Helsinki University of Technology Networking Laboratory
Teknillinen korkeakoulu Tietoverkkolaboratorio
Espoo 2005 Report 6/2005

PERFORMANCE EVALUATION OF MULTICAST NETWORKS AND SERVICE DIFFERENTIATION MECHANISMS IN IP NETWORKS

Eeva Nyberg-Oksanen

Dissertation for the degree of Doctor of Science in Technology to be presented with due permission for public examination and debate in Auditorium S5 at Helsinki University of Technology (Espoo, Finland) on the 13th of January, 2006, at 12 o'clock noon.

Helsinki University of Technology
Department of Electrical and Communications Engineering
Networking Laboratory

Teknillinen korkeakoulu
Sähkö- ja tietoliikennetekniikan osasto
Tietoverkkolaboratorio

Distribution:
Helsinki University of Technology
Networking Laboratory
P.O.Box 3000
FIN-02015 TKK
Tel. +358-9-451 2461
Fax. +358-9-451 5302

ISBN 951-22-7992-4
ISSN 1458-0322

Picaset Oy
Espoo 2005

HELSINKI UNIVERSITY OF TECHNOLOGY P.O.BOX 1000, FIN-02015 TKK http://www.tkk.fi/	ABSTRACT OF DOCTORAL DISSERTATION
Author: Eeva Nyberg-Oksanen	

Name of the dissertation
Performance Evaluation of Multicast Networks and Service Differentiation Mechanisms in IP Networks

Date of manuscript 2nd of December 2005	Date of the dissertation 13th of January 2006
☐ Monograph	☒ Article dissertation (summary + original articles)

Department	Department of Electrical and Communications Engineering
Laboratory	Networking Laboratory
Field of Research	teletraffic theory, performance evaluation of multicast and IP networks
Opponents	Professor U. Körner (Lund Institute of Technology, Sweden) Professor J. L. van den Berg (TNO Information and Communication Technology, The Netherlands)
Supervisor	Professor Jorma Virtamo (Helsinki University of Technology)
Instructor	Ph.D. Samuli Aalto (Helsinki University of Technology)

The performance of a communication network depends on how well the network is designed in terms of delivering the level of service required by a given type of traffic. The field of teletraffic theory is concerned with quantifying the three-way relationship between the network, its level of service and the traffic arriving at the network. In this thesis, we study three different problems concerning this three-way relationship and present models to assist in designing and dimensioning networks to satisfy the different quality of service demands.

In the first part of the thesis, we consider service differentiation mechanisms in packet-switched IP networks implementing a Differentiated Services (DiffServ) architecture. We study how bandwidth can be divided in a weighted fair manner between persistent elastic TCP flows, and between these TCP flows and streaming real-time UDP flows. To this end, we model the traffic conditioning and scheduling mechanisms on the packet and the flow level. We also model the interaction of these DiffServ mechanisms with the TCP congestion control mechanism and present closed-loop models for the sending rate of a TCP flow that reacts to congestion signals from the network.

In the second part, we concentrate on non-persistent elastic TCP traffic in IP networks and study how flows can be differentiated in terms of mean delay by giving priority to flows based on their age. We study Multi Level Processor Sharing (MLPS) disciplines, where jobs are classified into levels based on their age or attained service. Between levels, a strict priority discipline is applied; the level containing the youngest jobs has the highest priority. Inside a particular level, any scheduling discipline could be used. We present an implementation proposal of a two-level discipline, PS+PS, with the Processor Sharing discipline used inside both levels. We prove that, as long as the hazard rate of the job-size distribution is decreasing, which is the case for Internet traffic, PS+PS, and any MLPS discipline that favors young jobs, is better than PS with respect to overall mean delay.

In the final part, we study distribution-type streaming traffic in a multicast network, where there is, at most, one copy of each channel transmission in each network link, and quantify the blocking probability. We derive an exact blocking probability algorithm for multicast traffic in a tree network based on the convolution and truncation algorithm for unicast traffic. We present a new convolution operation, the OR-convolution, to suit the transmission principle of multicast traffic, and a new truncation operator to take into account the case of having both unicast and multicast traffic in the network. We also consider different user models derived from the single-user model.

Keywords	TCP, QoS, DiffServ, fairness, scheduling, PS, MLPS, delay, multicast, blocking probability, OR-convolution		
Number of pages	64 pp. + app. 112 pp.	ISBN (printed)	951-22-7992-4
ISBN (pdf)	951-22-7993-2	ISBN (others)	
ISSN (printed)	1458-0322	ISSN (pdf)	
Publisher	Networking Laboratory / Helsinki University of Technology		
Print distribution			
☒ The dissertation can be read at http://lib.hut.fi/Diss/			

TEKNILLINEN KORKEAKOULU PL 1000, FIN-02015 TKK http://www.tkk.fi/	VÄITÖSKIRJAN TIIVISTELMÄ
Tekijä: Eeva Nyberg-Oksanen	

Väitöskirjan nimi
Jakeluliikenneverkkojen ja IP-verkkojen palvelun eriytysmenetelmien suorituskykyanalyysi

Käsikirjoituksen jättämispäivämäärä 2. joulukuuta 2005	Väitöstilaisuuden ajankohta 13. tammikuuta 2006
☐ Monografia	☒ Yhdistelmäväitöskirja (yhteenveto + erillisartikkelit)

Osasto	Sähkö- ja tietoliikennetekniikan osasto
Laboratorio	Tietoverkkolaboratorio
Tutkimusala	teleliikenneteoria, jakeluliikenneverkkojen ja IP-verkkojen suorituskykyanalyysi
Vastaväittäjät	Professori U. Körner (Lundin teknillinen korkeakoulu, Ruotsi)
	Professori J. L van den Berg (TNO Information and Communication Technology, Alankomaat)
Työn valvoja	Professori Jorma Virtamo (Teknillinen korkeakoulu)
Työn ohjaaja	Fil.tri Samuli Aalto (Teknillinen korkeakoulu)

Verkon suorituskyky määräytyy sen mukaan, kuinka hyvin verkko on suunniteltu tarjoamaan verkkoa käyttävän liikennetyypin vaatimaa palvelunlaatua. Teleliikenneteoria pyrkii kvantifioimaan kyseisen verkon, palvelunlaadun ja verkkoon saapuvan liikenteen välistä kolmisuuntaista riippuvuutta. Väitöskirjassa käsitellään tätä riippuvuutta kolmen eri tutkimusongelman avulla. Väitöskirjaa varten on luotu malleja, joiden avulla verkkoja voidaan suunnitella ja mitoittaa siten, että ne pystyvät tarjoamaan erilaisten laatuvaatimusten mukaista palvelua.

Työn ensimmäinen osa käsittelee eriytetyn palvelun mekanismeja pakettikytkentäisissä IP-verkoissa, joissa on käytössä eriytetyt palvelut (DiffServ) -arkkitehtuuri. Työssä tutkitaan, kuinka linkin kaista voidaan jakaa painotetun reiluuden mukaisesti sekä keskeytymättömien elastisten TCP-voiden välillä että näiden TCP-voiden ja virtaustyyppisten UDP-voiden välillä. Tätä varten mallinnetaan liikenteen luokittelu- ja vuoronjakomekanismeja paketti- ja vuotasolla. Työssä mallinnetaan myös näiden DiffServ-mekanismien ja TCP-ruuhkanhallintamenetelmien välistä vuorovaikutusta. Tuloksena on suljetun järjestelmän malli TCP-lähetysnopeuden ja verkon ruuhkasignaalien väliselle yhteydelle.

Työn toinen osa keskittyy pelkästään lyhytkestoisiin TCP-voihin IP-verkoissa. Työssä tutkitaan, kuinka vuot voidaan eriyttää toisistaan, kun vuon prioriteetti on verrannollinen vuon ikään ja kriteerinä on vuon kokema keskiviive. Työssä tutkitaan monitasoisia prosessorinjako- eli MLPS-jonokureja, joissa työn ikä tai jo saama palvelu määrää, mille tasolle työ luokitellaan. Eri tasojen välillä on tiukka prioriteettijako, ja nuorin työ on korkeimmalla prioriteettitasolla. Tason sisällä voidaan käyttää mitä tahansa jonokuria. Työssä ehdotetaan implementaatiota PS+PS -jonokurille, joka on kaksitasoinen prosessorinjakojonokuri, jossa prosessorinjako- eli PS-jonokuria käytetään molemmilla tasoilla. Työssä todistetaan, että niin kauan kuin töiden kokojakauman hasardifunktio on laskeva, PS+PS tai mikä tahansa MLPS-jonokuri, joka suosii nuoria töitä, on parempi kuin PS keskiviiveen suhteen. Internet-liikenteellä on laskeva hasardifunktio.

Työn viimeisessä osassa tutkitaan virtaustyyppistä liikennettä jakeluliikenneverkossa ja kvantifioidaan verkon estoa. Tällaisessa verkossa jokaisella verkon linkillä on enintään yksi kopio kustakin tarjottavasta kanavasta. Työssä johdetaan tarkka estonlaskenta-algoritmi jakeluliikenteelle puumaisessa verkossa. Algoritmi perustuu tavallisen liikenteen konvoluutio- ja katkaisualgoritmiin. Työssä määritellään uusi konvoluutio-operaatio, OR-konvoluutio, joka huomioi jakeluliikenteen siirtotavan erityispiirteet. Työssä esitetään myös uusi katkaisuoperaatio käytettäväksi silloin kun verkossa on sekä tavallista että jakeluliikennettä. Työssä tutkitaan eri käyttäjämalleja, jotka ovat johdettavissa yhden käyttäjän mallista.

Avainsanat	TCP, palvelunlaatu, DiffServ, reiluus, vuoronjako, PS, MLPS, viive, jakeluliikenne, estotodennäköisyys, OR-konvoluutio		
Sivumärä	64 s. + liit. 112 s.	ISBN (painettu)	951-22-7992-4
ISBN (pdf)	951-22-7993-2	ISBN (muut)	
ISSN (painettu)	1458-0322	ISSN (pdf)	
Julkaisija	Tietoverkkolaboratorio / Teknillinen Korkeakoulu		
Painetun väitöskirjan jakelu			
☒ Luettavissa verkossa osoitteessa http://lib.hut.fi/Diss/			

PREFACE

Work on this thesis started when I joined the Networking Laboratory in 1999 to complete my master's thesis. During the following six years, I had the pleasure of continuing to work in the laboratory under the supervision of Professor Jorma Virtamo and Dr. Samuli Aalto. When working on my Licentiate thesis, I also received valuable insight from Dr. Kalevi Kilkki. After completing my Licentiate thesis, I visited INRIA Sophia Antipolis in Southern France, where I worked with Dr. Konstantin Avrachenkov, Dr. Urtzi Ayesta (at the time, still a student) and Patrick Brown. Since my first visit, I have visited INRIA twice and Urtzi Ayesta has visited TKK. Visiting the Mistral (now called Maestro) group was one of the highlights of my Ph.D. studies; I thank Professor Virtamo for supporting my visit and Konstantin Avrachenkov for hosting it.

I am indebted to Professor Virtamo for giving me the opportunity to do my Ph.D. thesis under his supervision. The thesis, in the format it is today, would never have been possible without the dedication and patience of my instructor Dr. Samuli Aalto. Urtzi Ayesta deserves a special thank you for encouraging me in our common research during the past three years. I am grateful for having had a chance to work with such inspiring people.

The work for the thesis was conducted under several projects: Cost257, Cost279 and PAN-NET. Funding for the projects came from Tekes, Nokia, Elisa, Siemens and Sonera. I have also been a student of the Graduate School in Electronics, Telecommunications and Automation (GETA), funded by the Academy of Finland. I thank the Nokia Foundation and the TES-foundations for supporting my work through personal scholarships.

Finishing this thesis coincides with the end of an important period in my life. I would like to thank all the people at work who have been part of my life and listened to me pondering my identity as a researcher, a student, a teacher and a person. I hope that at least someone has listened to me trying to encourage more discourse and openness in the lab. I thank, of course, our work floorball gang for giving me a well-deserved break from work.

I wish to thank my friends and family for their support and company. I thank my husband Olli for our wonderful son Leo, and for taking care of him during the few months it took me to finish my thesis after my maternity leave. I am thankful for having such wonderful people around me and for the life I have.

CONTENTS

LIST OF PUBLICATIONS

[1] Eeva Nyberg and Samuli Aalto. How to achieve fair differentiation. In *Proceedings of Networking 2002*, pages 1178–1183, Pisa, Italy, May 2002. Springer-Verlag.

[2] Samuli Aalto and Eeva Nyberg. Flow level models of DiffServ packet level mechanisms. In *Proceedings of the 16th Nordic Teletraffic Seminar*, pages 194–205, Espoo, Finland, August 2002.

[3] Eeva Nyberg and Samuli Aalto. Differentiation and interaction of traffic: a flow level study. In *Proceedings of International Workshop, Art-Qos 2003*, pages 276–290, Warsaw, Poland, March 2003. Springer-Verlag.

[4] Konstantin Avrachenkov, Urtzi Ayesta, Patrick Brown, and Eeva Nyberg. Differentiation between short and long TCP flows: predictability of the response time. In *Proceedings of IEEE INFOCOM 2004*, volume 2, pages 762–773, March 2004.

[5] Samuli Aalto, Urtzi Ayesta, and Eeva Nyberg-Oksanen. Two-Level Processor-Sharing scheduling disciplines: Mean delay analysis. In *Proceedings of ACM SIGMETRICS - PERFORMANCE 2004*, pages 97–105, June 2004.

[6] Samuli Aalto, Urtzi Ayesta, and Eeva Nyberg-Oksanen. M/G/1/MLPS comapred to M/G/1/PS. *Operations Research Letters*, 33(5):519–524, September 2005.

[7] Eeva Nyberg, Jorma Virtamo, and Samuli Aalto. An exact algorithm for calculating blocking probabilities in multicast networks. In *Proceedings of Networking 2000*, pages 275–286, Paris, France, May 2000. Springer-Verlag.

[8] Eeva Nyberg, Jorma Virtamo, and Samuli Aalto. An exact end-to-end blocking probability algorithm for multicast networks. *Performance Evaluation*, 54(4):311–330, December 2003.

1 INTRODUCTION

Performance analysis of communication networks is based on the three-way relationship between the network system, its level of service and the traffic arriving at the system. Teletraffic theory has concentrated on modeling and analyzing this relationship. Presently, there is a variety of traffic to be serviced by communication networks, and though the relationship between the traffic load, design of the system and quality of service still remains, the type of traffic in question imposes new demands on the modeling and design of networks.

Traditionally, networks are divided into circuit-switched and packet-switched networks. In a circuit-switched network, the resources, i.e. circuits, are reserved and admission control functions are performed on traffic before a given service level can be guaranteed. In a packet-switched network, as originally designed, all traffic is admitted to the network and traffic receives best-effort service dependent on the load and congestion level of the network.

With the success of packet switching for transportation of all kinds of traffic, concerns on the quality requirements of real-time traffic have brought on proposals for packet switched Quality of Service (QoS) networks. Packet-switched networks are then not necessarily best-effort networks anymore. It is thus more appropriate to make a distinction between whether a packet switched network uses resource reservations and admission control to give delay guarantees at the expense of blocking some connections and whether no blocking occurs, but service is offered to traffic classes by applying class specific traffic handling mechanisms at the packet level. As the different traffic types have different service requirements, we concentrate on the traffic types and their service requirements and study how these can be achieved, given the present network architectures. We divide traffic types in the packet-switched Internet to streaming traffic and elastic traffic. Streaming traffic is real-time traffic and the transmission rate is independent of the traffic conditions of the network. Elastic traffic is data traffic, which adapts its transmission rate to the network traffic conditions by a congestion control mechanism.

In this thesis, we consider packet-switched QoS networks and multicast networks that apply resource reservations. The networks service either elastic traffic or streaming traffic or both. We study the offered service under three different scenarios. For persistent elastic traffic and streaming traffic differentiated without using admission control, we evaluate the achieved service in terms of flow bandwidth shares. We study the mean delay, when non-persistent elastic traffic is differentiated based on age. When distribution-type streaming traffic is serviced using multicast connections, we give algorithms for calculating the blocking probability.

1.1 Communication networks

A network provides a platform for setting up connections between subscribers desiring to communicate with each other. A subscriber is typically represented by a terminal, e.g. a telephone or a computer. In this work we consider two networks: the Internet, a set of interconnected networks using packet switching, and a circuit-switched network for distribution-type applications. The networks are characterized by the traffic type they are designed for and whether they give or do not give performance guarantees through resource reservations and admission control.

When circuit switching is used, a whole physical line or channel is reserved for the call. The whole line is reserved until the call ends, even if pauses occur during transmission. In packet switching, the data to be sent, called a flow, is divided into packets, and the packets are transmitted through the network sharing the physical links with packets of other flows. The physical transmission medium is used more efficiently, as there is no need to reserve a circuit for each connection. Instead, packets are sent when there are data to be sent, and, during idle periods, other flows may use the capacity of the links. Because each router handles the packets in a different way, the packets may be routed to different links, they may arrive in a different order, and delays between them may occur. Packet switching is appropriate for data transmission, as data can easily be constructed from the packets, as long as no packets have been lost. Packet switching may not be appropriate for transmitting real-time voice or video if the transmission delays of packets exceed a limit that the human ear and eyes can tolerate.

Circuit-switched network

In a circuit-switched network, end systems are connected to a switch and the switches are connected to each other by links. The links are divided into a given number of circuits. When two end hosts desire to communicate with each other, a circuit is formed between the end hosts for the duration of the connection, even if, at times during the connection, the end hosts do not have traffic to send. In this way, performance guarantees on delays can be given, but some arriving calls may not be admitted to the network due to lack of resources.

Circuit-switched networks are modeled as loss networks. In a loss system, flows or connections arrive stochastically to the system and have holding times given by a probability distribution. Because the flow reserves a circuit for the whole duration of the call, in a loss system the performance measure of interest is the probability that an arriving call finds all circuits reserved and is blocked.

Packet-switched network

The original packet-switched IP network offers only best-effort service, i.e. all packets are admitted to the network without giving performance guarantees. If resources are scarce, packets waiting to be serviced are delayed due to queuing. Best-effort service has been designed to be undemanding and is suited for data delivery in IP where any underlying physical network can provide the service. The delivery is achieved through a connectionless data-

gram mode, where every packet contains the complete address to enable a router to decide how the packet is forwarded to its destination. The packet can be sent anywhere at any time. There are no guarantees of delivery, as packets are forwarded independently of each other and the network capacity is not reserved beforehand. As a consequence, packet-switched networks are more efficient in the use of network capacity. The delivery is unreliable, as messages may be dropped, reordered, delayed and delivered as duplicate copies. However, the delivery is also resilient: if a failure occurs at a router or on a link, an alternate route can be found, as each packet contains the complete address of the destination.

