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

Johanna Antila

Scheduling and quality differentiation in Differentiated Services

Thesis submitted in partial fulfillment of the requirements for the degree of Master of Science in Technology

Espoo, Finland, 7th April 2003

Supervisor: Professor Samuli Aalto (pro tem)
Instructor: Lic.Tech. Marko Luoma

HELSINKI UNIVERSITY OF TECHNOLOGY

ABSTRACT OF THE MASTER'S THESIS

Author:	Johanna Antila		
Name of the thesis:	Scheduling and quality differentiation in Differentiated Services		
Date:	7th April 2003	**Number of pages:**	81
Department:	Department of Electrical and Communications Engineering		
Professorship:	S-38 Teletraffic Theory		
Supervisor:	Prof. Samuli Aalto (pro tem)		
Instructor:	Lic.Tech. Marko Luoma		

During the last decade the Internet has developed into a public multi service network that should be able to support heterogeneous applications and customers with diverse requirements. For this reason, quality of service (QoS) provisioning in the Internet has gained increasing attention.

General service architectures have been proposed in the literature for providing QoS. At the moment, the Differentiated Services (DiffServ) architecture seems to be the most promising solution due to its scalability. In DiffServ, common resources are allocated among service classes. The mechanism that is primarily responsible for the allocation is packet scheduling.

The DiffServ architecture sets certain limits to the services that can be offered to customers. However, a lot of freedom is still left for the operators. For example, the operators can choose what kind of differentiation models they will use and how these models will be implemented.

This thesis studies issues related to quality differentiation and scheduling in Differentiated Services networks. The first part of the thesis provides a literature survey about QoS guarantees, architectures and mechanisms, with the focus on scheduling disciplines.

In the second part of the thesis possible service approaches and differentiation models for an operator are examined and schedulers for implementing these models are proposed based on a qualitative analysis. Some of the proposed differentiation models and schedulers are evaluated with simulations. The simulations study both at aggregate and session/flow level what kind of service the schedulers provide and how well they are able to support the differentiation model that they are designed for. In addition, the simulations give insight into the effect of traffic mapping on the differentiation.

Keywords: Scheduling, QoS, Differentiation, Simulation

TEKNILLINEN KORKEAKOULU DIPLOMITYÖN TIIVISTELMÄ

Tekijä:	Johanna Antila		
Työn nimi:	Skedulointi ja laadun eriyttäminen eriytettyjen palvelujen verkossa		
Päivämäärä:	7. huhtikuuta 2003	**Sivuja:**	81
Osasto:	Sähkö- ja tietoliikennetekniikan osasto		
Professuuri:	S-38 Teleliikenneteoria		
Työn valvoja:	Ma. professori Samuli Aalto		
Työn ohjaaja:	Tekn. Lis. Marko Luoma		

Viimeisen vuosikymmenen aikana Internet on kehittynyt julkiseksi monipalveluverkoksi, jonka pitäisi pystyä tukemaan sovelluksia ja asiakkaita, joiden vaatimukset vaihtelevat. Tästä syystä palvelun laadun eriyttäminen Internetissä on herättänyt paljon kiinnostusta.

Palvelun laadun mahdollistamiseksi on ehdotettu erilaisia palveluarkkitehtuureja. Tällä hetkellä eriytettyjen palveluiden (DiffServ) arkkitehtuuri vaikuttaa lupaavimmalta ratkaisulta skaalautuvuutensa ansiosta. DiffServ-arkkitehtuurissa yhteiset resurssit jaetaan palveluluokkien kesken. Mekanismi, joka pääasiassa vastaa resurssien allokoinnista, on pakettien skeduloija.

DiffServ arkkitehtuuri asettaa tietyt rajoitukset asiakkaille tarjottaville palveluille. Operaattoreille jää kuitenkin runsaasti vapausasteita. Operaattorit voivat esimerkiksi päättää, millaisia eriyttämismalleja käytetään ja miten mallit toteutetaan.

Tässä työssä tutkitaan laadun eriyttämistä ja skedulointia DiffServ-verkoissa. Työn ensimmäinen osa on kirjallisuustutkimus palvelunlaatutakuista, arkkitehtuureista sekä mekanismeista pääpainon ollessa skeduloinnissa.

