEE/ACM Transactions on Networking*, 1:397–413, 1993.

[28] D. Gaver and J. Lehoczky. Channels that cooperatively service a data stream and voice messages. *IEEE Transactions on Communications*, 30(5):1153–1162, 1982.

[29] A. Ge and G. Chiruvolu. DiffServ compatible extended pause (DiffPause) for fair congestion control in Metro-Ethernet. In *Proceedings of the IEEE International Conference on Communications (ICC)*, volume 2, pages 1248–1252, 2004.

[30] M. Gribaudo and M. Telek. Fluid models in performance analysis. In *Proceedings of the 7th International School on Formal Methods for the Design of Computer, Communication, and Software Systems, (SFM)*, pages 271–317, 2007.

[31] M. Gribaudo and M. Telek. Stationary analysis of fluid level dependent bounded fluid models. *Performance Evaluation*, 65:241–261, 2007.

[32] J. Heinanen, T. Finland, and R. Guerin. The rate control RFC: RFC2698: A two rate three color marker. *IETF RFC2698*, `http://www.faqs.org/rfcs/rfc2698.html`, 1999.

[33] M. Howard. Metro Ethernet equipment biannual worldwide market forecast and equipment: 1st edition. *Market Report*, April 2007.

[34] M. Howard. Service provider routers and switches: IP, Ethernet, and ATM. *Quarterly Market Report*, May 2007.

[35] L. Hui-Lan and I. Faynberg. An architectural framework for support of quality of service in packet networks. *IEEE Communications Magazine*, 41(6):98–105, 2003.

[36] J. Jiang and R. Jain. Analysis of Backward Congestion Notification (BCN) for Ethernet in datacenter applications. In *Proceedings of the 26th IEEE International Conference on Computer Communications, INFOCOM*, pages 2456–2460, 2007.

[37] J. Jiang and R. Jain. Forward Explicit Congestion Notification (FECN) for data center Ethernet networks. *IEEE 802.1au*, Interim meeting, Monterey, CA, January 2007.

[38] S. Jung, J. Kwak, and O. Byeon. Performance analysis of queue scheduling mechanisms for EF PHB and AF PHB in Diffserv networks. In *Proceedings of the 5th IEEE International Conference on High Speed Networks and Multimedia Communications*, pages 101–104, 2002.

[39] K. Kawahara, Y. Oie, M. Murata, and H. Miyahara. Performance analysis of backpressure congestion control: preliminary case. *IEEE GLOBECOM*, 1:304–309, 1995.

[40] J. Kidambi, D. Ghosal, and B. Mukherjee. Dynamic Token Bucket (DTB): a fair bandwidth allocation algorithm for high-speed networks. *Journal of High Speed Networks*, 9(2):67–87, 2000.

[41] L. Kosten. Stochastic theory of a data handling systems with groups of multiple sources. *Performance of Computer Communication Systems*, pages 321–331, 1984.

[42] S. Kumar and L. Massoulié. Integrating streaming and file-transfer Internet traffic: fluid and diffusion approximations. *Queueing Systems: Theory and Applications*, 55(4):195–205, 2007.

[43] M. MacFarland, S. Salam, and R. Checker. Ethernet OAM : key enabler for carrier class Metro Ethernet services. *IEEE Communication Magazine*, 43(11):152–157, 2005.

[44] R. Malhotra, M.R.H. Mandjes, W.R.W. Scheinhardt, and J.L. van den Berg. Design issues of a backpressure based congestion control mechanism. *Submitted*, 2008.

[45] R. Malhotra, M.R.H. Mandjes, W.R.W. Scheinhardt, and J.L. van den Berg. A fluid queue with two congestion control thresholds. *To appear in: Mathematical Methods of Operations Research*, 2008.

[46] R. Malhotra and J.L. van den Berg. Flow level performance approximations for elastic traffic integrated with prioritized stream traffic. In *Proceedings of the 12th International Telecommunications Network Strategy and Planning Symposium, NETWORKS*, pages 1–9, 2006.

[47] R. Malhotra, R. van Haalen, R. de Man, and M. van Everdingen. Managing service level agreements in Metro Ethernet networks using backpressure. *Bell Labs Technical Journal*, 8(2):83–95, 2003.