On the packet level, packet-switched networks can be modeled as a queuing network, where packets arrive stochastically to the system and have a given size distribution. The service time is then the time the link with a given capacity processes the packet with a probabilistic packet size distribution. The total time spent in the system depends on the service time and the waiting time of the packet. If routers have infinite buffers, i.e. infinite waiting space, the system can be modeled by a queuing system, where the performance measure of interest is the expected delay of a packet traversing the system. In a real packet-switched network the capacity of the buffer is finite, and thus a mixed model is used, where packets may be delayed due to waiting time in queues and losses may occur due to insufficient buffer capacity.

Internet QoS network

The success of the Internet has also increased the transportation of real-time traffic in packet-switched networks. However, real-time applications suffer from variable queuing delays and large losses due to congestion. On the other hand, aggressive real-time applications may starve bandwidth from elastic TCP traffic. Quality of Service (QoS) networks are designed in order to use packet switching and at the same time bring quality guarantees to the various traffic types using the Internet.

Two main architectures for Internet QoS networks are IntServ and DiffServ. The Integrated Services (IntServ) architecture [BCS94, Wro97b, Wro97a, SPG97] aimed at providing control over end-to-end packet delays. The QoS requirements of the admitted flows are met through admission control, an explicit resource reservation setup mechanism, per flow state in the routers and advanced packet scheduling mechanisms. The performance measures used are latency, fidelity, reordering and/or delay of packets. The Differentiated Services (DiffServ) architecture [BBC+98] was designed to provide service in a scalable way. Service is defined as a set of significant characteristics of packet transmission in one direction across a set of one or more paths within a network. The characteristics may be in terms of relative priority in accessing network resources or in terms of quantitative or statistical performance measures, such as throughput, delay, jitter and/or loss. Scalability of DiffServ, compared to the less scalable requirement of per flow state in IntServ routers, is achieved by performing complex classification and conditioning functions at network boundary nodes to flows that are aggregated to forwarding classes. Simple forwarding and discarding functions are then performed on the aggregates inside the

network.

The objective of IntServ, as given in [BCS94], is to offer guaranteed and predictive services in addition to best-effort service, and thus implicitly change the Internet best-effort service model. The new model uses controlled link sharing, implemented by resource reservations and admission control. The service model of IntServ was created to integrate real-time services with existing elastic services of the Internet. The service model is thus concerned with the time of delivery of packets, i.e. the per packet delay. Quantitative service commitments are given as bounds on the maximum and minimum delays. To calculate the delay bounds and make the admission control decision, flows need to be characterized and the service requirements of the flow need to be specified. The flow specification is carried by the reservation setup protocol and is passed to the admission control. If the flow is admitted, the flow specification is used to parameterize the packet scheduling mechanisms in order to meet the given delay bounds. Scheduling packets flow-by-flow ensures that the requirements of the admitted traffic are met to the required precision. Packet scheduling and admission control are also used for resource sharing between entities of many flows.

Differentiated Services is designed to achieve assured or relative service between flows without scheduling packets using a per flow state. Traffic classification and conditioning is achieved at the boundary nodes, where flows are divided into per hop behaviors (PHB) and drop precedences inside PHBs. Inside a DiffServ node, scheduling and discarding are performed on aggregates based on the given PHB. In general, a contracted or assured rate is assigned to the flows. With this contracted rate, a charge may be associated. Based on this contracted rate, flows are classified into priority aggregates at the boundary nodes and are then scheduled packet-by-packet inside the core network. In addition, congestion control mechanisms designed for the Internet, such as TCP, and active queue management algorithms, such as Random Early Detection (RED) [FJ93], may be used to achieve Quality of Service in the Internet. In this work, we assume that classification of packets into PHBs corresponds to classification of packets into delay aggregates, and metering and marking packets into drop precedence levels corresponds to marking into priority aggregates. In the scheduling unit, two main elements affect the service experienced by the PHBs: the scheduling policy of the buffers and the realization and relationship between the discarding thresholds of the buffers.

The DiffServ proposals considered here are Expedited Forwarding (EF) [JNP99], Assured Forwarding (AF) [HBWW99] and the Simple Integrated Media Access (SIMA) proposal [RK98]. EF and AF have the status of an IETF RFC and give a conceptual service model of the PHB groups that could be implemented. SIMA has not gained RFC status, but gives a more thorough implementation and end-to-end view on how to achieve service differentiation.

We can categorize EF and AF as assured-services type and SIMA as the type that offers relative services. AF can, however, due to its broad definition, be implemented to be of the relative-services type also. In assured services, the flow should receive a rate at least equal to the contracted rate,

while, in relative services, the rate of the flow should be proportional to the contracted rate. Relative services are easier to realize in the sense that no capacity reservation is needed; in overload situations, it suffices that capacity is divided into shares based on the contracted rate. In this case, each flow receives less than its contracted rate, but by an amount proportional to its contracted rate. With assured services, capacity reservations are needed. When reservations are not used and conditioning is performed only at the edge of the network, it may happen that, at a core link, the total capacity is less than the sum of assured rates. Then capacity is divided equally and no differentiation results.

Multicast network

A unicast transmission is designed for point-to-point communication, where a source sends a message to only one receiver. If the message is intended to be received by a group, using one of the point-to-multipoint transmissions, either multicast or broadcast, is more effective. In broadcasting, a message is transmitted to all users on the network and may therefore require unnecessary bandwidth and/or a limit to the number of recipients. A multicast transmission originating at a source is selectively replicated at various network nodes to form a tree-and-branch structure. The transmission reaches the end-users requesting the transmissions without a separate transmission required for each user, as would be the case in a unicast transmission. A multicast connection has therefore a bandwidth saving nature (figure 1.1). A multicast transmission is sent to a multicast group, a group

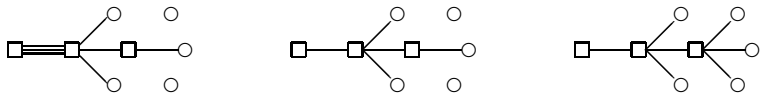

Figure 1.1: Difference between unicast, multicast, and broadcast. Circles represent end-users and squares network nodes. In all three networks, the source is the leftmost square.

of users, requesting the transmission. The multicast groups are dynamic, receiver-controlled groups, where a host can join or leave the group at any time. Traditionally, the use of multipoint connections has been limited to Local Area Network (LAN) applications. Applications on the Internet relying on multicast transmission have increased in the past few years. Due to the Internet Multicast Backbone (MBone), IP-multicast has become a widely-used multicast protocol.

1.2 Traffic types in communication networks

We concentrate here on the different traffic types in the Internet. On the flow level, traffic can be divided into two categories, based on the transport layer protocol used: elastic traffic using the Transmission Control Protocol (TCP) or streaming traffic using the User Datagram Protocol (UDP).

Streaming traffic

Streaming traffic is real-time traffic and the transmission rate is independent of the traffic conditions in the network. In general, real-time traffic uses UDP instead of TCP, as TCP flow control and congestion control require the flow to accept drops and retransmits and to control its sending rate. TCP is thus not suited to transporting delay-critical streaming traffic. UDP is a transport protocol that adds demultiplexing functionality, allowing multiple application processes on each host to share the network. Processes indirectly identify each other using an abstract locater, e.g. the port. The UDP header includes the source and the destination ports. UDP thus provides a connectionless datagram service, as opposed to TCP, which is connection oriented. For example, multicast transmissions use UDP, as the sender of the multicast transmission does not know the address of the receiver and therefore cannot establish a connection at the transport level.

Elastic traffic

TCP traffic is elastic traffic, as it uses flow control and congestion control to adjust its sending rate according to the feedback signals, e.g. missing acknowledgments of a lost packet, from the receiver or the network. TCP is a connection-oriented protocol that assures that the service is reliable, by guaranteeing message delivery, delivering the messages in the order they were sent and delivering, at most, one copy. TCP also supports arbitrarily large messages, synchronization between sender and receiver, multiple application processes on each host and allows the receiver to apply flow control to the sender. By introducing congestion control, the source can adapt the sending rate to the congestion level of the network. On the flow level, elastic traffic is modeled as traffic that divides bandwidth equally. This is due to the interaction of the source with the network feedback signals; ideally, if all sources adapt their sending rate to the network state and all flows receive information on congestion at the same time, all flows have the same sending rate.

The equilibrium sending rate of elastic traffic depends inversely on the Round Trip Time (RTT) of the connection and, inversely, on the square root of the packet loss probability of the route. This is based on the additive increase multiplicative decrease sawtooth model for TCP congestion control. When a TCP connection is in congestion avoidance mode, the congestion window is increased by one segment per RTT and halved upon a packet drop notification.

The TCP models in the literature are often based on the ideal and canonical form of the TCP congestion control algorithm [MSMO97]. It is assumed, for example, that the sender is persistent, it has always a congestion window amount of traffic to send, all segments are of size Maximum Segment Size (MSS), the connection does not contribute to delays and that the RTT is independent of the window size and that loss intervals are from a defined process, e.g. deterministic.

As the research has advanced, some modeling assumptions have been relaxed, e.g., recovery from lost segments does not interact with congestion avoidance and that the network is the bottleneck not the receiver. At the same time, the focus of the design of TCP congestion control has been

on pushing the algorithms towards the ideal model by introducing fast recovery and retransmit algorithms, limited transmit and Delayed Selective Acknowledgments (D-SACK).

It can be further noted that the ideal model assumes that the connection is in congestion avoidance mode. The slow start phase, where the sending window is initially one segment and increased exponentially, is ignored as it is usually of short duration; with new developments to TCP congestion control, slow start should only occur at the beginning of a connection.

When traffic is not persistent, the slow start phase at the beginning of the connection cannot be ignored. However, once the flow is in congestion avoidance mode, the bandwidth is divided equally among flows and is often modeled as an M/G/1/PS system [HLN97, FBP+02], where the flow arrivals constitute a Poisson process and flow sizes are generally distributed. Then the delay of the flow depends linearly on the flow size, and the throughput is then the same for all flows.

Recently, however, it has been pointed out that the multiplicative decrease is too drastic, and the additive increase too slow, in the case of large congestion windows, which require very small packet loss probabilities to keep the equilibrium sending rate large enough for full link utilization. New versions of TCP for high-speed links have been proposed in [Flo03, Kel03, JWL04, CGM+02], resulting in a linear or near-linear relationship between the TCP sending rate and the inverse of the loss probability.

1.3 Outline of the thesis

In this thesis, we study three different problems concerning the relation between the arriving traffic, the network system and the level of service. For persistent elastic traffic in a packet-switched Internet with a Differentiated Services (DiffServ) architecture, we compare the weighted bandwidth shares of flows when flows are differentiated and grouped to aggregates according to weights given to the flows. We extend the study of the network to the case where both streaming UDP traffic and persistent elastic TCP traffic are present. For non-persistent elastic TCP traffic in the Internet, we study the mean delay when flows are differentiated based on their age. For distribution-type streaming traffic in a multicast network with resource reservations, we calculate blocking probabilities when there is a set of channels the users can choose from.

The thesis is organized as follows. In chapter 2, we consider persistent elastic and streaming traffic in the packet-switched Internet implementing a DiffServ architecture. We model the proposed DiffServ mechanisms and their interaction with TCP control functions, both on the packet level and the flow level. We study how the proposed DiffServ traffic conditioning and handling mechanisms affect the sending rate of elastic TCP flows and how the mechanisms can be used to divide bandwidth in a weighted fair manner between persistent elastic TCP traffic and between elastic TCP and streaming UDP traffic.

In chapter 3, we consider non-persistent TCP traffic in the Internet. We introduce differentiation by favoring flows at the beginning of a connection,

i.e. with small attained service, and study how scheduling of flows based on their age can reduce the overall mean delay of the network and favor short TCP flows, which are more prone to time-outs than longer flows. We propose an implementation of a PS+PS discipline, a two-level age-based scheduling discipline using PS discipline inside both levels, and study its performance by simulations. We are also able to prove that the PS+PS discipline reduces the mean delay compared to the PS discipline. The proofs are extended to include multi-level PS disciplines.

In chapter 4, we study streaming traffic in a multicast network for distribution-type applications offering a set of channels to choose from. We study the blocking probability of a user trying to subscribe to a channel, assuming that there is no more than one copy of each channel transmission in a link of the network. Traditionally, there is a separate end-to-end connection and thus multiple copies of the same channel transmission for each user. As a result, we present an exact blocking probability algorithm for multicast networks. The algorithm is based on a convolution and truncation algorithm similar to that for unicast traffic. We present a new convolution principle, OR-convolution, suited to multicast traffic, and modify the truncation operators for the case of both unicast and multicast traffic in the network. We also consider a set of user models based on the model for a single user.

2 DIFFERENTIATION AND INTERACTION OF INTERNET TRAFFIC

2.1 Introduction

Internet is a packet-switched network that is best suited to the transport of elastic data. When streaming real-time traffic is also transported, rules and service characteristics of the packet transmission are needed. Differentiation of traffic and Quality of Service (QoS) guarantees are designed to accommodate heterogeneous traffic types and quality requirements. The service requirements can be for packets or flows, they can be in terms of quality guarantees for throughput, delay, jitter and loss, they can be in terms of statistical bounds or in terms of hard guarantees, and they can be in terms of bandwidth shares or in terms of price paid for transmission.

We consider here differentiation using the Differentiated Services (DiffServ) architecture [BBC+98]. In DiffServ, flows are conditioned and classified at the edge of the network to aggregates based on some reference weight, e.g. a contracted rate, price paid or a classification decision made by a network administrator. Inside the network, forwarding and discarding are performed to these aggregates. In the best-effort Internet, each packet is treated similarly. In DiffServ, each packet of an aggregate is treated similarly, but, between aggregates service differentiation may occur. The resulting differentiation should then depend on the reference weight of the flow.

Previous research in the field of DiffServ has had two approaches: the mechanism-oriented approach, which aims to create new traffic handling mechanisms such as conditioning and scheduling inside the network, and the traffic-oriented approach, which aims to study how differentiated congestion signals could be used to influence the sending rate of TCP traffic. The approaches have been studied in isolation. However, as TCP traffic is elastic adjusting its sending rate according to implicit or explicit congestion notifications, any new mechanism that marks, delays or discards packets must take into account the reaction of TCP flow and congestion control mechanisms to the congestion signals it produces. Furthermore, the interaction of different traffic types, e.g. elastic and streaming, with each other, and with the proposed QoS mechanisms, also needs to be quantified.

Of great importance in evaluating DiffServ mechanisms is the positioning of the mechanisms relative to the service requirements. We consider the share of bandwidth a flow receives, i.e. the relationship between the sending rate and the contracted rate of a flow, and distinguish two main service requirements of DiffServ: relative services and assured services. In relative services, the rate of the flow should be proportional to the contracted rate, while, in assured-services, the flow should receive a rate at least equal to the contracted rate. We claim that the relative services requirement is a more appropriate DiffServ service requirement than the assured services requirement.

We model service differentiation mechanisms, study how bandwidth is divided between different traffic groups and how design choices affect the

resulting Quality of Service. We divide the mechanisms into flow-level mechanisms at the boundary nodes for metering and marking of flows into priority aggregates and packet-level mechanisms inside the Diffserv nodes for forwarding and discarding packets of aggregates. We model service differentiation on two levels of abstraction, packet level and flow level, and compare the results given by the two models.

On the packet level, we present models for the TCP sources that adjust their sending rate, for the conditioner that marks the flows to aggregates and for network routers that forward and handle the aggregates. We study how the sending rate of TCP flows affects the dropping probabilities of the network and vice versa. Our models also include the effect of UDP flows in the network. We then evaluate the differentiation mechanisms based on the relationship between the contracted rate, which will be called the weight, of the flow and the share of bandwidth the flow actually achieves.

On the flow level, we study the fairness of greedy TCP flows. We assume that TCP flows try to realize max-min fair bandwidth allocation, i.e. tend to maximize the minimum bandwidth allocation. In a best-effort flow-level setting with no priorities, it is then optimal for greedy TCP flows to divide bandwidth equally among themselves. When a priority mechanism based on relative services is added to the network, a flow is classified into a priority level based on a given weight, such that, ideally, the achieved bit rate of the flow should be proportional to its weight. Our goal is to study what kind of weighted fairness is achieved between TCP flows, with the effect of UDP flows also included in the model.

We present closed-loop models with an end-to-end view, which, to our knowledge, has not been done before. As a result, we identify the key differentiation mechanisms, provide design guidelines for delivering relative services and point out some shortcomings of the proposed mechanisms.

2.2 Related research

Previous analytical research has focused on modeling either the mechanism-oriented or traffic-oriented approach: buffer models with various scheduling and discarding mechanisms to quantify delay and loss probabilities as a function of given sending rates for different priority classes, or quantifying TCP sending rate as a function of marking decisions and given loss probabilities for different priority classes. The problem with these models is that they do not quantify the interaction of both the QoS mechanisms and the TCP congestion control.

A TCP traffic-oriented approach is taken, for example, in [SNT+00] and [YR01], where an analytical relationship between the contracted rate and the transmission rate of TCP flows is studied. The works include a model for TCP and a model for the priority marker, under the assumption of deterministic loss rates for different priorities, without an explicit buffer model for how the loss probabilities depend on the TCP rates.

The mechanism-oriented approach is taken, for example, in [MBDM99], [STK99] and [NEC00]. In [MBDM99], a model for assured service that includes a buffer model with different discarding levels for different priorities is studied, but the paper does not include a TCP model nor

a marking model for the traffic. A similar model is given in [STK99] and [NEC00], with some considerations on modeling TCP flows. These papers, however, still lack combining both a TCP model and a buffer model.

The effect of having both TCP and UDP flows in the network have only be seen studied by simulations. The study by Goyal et al. [GDJL] simulate different factors related to differentiation using Assured Forwarding (AF). These include number of priority levels, percentage of highest priority traffic, buffer management and traffic types. The simulations assume many, but a fixed number of TCP flows. The system model is similar to the ones of assured-services presented in the analytical studies. Elloumi et al. [ECP] give similar results. Pieda et al. [PSN99] study the claim made by Goyal et al. and Elloumi et al. that TCP flows can be protected from UDP flows by marking out-of-profile UDP flows to lowest priority, but out-of-profile TCP flows to middle priority. They divide the study of priority levels into six scenarios. In each scenario, three priority levels are used, but packets of a flow are only marked in or out of profile. As a result they show that none of the six scenarios is able to meet all three differentiation targets: that in an over-provisioned network both UDP and TCP target rates are achieved, that UDP and TCP out-of-profile packets should have a reasonable share of excess bandwidth, and that in an under-provisioned network TCP and UDP flows experience degradation in proportion to the assured rate.