Työn toisessa osassa analysoidaan kvalitatiivisella tasolla operaattorin mahdollisia palvelutapoja ja eriyttämismalleja sekä ehdotetaan erilaisia skedulointimenetelmiä mallien toteuttamiseksi. Malleja ja skedulereita arvioidaan myös simulaatioiden avulla. Simulaatioissa tutkitaan sekä aggregaatti- että sessio/vuotasolla, millaista palvelua eri skedulerit pystyvät tarjoamaan ja miten hyvin ne tukevat valittua eriyttämismallia. Lisäksi tarkastellaan liikenteen luokittelun vaikutusta eriyttämiseen.

Avainsanat: Skedulointi, QoS, Eriyttäminen, Simulaatio

Acknowledgements

This Master's thesis was written in the Networking Laboratory of Helsinki University of Technology. The research was conducted in the FIT project funded by Academy of Finland.

I want to express my gratitude to my instructor, Lic.Tech Marko Luoma, for many inspiring discussions and debates. It has been a pleasure to work with a person who has such an enthusiastic attitude and provocative ideas. I also want to thank my supervisor, Professor Samuli Aalto, for his patient guidance and flexibility during the course of this work, and professor Jorma Virtamo for comments at the initial phase of the thesis.

Also other people in the laboratory deserve thanks for creating a cosy working atmosphere. In particular I want to thank my office mates, M.Sc Laura Nieminen and M.Sc Riikka Susitaival for helpful discussions and guidance in practical issues.

Finally, I would like to thank my family and my boyfriend Tuomas for their endless love and support.

Otaniemi, 7th April 2003

Johanna Antila

Contents

Abbreviations

Abbreviations

ACK	Acknowledgement
AF	Assured Forwarding
CAC	Connection Admission Control
CBQ	Class Based Queueing
CNCL	Communication Networks C++ Library
DDP	Delay Differentiation Parameter
Delay EDD	Delay Earliest Due Date
DiffServ	Differentiated Services
DNS	Domain Name Server
DRR	Deficit Round Robin
DSCP	Differentiated Services Code Point
EDD	Earliest Due Date
EDF	Earliest Deadline First
EF	Expedited Forwarding
FCFS	First Come First Served
FIFO	First In First Out
FTP	File Transfer Protocol
GPS	Generalized Processor Sharing
HPD	Hybrid Proportional Delay
HTTP	Hypertext Transfer Protocol
IntServ	Integrated Services
IP	Internet Protocol
Jitter EDD	Jitter Earliest Due Date
Jitter VC	Jitter Virtual Clock
LIFO	Last In First Out
MSS	Maximum Segment Size
PAD	Proportional Average Delay
PDD	Proportional Delay Differentiation
PGPS	Packetized GPS
PHB	Per Hop Behavior
QoS	Quality of Service
RED	Random Early Detection

RSVP	Resource Reservation Protocol
RTT	Round Trip Time
SCFQ	Self Clocked Fair Queueing
Sstresh	Slow Start Threshold
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
VC	Virtual Clock
VoIP	Voice over IP
VTRS	Virtual Time Reference System
WEDD	Weighted Earliest Due Date
WFQ	Weighted Fair Queueing
WF^2Q	Worst Case Weighted Fair Queueing
WRR	Weighted Round Robin
WTP	Waiting Time Priority
WWW	World Wide Web

Chapter 1

Introduction

1.1 Background

For a long time, the Internet was primarily used in the academic world for information exchange. File transfer, e-mail and remote access were the predominant applications. However, the emergence of the World Wide Web changed the nature of the Internet in a profound way: it started to develop from a research network into a public, commercial multi service network where a variety of applications, such as file transfer, VoIP and video-conferencing co-exist. Since these applications have fundamentally different characteristics and quality of service (QoS) requirements, it is a major challenge to support them in a single network. Furthermore, in the current Internet heterogeneity exists not only among applications but also among customers: some customers are satisfied even with poor quality while others would be willing to pay significantly higher prices in order to receive better service, which opens up new business opportunities for the network operators. These are the main reasons why QoS provisioning in the Internet has been - and still is - a hot research topic.