[48] R. Malhotra, R. van Haalen, M.R.H. Mandjes, and R. Núñez-Queija. Modeling the interaction of IEEE 802.3x flow control and TCP end-to-end flow control. In *Proceedings of Euro NGI 1st Conference on Next Generation Internet Networks: Traffic Engineering*, pages 260–267, 2005.

[49] M. Mandjes, D. Mitra, and W. Scheinhardt. Models of network accesssing feedback fluid queues. *Queueing Systems*, 44:365–398, 2003.

[50] G. McAlpine, M. Wadekar, T. Gupta, A. Crouch, and D. Newell. An architecture for congestion management in Ethernet clusters. In *Proceedings of the 19th IEEE International Parallel and Distributed Processing Symposium (IPDPS'05) - Workshop 9*, page 211.1, 2005.

[51] D. Mendes, K. Fonseca, and C.M. Pedroso. Bandwidth fairness of a single rate three color marker algorithm implementation. In *Proceedings of the 8th International Conference on Communication Systems (ICCS)*, pages 609–611, 2002.

[52] P. Mishra and H. Kanakia. A hop by hop based congestion control scheme. In *Proceedings of Communications Architectures and Protocols*, pages 112–123, 1992.

[53] R. Núñez-Queija. Sojourn times in a processor sharing queue with service interruptions. *Queueing Systems*, 34:351–386, 2000.

[54] R. Núñez-Queija, H. van den Berg, and M. Mandjes. Performance evaluation of strategies for integration of elastic and stream traffic. In *Proceedings of ITC 16, Edinburgh*, pages 1–17, 1999.

[55] W. Noureddine and F. Tobagi. Selective back-pressure in switched Ethernet LANs. In *Proceedings of IEEE GLOBECOM*, pages 1256–1263, 1999.

[56] W. Noureddine and F. Tobagi. Selective back-pressure in switched Ethernet LANs. In *Proceedings of the Global Telecommunications Conference 2, GLOBECOM*, pages 1256–1263, 1999.

[57] C.G. Omidyar and G. Pujolle. Guest editorial - Introduction to flow and congestion control. *IEEE Communications Magazine*, 34(11):30–32, 1996.

[58] J. Padhye, V. Firoiu, D. Towsley, and J. Kurose. Modeling TCP throughput: A simple model and its empirical validation. In *Proceedings of ACM SIGCOMM*, pages 303–314, 1998.

[59] C.M. Pazos and M. Gerla. A rate based backpressure flow control for the Internet. In *Proceedings of High Performance Networking (HPN)*, pages 555–573, 1998.

[60] C.M. Pazos, J.C. Sanchez-Agrelo, and M. Gerla. Using back-pressure to improve TCP performance with many flows. In *Proceedings of IEEE INFOCOM*, pages 431–438, 1999.

[61] K. Perumalla. Libsynk. `http://www.cc.gatech.edu/fac/kalyan/libsynk.htm`.

[62] J.B. Pipas, I.S. Venieris, and J.-A. Sanchez-Papaspiliou. On the extension of ABR flow control to legacy LANs. In *Proceedings of the IEEE International Conference on Communications (ICC)*, pages 832–837, 1997.

[63] B. Rahemi, G. Chiruvolu, A. Ge, and M. Ali. Metro Ethernet quality of services. *Alcatel Telecommunications Review*, December,2004.

[64] K. Ramanan and A. Weiss. Sharing bandwidth in ATM. *Proceedings of the Allerton Conference*, pages 732–740, 1997.

[65] ITU-T Recommendations. Y.1731, OAM functions and mechanisms for Ethernet based networks. 2006.

[66] J.F. Ren and R. Landry. Flow control and congestion avoidance in switched Ethernet LANs. In *Proceedings of the IEEE International Conference on Communications (ICC)*, pages 508–512, 1997.

[67] J. Roberts. *Engineering for Quality of Service.* Chapter 16, Wiley-Interscience, UK, 2000.

[68] J.W. Roberts and L. Massoulié. Bandwidth sharing and admission control for elastic traffic. In *Proceedings of the ITC Specialist Seminar*, Yokohama 1998.

[69] L. Rogers. Fluid models in queueing theory and Wiener-Hopf factorization of Markov chains. *Annals of Applied Probability*, 4:390–413, 1994.