The Simple Integrated Media Access (SIMA) proposal [RK98] does not aim at assuring a certain contracted rate for each flow, but aims at assuring that the bandwidth is divided in relation to the contracted rate. SIMA with TCP flows has been studied with the help of simulations in [KR98] and using a test network in [LSLH00] and [HKL+00]. The above studies show that SIMA is able to achieve differentiation, and that the order relation between flows in terms of the contracted rates is kept in the level of quality received by the flows.

In [HA02], an approach similar to Publication 1 is taken. The paper concentrates on studying different buffer management policies on the packet level in a weighted fair queuing scheduler, where loss feedback to TCP traffic is also taken into account.

Recently new buffer management and marking policies have been proposed to overcome the shortcomings of assured rate based mechanisms, see e.g. [HG04], [LC02] and [PC04]. Though these mechanisms are designed to be scalable and robust, they are much more complicated than mechanisms based on relative services.

Studies on how to provide differentiated services in other ways have also been proposed, such as congestion pricing [GK99] and proportional delay differentiation [DSR99] to name but a few.

2.3 Contribution

Publications 1–3 contribute in three main ways to the research on Differentiated Services. Firstly, to address the lack of models that combine both a traffic and a buffer mechanism model, the emphasis of both the packet and flow level studies is in modeling the interaction between the various differentiation mechanisms and traffic flows of type TCP and non-TCP, e.g.

UDP.

Secondly, previous research on AF has concluded that without admission control assured services cannot be achieved. Early on in our research, we focused on studying schemes, such as SIMA, where the objective is to divide bandwidth relative to flow weights, in order to broaden the view on how bandwidth can be divided in a packet switched QoS network.

Thirdly, we propose a novel flow level model as an abstraction of the packet level model. By introducing abstract flow level models for the packet level differentiation mechanisms, we concentrate on studying the interaction of the flows and the network. Then the optimal bandwidth share of flows is the result of a game, where the TCP flows optimize their priority level based on the network conditions, i.e. number of other TCP flows and the load of non-TCP flows.

In Publication 1, a packet level model is proposed to show how marking at the DiffServ boundary node and scheduling and discarding inside a DiffServ node affect the division of bandwidth between two delay classes: elastic TCP flows and streaming non-TCP flows. As an abstraction of the packet level model, a flow level model for TCP traffic is given in Publication 2. In Publication 3, the flow level model is extended to include non-TCP traffic.

2.4 Models for differentiation mechanisms

Introducing differentiation to the Internet has two main motives: separating delay sensitive real-time traffic from elastic non-real-time traffic and dividing available bandwidth in proportion to given weights. The purpose of DiffServ is to introduce delay and priority aggregates, instead of per flow state, to achieve the differentiation goals of flows. Inside an aggregate, which we call a class, we assume that the received service is equal. Adopting the relative-services approach introduced in [RK98], the goal is to obtain a clear dependency between the marking of flows into priorities and the resulting actual sending rate or share of bandwidth.

To identify how delay and priority aggregates can be used to achieve relative differentiation of flows, we formulate a generic DiffServ architecture based on such DiffServ proposals as Expedited Forwarding (EF)[JNP99], Assured Forwarding (AF)[HBWW99] and SIMA [RK98], and identify the main QoS mechanisms used to achieve differentiation.

The main elements of DiffServ, depicted in figure 2.1, are conditioning of flows and classification to priority aggregates at the boundary nodes and flow aggregate forwarding through scheduling and discarding of packets at the DiffServ interior nodes. Considering the requirements of real-time traffic, we assume that UDP traffic is in one delay class and TCP traffic in another. TCP traffic could further be divided into two delay classes, one for interactive short transfers and another for long file transfers, but we do not make such a distinction in this section, as we assume that all TCP flows are persistent. Inside a delay class, traffic is divided into priorities. For TCP traffic that reacts to packet drops, the priority class of the flow determines at what buffer load level a packet of the flow is dropped and how often a TCP source has to adjust its sending window due to a missing acknowledgment.

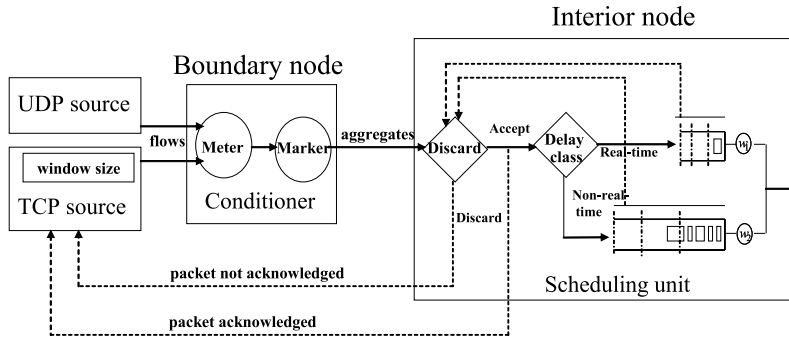

Figure 2.1: Components of a DiffServ network including feedback to TCP sources

The interaction between the elastic TCP traffic and the service differentiation mechanisms can be modeled on the packet level by a fixed point equation on the relationship between traffic class sending rate, resulting priority level mark and discarding probability. On the flow level, we equate the relationship between traffic class bandwidth share, resulting priority level and number of flows. As the focus is on understanding how the separate mechanisms interact and influence the resulting differentiation, submodels for these mechanisms are needed. Conditioning mechanisms at the DiffServ boundary nodes are performed per flow, therefore sub-models for the metering and marking of flows to priority aggregates do not depend on the level of abstraction. Inside the DiffServ domain, the level of abstraction, packet or flow level, determines in which way the scheduling unit and its interaction with TCP traffic sources, is modeled.

We consider two delay classes: real-time streaming traffic and non-real-time elastic traffic, $d = 1, 2$, respectively. Within each delay class there are I priority levels, $i = 1, \ldots, I$. Level I refers to the highest priority, i.e. flows at that level encounter the smallest packet loss probability. Each flow is given a weight ϕ that reflects the value of the flow, i.e. the contracted rate of the flow, and flows are grouped according to this weight. There are L^d different groups of flows of delay class d, each group l with a characteristic packet sending rate $\nu(l)$ and weight $\phi(l)$. Let $\mathcal{L}^d$ denote the set of such flow groups. Finally, let $n(l)$ denote the number of flows in any group l. For TCP flows the equilibrium sending rate $\nu(l)$ depends on the network state, while for UDP flows $\nu(l)$ is fixed and does not change even if the network congestion level changes.

Conditioning flows at the boundary nodes

In order to mark packets of flows into priority classes, we need a flow metering and marking mechanism. According to the DiffServ proposal, this conditioner function is situated at the boundary nodes of the network, allowing per flow state information to be used. We assume here that the delay class of a flow is given and is based on the transport protocol used by the flow.

Flow sending rate is metered and compared to a marking threshold rate.

In DiffServ proposals and in Publication 1, two metering and marking alternatives to mark flows to priority levels are presented: *Token Bucket* and *Exponential Weighted Moving Average* (EWMA).

- *Token Bucket*: To mark flows into I priorities, $I-1$ buckets are needed. Buckets are filled with tokens at a given marking rate. Packets are marked in-profile if the bucket holds enough tokens upon arrival and out-of-profile otherwise.

- *EWMA*: The measured bit rate of previous time instants are exponentially dampened according to the time interval that has passed since the measurement was done multiplied by a parameter α. This bit rate is then compared to a given marking rate threshold.

To determine the marking rate thresholds, we consider conditioning alternatives that mark flows belonging to a group l to priority levels based on the relationship of the contracted rate $\phi(l)$ and the sending rate $\nu(l)$. More specifically, we adopt the relative-services approach, and assume that a flow sending at its contracted rate has middle priority [RK98]. Analogous to the TCP congestion control model where upon a packet drop a TCP flow halves its sending rate, a flow is able to move up in priority by halving its sending rate. The corresponding marking thresholds $t(l,i)$ for priority level i are

$$\begin{aligned} t(l,0) &= \infty, \\ t(l,i) &= \phi(l)a(i),\ i=1,...,I-1, \\ t(l,I) &= 0, \end{aligned} \tag{2.1}$$

where, $a(i-1)/a(i)=2$ for all i and is, for example, in the SIMA specification [RK98], defined as $a(i)=2^{I/2-i-0.5}$.

In [NAS02], we show through simulations that the Token Bucket can be modeled as per packet marking and that EWMA can be modeled as per flow marking:

- *Per packet marking:* Only those packets of a flow that exceed the marking threshold $t(l,i)$ are marked to the priority level i. The packets that have priority i form a substream and their proportion of the sending rate $\nu(l)$ is

$$\frac{\min[\nu(l),t(l,i-1)]-\min[\nu(l),t(l,i)]}{\nu(l)}. \tag{2.2}$$

- *Per flow marking:* Once the measured rate of a flow exceeds a marking threshold $t(l,i)$, all packets of the flow are marked to the priority level i,

$$pr(l)=\arg\min_i[\nu(l)\geq t(l,i)]. \tag{2.3}$$

Figure 2.2 depicts the resulting marks given to the packets of a flow. When per packet marking is used, the flow is divided into $I+1-i$ substreams each having a different priority, with per flow marking all packets of a flow have the same priority mark.

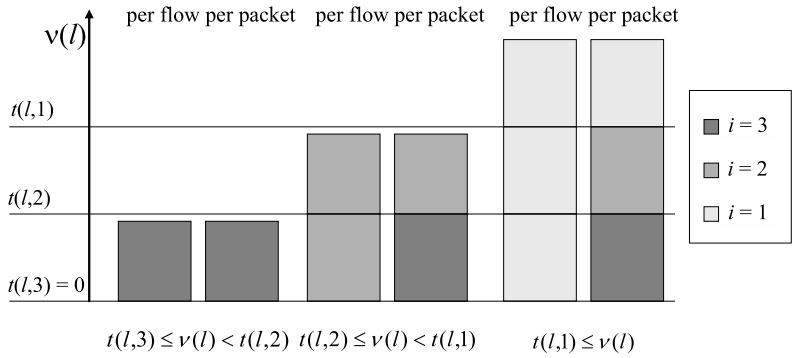

Figure 2.2: Differences in marking, for three priority levels

Scheduling packets inside the DiffServ nodes

After flows and their packets are divided into delay and priority classes, scheduling and discarding functions are performed to these aggregate classes. Note that, the weight or contracted rate of a flow affects the priority class, which in turn determines the discarding decision, and, as a result, the bandwidth share of the flow. The delay requirements, on the other hand, are satisfied by scheduling the delay classes into separate buffers.

On the packet level, we have a system with two delay classes, serviced by two separate buffers, where the buffer sizes are chosen according to the delay requirements of the delay aggregates. Both buffers have I discarding thresholds, one for each priority class. Traffic that is not discarded is placed in either one of the two buffers. The buffers can be serviced using a given scheduling mechanism. We restrict the analysis of scheduling mechanisms to Weighted Fair Queuing (WFQ) and the different weights possible in the WFQ scheduling principle. WFQ is a packet level approximation of Generalized Processor Sharing (GPS) [PG93]. In WFQ, the capacity of the link is divided among the delay classes according to predetermined weights w^1 and w^2, with $w^1 + w^2 = 1$. If one of the buffers is empty, the other buffer has use of total link capacity.

On the packet level, we consider two different discarding mechanisms:

- *Independent discarding:* Each buffer acts locally as a separate buffer, discarding appropriate priority levels according to its buffer content.
- *Dependent discarding:* The content of both buffers determines which priority level is discarded, in both buffers.

For example, when one buffer is heavily loaded and the other is almost empty, *independent discarding* would discard traffic belonging to high priority level in the heavily loaded buffer and only low priority traffic or no traffic in the lightly loaded buffer. *Dependent discarding*, on the other hand, would discard traffic of high priority also from the buffer that is almost empty. Figure 2.3 illustrates this difference with the help of a two-dimensional plot of the buffer occupancies. Instead of fixed discarding thresholds a push-out buffer management could also be considered, as was done in [HA02].

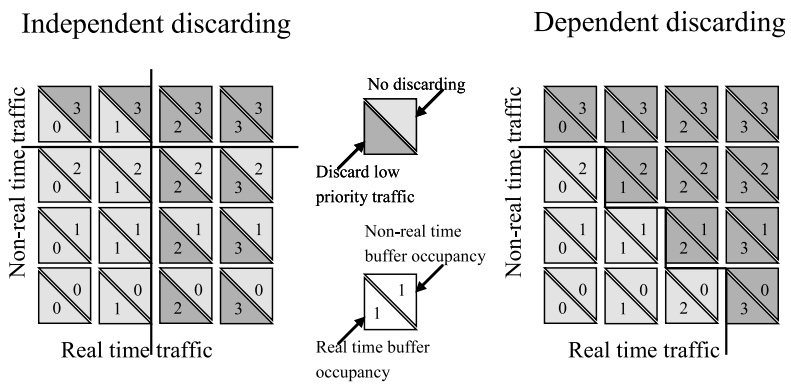

Figure 2.3: Discarding alternatives based on buffer occupancy of two delay classes

On the flow level, we model the bandwidth share that results from the scheduling decisions of the DiffServ interior node, without a detailed packet level model of the scheduling unit. We model the division of bandwidth between priority levels as a strict priority, where a higher priority level has strict priority over the lower class. Note that, this does not cause a flow of the higher class to use up all the bandwidth and starve the flows of the lower class, as the assignment to priority level depends both on the contracted rate $\phi(l)$ and the sending rate $\nu(l)$ of a flow in group l. Between delay aggregates, we model a priority system assuming dependent discarding, and the bandwidth is divided so that UDP sources always first divide among themselves the remaining bandwidth, but only up to the given boundary rate of their priority level. Elastic TCP flows then divide among themselves the capacity that is left over by higher priority flows and UDP flows of the same priority level. We assume that dependent discarding is used, then UDP flows of lower priority levels do not affect the bandwidth share of TCP flows of higher priorities.

2.5 Packet and flow level models for differentiation

To model the interaction of TCP traffic with QoS mechanisms of the Internet, we build on models for TCP with congestion control. Elastic TCP traffic is greedy in the sense that it tries to fill all the available capacity, but it is conforming as it adjusts the sending rate according to congestion notifications. On the flow level, assuming one bottleneck link, TCP flows then divide bandwidth equally. The resulting bandwidth share θ of an elastic TCP connection is modeled as Processor Sharing (PS), where bandwidth C is divided equally among n competing TCP flows,

$$\theta = \frac{C}{n}.$$

Figure 2.4 depicts the resulting relationship.

On the packet level, we are able to characterize the relationship in more detail, where the sending rate of TCP packets is expressed in terms of the

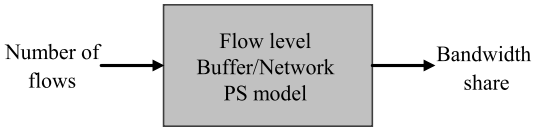

Figure 2.4: Abstraction model of TCP equilibrium, when the network is modeled on the flow level

round trip time and the probability that packets are lost due to congestion. The most common equilibrium models for TCP congestion control are the heuristic model by Floyd [Flo91] and [FF99], a more detailed model by Padhye et al. [PFTK98] and the model by Kelly [Kel99] for a collection of TCP connections. These steady state models for the TCP congestion control all have in essence the same $\frac{1}{RTT\sqrt{p}}$ relation for the TCP sending rate. The basic assumptions behind these models are that the TCP flow is persistent, it is in the congestion avoidance mode and has always a congestion window amount of traffic to send.

We use the model by Kelly as the differential equation and rate control framework is best suited to our work. The TCP equilibrium rate reads,

$$\nu = \frac{1}{RTT}\sqrt{2\frac{1-q}{q}}, \tag{2.4}$$

where ν is the resulting sending rate, when the loss probability is q and the round trip time is RTT. Assuming that the loss probability of the network depends on the sending rates of the TCP connections through a given buffer model $q = q(\nu)$, we obtain a fixed point equation, where the sending rate satisfies $\nu = f(\nu)$. Figure 2.5 depicts the resulting closed-loop model

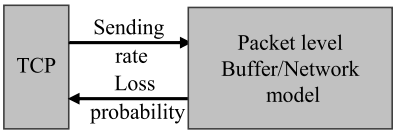

Figure 2.5: Abstraction model of TCP equilibrium, when the network is modeled on the packet level

Referring to figures 2.4 and 2.5, if we introduce differentiation inside a non-real-time traffic class using TCP, we condition flows to given priority levels and these priority levels in turn determine the level of service of the flows. The conditioning is based on comparing the sending rate to the contracted rate of the flow. The priority levels assigned to the packets of the flow depend on the ratio of these rates. On the other hand, the way we mark flows to priorities and how we handle the resulting priority aggregates inside the network, determines the packet drops experienced by the TCP flows and thus affects the resulting sending rate and bandwidth share.

Figure 2.6 depicts the modeling approach on the flow level. Flows in class l choose the priority level i so that the share of bandwidth $\theta(l, i)$ is maximized. The result is a game, where each flow tries to maximize its

sending rate constrained by the network state, i.e. the number of flows in each priority class $n(l,i)$. The maximization is constrained by the maximum sending rate allowed inside a priority level, e.g. for priority level $i+1$ it is $t(l,i)$.

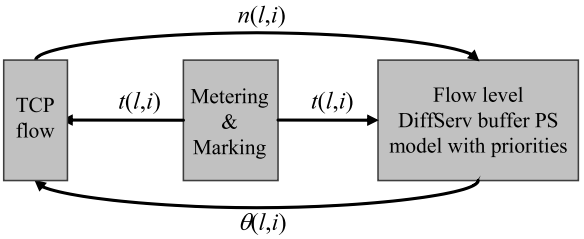

Figure 2.6: Abstraction model of TCP equilibrium, when the DiffServ network is modeled on the flow level

Figure 2.7 depicts the modeling approach on the packet level. The loss probability $q(l)$ of a flow group depends on the sending rate $\nu(l)$ of the TCP connections through a given buffer model and through a given metering and marking model assigning priority levels to achieve differentiation. The sending rate and loss probability for priority aggregate i are $\lambda(i)$ and $p(i)$, respectively.

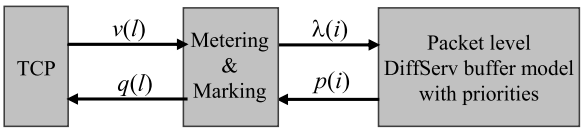

Figure 2.7: Abstraction model of TCP equilibrium, when the DiffServ network is modeled on the packet level

Using two different modeling levels, packet level and flow level, it is possible to study and quantify how delay and priority aggregates can be used to achieve relative differentiation of flows, how traffic with different service requirements, e.g., delay and drops, can be protected from each other and what the resulting bandwidth share or throughput of a flow is.

Packet level model for TCP and UDP flows

On the packet level, we construct a fixed point equation to calculate the TCP equilibrium sending rate, given that packets are classified into priority aggregates and that forwarding and discarding decisions at the buffer depend on the packet's priority level. Non-TCP flows are assumed to be streaming flows that do not adjust their sending rate to network conditions. However, the loss probability of non-TCP flows is dependent on the network condition and thus dependent on flow and packet priority level. Figure 2.8 gives a more detailed picture of the modeling approach introduced in figure 2.7.