During the last decade the research in QoS provisioning has primarily focused on the development of service architectures and QoS mechanisms. A service architecture provides a general framework for service provisioning by giving an abstraction of the services and functionality of the network. However, it does not define in detail what services can be offered to the customers or what mechanisms should be used for implementation. The most notable service architectures are the Integrated Services and the Differentiated Services architectures. The Integrated Services architecture aims to provide absolute per flow guarantees by resource reservation. The Differentiated Services architecture is a more scalable solution for QoS provisioning: traffic is divided into a limited number of forwarding classes and resources are then allocated to these classes instead of individual flows.

In order to be able to construct different services, QoS mechanisms must be deployed in the network. During the last decade a vast amount of these mechanisms have been proposed, such as traffic classification, packet scheduling, admission control and QoS routing. Packet scheduling is one of the most crucial of these mechanisms because it is responsible for sharing the common resources according to a predefined policy. In practice, a packet scheduler enforces resource allocation by deciding the order at which individual packets are served. These decisions have a great impact on performance parameters such as throughput, delay or delay jitter.

The available service architectures and mechanisms set certain limits to the services that can be offered to customers. However, every operator that decides to build a QoS system has still many degrees of freedom: First, the operator can decide what is the target for service provisioning. Second, the operator can choose what kind of differentiation models it will use to meet this target. Finally, the operator has the freedom to decide what mechanisms (e.g. packet schedulers, routing) will be used in implementing the differentiation models.

1.2 Research problem

The research problem in this thesis is to analyze what kind of differentiation models the operator can use to support its service approach and what kind of scheduling disciplines would be suitable for implementing these models. The focus is on scheduling disciplines designed for capacity and delay differentiation since they are the quality parameters that are most affected by the choice of a scheduling discipline.

The methodology adopted in this thesis is qualitative analysis and simulations: The first part of the thesis provides a literature survey about scheduling disciplines that can be used for service differentiation. Based on the literature survey the scheduling disciplines are compared qualitatively in order to get an idea of which scheduling disciplines could be suitable for implementing the desired differentiation models. Finally, the performance of the selected scheduling disciplines is evaluated through simulations. The goal of the simulations is to study how well the proposed scheduling disciplines are able to support the differentiation model that they are designed for. Another important goal is to study the effect of the traffic mix on the results.

1.3 Structure of the thesis

This thesis is structured in the following way: The second chapter begins with a discussion about the motivations for QoS. Also terminology related to services, QoS and differentiation is introduced and different types of guarantees are described. Finally, background is provided about QoS mechanisms and architectures.

The third chapter provides a general description of scheduling disciplines, which are key components in a resource allocation system. This chapter introduces terminology related to scheduling disciplines, desirable properties of scheduling disciplines and ways to classify scheduling disciplines.

In the fourth chapter several scheduling algorithms are presented in detail.

The fifth chapter focuses on quality differentiation. First, service approaches and differentiation models for an operator are introduced. Then, possible packet schedulers for implementing these models are analyzed at a qualitative level. Finally, issues related to traffic mapping are discussed.

The sixth chapter describes the simulation methodology used to evaluate different packet schedulers. In this chapter the simulation environment, simulation model, used traffic models and simulation scenarios are introduced.

The seventh chapter presents the simulation results of different packet schedulers. Summary and conclusions about the results are made in the eighth chapter. The eighth chapter provides also directions for future work.

Chapter 2

Background of QoS and service architectures

2.1 Motivations for QoS

The current Internet offers a common best effort service for all. There is no differentiation between users, applications or even protocols: the network does its best to deliver packets but no guarantees for quality of service are made. If there is no congestion in the network, this kind of simple operation may be enough. However, with current traffic volumes congestion situations can not be avoided. This means that the amount of resources available for different users is limited, which creates need for quality differentiation.

There are basically two motivations for realizing QoS in the Internet. The most evident motivator is that different applications require different service from the network in order to function properly. For instance, a file transfer is not so sensitive to transfer delay but the correctness of information is important. On the other hand, real time applications such as VoIP or video-conferencing set stringent requirements for end to end delay and jitter. Another motivator is purely economic: if an operator provides quality differentiation, it can also charge the customers differently. At first it may seem that these two motivators are closely interrelated. It should be noted however, that the actual requirements of applications are not necessarily the basis for quality differentiation. The operator may for instance differentiate the customers based solely on the prices that they are willing to pay independent of the applications that they are using.