[70] S. Sahu, P. Nain, D. Towsley, C. Diot, and V. Firoiu. On achievable service differentiation with token bucket marking for TCP. In *Proceedings of the 2000 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems*, pages 23–33, 2000.

[71] S. Salam and A. Sajassi. Provider backbone bridging and MPLS: complementary technologies for next-generation carrier Ethernet transport. *IEEE Communications Magazine*, 46(3):77–83, 2008.

[72] W. Scheinhardt, N. van Foreest, and M. Mandjes. Continuous feedback fluid queues. *Operations Research Letters*, 33:551–559, 2005.

[73] IEEE Std 802.1ad 2005. IEEE standard for local and metropolitan area networks - virtual bridged local area networks, amendment 4: Provider bridges. *IEEE Std 802.1ad-2005 (Amendment to IEEE Std 802.1Q-2005)*, 2005.

[74] IEEE Std 802.1D 2004. IEEE standard for local and metropolitan area networks - Media Access Control (MAC) bridges. *Revised version incorporating existing published amendments 802.1t and 802.1w*, 2004.

[75] IEEE Std 802.1Q 2005. IEEE standard for local and metropolitan area networks - virtual bridged local area networks – revision. 2005.

[76] IEEE Std 802.3. Carrier Sense Multiple Access with Collision Detection (CSMA/CD) access method and physical layer specification. *Annex 31 B*, 1998 Edition.

[77] R.W. Stevens. *TCP/IP Illustrated*. Addison-Wesley, 1994.

[78] H.C. Tijms. *Stochastic Modeling and Analysis*. Wiley, New York, 1986.

[79] R. van der Mei, J.L. van den Berg, R. Vranken, and B.M.M. Gijsen. Sojourn times approximations for a multi-server processor sharing system with priorities. *Performance Evaluation*, 54:249–261, 2003.

[80] E. van Doorn, A. Jagers, and J. de Wit. A fluid reservoir regulated by a birth death-process. *Stochastic Models*, 4:457–472, 1988.

[81] M. van Everdingen. Method and device for controlling source specific data flow. *European Patent EP1187399*.

[82] N. van Foreest, M. Mandjes, and W. Scheinhardt. A versatile model for asymmetric TCP sources. In *Proceedings of the 18th International Teletraffic Congress*, pages 631–640, 2000.

[83] R. van Haalen and R. Malhotra. Improving TCP performance with bufferless token bucket policing: A TCP friendly policer. In *Proceedings of the 15th IEEE workshop on Local and Metropolitan Area Networks (LANMAN)*, pages 72–77, 2007.

[84] R. van Haalen, R. Malhotra, and A. de Heer. Optimized routing for providing Ethernet LAN services. *IEEE Communications Magazine*, 43(11):158–164, 2005.

[85] J. Wechta, A. Eberlein, and F. Halsall. The interaction of the TCP flow control procedure in the end nodes on the proposed flow control mechanism for use in IEEE 802.3x switches. In *Proceedings of the Eigth IFIP Conference on High Performance Networking (HPN)*, pages 515–534, 1998.

[86] J. Wechta, M. Fricker, and F. Halsall. Hop-by-hop flow control as a method to improve QoS in 802.3 LANs. In *Proceedings of the IEEE/IFIP International Workshop on Quality of Service (IWQoS)*, pages 239–247, 1999.

[87] Y.R. Yang and S.S. Lam. General AIMD congestion control. In *Proceedings of the International Conference on Network Protocols*, pages 187–198, 2000.

[88] S.F. Yashkov. Processor-sharing queues: some progress in analysis. *Queueing Systems*, 2(1):1–17, 1987.

[89] S.F. Yashkov. Mathematical problems in the theory of processor-sharing queueing systems. *Journal of Soviet Mathematics*, 58:101–147, 1992.

[90] I. Yeom and A.L. Narasimha Reddy. Realizing throughput guarantees in a differentiated services network. In *Proceedings of the IEEE International Conference on Multimedia and Computing Systems*, pages 372–376, 1999.

Publications and patents by the author

Publications

- R. Malhotra, M.R.H. Mandjes, W.R.W. Scheinhardt, J.L. van den Berg, "Design issues of a Ethernet backpressure flow control mechanism". Submitted, 2008.