In modeling the buffer, we assume that the packet arrival process of

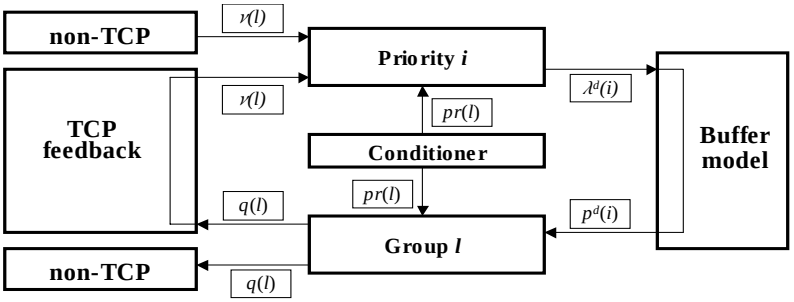

Figure 2.8: Packet level model

delay class d and priority aggregate i is Poisson, with intensity $\lambda^d(i)$. Therefore, our model does not cover the case of bursty packet arrivals. This restrictive assumption has to be made, in order to be able to derive the fixed point equation. Furthermore, we also assume that packet lengths are exponentially distributed, which is not necessarily the case for Internet packet traffic. However, the buffer model should be a reasonable choice, as we are studying aggregated arrivals of a fixed, sometimes large, number of flows to the buffer. The packet bursts of individual flows are evened out when the flows are aggregated and packet arrival intervals at the buffer can be considered exponentially distributed. As the focus of our packet level model is in modeling the interaction of the buffer forwarding and discarding mechanisms with TCP congestion control, the relationship between buffer management policies and the marking to priorities of packets plays a greater role in our model than the details of the packet arrival process to the buffer.

Let us, first consider UDP flows, to better understand the packet level modeling approach. UDP traffic is sent at a constant rate $\nu(l)$ and classified into appropriate priorities, resulting in aggregate arrival intensities $\lambda^{\mathrm{UDP}}(i)$. As an example, consider the case when all packets of a flow are marked to the same priority level, then $\lambda^{\mathrm{UDP}}(i)$, when the priority level is i, is

$$\lambda^{\mathrm{UDP}}(i) = \sum_{l \in \mathcal{L}^{\mathrm{UDP}}: pr(l)=i} n(l)\nu(l), \tag{2.5}$$

for all $i = 1, \ldots, I$. For the definition of variables, see section 2.4. The buffer either forwards or discards the arriving traffic. Depending on the buffer content some packets with a given priority level are discarded. The loss probabilities $p^{\mathrm{UDP}}(i)$ for priority levels $i = 1, \ldots, I$ are then solved from the buffer model assuming state dependent arrival intensities. If we have one buffer, we apply an $M/M/1/K$ model with state dependent arrival intensities. For the two buffer case, we apply a model of two dependent $M/M/1/K$ queues and the loss probabilities can only be solved numerically. We then need to regroup the loss probabilities that are a function of the priority levels i to loss probabilities as a function of user group l. To do this, we need to know how the marking scheme divided the flow into priority levels. Assuming per flow marking, the packet loss probability, $q(l)$, for a flow belonging to group l, is

$$q(l) = p^{\mathrm{UDP}}(pr(l)),\ l \in \mathcal{L}^{\mathrm{UDP}}. \tag{2.6}$$

For TCP flows that elastically adjust their sending rate according to the loss probability feedback of the buffer, we have a fixed point equation that we solve,

$$\nu(l) = \frac{1}{RTT(l)}\sqrt{2\frac{1-q(l)}{q(l)}},\ l \in \mathcal{L}^{\mathrm{TCP}}, \tag{2.7}$$

where $RTT(l)$ is the round-trip time of flows in group $l \in \mathcal{L}^{\mathrm{TCP}}$.

To solve this equation, we note that under per flow marking the priority levels, as given in equations (2.1) and (2.3), have discrete values depending on the sending rate $\nu(l)$ and thus the aggregate arrival intensity $\lambda^{\mathrm{TCP}}(i)$ is a discontinuous function. If we have per packet marking, equations (2.1) and (2.2), all packets of a flow do not jump to a new priority level, as only overflow packets are marked to lower priorities. Then packets are divided to substreams and the resulting functions are continuous.

In order to solve equation (2.7), when per flow marking is used, we need to make the aggregate arrival intensities piecewise continuous by parameterizing $\nu(l)$ and introducing a linear function between adjacent priorities. For a detailed explanation see [Nyb02]. Let $\mathbf{x} = \{x_l, l \in \mathcal{L}\}$, where x_l is an auxiliary variable used for parametrization. The priority function, equation (2.3), is made smooth and piecewise continuous by introducing a linear function between adjacent priorities levels. As an example, consider the parameterized expression $pr(l, x_l)$ for the priority level of group l. If $\lceil pr(l, x_l) \rceil > pr(l, x_l)$, the fraction $\lceil pr(l, x_l) \rceil - pr(l, x_l)$ of the flows' traffic is in priority level $pr(l, x_l)$ and the rest in priority level $pr(l, x_l) + 1$. Let $\nu(\mathbf{x}) = \{\nu(l, x_l), l \in \mathcal{L}^{\mathrm{TCP}}\}$, $\lambda^{\mathrm{TCP}}(i, \mathbf{x})$, let $q(l, \mathbf{x})$ denote the packet loss probability of flow group l and $p^d(i, \mathbf{x})$ the packet loss probability for delay class d at priority level i. The fixed point equation is then

$$\nu(l, x_l) = \frac{1}{RTT(l)}\sqrt{2\frac{1-q(l,\mathbf{x})}{q(l,\mathbf{x})}},\ l \in \mathcal{L}^{\mathrm{TCP}}. \tag{2.8}$$

The arrival intensity of priority level i using per flow marking is,

$$\lambda^d(i, \mathbf{x}) = \sum_{l \in \mathcal{L}^d : |pr(l,x_l) - i| < 1} n(l)\nu(l, x_l)(1 - \mid pr(l, x_l) - i \mid). \tag{2.9}$$

In the same fashion, the loss probability of flow group l is obtained from the loss probability $p^d(i)$ given by the buffer model,

$$\begin{aligned} q(l, \mathbf{x}) &= p^d(\lfloor pr(l, x_l) \rfloor, \mathbf{x})(\lfloor pr(l, x_l) \rfloor + 1 - pr(l, x_l)) \\ &+ p^d(\lfloor pr(l, x_l) \rfloor + 1, \mathbf{x})(pr(l, x_l) - \lfloor pr(l, x_l) \rfloor), l \in \mathcal{L}^d. \end{aligned} \tag{2.10}$$

In order to study how relative services is achievable by the given mechanisms, we solve equation (2.8) for the TCP equilibrium sending rate and calculate the throughput $\theta(l)$ of the flow

$$\theta(l) = \nu(l) \cdot (1 - q(l)).$$

By studying two flow groups, we are able to study the relationship between the ratio of contracted rates of flows to the ratio of throughputs of the flows

under varying scenarios. We study one buffer with TCP traffic, two buffers with TCP traffic and two buffers with TCP and UDP traffic. Furthermore, we compare per flow and per packet marking and dependent and independent discarding.

Flow level model for TCP and UDP flows

Assume that the packet handling in a bottleneck buffer can be approximated on the flow level as a Processor Sharing mechanism, which divides capacity equally among all flows, $\beta = 1/n$. On the flow level, we are not concerned with a more detailed model of packet arrivals to the buffer. The flows with the highest priority level I have, in our flow level model, a strict priority over all the other flows. Among these high priority flows, the bandwidth is divided as fairly as possible, i.e. each flow receives an equal share unless this exceeds the corresponding threshold rate. We are then able to study the conditioning mechanisms by modeling how the introduction of priority levels i and flow groups l affect the actual bit rate of a flow, $\beta(l,i)$.

Assume two TCP flow groups, $\phi(1) > \phi(2)$ and one buffer. The flows in the same priority level share bandwidth equally up to their threshold rate. Because the threshold rate for group 1 is higher than for group 2, $t(1,i) > t(2,i)$, group 1 flows share also among themselves the extra bandwidth left over by group 2 flows in the same priority level. If *per flow* marking is used, then the general rule to determine the bandwidth shares $\beta(l,i)$ for all the $n(l,i)$ flows in group l and at level i is, cf. Publication 3,

$$\begin{cases} \beta(1,i) = \min\{\max\{\dfrac{C(i)}{n(i)}, \dfrac{C(i) - n(2,i)t(2,i-1)}{n(1,i)}\}, t(1,i-1)\}, \\ \beta(2,i) = \min\{\dfrac{C(i)}{n(i)}, t(2,i-1)\}, \end{cases} \tag{2.11}$$

where $C(I) = C = 1$ and

$$C(i) = \max\{C(i+1) - n(1,i+1)t(1,i) - n(2,i+1)t(2,i), 0\}$$

refers to the remaining capacity for the flows with mark i, $n(i) = n(1,i) + n(2,i)$. Under *per flow* marking, since $t(l,0) = \infty$ all flows in the lowest priority receive the same share $\beta(l,1) = \frac{C(1)}{n(1)}$.

Assume now that, instead of two TCP flow groups sharing the same buffer, there is one group in each delay class, so that group 1 consists of UDP flows, sending at a fixed rate of $\nu(1)$ and group 2 consists of TCP flows. Assume further that packet discarding in the buffers is dependent. Now the UDP flows in group 1 divide the remaining capacity among themselves, but never receive more than their boundary rate. The TCP flows in group 2 then divide among themselves the capacity that is left over by the flows in higher priority levels and by the UDP flows or substreams of the same priority level. Assuming dependent discarding, UDP flows on lower priority levels than the TCP flows do not affect the bandwidth share of the

TCP flows. The equations are then for *per flow* marking[1],

$$\begin{cases} \beta(1,i) = \min\{\dfrac{C(i)}{n(1,i)}, t(1,i-1), \nu(1)\}, \\ \beta(2,i) = \min\{\max\{\dfrac{C(i) - n(1,i)\nu(1)}{n(2,i)}, 0\}, t(2,i-1)\}, \end{cases} \tag{2.12}$$

where

$$C(i) = \max\{C(i+1) - n(1,i+1)t(1,i) - n(2,i+1)t(2,i), 0\}$$

as before.

In general, to model the interaction between TCP and the marking mechanisms the priority levels $pr(l)$ need to be determined as a function of the number of flows $n(l)$ in each group l. These priority levels, in turn, determine uniquely the *network state* $\mathbf{n} = (n(l,i);\, l = 1,2;\, i = 1,2,\ldots,I)$, which in turn determines the actual bit rate of a flow, $\beta(l)$.

Flows belonging to the UDP group do not adjust their sending rate, while at level $pr(l)$, TCP group $l \in \mathcal{L}^{\mathrm{TCP}}$ decides to raise the level by one if $pr(l) < I$ and the resulting bandwidth share $\beta'(l, pr(l)+1)$ is higher than the original one $\beta(l, pr(l))$. If the level is not raised, group l decides to lower the level by one if $pr(l) > 1$ and the resulting bandwidth share $\beta'(l, pr(l)-1)$ is higher than the original one. Equilibrium is found whenever it is beneficial for all TCP groups to keep the current priority levels. The final bandwidth share $\theta(l)$ for a flow in group l will be $\beta(l,i)$ corresponding to the final level $pr(l) = i$. In principle, it may happen that the iteration does not end, but remains in a loop consisting of a number of states. However, our numerical experiments suggest that such a unique final state is always achieved.

Similar to the packet level, we study two flow groups and characterize the relationship between the ratio of contracted rates of flows to the ratio of bandwidth shares $\theta(l)$ of the flows under varying scenarios. We study bandwdith shares for two competing TCP traffic groups and for two TCP and UDP traffic groups. Furthermore, using the above models, we can compare the packet and flow level models to each other.

2.6 Summary of work and Publications 1–3

Publications 1–3 consider how traffic differentiation can be performed in the Internet when traffic interacts with differentiation mechanisms. On the packet level we model the interaction with the help of a detailed packet level TCP model and packet level buffer model of the traffic handling mechanism. On the flow level we base the models on the Processor Sharing model of TCP, assuming that flows inside a class divide bandwidth equally. With the help of these two models, we study how flow weights and traffic classification based on these weights at the edge of the network influence the resulting share of bandwidth. The investigated scenarios in each of the

[1]The equation in Publication 3 had $C(i) - n(1,i)t(1,i-1)$ instead of $C(i) - n(1,i)\nu(1)$. The original equation did not take into account the case where $t(1,i) < \nu(1) < t(i,i-1)$. See Appendix A.1 for more detail.

three publications are comparable and Publication 3 includes, compares and extends the packet level model of Publication 1 and the flow level model of Publication 2.

The packet level and flow level models both conclude that, in a network with both TCP and UDP traffic, to provide relative services the key differentiation mechanisms are per flow marking and dependent discarding. When modeling only TCP traffic, we assume one buffer and we do not consider discarding alternatives. Then, when modeling on the flow level, we conclude that packet marking is better, while on the packet level we cannot make such a clear distinction. The two modeling levels give qualitatively similar results and the minimum of the ratio $\theta(1)/\theta(2)$ is similar, but the differences are due to the numerical values of the maximum of $\theta(1)/\theta(2)$. When per flow marking is used, capacity is more often divided equally between flow groups, as at each priority level there is a network condition when flows belonging to different flow groups will have the same priority and bandwidth is divided equally. Under per packet marking this occurs only at the highest priority level and thus if we consider the minimum of the ratio $\theta(1)/\theta(2)$, per packet marking is better. However, when modeling on the packet level and assuming per flow marking, bandwidth is divided at times according to the ratio of the flow group weights while under per packet marking the ratio $\theta(1)/\theta(2)$ is always less than that of the weights. Thus no definite conclusion can be made. On the flow level, the maximum ratios are almost the same or at times under per packet marking closer to the ratio of the weights, and thus per packet marking seems to divide capacity better as shown by the numerical results in Publication 3. For more numerical results to illustrate this and a more thorough analysis of the effect of parameter choices on the results, see [Nyb02].

The basic assumptions in each publication is that the number of TCP flows is fixed and that a single backbone link is considered. When the buffer is modeled on the packet level, the packet arrival process is assumed to be Poisson. Further research could relax these assumptions, namely introducing a dynamic flow model, where the number of flows varies randomly. Extending the TCP model to a more detailed and dynamic traffic model would allow for the study of the stability of the interaction of the DiffServ mechanisms and the TCP congestion control. Considering larger networks on the flow level would also be an interesting topic for further research. Validation of the models through detailed packet level TCP simulations have not been done; in [NAS02] the simulations used a simplified TCP model of the congestion avoidance phase, and the main focus was in validating that the EWMA and Token Bucket marking mechanisms can be modeled as per flow and per packet marking, respectively.

Each publication is separately summarized below.

Publication 1: How to achieve fair differentiation

Publication 1 presents a packet level model to study how marking at the DiffServ boundary node and scheduling and discarding inside a DiffServ node affect the division of bandwidth between two delay classes: elastic TCP flows and streaming non-TCP flows. A static case, where the number of TCP flows is fixed, on a single backbone link is considered.

The packet level model includes expressions for aggregated arrival intensities based on the marking to priority levels, loss probabilities of these aggregates and resulting loss probabilities as seen by flows in different delay classes. These expressions together with a fixed point equation for TCP sending rate are used to solve the resulting equilibrium TCP sending rates and priority levels given the number of flows in the network.

The results show that, in a network with both TCP and non-TCP flows, only per flow marking together with dependent discarding thresholds across both delay classes is able to divide bandwidth fairly and in a TCP friendly way. Furthermore, the numerical results illustrate that there is no clear one-to-one relationship between the ratio of WFQ scheduler weights and flow group weights. Per flow marking and dependent discarding seems to be more powerful in controlling the resulting bandwidth shares. When non-TCP flows are in the highest priority level, i.e. the marking and discarding have little effect, the WFQ weights play a larger role, and they give a fixed upper bound for the bandwidth share achievable by non-TCP flows.

Publication 1 is written by the present author. The packet level model in section 2 is mainly by the present author, but joint work with the co-author. Numerical calculations are by the present author.

Publication 2: Flow level models of DiffServ packet level mechanisms

Publication 2 presents a flow level model to illustrate the effect of various DiffServ traffic conditioning methods on bandwidth sharing among elastic TCP flows. A static case, where the number of TCP flows is fixed, on a single backbone link is considered.

The flow level model includes expressions for resulting bandwidth share in terms of the number of flows in a priority level and the boundary rate of a priority level. The resulting equilibrium bandwidth share is solved by iteration, until the bandwidth share of TCP flows is in accordance with the number of flows in the network.

The results show that per packet marking yields a slightly better approximation of the relative-services approach compared to per flow marking. With per packet marking bandwidth shares are equal only when both flow groups are in the highest priority level. Under per flow marking, bandwidth shares may be equal whenever both flow groups are in the same priority level. For both marking schemes, however, the approximation is the better, the more there are priority levels.

Publication 2 is co-authored by the present author. The flow level model was proposed and section 2 written by the principal author, Dr. Samuli Aalto, but is joint work with the present author.

Publication 2 has not been subject to a peer review.

Publication 3: Differentiation and interaction of traffic: a flow level study

Publication 3 first reviews the main results of Publication 1, a packet level model and then introduces the flow level model of Publication 2 including the case of two delay classes. The models are compared with each other, with emphasis on evaluating the differentiation that can be achieved using DiffServ without admission control. The relative-services approach, i.e. the

rate of the flow should be in proportion to the contracted rate, is adopted. Differentiation mechanisms proposed, such as AF and Simple Integrated Media Access (SIMA) are modeled analytically with emphasis on modeling the interaction between various differentiation mechanisms and traffic flow types TCP and UDP.

The packet level model is presented in more detail including some considerations on the parametrization form needed to solve the resulting fixed point equation. For the flow level model, expressions for bandwidth shares is given for both the case of one delay class, i.e. only TCP traffic, and two delay classes, i.e. TCP and non-TCP traffic.

The results show that the flow level models are powerful in explaining how marking thresholds and mechanisms determine the differentiation in a network. The flow level models are able to produce results similar to the packet level models, and can be used as an abstraction of a DiffServ network and give system level guidelines for both network and differentiation design.

Publication 3 is written by the present author. The publication is based on Publications 1 and 2, with the flow model extended to include non-TCP traffic. The extended flow model is joint work by the authors. Numerical calculations are by the present author.