2.2 Elements of a QoS system

In Figure 2.1 the basic elements of a QoS system are illustrated. Also the interfaces between these elements are shown. In order to understand the roles of the elements

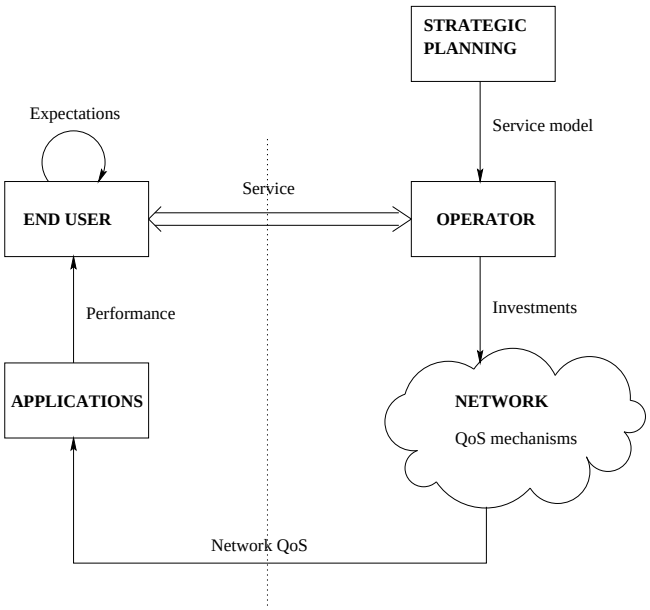

Figure 2.1: Elements of a QoS system

and the relationships between them, some terminology related to services, QoS and differentiation is introduced:

Service approach

The purpose of the service approach is to support the mission and objectives of the operator. It defines the goal of service provisioning, i.e. who or what the operator wishes to serve.

Service model

Service model is closely related to the visions and strategic planning of an operator. It defines at a rather high level what services will be offered or could be offered to the customer. On one hand, the available technology sets limitations for the selection of a service model. On the other hand, the service model may have influence on how the technology develops in the future.

Service

Service is a realization of the service model. It defines what the operator is guaranteeing for the customer (if anything) and what the customer is expected to pay for the service.

QoS

There are many different definitions for Quality of Service (QoS). Originally the term has been used to express that the performance of the service meets some predefined characteristics. However, nowadays QoS is often understood in a broader sense: QoS means that some user, application etc. receives better service than the others. When talking about QoS, it is useful to distinguish between network QoS and user perceived QoS:

Network QoS

Network QoS is described with parameters such as bandwidth, delay, jitter and packet loss that can be measured directly or undirectly from the network . The technical functioning of an application depends largely on the level of network QoS.

User perceived QoS

User perceived QoS entails the whole service experience of the end user. Both technical and psychological aspects have an effect on this experience. For example, the user compares the performance of an application with his previous experiences and expectations. Even if the QoS delivered by the network would be good, the user may not be satisfied if the quality is not what he had expected.

QoS mechanisms

QoS mechanisms are building blocks that are used to construct the desired services. They contain the logic to differentiate for example users, applications or protocols.

Quality differentiation

Differentiation is a somewhat looser concept than QoS. It essentially means that users, applications etc. are somehow distinguished from each other and treated in different ways in order to produce different levels of quality.

Differentiation model

Differentiation model defines the quality aspect related to differentiation. The quality aspect may be for example delay, bandwidth or importance.

2.3 Types of QoS Guarantees

In the previous section the elements of a QoS system were described. It was stated that the key relationship between the end user and the operator is the service that defines what is guaranteed for the customer. When speaking about QoS and differentiation the term guarantee must be considered in a broad sense. It does not necessarily mean that predefined performance parameters are guaranteed for the

customer all the time. In fact, the guarantees may be very vague as long as this uncertainty is clearly expressed in the service agreement.

QoS guarantees can be divided roughly into three groups based on the degree of assurance that they provide.