- R. Malhotra, M.R.H. Mandjes, W.R.W. Scheinhardt, J.L. van den Berg, "A feedback fluid queue with two congestion control thresholds". To appear in Mathematical Methods of Operations Research, 2008.

- R. Malhotra, M.R.H. Mandjes, W.R.W. Scheinhardt, J.L. van den Berg, "A feedback fluid queue with two congestion control thresholds". Presented at the Fourteenth INFORMS Applied Probability Conference, July 2007.

- R. van Haalen, R. Malhotra, "Improving TCP performance with a dynamic token bucket policer: A TCP friendly policer". In Proceedings of the 15th IEEE Workshop on Local and Metropolitan Area Networks, LANMAN, 2007.

- R. Malhotra, J.L. van den Berg, "Flow level performance approximations for elastic traffic integrated with prioritized stream traffic". In Proceedings of the 12th International Telecommunications Network Strategy and Planning Symposium, Networks, 2006, pp 1-9.

- R. van Haalen, R. Malhotra, A. de Heer, "Optimized routing of Ethernet LAN services". IEEE Communications Magazine, vol 43 (11), 2005, pp 158-164.

- R. Malhotra, R. van Haalen, M.R.H. Mandjes, R. Núñez Queija, "Modeling the interaction of IEEE 802.3x flow control and TCP end-to-end flow control". In Proceedings of Euro NGI 1st Conference on Next Generation Internet Networks: Traffic Engineering, Rome, April 2005, pp 260-267.

- R. Malhotra, R. van Haalen, R. de Man, M. van Everdingen, “Managing service level agreements in Metro Ethernet networks using backpressure”. Bell Labs Technical Journal, vol 8(2), 2003, pp 83-95.

- R. van Haalen, R. Malhotra, R. de Man, M. van Everdingen, “Back-pressure based traffic policing mechanism for Metro Ethernet networks”. In Proceedings of the 12th IEEE Workshop on Local and Metropolitan Area Networks, LANMAN, Stockholm, August 2002, pp 217-220.

- R. Malhotra, D. Dey, E.A. van Doorn, A.M.J. Koonen, “Traffic modeling in a reconfigurable broadband nomadic computing environment”. Performance Evaluation, vol 47, 2002, pp 255-267.

- R. Malhotra, P. Busch, “Dynamic channel selection for wireless LANs”. In Proceedings of the Joint International Conference on Wireless LANs and Home Networks (ICHLWN 2002) and Networking (ICN 2002), Networks, August 2002, pp 3-14.

- R. Malhotra, D. Dey, E.A. van Doorn, A.M.J. Koonen, “ Traffic modelling in a reconfigurable broadband nomadic computing environment”. In Proceedings of SPIE 4211, Internet Quality and Control of Network Systems, Nov 2000, pp 82-92.

Patents issued

- R. Malhotra, P. Busch, “Dynamic channel selection for wireless LANs”, United States patent 20020181417.

- P. Busch, R. Malhotra, “Channel swapping for wireless LANs”, United States patent 20020176437.

- P. Busch, R. Malhotra, “Corrected predictions based dynamic frequency selection”, European patent EP1257091.

- A. de Heer, S. Roijakkers, R. Malhotra, R. de Man, “Method for establishing a loop-free path for transfer of data packets within a circular network”, European patent EP1359715.

Patents pending

- R. Malhotra, “Controlling congestion in a packet switched data network”. Patent application filed 12/06/2007.

- R. Malhotra, “Method and apparatus for flow control of data in a network”. Patent application filed 03/23/2005.
- R. van Haalen, R. Malhotra, “Method for dynamically adjusting token bucket sizes”. Patent application filed 09/30/2005.
- R. Malhotra, R. van Haalen, “Method for policing-based adjustments to transmission window size”. Patent application filed 09/30/2005.
- J. van Bemmel, A. de Heer, R. Malhotra, “Congestion control for improved management of service level agreements in switched networks”. Patent application filed 11/02/2004.
- R. Malhotra, N. van Foreest, “A fast converging spanning tree protocol for LAN VPNs”, Patent application filed 01/16/2002.