3 SCHEDULING AS MEANS TO DIFFERENTIATE INTERNET TRAFFIC

3.1 Introduction

In the previous chapter, we considered persistent TCP flows that always have data to send. Given such flows, we studied the resulting equilibrium state of the network, assuming that differentiation based on given reference weights is deployed in the network. In this chapter, we consider the service requirements of TCP flows that have a finite flow size. Such flows are called non-persistent flows. The resulting long-term equilibrium differentiation of the previous section does not address the difference in short-term delay requirements between short and long TCP flows and the effect of random arrivals of flows. Randomly arriving TCP flows may be long FTP or E-mail data transfers or short Web and Telnet transactions that are interactive, therefore, inside a TCP traffic delay class, the transfer time and loss requirements of flows may vary.

The perceived quality in terms of transfer time is more prone to high variations for short flows than for long flows. TCP applications react to congestion and losses by reducing their window sizes either with certain fluidity through fast retransmit procedures or after a timeout. For short connections with small window sizes, there are not enough packets to activate the duplicate ACK mechanism; a loss is then often detected only after a timeout and possibly after all data have been sent to the network [BPS+98, PA00]. A loss occurrence for a short TCP transfer may thus increase the transfer time manyfold, while, on the other hand, a reduction in the delay of the order of one second would be a significant improvement for short transactions.

We propose to differentiate between non-persistent TCP flows on the basis of the flow size. From a queuing theory point of view, it has been shown that choosing an appropriate scheduling policy may significantly improve the performance of the system. One of the classical results of queuing theory says that the Shortest Remaining Processing Time (SRPT) policy is optimal as it is able to reduce the overall mean delay of the flows [Sch68]. This is only applicable, however, if the flow sizes are known, which obviously is not the case in the current TCP/IP architecture.

When the flow sizes are not known, age-based scheduling policies can be used as an approximation. Age-based scheduling policies infer the remaining size of a job from the attained service of the job. Their optimality is critically dependent on the distribution of the job sizes. For example, if the distribution has a decreasing hazard rate, the more service a job has attained the smaller the probability it has received all the service it requires. One can then reduce the mean delay of the system by favoring short jobs that have a high probability of finishing. Internet flow-size distributions exhibit heavy tailed behavior and can often be modeled by a distribution with a decreasing hazard rate, e.g. Pareto distribution. For instance most TCP sessions, e.g. interactive sessions, are of small size but a small number of large flows, e.g. from data applications, are responsible for the largest amount of transferred data. As a consequence, age-based scheduling poli-

cies that favor jobs with small attained services would increase the overall mean throughput of jobs by reducing the mean delay of flows.

In designing scheduling mechanisms for Internet traffic, or for any other implementation in general, optimizing the overall mean delay of flows may not be enough. Predictability and fairness of the scheduling policy must also be taken into account. The reduction in the mean delay may come at the expense of larger variance, where large flows see a manifold increase in their delay. We propose to use threshold-based mechanisms and show through simulations and analytical considerations that they are able to reduce the maximum delay of flows compared to the Foreground Background (FB, also called Least Attained Service (LAS)) discipline, which strictly schedules jobs according to their age.

To implement age-based differentiation in the Internet, one needs to mark flows into priorities based on their age and schedule the flows based on this priority marking. We propose a stateless implementation for such a mechanism and show that threshold-based mechanisms are also easier to implement than mechanisms based on FB, which can be considered a threshold-based mechanism with an infinite number of levels.

We study more closely the family of age-based scheduling mechanisms called Multi Level Processor Sharing (MLPS) disciplines with thresholds, a given scheduling discipline inside a level and strict priority between levels. We model TCP traffic at the flow level and assume that resulting best-effort scheduling of TCP flows can be modeled as an M/G/1/PS queue. Building on this M/G/1/PS model, the packet-level priority discipline for TCP flows can be modeled on the flow level as an MLPS discipline, with PS used as the internal discipline within each level. We compare MLPS disciplines to the Processor Sharing (PS) discipline, and show that MLPS disciplines are always better than the PS discipline when the hazard rate of the flow-size distribution is decreasing. Furthermore, we give guidelines and proofs for choosing the thresholds and scheduling disciplines inside a level.

3.2 Related Research

Seminal work on one server scheduling disciplines was made in the 1960's and 1970's. Schrage [Sch68] proves that the Shortest Remaining Processing Time (SRPT) discipline minimizes the mean delay, i.e., the expected time in the system. The SRPT discipline, however, requires the knowledge of the remaining service times of all jobs. Recently SRPT has been proposed to be used in scheduling web requests [HBSBA03] and [SHB05], as there the size of the request is often known.

Harchol-Balter et al. have studied SRPT extensively in [BHB01] and more generally together with different scheduling policies in [HBSW02] [WHB03] and [WHB05]. In these papers, scheduling policies are classified according to slowdown, fairness and predictability. The work by Harchol-Balter et al. has highlighted the need for different optimality metrics in evaluating scheduling policies.

A new scheduling discipline, Fair Sojourn Policy (FSP), has been proposed in [FH03] by Friedman and Henderson. It is a preemptive scheduling policy, where the priority of a job is determined by the remaining pro-

cessing time the job would have if all jobs were scheduled under PS. The FSP scheduler then devotes full capacity to the job that would, under PS, have the smallest remaining processing time until it departs or until a new job with higher priority arrives. FSP is claimed to be fair and efficient, as no job performs worse under FSP than under PS.

In several recent works [NT02, GM01, GM02a, GM02b, RUKB02, RUKB03], the authors address the differentiation between short and long flows in the Internet. In [NT02], the authors suggest two approaches based on simulation studies. The first approach is application-based and it is proposed in the framework of Differentiated Services (DiffServ), with Assured Forwarding (AF) and RED with In and Out (RIO). This approach requires a non-trivial choice of the numerous AF and RIO parameters. The second approach is TCP state based, using each connection's window size and relying on the compliance of the end hosts. Furthermore, this approach requires tuning of weighted round robin (WRR) parameters, which again is neither evident nor robust.

In [GM01, GM02a, GM02b], the authors propose a two-class based architecture to provide better service to short TCP flows. At the edge router, state information is kept for active flows. Packets are marked with high priority if the current length of the flow is below some threshold and inside the network service differentiation is performed by RIO routers or WRR scheduling. They evaluate the gain obtained on mean response times through simulations. The results presented show reasonable gain in the average performance, but without an indication to the worst case performance or on the variance of the performance. In [GM02a, GM02b], the authors also discuss analytical modeling of their approach, but are only able to give approximate numerical results based on the use of the Kleinrock's conservation law [Kle76].

In [RUKB02], and ensuing work [RUKB03], the authors study the FB scheduling policy on the flow level and what the FB policy would produce in the context of a TCP network if packets from TCP flows were sorted in decreasing order of attained service.

Theoretical work on MLPS disciplines and scheduling disciplines based on the attained service of jobs is studied in [Yas87], [RS89], [RSY90] and [WBHB04]. Yashkov [Yas87] proves that FB minimizes the mean delay among such disciplines when the service time distribution is of type DHR (Decreasing Hazard Rate). Righter and Shantikumar [RS89] prove that, under the DHR condition, FB minimizes the queue length even stochastically. Righter et al. [RSY90] show that FB minimizes the mean delay when the service time distribution is of type IMRL (Increasing Mean Residual Life), which is a weaker condition than DHR.[1] Recently, Wierman et al. [WBHB04] prove that FB is better than PS with respect to the mean delay whenever the service time distribution is of type DHR, and vice versa if the service time distribution is of type IHR (Increasing Hazard Rate).

[1]There seems, however, to be a subtle deficiency in [RSY90] regarding the proof of FB's optimality. Instead of the truncated unfinished work $U_x^\pi(t)$, Righter et al. [RSY90] consider the corresponding untruncated random variable $V_x^\pi(t)$. Therefore, while FB minimizes $U_x^\pi(t)$, see Proposition 5 in Publication 5, FB does not minimize $V_x^\pi(t)$ nor $\overline{V}_x^\pi$. See Publication 5, section 3 for more detail.

3.3 Contribution

FB has been compared to the PS scheduling discipline, and it is a known result that FB is optimal in terms of minimizing the overall mean delay when service time distributions have a decreasing hazard rate. However, similar comparisons among age based disciplines in general have not been done. In Publication 5, we prove that the MLPS disciplines with two levels are better than PS with respect to the mean delay whenever the hazard rate of the service time distribution is decreasing, and vice versa if the hazard rate is increasing and bounded. In Publication 6, we show that these results are valid for a general MLPS discipline with any number of levels.

Implementation proposals to favor short flows exist, but the previous work has not thoroughly analyzed the underlying models nor simulated all relevant metrics, such as maximum delay. In Publication 4, we propose a stateless packet level implementation of a two-level MLPS discipline, called RuN2C, where the age of a TCP flow is inferred from the sequence numbers of TCP packets. Through simulations, we show that RuN2C is able to both reduce the mean delay compared to traditional tail drop routers and the maximum delay of TCP flows compared to a packet level implementation of FB.

In Publication 5 and Publication 6 we study and prove some of the theory on MLPS disciplines introduced in Publication 4. The numbering of the publications represents the order they were written and published, but it seems more natural, in the next section, to first present the theory behind the implementation proposal.

3.4 Age-based scheduling

Full information on flow size is not necessarily available, and age-based policies have therefore been proposed for differentiation in the Internet. Furthermore, due to the slow start function of TCP congestion control, packets of TCP flows should receive priority at the beginning of a connection when window sizes are small and TCP performance is sensitive to losses, regardless of the total size of the flow.

To study the effect of scheduling on reducing the overall mean delay of flows in the Internet, we need to consider how a new scheduling mechanism compares to the best-effort Internet and how it changes TCP fairness. On the flow level the Processor Sharing (PS) discipline and M/G/1/PS model is a natural point of comparison, since it has been proposed as an ideal model for the bandwidth sharing among TCP flows in a bottleneck router [HLN97, FBP+02]. PS is a fair policy, as all flows receive a mean delay in proportion to their size. Other scheduling policies that favor short flows to reduce the overall mean delay achieve it usually at the expense of long flows. Thus, in introducing a scheduling policy for TCP traffic, one needs to consider the fairness and predictability of the policy. Studying the mean delay, the maximum delay and the variance of the delay for flows, conditioned on the flow size, can give indications on fairness.

It is not enough to compare an age-based scheduling policy to PS, one needs to consider which age-based scheduling policy is best suited to TCP

traffic. Necessarily the policy that is optimal in terms of the mean delay, is not optimal in terms of fairness and predictability. Simulations and numerical results indicate that threshold-based policies, i.e. ones with a finite number of priority levels, are able to reduce the variance of the delay. A topic for further research is to quantify how the benefits in reducing the mean delay by an infinite level discipline is achieved by a finite class or threshold-based scheduling policies and, furthermore, how threshold-based policies offer better predictability for the mean delay of flows.

Multi Level Processor Sharing disciplines

To understand how age-based scheduling mechanisms can be used to reduce the mean delay, we consider a family of Multi Level Processor Sharing (MLPS) scheduling disciplines. Building on the M/G/1/PS model, we assume that inside a priority class bandwidth is shared at the flow level equally according to the PS discipline, thus the packet level priority discipline for TCP flows can be modeled on the flow level as an MLPS discipline with PS used as the internal discipline within each level. More precisely, an MLPS discipline is defined by a set of thresholds

$$0 = a_0 < a_1 < \cdots < a_N < a_{N+1} = \infty,$$

which define $N + 1$ levels. Jobs are classified into levels based on their age or attained service. Between these levels, a strict priority discipline is applied at the level with the lowest index, having the highest priority. Thus, those jobs that have attained service less than a_1 time units are served first. Inside a level any scheduling discipline could be used, but we restrict the analysis to Processor Sharing (PS) and Foreground Background (FB). As $N \to \infty$ and $a_i - a_{i-1} \to 0 \ \forall i$ the discipline becomes the Foreground Background (FB) discipline.

In evaluating proposed scheduling mechanisms, we have studied the mean delay $\overline{T}(x)$, conditioned on the job size x, and the overall mean delay $\overline{T}$. We have studied by numerical means some qualitative arguments considering the variance of the delay. Wierman and Harchol-Balter have classified scheduling policies according to fairness [WHB03] and predictability [WHB05], which they define in terms of mean and higher moments of the conditional delay respectively. For age-based policies only FB is included, as the analytical expressions for MLPS policies have only been solved for exponential job-size distributions.

To illustrate the range of policies under MLPS, we consider, as an example, a Two-Level Processor Sharing (TLPS) policy: PS+PS. In PS+PS the PS scheduling discipline is used at both levels. When appropriate we indicate the value of the threshold a between the two levels by using the notation PS+PS(a). Figure 3.1 shows the mean delay for flows for PS, PS+PS, and FB scheduling policies, as a function of their size. The flow-size distribution is bounded Pareto with parameters $\mathrm{BP}(13, 3500, 1.1)$. We observe that, even though large flows do not suffer much with PS+PS, the average time in the system is reduced significantly (see Figure 3.2). Even though the benefit FB and PS+PS provide to short flows are comparable, PS+PS causes a smaller degradation of the performance for large flows than FB.

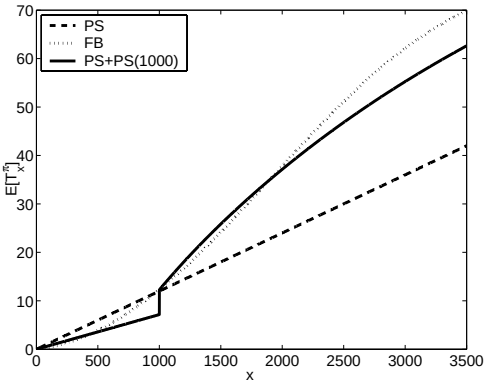

Figure 3.1: Mean conditional delay $\overline{T}^{\pi}(x)$, $\pi = \{\mathrm{PS}, \mathrm{FB}, \mathrm{PS}+\mathrm{PS}(1000)\}$ under service time distribution $\mathrm{BP}(13, 3500, 1.1)$ and load $\rho = 0.9$

In Figure 3.2, the mean delays of the two-level PS disciplines PS+PS(a) and FB+PS(a) are depicted as a function of the threshold a. In addition, the mean delay values for the PS and FB disciplines are also depicted. The service time distribution is bounded Pareto $\mathrm{BP}(13, 3500, 1.1)$.

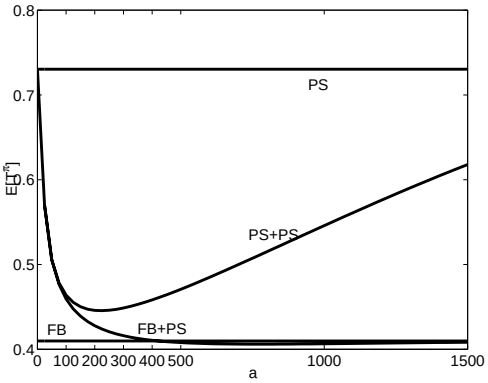

Figure 3.2: Mean delay $E[T^{\pi}]$, for $\pi = \{\mathrm{PS}, \mathrm{PS}+\mathrm{PS}(a), \mathrm{FB}+\mathrm{PS}(a), \mathrm{FB}\}$, as a function of a under service time distribution $\mathrm{BP}(13, 3500, 1.1)$ and load $\rho = 0.9$

These numerical examples show that MLPS disciplines should be better than PS, but worse than FB in minimizing the mean delay, when the hazard rate of the service time distribution is decreasing. This is indeed the case and has been proven in Publications 5 and 6 and is discussed below.

Mean delay analysis

We study the relationship between MLPS disciplines in terms of the mean delay. We present a summary of the main results of Publications 5 and 6 marked by a bullet. The proofs are based on a mixed approach of both

meanwise and pathwise arguments on the unfinished truncated work. Consider any work-conserving discipline π. Let $A(t)$ denote the number jobs that have arrived up to time t, $N^\pi(t)$ denote the number of jobs in the system at time t when discipline π is used, and denote the service times of these jobs by S_i and their attained services by $X_i^\pi(t)$, $i \in \{1, \ldots, N^\pi(t)\}$ under discipline π. Let us introduce the notation $a \wedge b = \min\{a, b\}$. Then $U_x^\pi(t)$, the unfinished truncated work at time t, is the sum of remaining truncated service times of those jobs in the system at time t with attained service less than x units,

$$U_x^\pi(t) = \sum_{i=1}^{A(t)} ((S_i \wedge x) - (X_i^\pi(t) \wedge x)). \tag{3.1}$$

For age-based scheduling disciplines in an M/G/1 queue, the mean unfinished truncated work can be described in terms of the mean conditional delay $\overline{T}(t)$ and the complementary probability distribution function $\overline{F}(t) = 1 - F(t)$, where $F(t) = \int_0^t f(y)\, dy$, and $f(t)$ is the density function (cf. [Kle76] equation (4.60)),

$$\overline{U}_x = \lambda \int_0^x \overline{F}(t)\overline{T}(t)\, dt. \tag{3.2}$$

Differentiating and averaging over x, we have the following result for the mean delay $E[T]$:

$$E[T] = \int_0^\infty \overline{T}(x) f(x)\, dx = \frac{1}{\lambda} \int_0^\infty (\overline{U}_x)' h(x)\, dx, \tag{3.3}$$

where $h(x) = f(x)/\overline{F}(x)$ is the hazard rate of the distribution.

We can then study the mean delay of scheduling disciplines in terms of unfinished truncated work and the hazard rate of the job-size distribution. The following results follow from equation (3.3). Let $\pi_1, \pi_2 \in \Pi$ be two scheduling disciplines (cf. Publication 5).

- If $\overline{U}_x^{\pi_1} \leq \overline{U}_x^{\pi_2}$ for all $x \geq 0$ and the hazard rate $h(x)$ is decreasing, then
$$E[T^{\pi_1}] \leq E[T^{\pi_2}].$$
- If $\overline{U}_x^{\pi_1} \leq \overline{U}_x^{\pi_2}$ for all $x \geq 0$ and the hazard rate $h(x)$ is increasing and bounded, then
$$E[T^{\pi_1}] \geq E[T^{\pi_2}].$$

A numerical example of $U_x^\pi(t)$ for a bounded Pareto service time distribution BP(13,3500,1.1) is depicted in figure 3.3. Here we see that for FB the mean unfinished truncated work is minimized and that for PS+PS it is always smaller than for PS.

A sufficient condition for $\overline{U}_x^{\pi_1} \leq \overline{U}_x^{\pi_2}$ for all $x \geq 0$ is that the mean delay curves, see e.g. figure 3.1, cross each other at most once (cf. Publication 5).

- If there exists some $x^* \geq 0$ such that $\overline{T}^{\pi_1}(x) \leq \overline{T}^{\pi_2}(x)$ for all $x < x^*$ and $\overline{T}^{\pi_1}(x) \geq \overline{T}^{\pi_2}(x)$ for all $x > x^*$, then $\overline{U}_x^{\pi_1} \leq \overline{U}_x^{\pi_2}$ for all $x \geq 0$.

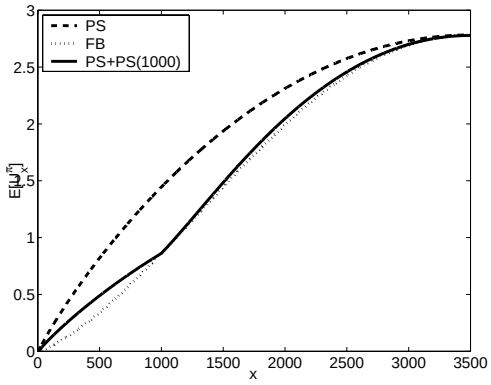

Figure 3.3: Mean unfinished truncated work $\overline{U}_x^\pi$, $\pi = \{\mathrm{PS}, \mathrm{FB}, \mathrm{PS}+\mathrm{PS}(1000)\}$ under service time distribution $\mathrm{BP}(13, 3500, 1.1)$ and load $\rho = 0.9$

The following result then follows for a TLPS discipline (cf. Publication 5).

- The mean unfinished truncated work under TLPS disciplines is less than under PS.

For MLPS disciplines, at every threshold there is a discontinuity in the mean delay curve, and though the above sufficient condition seems to hold for MLPS disciplines in general, the proofs are based on another approach (cf. Publication 6).

- For $x \geq a_N$, there exists $x^* \geq a_N$ such that $\overline{T}^{\pi_1}(x) \geq \overline{T}^{\pi_2}(x)$ implies $\overline{U}_x^{\pi_1} \leq \overline{U}_x^{\pi_2}$ for all $x \geq x^*$. For $x \leq a_N$ we can show that for the sample paths $U_x^{\pi_1} \leq U_x^{\pi_2}$ (see below). Then by induction and the proof for the TLPS case above, the meanwise case follows.

The following result then holds (cf. Publication 6).

- The mean unfinished truncated work under all MLPS disciplines is less than under PS.

For some cases, we can relax the assumption of Poisson arrivals and give pathwise arguments on the unfinished truncated work U_x^π, which are stronger than the meanwise arguments $\overline{U}_x^\pi$ above. All proofs considering delay are based on the mean delay (cf. equation (3.3)).

Consider equation (3.1), which can be written as

$$U_x^\pi(t) = \sum_{i=1}^{A(t)} (S_i \wedge x) - \int_0^t \sigma_x^\pi(u)\, du, \tag{3.4}$$

where $\sigma_x^\pi(t)$ refers to the rate at which such jobs that have attained service less than x units are served at time t. The first term in (3.4) is independent of the scheduling discipline and it suffices to study $\sigma_x^\pi(t)$ in order to compare scheduling disciplines. We have, for any t,

$$\begin{cases} \sigma_x^\pi(t) = 0, & \text{if } N_x^\pi(t) = 0, \\ \sigma_x^\pi(t) \leq 1, & \text{if } N_x^\pi(t) > 0. \end{cases} \tag{3.5}$$

Here $N_x^\pi(t)$ is the number of jobs with attained service less than x at time t and $N^\pi(t)$ is the total number of jobs in the system at time t.

For example, for the PS discipline this rate is as follows:

$$\sigma_x^{\mathrm{PS}}(t) = \begin{cases} 0, & \text{if } N_x^{\mathrm{PS}}(t) = 0, \\ \dfrac{N_x^{\mathrm{PS}}(t)}{N^{\mathrm{PS}}(t)} \leq 1, & \text{if } N_x^{\mathrm{PS}}(t) > 0. \end{cases} \tag{3.6}$$

FB is a work conserving scheduling discipline that gives full priority to jobs with attained service less than x. Thus

$$\sigma_x^{\mathrm{FB}}(t) = \begin{cases} 0, & \text{if } N_x^{\mathrm{FB}}(t) = 0, \\ 1, & \text{if } N_x^{\mathrm{FB}}(t) > 0. \end{cases} \tag{3.7}$$

From this it follows that FB is always better than PS or any MLPS discipline regarding the sample path of the truncated unfinished work.

By defining order relations between disciplines used inside an MLPS level n, $\pi_n \preceq \pi'_{n'}$, we can compare MLPS disciplines that have the same thresholds, but different scheduling disciplines inside a given level (cf. Publication 5).

- Assume that $\pi \in \mathrm{MLPS}(a_1, \ldots, a_N)$, $\pi' \in \mathrm{MLPS}(a'_1, \ldots, a'_{N'})$, $n \in \{1, \ldots, N+1\}$ and $n' \in \{1, \ldots, N'+1\}$ such that for a fixed $x > 0$,
$$a_{n-1} = a'_{n'-1} \leq x \leq a_n = a'_{n'}.$$
  (i) If $\pi_n = \pi'_{n'}$, then $U_x^\pi(t) = U_x^{\pi'}(t)$ for all $t \geq 0$.
  (ii) If $\pi_n \preceq \pi'_{n'}$, then $U_x^\pi(t) \leq U_x^{\pi'}(t)$ for all $t \geq 0$.

In our work, we study FB and PS scheduling policies inside a level, obeying the ordering

$$\mathrm{FB} \preceq \mathrm{FB}, \quad \mathrm{FB} \preceq \mathrm{PS}, \quad \mathrm{PS} \not\preceq \mathrm{FB}, \quad \mathrm{PS} \preceq \mathrm{PS}.$$

Thus, if $\pi_n = \mathrm{FB}$ and $\pi'_{n'} = \mathrm{PS}$, then $U_x^\pi(t) \leq U_x^{\pi'}(t)$ for all $t \geq 0$.

If we restrict the analysis to MLPS disciplines with N levels, and PS at every level, denoted by NPS, we can show that adding a new level, with threshold a_N reduces the unfinished truncated work for all thresholds $x \leq a_N$, assuming that the original and new discipline follow the same scheduling rule for jobs with attained service time less than a_{N-1} (cf. Publication 6).

- Let $N \geq 1$, $\pi \in (N+1)$PS with thresholds $\{a_1, \ldots, a_N\}$, and $\pi' \in N$PS with thresholds $\{a_1, \ldots, a_{N-1}\}$. Then $U_x^\pi(t) \leq U_x^{\pi'}(t)$ for all $x \leq a_N$ and $t \geq 0$.

Though PS+PS is better than PS regarding the mean value of the unfinished truncated work, the pathwise version of this result is not true.

MLPS scheduling policies are able to reduce the mean delay compared to the PS policy, with FB being the optimal policy under decreasing hazard rates of the job-size distribution. Furthermore, for PS+PS it has been shown in [AAB05] that the mean conditional delay has an asymptote as the job size

goes to infinity, which is not the case for FB, as shown in Publication 4. In view of this, the theoretical considerations on scheduling policies show that PS+PS is able to reduce the mean delay compared to PS and has better predictability for large jobs than FB, even though FB is optimal in terms of the mean delay.

3.5 Scheduling TCP packets based on sequence number

TCP sequence numbers are used for reliable transfer to keep track of which bytes have been sent and which have been acknowledged. The sequence number of a TCP segment is the byte-stream number of the first byte contained in the packet incremented from one segment to the next by the number of bytes in the packet's workload. The initial sequence number is chosen at random from a 32 bit field to minimize the possibility that a segment from a terminated connection between two hosts, still present in the network, is mistaken for a valid segment of an ongoing connection between the two hosts. Furthermore, for security reasons, misbehaving users should not be able to infer the initial sequence numbers of other connections.

Sequence number lookup

The difference between the initial sequence number and the sequence number of the newest TCP packet represents the age of the connection. We thus propose that routers infer the served amount of bytes by only looking at the current TCP sequence number. If the mechanism is to be used in routers which may not allow for sequence number lookup, the lookup may be performed already in the edge routers that may then set the Type of Service (TOS) bits in the IP header to indicate the age of the connection by using priorities. As our implementation retains the per byte nature of TCP flow control, the number of high priority bytes of each connection will be fixed and independent of the Maximum Segment Size (MSS) of the connection. In packet count based schemes, e.g. [GM01], the number of high priority segments is fixed and the number of high priority bytes depends on the MSS. Sequence number lookup provides well-defined fairness with respect to other users and at the same time it prevents misbehaving users from getting preferential treatment since, whatever the value of MSS a connection chooses, only the number of bytes defines the degree of preferential treatment obtained.

Sequence number lookup provides a method to infer the age of the flow, while retaining the characteristics of TCP transfer. With this in mind, to propose age-based scheduling, we wish also to take into account how the use of one or a finite number of threshold compares to FB, a policy with an infinite number of levels. We propose the use of threshold-based mechanisms. There are three reasons for this. First, TCP transfers are byte oriented but scheduling is performed per packet, the priority mechanism should therefore be coarse grained with a finite number of levels so that the number of bytes falling inside a level are at least larger than the largest MSS possible in the network. Secondly, as discussed in the previous section, reducing the overall mean delay is not the only criterion in proposing scheduling disciplines. The reduction in the overall mean delay

is achieved almost always at the expense of longer delays for long flows and higher variability in delay. Threshold-based policies penalize long flows less than FB, but seem to be able to reduce the overall mean delay compared to PS. Thirdly, a two-level policy is easier to implement than FB, as one bit is needed to keep priority information in the header and the routers need only to maintain two queues and consider if a packet has priority or not.

Implementation at routers

We propose a scheduling mechanism based on PS+PS, with one threshold th. The threshold th must be chosen in such a way that short flows benefit from the differentiation mechanism while keeping the load of high priority low in order not to harm longer flows. To find a compromise solution, we note two facts. First, short TCP flows are prone to timeouts upon packet losses, see e.g. [BPS+98]. The impact of timeouts can be extremely important on the response time of short flows since its minimum value is 1 s [PA00]. Thus, to avoid timeouts, packets should be given priority until the congestion window reaches a value of 3 or 4. This is the case if approximately 8 packets are transmitted. This corresponds to a threshold th of 12 KBytes for MSS of 1460 KBytes. Second, since TCP flow sizes are heavy tailed, even though flows shorter than 8 packets may represent a significant number of TCP flows, they will account for only a small proportion of the total load. Hence, giving priority to short flows, will not lead to the starvation of longer flows.

In order to infer from the TCP sequence number the amount of bytes sent, we need to know the initial sequence numbers of the TCP flows. We propose to divide the 32 bit TCP sequence number space into R bits, giving use 2^R Possible Initial Numbers (PIN). They are equally spaced in the sequence number field ranging from 0 to $2^{32} - 1$, and denoted by PIN_i where $i \in \{1, \ldots, 2^R\}$. These numbers should be spaced not too far to allow for the initial sequence number of a TCP connection to be picked sufficiently at random. They must be spaced far enough to reduce the probability (or the occurrence rate) of running over to the next PIN.

Let th be the value in bytes of the threshold, packets for which the sequence number is between PIN_i and $\text{PIN}_i + th$ will be classified as priority packets in contrast to packets for which the sequence number lies between $\text{PIN}_i + th$ and PIN_{i+1}, where PIN_{i+1} is the next possible initial number (see Figure 3.4).

With this structure, the sequence number expressed in binary code is divided into three parts (see Figure 3.4). The $R = 32 - (L + TH)$ most significant bits are picked at random, providing 2^R different PIN values. The next L bits and the following TH bits, where $TH = \log_2 th$, are set to zero when the TCP connection is established. This scheme permits one to infer the priority of the packet by a simple mask-based comparison, since when the sequence number belongs to the low priority range $[\text{PIN}_i + th, \text{PIN}_{i+1}]$ the L intermediate bits will be equal to 0. Packet sequence numbers from a given connection will overflow to the next PIN after 2^{L+TH} bytes which can be chosen quite large.

Note that, since the sequence number is counted in bytes, the interval 2^{TH} is divided into MSS (Maximum Segment Size) disjoint sets. This

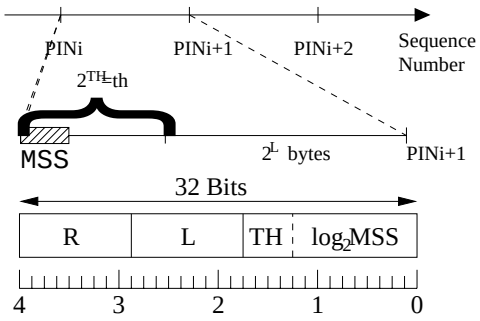

Figure 3.4: Structure of the sequence number for RuN2C

allows us to choose at random also the first $\log_2$ MSS bits of the initial sequence number. We can thus increase the randomness of the scheme allowing both the first R bits and the $\log_2$ MSS least significant bits to be random.

The packet level implementation, which we call RuN2C requires maintaining a two-class priority queue and knowledge of the value of the threshold parameter between high and low priority queues. If standard TCP connections share RuN2C enabled routers, the TCP connections will not benefit from the priority queue except if they randomly start their sequence number to do so. Our analytical models and simulation results show that long TCP connections obtain equivalent performance with tail drop routers and RuN2C routers as long as the load of the priority traffic remains small. One may expect for a similar result to stand for non-adapted short TCP flows if the volume of priority traffic is sufficiently small. RuN2C routers can thus be considered compatible with current TCP implementations. In addition, TCP connections implementing RuN2C are not affected by tail drop routers. We thus conclude, that RuN2C can be progressively deployed in a network, as it is always beneficial for all adapted TCP connections and never worse than the current implementation for non-adapted connections.

3.6 Summary of work and Publications 4–6

In Publications 4–6 we consider MLPS disciplines with thresholds and compare them to the PS discipline. We show that MLPS disciplines are always better than the PS discipline when the hazard rate of the flow-size distribution is decreasing. Through theoretical considerations and simulations we give guidelines and proofs for choosing the thresholds and scheduling disciplines inside a level.

Recently, it has been proven in [AA06], that for a flow-size distribution with decreasing hazard rate the more levels there are in an MLPS discipline the better it is. Most recent unpublished results on the mean delay show that FB is not always optimal when the flow-size distribution is of type Increasing Mean Residual Life (IMRL), but MLPS disciplines with FB or PS inside a level are better than PS also for distributions of type IMRL.

Thus far we have considered optimality in relation to overall mean delay and job-size distributions with decreasing hazard rate, then FB is optimal in minimizing the mean delay. Two interesting questions to answer remain. Are MLPS policies with a finite number of thresholds better than FB in relation to the variance or higher moments of the conditional delay, i.e., should a resulting larger mean delay for jobs under TLPS be accepted if at the same time the variance of FB can be reduced? Our simulation results seemed to hint towards this, but it remains to be proven analytically.

Secondly, could age-based policies be defined in terms of general classes of distributions and loss measures, and thus prove the optimality of the policies in a more general way? An example of a general definition is the Smallest Rank (SR) sequencing policy named Shortest Imminent Processing Time (SIPT) in Kleinrock [Kle76]. Sevcik has proven in [Sev74] that SR is optimal in minimizing expected total loss, defined as the sum of expected completion times of jobs, under full information of arrivals and conjectured that it is also optimal under random arrivals. However, as this is an age-based policy the job-size distribution or at least some relevant moments of the distribution beyond the mean have to be known, in order to determine the optimal priority scheme. In SR, the processor services the job that has the current minimum rank. Under full preemption the rank of a job is updated at regular preemption intervals ϵ or at every instant $\epsilon \to 0$. At time t, the rank of a job $R(\epsilon, t)$ is the ratio of the expected investment of processor time $I(\epsilon, t)$ to the expected payoff $P(\epsilon, t)$, i.e. given that the job has received t units of service, the rank is the ratio of expected remaining service time in the next interval ϵ and the probability that the job finishes during this interval of length ϵ,

$$R(\epsilon, t) = \frac{I(\epsilon, t)}{P(\epsilon, t)} = \frac{\overline{X}_{\epsilon+t} - \overline{X}_t}{F(\epsilon + t) - F(t)}.$$

For a job that requires a total service time of t, $F(t)$ is the distribution function of the flow service times and $\overline{X}_t$ is the mean service time of a job truncated at t units of service with limit $\overline{X}$.

Assuming that $\epsilon \to \infty$, the rank becomes the mean residual life time of the job

$$R(t) = \frac{\overline{X} - \overline{X}_t}{1 - F(t)} = E[X - t \mid X > t].$$

It remains to be shown, how the rank and loss measure of SR relate to the mean delay under FB and other age-based scheduling disciplines, given conditions on the job-size distribution. Classes of interest include distributions with decreasing hazard rate, and distributions with weaker conditions such as New Worse than Used in Expectation (NWUE) or Increasing Mean Residual Life (IMRL). Note that, a distribution with decreasing hazard rate is of type NWUE and IMRL, but the converse is not true.

To study more thoroughly the implementation proposal, simulations of the network in the case of RuN2C compliant traffic and non-compliant, i.e. standard TCP flows, would be interesting. It could be that even though some flows do not start their initial sequence numbers at predefined PINs, they might still experience shorter delays when sufficiently many connections are RuN2C compliant.

Publication 4: Differentiation Between Short and Long TCP Flows: Predictability of the Response Time

Publication 4 proposes a packet level, stateless, and threshold-based scheduling mechanism for TCP flows, called RuN2C. An implementation of this mechanism is considered. The behavior of RuN2C is compared with FB-based mechanisms through analytical models and simulations. As an analytical model for RuN2C, a two-level priority Processor Sharing PS+PS discipline is used.

Both simulations and analytical results show that RuN2C has a very beneficial effect on the mean delay of short flows, while treating large flows as the current TCP implementation does. In contrast, FB-based mechanisms can lead to pathological behavior in extreme cases, as illustrated by considering the maximum delay of flows and the number of flows in the network.

Publication 4 is a result of joint work with all the co-authors. The implementation proposal of section 3 is by the co-authors. Simulations in section 4 and all numerical calculations are by the present author, based on joint planning of simulation scenarios.

Publication 5: Two-Level Processor-Sharing Scheduling Disciplines: Mean Delay Analysis

Publication 5 considers a mean delay analysis of MLPS scheduling disciplines in the context of M/G/1 queues.

The proofs are based on a mixed approach of both meanwise and pathwise arguments on the unfinished truncated work. FB is always better than any PS or any MLPS discipline regarding the sample path of the unfinished truncated work. On the other hand, PS+PS is better than PS regarding the mean value of the unfinished truncated work; the pathwise version of this result is, however, not true. The analysis shows that two-level PS (TLPS) disciplines, e.g. FB+PS and PS+PS, are better than PS, but worse than FB scheduling when the hazard rate of the job-size distribution is decreasing. The analysis is further extended to study local optimality within a level of an MLPS scheduling discipline.

We also point out a common mistake of earlier proofs, which instead of considering the unfinished truncated work, have considered the unfinished untruncated work.

Publication 5 is joint work by the authors. Numerical calculations are by the present author.

Publication 6: M/G/1/MLPS compared to M/G/1/PS

Publication 6 extends the pathwise and meanwise results of TLPS disciplines of Publication 5 to the general case of MLPS disciplines. We prove that, for M/G/1 queues, MLPS disciplines are better than the Processor Sharing discipline with respect to the mean delay whenever the hazard rate of the service time distribution is decreasing.

Publication 6 is a follow-up to publication 5. It is joint work by the authors, with the main contribution to the new proofs by the co-authors.

4 BLOCKING IN MULTICAST NETWORKS

4.1 Introduction

In the previous sections, we considered IP networks where capacity is not reserved for a flow. In this case, scarce capacity may delay the transfer of some number of individual packets while other segments of the flow are transferred; however, in theory, no connection is entirely blocked. In this section, we consider streaming traffic in a network, where capacity is reserved for the whole route. In such a case, we are interested in the probability that a connection is blocked, i.e. that there is not enough capacity for the connection on at least one link on the route.

Bandwidth may be saved when one transmission reaches many different end-users. For a unicast transmission, the same information streams are replicated separately for each user. A multicast transmission uses less bandwidth, as a single copy of the information stream is delivered to each branch leading to at least one user. This kind of streaming multicast transmission is particularly suited to distribution-type applications, such as distribution of radio or TV programs, or, for example, push-type services in 3G mobile networks, where certain information is delivered to all the subscribers of the service.

The multicast one-to-many connections form a tree-type structure. For unicast traffic, there exists algorithms for calculating blocking probabilities in hierarchical multiservice access networks [Ros95]. The basic idea behind the algorithms is the convolution truncation recursion. The truncation principle states that we can convolve independent leaf link distributions assuming no capacity constraints, and then truncate and normalize the distribution according to the capacity constraints. For multicast traffic, there is always at most one copy of the transmission on a given link, while, for a unicast transmission we need to know how many copies are on the link. Thus, to suit multicast traffic characteristics, the convolution operation is modified, and we introduce the OR-convolution operation. If both unicast and multicast traffic are present in the network, the truncation step is also modified. As a result, we present new algorithms suited to calculating blocking probabilities in multicast networks.

The algorithm can be used with many different user population models. All user models can be expressed in terms of a single-user model that we introduce. The finite population models can then be constructed using our proposed algorithm and, more specifically, using the OR-convolution operation on a given set of single users. We are thus able to broaden the class of finite user population models. Furthermore, at the limit, we obtain the infinite user population model with exponentially distributed arrivals.

4.2 Related research

Most of the previous research has focused on blocking probabilities in multicast capable switches. Kim [Kim96] studies blocking probabilities in a

multirate multicast switch. Three stage switches are studied by Yang and Wang [YW98] and Listanti and Veltri [LV98]. Stasiak and Zwierzykowski [SZ98] study blocking in an ATM node with multicast switching nodes carrying different multi-rate traffic, unicast and multicast, using Kaufman-Roberts recursion and Reduced-Load Approximation. Admission control algorithms are studied in [SY97].

The paper by Almeroth and Ammar [AA97] investigates multicast group behavior in the MBone. From this data, they conclude that interarrival times are exponentially distributed while group membership duration times are exponentially distributed for small networks and Zipf distributed for larger networks. The study of intersession data suggests that simultaneous sessions, where a user subscribes to more than one channel, occur, but not frequently.

Multicast is also used in parallel computing applications, e.g, the parallel algorithm for the Fast Fourier Transform (FFT) and write update/invalidate in directory based cache coherence protocols. The paper by Yang [Yan96] discusses multicast for parallel computing applications.

The model by Chan and Geraniotis [CG96] is a multipoint-to-multipoint model for a network with subscriptions arriving from single users. They explore the tradeoff between blocking and dropping in multicast networks. The model is based on two main characteristics of video transmission in a multicast network: receivers share part of the connections and a source may transmit video signals at the same time to a group of receivers with different receiving capabilities and/or requirements. The model is based on subband coding, where a signal is encoded into several layers each containing a part of the information. The lowest layer contains essential information for transmitting a low quality version of the video signal. Higher layers add information to the signal. As long as all the lower level signals are received, a higher layer adds to the resolution of the signal. The traffic class is defined by the triplet: physical path (p), source node (s), and class of video quality (t). The behavior of each user is modeled as a two state Markov chain, with unique transition rates defined for each traffic class triplet. Chan and Geraniotis give a closed form expression for the time blocking probability in the network, but as prohibitive computational effort would be required they use the Reduced Load Approximation (RLA) for numerical calculations.

The model by Karvo et al. [KVAM98] is a point-to-multipoint model for a network, with subscriptions arriving from an infinite user population. The source is called the service center and it can offer a variety of channels, e.g. TV-channels. The users subscribing to the network may, at any time, join or leave any of the several multicast trees, each carrying a separate multicast transmission or channel offered by the source. The behavior of the user population defines the state probabilities at the links of the tree-structured network. The user population is assumed infinite and the requests to join the network arrive according to a Poisson process. Karvo et al. give exact solutions for blocking probabilities for the special case of all but one link in the network having infinite capacity. A network with more than one link having finite capacity is considered in [KVAM01], there RLA is used as well. The single link case was further broadened by Bousseta and Beylot

[BB99] by including both multirate multicast and unicast traffic.

After the present author's contribution to the field, papers based on Publication 8 have been published mainly by Karvo and the other two co-authors of Publication 8. In [AV00], the algorithm from Publication 8, called the basic algorithm, is simplified, by considering that the channels are statistically indistinguishable and no specific channel requirements are needed. In [AKV02], the combinatorial algorithm of [AV00] is extended to include multiple groups of statistically indistinguishable channels. The basic algorithm is extended in [KAV01] and [KAV02] to the case where multicast transmissions use two-layer and multiple layer hierarchical coding, respectively. Fast simulation techniques to simulate blocking in multicast networks are presented in [LKV01] and [Kar02].

4.3 Contribution

The analytical model introduced in [KVAM01] is extended to the network case in Publication 7 by introducing an exact algorithm. The new algorithm for calculating blocking probabilities in multicast networks is based on the known algorithm for unicast hierarchical networks [Ros95]. In Publication 7, we present a mathematical model for calculating blocking probabilities in a multicast network with any number of finite capacity links and an infinite user population. We also consider the case with background unicast traffic on the links of the network. The main result is an exact algorithm for calculating the time and call blocking probabilities. The accuracy of the RLA algorithm, considered in [KVAM01], is compared to the exact algorithm. In Publication 8, we extend the user population model to include arbitrary sized, i.e. also finite, user populations, give a more unified account of the algorithm and provide a proof for the insensitivity properties of the results.

4.4 Convolution and truncation in tree-structured multicast networks

Assume a tree-structured multicast network depicted in figure 4.1, with one root link J, leaf links and intermediate links connecting the root link to the leaf links. Behind a leaf link, we have a user or a group of users $u \in \{1, \ldots, U\}$. Each user has a selection of I channels to choose from and the capacity requirement of channel i is d_i, $i \in \{1, \ldots, I\}$. A new subscription to channel i requires the capacity d_i only on those links where channel i is not yet carried. The state space is then of size 2^{UI}, as any channel at any user may be on or off.

Consider first unicast traffic in a tree network, then the state space is only of the size 2^U, as we only need to consider the link occupancy in terms of the capacity required by each user.

A new unicast connection may be blocked if there is not enough capacity on one or more links to accept the connection. The blocking probability is a function of two sets of state probabilities: the set of non-blocking states and the set of allowed states. In order to calculate the blocking probability in a network with finite link capacities we would need to keep track of all the possible states in the network. By using recursive convolution and

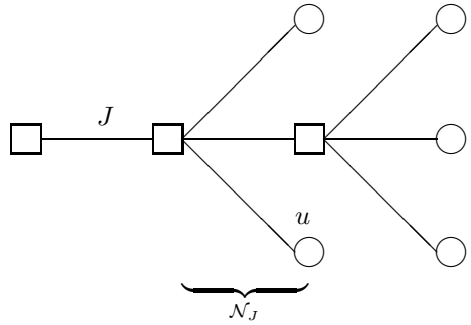

Figure 4.1: Example tree-structured multicast network. Users u are represented by circles, network nodes by boxes, and the source is the leftmost node, from where the root link J originates. $\mathcal{N}_J$ denotes the set of downstream links connected to the root link J.

truncation steps, the computational complexity can be reduced.

Assume all links have infinite capacity. We can deduce the link occupancy probabilities by convolving links pairwise starting from the leaf link up to the root link. Then the complexity is linear in terms of the number of users, as we need $U-1$ convolution operations to calculate the root link occupancy probabilities.

When the link capacities are finite, we use the truncation principle to recursively convolve and truncate the state probabilities. The truncation principle states that we can convolve independent leaf link distributions assuming no capacity constraints and then truncate according to the capacity constraints. This procedure of convolution and truncation is repeated step by step until the root link J. The truncated state probabilities at the root link are the joint state probabilities of all links in the network with the capacity restrictions taken into account. The original unicast algorithm [Ros95] for a tree network with U users is then able to calculate the blocking probability in $U-1$ convolution operations compared to the brute force approach of going through all 2^U states.

For multicast traffic we can apply the same procedure, with the main exception that for each user we need to keep track of which individual channel the user is subscribed to, and thus the computational complexity is still exponential in terms of the number of channels. For unicast traffic, we are solely interested in the probability that the end user occupies a given amount of capacity, but for multicast traffic, we only have at most one copy of each channel on each link. Throughout the network, we thus need the individual channel information, in order not to calculate the required capacity of a channel more than once.

The algorithms we consider are all based on this truncation principle, the approach requires calculating two state probability sums: one over the set of non-blocking states $\tilde{\mathbf{\Omega}}_k$ and another one over the set of allowed states

$\tilde{\mathbf{\Omega}}$. The time blocking probability b_k^t for traffic class k is then,

$$b_k^t = 1 - P(\tilde{\mathbf{X}} \in \tilde{\Omega}_k) = 1 - \frac{P(\mathbf{X} \in \tilde{\Omega}_k)}{P(\mathbf{X} \in \tilde{\Omega})} = \frac{\sum_{\mathbf{y}} Q_j^k(\mathbf{y})}{\sum_{\mathbf{y}} Q_J(\mathbf{y})}, \qquad (4.1)$$

where

$\mathbf{X}$	=	network state without capacity constraint,
$\tilde{\mathbf{X}}$	=	network state with capacity constraint,
$\tilde{\mathbf{\Omega}}$	=	set of states satisfying capacity constraints,
$\tilde{\mathbf{\Omega}}_k$	=	set of non-blocking states for traffic class k,

$\mathbf{y}$ is the link state and J is the root link. In the case of unicast traffic, $k = u$ and in the case of multicast traffic, $k = (u, i)$. The probabilities $Q_J(\mathbf{y})$ and $Q_j^k(\mathbf{y})$ are calculated recursively for traffic class k,

$$Q_j(\mathbf{y}) = \begin{cases} T_j \pi_j(\mathbf{y}) & , \text{ if } j \text{ leaf link} \\ T_j[\bigotimes_{n \in \mathcal{N}_j} Q_n](\mathbf{y}) & , \text{ otherwise,} \end{cases}$$

and

$$Q_j^k(\mathbf{y}) = \begin{cases} T_j^k \pi_j(\mathbf{y}) & , \text{ if } j \text{ leaf link} \\ T_j^k[\bigotimes_{n \in \mathcal{N}_j} Q_n^k](\mathbf{y}) & , \text{ otherwise.} \end{cases}$$

The truncation operators T_j and T_j^k and the OR-convolution $\bigotimes$ are explained below, $\pi_j(\mathbf{y})$ are the leaf link state probabilities and $\mathcal{N}_j$ is the set of downstream links connected to link j.

As mentioned above, the convolution operation is different for multicast traffic than for unicast traffic. Our contribution is an exact algorithm to calculate blocking probabilities in a multicast network with only multicast traffic or multicast and background unicast traffic. The algorithm is based on equation (4.1) with modifications to the convolution operation and the truncation operator.

For unicast traffic, the convolution operation is defined in terms of the link occupancy c instead of the link state $\mathbf{y}$,

$$[Q_{j_1} \otimes Q_{j_2}](c) = \sum_{c_{j_1}=0}^{c} Q_{j_1}(c_{j_1}) Q_{j_2}(c - c_{j_1}).$$

For multicast traffic, we define the OR-convolution as

$$[Q_{j_1} \otimes Q_{j_2}](\mathbf{y}) = \sum_{\mathbf{y}_{j_1} \oplus \mathbf{y}_{j_2} = \mathbf{y}} Q_{j_1}(\mathbf{y}_{j_1}) Q_{j_2}(\mathbf{y}_{j_2}),$$

where $\mathbf{y}_{j_1} \oplus \mathbf{y}_{j_2} = \mathbf{y}$ is the OR-operation taken componentwise.

The truncation operator T_j discards the state probabilities of those states that do not satisfy the capacity constraint of the link j. The truncation operator T_j^k discards the state probabilities of those states that do not satisfy the capacity constraint when a new connection k is admitted on the link j.

Set $\tilde{\mathcal{S}}_j$ refers to those states of link j for which the capacity constraint is satisfied. The corresponding truncation operator acting on any real valued function f is defined as

$$T_j f(\mathbf{y}) = f(\mathbf{y}) 1_{\mathbf{y} \in \tilde{\mathcal{S}}_j}.$$

For T_j^k, the set is replaced by a tighter set $\tilde{\mathcal{S}}_j^k$ where the extra capacity required by the new connection k is taken into account.

If independent background unicast traffic is present in the network, the truncation operation must be modified to alter the state probabilities of the states that satisfy the capacity restriction of the link, as the available capacity on the link depends on the amount of non-multicast traffic on the link. This means altering the set of admissible states and associating a probability for each state in the set, based on the state probabilities of unicast traffic. Assume that multicast and unicast traffic are independent. The truncation operator is modified by taking into account the joint probability of the two traffic types, i.e. replacing $1_{\mathbf{y} \in \tilde{\mathcal{S}}_j}$ by the probability $P(Z_j \leq C_j - \mathbf{d} \cdot \mathbf{y})$ that unicast traffic occupies the capacity left over by multicast traffic. Vector $\mathbf{d}$ holds the capacity requirements of all multicast channels. The modified truncation operator is then

$$\hat{T}_j f(\mathbf{y}) = f(\mathbf{y}) P(Z_j \leq C_j - \mathbf{d} \cdot \mathbf{y}),$$

and $\hat{T}_j^k$ is modified correspondingly. Recall that for multicast traffic $k = (u, i)$.

4.5 Multicast user models

In order to use the truncation principle and thus our algorithm, we need the state probabilities of the leaf links $\pi_j(\mathbf{y})$. We restrict the analysis to multicast user models. If the user model forms a reversible process, then the truncation of the state space only alters the normalization constant and the truncation principle applies. As an example if the idle and holding time distributions are exponential, the state vector is reversible. In the appendix of Publication 8, we show that the truncation principle applies for general channel holding times and user idle times resulting in a reversible semi-Markov process.

Consider a model where the user population consists of a single user. There is an idle state, where the user is not connected to any channel. The user u can either be in the idle state 0 or connected to some channel i. All transitions by user u are made via the idle state. Note that, the mean idle time can be arbitrarily small, in which case the user switches almost directly from one channel to another. However, by having the idle state, we emphasize the fact that the capacity reservation related to the current channel has to be first released, before a new reservation can be made.

The transition rate from state 0 to state i is denoted by $\lambda_{u,i} = \alpha_i \lambda_u$, where α_i is the probability of choosing channel i among the channel set $\mathcal{I}$, $\sum_{i \in \mathcal{I}} \alpha_i = 1$. The transition rate from state i to state 0 is denoted by μ_i. The model proposed is thus a Markov process with $I + 1$ states. The state transition diagram is shown in figure 4.2.

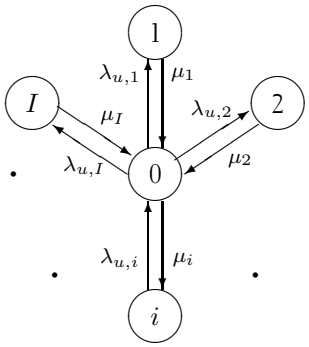

Figure 4.2: The Markov process used to model user behavior.

For the truncation principle to hold, the channel holding times as well as the user idle times can be generally distributed with means $1/\mu_i$ and $1/\lambda_u$, respectively.

Assume that behind each leaf link we have one user and the link at the next level combines N leaf links. Assume further that the leaf link has enough capacity to admit any channel request of the user and a user only requests one channel at a time. Then the second level link sees a user population of size N. Thus using our algorithm, we can define a set of user models based on the single-user model.

We consider the single-user model and the following derivatives

- The channel user model is constructed as a special case of the single-user model. The single user is defined to be connection specific, i.e. having only the possibility of choosing a given channel i. The leaf link distribution is then constructed by combining the I channel specific single users.

- The most general user population model, the finite user population model is a model for a population consisting of N users each having the whole selection of channels to choose from.

- Lastly the user population for a finite number of users results in the infinite user population as the number of users N tends to infinity.

4.6 Summary of work and Publications 7 and 8

Publications 7 and 8 present an exact algorithm for calculating blocking probabilities in a tree-structured multicast network. Using the convolution principle the computational complexity of the exact algorithm is linear in terms of the number of users. The exact algorithm, however, depends critically on the number of channels in the network, in contrast to the RLA algorithm. The two algorithms were compared in Publication 7. For the small networks considered the RLA algorithm overestimated the blocking probability by 15%. The larger the network and the more links there are on a route the larger the overestimation of the RLA algorithm is. Thus the

exact algorithm is powerful for even large networks, as long as the number of channels is sufficiently small.

Since Publications 7 and 8, various extensions have been developed to the original exact algorithm, as discussed in section 4.2. In [AV00], the algorithm is simplified, by considering that the channels are statistically indistinguishable and no specific channel requirements are needed. In [AKV02], the combinatorial algorithm of [AV00] is extended to include multiple groups of statistically indistinguishable channels. The basic algorithm is extended in [KAV01] and [KAV02] to the case where multicast transmissions use two-layer and multiple layer hierarchical coding, respectively. Fast simulation techniques to simulate blocking in multicast networks are presented in [LKV01] and [Kar02]. These aforementioned papers have studied the field in detail, not leaving much room for further research.

Publication 7: An Exact Algorithm for Calculating Blocking Probabilities in Multicast Networks

Publication 7 deals with tree-structured point-to-multipoint networks, where users from infinite user populations at the leaf nodes subscribe to a variety of channels, offered by one source. The users joining the network form dynamic multicast connections that share the network resources.

An exact algorithm for calculating end-to-end call blocking probabilities for dynamic connections in a multicast network is presented. The algorithm is based on the truncation principle for calculating blocking probabilities in hierarchical multiservice unicast traffic networks, where link occupancy distributions are alternately convolved and truncated. The resource sharing of multicast connections requires the modification of the convolution step by using a new type of convolution, the OR-convolution.

The exact algorithm for end-to-end call blocking probabilities enables the study of the accuracy of earlier results based on Reduced Load Approximation. The numerical results show, that the RLA algorithm overestimated the blocking probability in a network with many finite links.

The model is further extended, by modifying the truncation operation, to include background traffic, allowing the analysis of networks carrying mixed traffic e.g. multicast and unicast traffic.

The mathematical model and resulting algorithm in Publication 7 are joint work by the authors. The paper is written and the numerical analysis in sections 3 and 4 was performed by the present author.

Publication 8: An Exact End-to-End Blocking Probability Algorithm for Multicast Networks

Publication 8 considers the calculation of blocking probabilities in multicast trees with dynamic membership. The algorithm for calculating end-to-end call blocking exactly is from Publication 7. In Publication 8, it is presented in a more detailed manner and extended to be applied for several different user population models. The user population models are described in detail.

Publication 8 is based on Publication 7. It was written as joint work by the authors, with the appendix written by Dr. Samuli Aalto.

REFERENCES

[AA97] K.C. Almeroth and M.H. Ammar. Multicast group behavior in the Internet's multicast backbone (MBone). *IEEE Communications Magazine*, 35(6):124–129, 1997.

[AA06] S. Aalto and U. Ayesta. Mean delay analysis of Multi Level Processor Sharing disciplines. *Accepted to IEEE INFOCOM 2006*, Barcelona, April 2006.

[AAB05] K. Avrachenkov, U. Ayesta, and P. Brown. Batch arrival Processor Sharing with application to Multilevel Processor Sharing scheduling. *Queueing Systems*, 50(4):459-480, 2005.

[AKV02] S. Aalto, J. Karvo, and J. Virtamo. Calculating blocking probabilities in multicast loss systems. In *Proceedings of SPECTS 2002*, pages 833–842, San Diego, CA, July 2002.

[AV00] S. Aalto and J.T. Virtamo. Combinatorial algorithm for calculating blocking probabilities in multicast networks. In *Proceedings of the 15th Nordic Teletraffic Seminar*, pages 23–34, Lund, Sweden, August 2000.

[BB99] K. Bousseta and A.-L. Beylot. Multirate resource sharing for unicast and multicast connections. In *Proceedings of Broadband Communications '99*, pages 561–570, 1999.

[BBC+98] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, and W. Weiss. *An Architecture for Differentiated Service*, December 1998. RFC 2475.

[BCS94] R. Braden, D. Clark, and S. Shenker. *Integrated Services in the Internet Architecture: an Overview*, June 1994. RFC 1633.

[BHB01] N. Bansal and M. Harchol-Balter. Analysis of SRPT scheduling: Investigating unfairness. In *Proceedings of ACM SIGMETRICS 2001*, pages 279–290, 2001.

[BPS+98] H. Balakrishnan, V. Padmanabhan, S. Seshan, M. Stemm, and R. Katz. TCP behavior of a busy web server: analysis and improvements. In *Proceedings of IEEE INFOCOM 1998*, San Francisco, CA, 1998.

[CG96] W.C. Chan and E. Geraniotis. Tradeoff between blocking and dropping in multicasting networks. In *IEEE International Conference on Communications 1996*, pages 1030–1034, 1996.

[CGM+02] C. Casetti, M. Gerla, S. Mascolo, M. Sansadidi, and R. Wang. TCP Westwood: end-to-end congestion control for

wired/wireless networks. *Wireless Networks Journal*, 8:467–479, 2002.

[DSR99] C. Dovrolis, D. Stiliadis, and P. Ramanathan. Proportional Differentiated Services: Delay differentiation and packet scheduling. In *Proceedings of ACM SIGCOMM 1999*, pages 109–120, 1999.

[ECP] O. Elloumi, S. De Cnodder, and K. Pauwels. Usefulness of three drop precedences in Assured Forwarding service. IETF Draft July 1999.

[FBP+02] S. Ben Fredj, T. Bonald, A. Proutière, G. Régnié, and J. Roberts. Statistical bandwidth sharing: A study of congestion at flow level. In *Proceedings of ACM SIGCOMM 2002*, pages 111–122, Pittsburgh, PA, 2002.

[FF99] S. Floyd and K. Fall. Promoting the use of end-to-end congestion control in the Internet. *IEEE/ACM Transactions on Networking*, 7(4):458–472, August 1999.

[FH03] E.J. Friedman and S.G. Henderson. Fairness and efficiency in web server protocols. In *Proceedings of ACM SIGMETRICS 2003*, pages 229–237, San Diego, CA, 2003.

[FJ93] S. Floyd and V. Jacobson. Random early detection gateways in congestion avoidance. *IEEE/ACM Transactions on Networking*, 1(3):397–413, 1993.

[Flo91] S. Floyd. Connections with multiple congested gateways in packet-switched networks part 1: One-way traffic. *Computer Communication Review*, 21(5):30–47, October 1991.

[Flo03] S. Floyd. *HighSpeed TCP for Large Congestion Windows*, December 2003. RFC 3649.

[GDJL] M. Goyal, A. Durresi, R. Jain, and C. Liu. Performance analysis of Assured Forwarding. IETF Draft October 1999.

[GK99] R.J. Gibbens and F.P. Kelly. Resource pricing and the evolution of congestion control. *Automatica*, 35:1969–1985, 1999.

[GM01] L. Guo and I. Matta. The war between mice and elephants. In *Proceedings of the 9th IEEE International Conference on Network Protocols ICNP'01*, 2001.

[GM02a] L. Guo and I. Matta. Differentiated control of web traffic: A numerical analysis. In *Proceedings of SPIE ITCOM'2002: Scalability and Traffic Control in IP Networks*, Boston, MA, 2002.

[GM02b] L. Guo and I. Matta. Scheduling flows with unknown sizes: Approximate analysis. Technical Report BU-CS-2002-009, Boston University, March 2002.

[HA02] N. Hegde and K.E. Avrachenkov. Service differentiation and guarantees for TCP-based elastic traffic. In *QofIS/ICQT'02*, October 2002.

[HBSBA03] M. Harchol-Balter, B. Schroeder, N. Bansal, and M. Agrawal. Size-based scheduling to improve web performance. *ACM Transactions on Computer Systems*, 21(2):207–233, 2003.

[HBSW02] M. Harchol-Balter, K. Sigman, and A. Wierman. Asymptotic convergence of scheduling policies with respect to slowdown. *Performance Evaluation*, (49):241–256, 2002.

[HBWW99] J. Heinanen, F. Baker, W. Weiss, and J. Wroclawski. *Assured Forwarding PHB Group*, June 1999. RFC 2597.

[HG04] Y. Huang and R. Guérin. A simple FIFO-based scheme for differentiated loss guarantees. In *Proceedings of the 12th IEEE IWQOS 2004*, pages 96–105, 2004.

[HKL+00] J. Harju, Y. Koucheryavy, J. Laine, S. Saaristo, K. Kilkki, J. Ruutu, H. Waris, J. Forsten, and J. Oinonen. Performance measurements and analysis of TCP flows in a Differentiated Services WAN. In *Proceedings of the Local Computer Networks conference (LCN 2000), Tampa, Florida, USA*, pages 1 – 10, November 2000.

[HLN97] D.P. Heyman, T.V. Lakshman, and A.L. Neidhardt. A new method for analysing feedback-based protocols with applications to engineering web traffic over the Internet. In *Proceedings of ACM SIGMETRICS 1997*, pages 24–38, Seattle, WA, 1997.

[JNP99] V. Jacobson, K. Nichols, and K. Poduri. *An Expedited Forwarding PHB*, June 1999. RFC 2598.

[JWL04] C. Jin, D. Wei, and S.H. Low. FAST TCP: Motivation, architecture, algorithms, performance. In *Proceedings of IEEE INFOCOM 2004*, volume 4, pages 2490–2501, March 2004.

[Kar02] J. Karvo. Efficient simulation of blocking probabilities for multi-layer multicast streams. In *Proceedings of Networking 2002*, Pisa, Italy, May 2002.

[KAV01] J. Karvo, S. Aalto, and J. Virtamo. Blocking probabilities of two-layer statistically indistinguishable multicast streams. In *Proceedings of ITC-17*, pages 769–779, Salvador da Bahia, Brazil, December 2001.

[KAV02] J. Karvo, S. Aalto, and J. Virtamo. Blocking probabilities of multi-layer multicast streams. In *Proceedings of HPSR 2002*, pages 268–277, Kobe, Japan, May 2002.

[Kel99] F. Kelly. Mathematical modelling of the Internet. In *Proc. of Fourth International Congress on Industrial and Applied Mathematics*, pages 105–116, 1999.

[Kel03] T. Kelly. Scalable TCP: Improving performance in high-speed Wide Area Networks. *ACM SIGCOMM Computer Communication Review*, 33(2), 2003.

[Kim96] C.K. Kim. Blocking probability of heterogeneous traffic in a multirate multicast switch. *IEEE Journal on Selected Areas in Communications*, 14(2):374–385, 1996.

[Kle76] L. Kleinrock. *Queueing Systems, vol. 2.* John Wiley and Sons, 1976.

[KR98] K. Kilkki and J. Ruutu. Simple Integrated Media Access (SIMA) with TCP. In *the 4th INFORMS Telecommunications conference Boca Raton, FL, USA*, March 1998.

[KVAM98] J. Karvo, J. Virtamo, S. Aalto, and O. Martikainen. Blocking of dynamic multicast connections in a single link. In *Proceedings of Broadband Communications '98*, pages 473–483, 1998.

[KVAM01] J. Karvo, J. Virtamo, S. Aalto, and O. Martikainen. Blocking of dynamic multicast connections. *Telecommunication Systems*, 16(3–4):467–481, 2001.

[LC02] J. Liebherr and N. Christin. Rate allocation and buffer management for Differentiated Services. *Computer Networks*, 40(1), August 2002.

[LKV01] P. Lassila, J. Karvo, and J. Virtamo. Efficient importance sampling for monte carlo simulation of multicast networks. In *Proceedings of IEEE INFOCOM 2001*, pages 432–439, Anchorage, Alaska, April 2001.

[LSLH00] J. Laine, S. Saaristo, J. Lemponen, and J. Harju. Implementation and measurements of Simple Integrated Media Access (SIMA) network nodes. In *Proceedings for IEEE ICC 2000*, pages 796–800, June 2000.

[LV98] M. Listanti and L. Veltri. Blocking probability of three-stage multicast switches. In *IEEE International Conference on Communications*, pages 623–629, 1998.

[MBDM99] M. May, J. Bolot, C. Diot, and A. Jean Marie. Simple performance models for Differentiated Services schemes for the Internet. In *Proceedings of IEEE INFOCOM 1999*, pages 1385–1394, March 1999.

[MSMO97] M. Mathis, J. Semke, J. Mahdavi, and T. Ott. The macroscopic behavior of the TCP congestion avoidance algorithm. *Computer Communications Review*, 27(3), July 1997.

[NAS02] E. Nyberg, S. Aalto, and R. Susitaival. A simulation study on the relation of DiffServ packet level mechanisms and flow

level QoS requirements. In *International Seminar, Telecommunication Networks and Teletraffic Theory*, St. Petersburg, Russia, January 2002.

[NEC00] L.V. Nguyen, T. Eyers, and J.F. Chicaro. Differentiated service performance. In *Procedings of Fifth IEEE Symposium on Computers and Communications ISCC 2000*, pages 328–333, 2000.

[NT02] W. Noureddine and F. Tobagi. Improving the performance of interactive TCP applications using service differentiation. *Computer Networks Journal*, 40(1):19–43, September 2002.

[Nyb02] E. Nyberg. How to achieve fair differentiation: Relating flow level QoS requirements to DiffServ packet level mechanisms. Licentiate thesis, Helsinki University of Technology, Networking Laboratory, September 2002.

[PA00] V. Paxson and M. Allman. *Computing TCP's Retransmission Timer*, November 2000. RFC2988.

[PC04] E.-C. Park and C.-H. Choi. Proportional bandwidth allocation in DiffServ networks. In *Proceedings of IEEE INFOCOM 2004*, volume 3, pages 2038–2049, 2004.

[PFTK98] J. Padhye, V. Firoiu, D. Towsley, and J. Kurose. Modeling TCP throughput: A simple model and its empirical validation. In *Proceedings of ACM SIGCOMM 1998*, pages 303–314, 1998.

[PG93] A.K. Parekh and R.G. Gallagher. A generalized processor sharing approach to flow control in integrated services networks: The single-node case. *IEEE/ACM Transactions on Networking*, 1(3):344–357, 1993.

[PSN99] P. Pieda, N. Seddigh, and B. Nandy. The dynamics of TCP and UDP interaction in IP-QoS Differentiated Service networks. In *Proceedings of 3rd Canadian Conference on Broadband Research (CCBR)*, November 1999.

[RK98] J. Ruutu and K. Kilkki. Simple Integrated Media Access – a comprehensive service for future Internet. In *Performance of Information and Communications Systems (PICS)*, pages 321–332, 1998.

[Ros95] K.W. Ross. *Multiservice Loss Models for Broadband Telecommunication Networks*. Springer-Verlag, London, 1995.

[RS89] R. Righter and J.G. Shanthikumar. Scheduling multiclass single server queueing systems to stochastically maximize the number of successful departures. *Probability in the Engineering and Informational Sciences*, 3:323–333, 1989.

[RSY90] R. Righter, J.G. Shanthikumar, and G. Yamazaki. On extremal service disciplines in single-stage queueing systems. *Journal of Applied Probability*, 27:409–416, 1990.

[RUKB02] I. Rai, G. Urvoy-Keller, and E. Biersack. Size-based scheduling with differentiated services to improve response time of highly varying flows. In *Proceedings of the 15th ITC Specialist Seminar, Internet Traffic Engineering and Traffic Management*, Wurzburg, Germany, 2002.

[RUKB03] I. Rai, G. Urvoy-Keller, and E. Biersack. Analysis of LAS scheduling for job size distributions with high variance. In *Proceedings of ACM SIGMETRICS 2003*, pages 218–228, San Diego, CA, 2003.

[Sch68] L.E. Schrage. A proof of the optimality of the shortest remaining processing time discipline. *Operations Research*, 16(3):687–690, 1968.

[Sev74] K.C. Sevcik. Scheduling for minimum total loss using service time distributions. *Journal of the ACM*, 21(1):66–75, January 1974.

[SHB05] B. Schroeder and M. Harchol-Balter. Web servers under overload: How scheduling can help. *To appear in ACM Transactions on Internet Technologies*, 2005.

[SNT+00] S. Sahu, P. Nain, D. Towsley, C. Diot, and V. Firoiu. On achievable service differentiation with token bucket marking for TCP. In *Proceedings of ACM SIGMETRICS 2000*, pages 23–33, June 2000.

[SPG97] S. Shenker, C. Partridge, and R. Guerin. *Specification of Guaranteed Quality of Service*, September 1997. RFC 2212.

[STK99] S. Sahu, D. Towsley, and J. Kurose. A quantitative study of Differentiated Services for the Internet. In *Proc. IEEE Global Internet'99*, pages 1808–1817, December 1999.

[SY97] N. Shacham and H. Yokota. Admission control algorithms for multicast sessions with multiple streams. *IEEE Journal on Selected Areas in Communications*, 15(3):557–566, 1997.

[SZ98] M. Stasiak and P. Zwierzykowski. Analytical model of ATM node with multicast switching. In *Proceedings of Mediterranean Electrotechnical Conference*, pages 683–687, 1998.

[WBHB04] A. Wierman, N. Bansal, and M. Harchol-Balter. A note on comparing response times in the M/G/1/FB and M/G/1/PS queues. *Operations Research Letters*, 32(1):73–76, 2004.

[WHB03] A. Wierman and M. Harchol-Balter. Classifying scheduling policies with respect to unfairness in an M/GI/1. In *Proceedings of ACM SIGMETRICS 2003*, pages 238–249, San Diego, CA, 2003.

[WHB05] A. Wierman and M. Harchol-Balter. Classifying scheduling policies with respect to higher moments of conditional response time. In *Proceedings of ACM SIGMETRICS 2005*, pages 229–240, 2005.

[Wro97a] J. Wroclawski. *Specification of the Controlled-Load Network Element Service*, September 1997. RFC 2211.

[Wro97b] J. Wroclawski. *The Use of RSVP with IETF Integrated Services*, September 1997. RFC 2210.

[Yan96] Y. Yang. A class of interconnection networks for multicasting. In *Proceedings of IPPS '96*, pages 796 –802, 1996.

[Yas87] S.F. Yashkov. Processor-sharing queues: Some progress in analysis. *Queueing Systems*, 2:1–17, 1987.

[YR01] I. Yeom and A.L.N. Reddy. Modeling TCP behavior in a Differentiated Services network. *IEEE/ACM Transactions on Networking*, 9(1):31–46, 2001.

[YW98] Y. Yang and J. Wang. On blocking probability of multicast networks. *IEEE/ACM Transactions on Networking*, 46(7):957–968, 1998.

PUBLICATIONS

Publication 1 Eeva Nyberg and Samuli Aalto. How to achieve fair differentiation. In *Proceedings of Networking 2002*, pages 1178–1183, Pisa, Italy, May 2002.

Publication 2 Samuli Aalto and Eeva Nyberg. Flow level models of DiffServ packet level mechanisms. In *Proceedings of the 16th Nordic Teletraffic Seminar*, pages 194–205, Espoo, Finland, August 2002.

Publication 3 Eeva Nyberg and Samuli Aalto. Differentiation and interaction of traffic: a flow level study. In *Proceedings of International Workshop, Art-Qos 2003*, pages 276–290, March 2003.

Publication 4 Konstantin Avrachenkov, Urtzi Ayesta, Patrick Brown, and Eeva Nyberg. Differentiation between short and long TCP flows: predictability of the response time. In *Proceedings of IEEE INFOCOM 2004*, volume 2, pages 762–773, March 2004.

Publication 5 Samuli Aalto, Urtzi Ayesta, and Eeva Nyberg-Oksanen. Two-Level Processor-Sharing scheduling disciplines: Mean delay analysis. In *Proceedings of ACM SIGMETRICS - PERFORMANCE 2004*, pages 97–105, June 2004.

Publication 6 Samuli Aalto, Urtzi Ayesta, and Eeva Nyberg-Oksanen. M/G/1/MLPS comapred to M/G/1/PS. *Operations Research Letters*, 33(5):519–524, September 2005.

Publication 7 Eeva Nyberg, Jorma Virtamo, and Samuli Aalto. An exact algorithm for calculating blocking probabilities in multicast networks. In *Proceedings of Networking 2000*, pages 275–286, Paris, France, May 2000.

Publication 8 Eeva Nyberg, Jorma Virtamo, and Samuli Aalto. An exact end-to-end blocking probability algorithm for multicast networks. *Performance Evaluation*, 54(4):311–330, December 2003.