1. *Deterministic guarantees:* Deterministic (or absolute) guarantees ensure that the connection receives service according to predefined performance parameters all the time: the connection can be guaranteed for example with a minimum bandwidth of 5 Mbps and a maximum delay of 100 ms. Furthermore, no loss of information is allowed.

2. *Statistical guarantees:* Since many applications can tolerate some deviation from the absolute performance bounds, it is more efficient to guarantee that the bounds will be violated only with some small probability. For instance, statistical guarantee for delay could be of form: $\mathrm{P}(\text{Delay} > 100 \text{ ms}) < 0.05$. Then, 95% of the packets will experience a delay smaller than 100 ms in the specified time scale.

3. *Relative guarantees:* A softer form of QoS guarantees is to provide only relative differentiation among service classes. The difference in the service experienced by the classes can either be quantified or not. For example, it could be specified that higher classes experience smaller packet losses at a node without quantifying how much smaller the packet losses will be. On the other hand, the ratio of the performance parameters (such as mean delay) of different classes could be kept at a fixed constant.

These groups of QoS guarantees can also be combined. For example, by combining group 1 and group 3 it could be specified that the ratio of mean delays of service classes should be 5, and the delay of class 1 should not exceed 100 ms.

2.4 General Description of QoS control mechanisms

In order to realize services, different QoS control mechanisms must be implemented in the operations and management of the network. During the last decade various QoS control mechanisms have been proposed for different purposes. One useful way to classify these mechanisms is based on the time scale at which they operate. Starting from the shortest scale the time scale levels of QoS control mechanisms can be divided into packet level, round-trip-time level, session level and long-term level [FBTZ01].

Mechanisms operating at the packet level time scale ($\sim 1 - 100\mu s$) include traffic classifiers, policers, markers, shapers, packet schedulers, buffer management etc. The

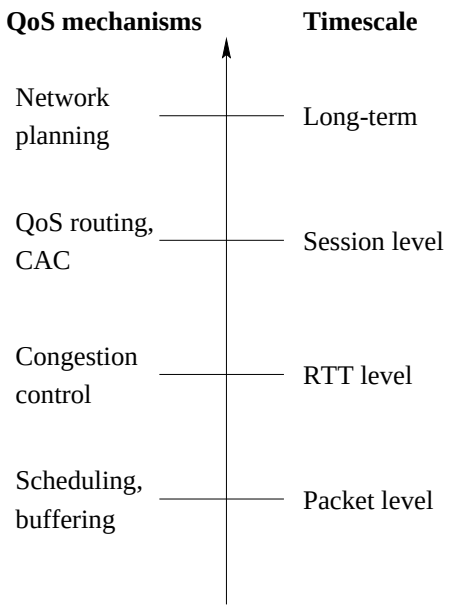

Figure 2.2: Time scale hierarchy of QoS control mechanisms

next fastest time scale, the round-trip-time time scale ($\sim 1 - 100ms$), is the time scale where feedback-based flow and congestion control operate. The session time scale (from seconds to minutes or longer) refers to the time that user sessions usually last and thus this is the time scale of admission control and QoS routing. Finally, the long-term time scale ranges from minutes to even months and mechanisms operating at this level are for instance traffic engineering and capacity planning.

The rest of this section gives a brief description of the most important QoS control mechanisms mentioned above:

1. *Traffic classification:* Classification is a process that recognizes which connection or class the incoming packets belong to. The recognition can be based for example on the information in the IP/TCP packet header, such as the source or destination IP address, DSCP-field or port number. As a result of the classification process, the total incoming traffic stream is divided into logically separate sub streams that can be treated in different ways.

2. *Traffic conditioning:* The task of a traffic conditioner is to measure the incoming traffic streams (from the classifier) and compare the measurement result against the customers' traffic profile [Wan01]. If the incoming traffic is in-profile, it can be allowed to enter the network. Otherwise the traffic may be remarked, shaped or even dropped.

3. *Packet scheduling:* Packet scheduling is one of the most important QoS mechanisms because it enforces resource allocation by deciding when packets are

transmitted. These decisions have a great impact on performance parameters such as throughput, delay or delay jitter. The objective of packet scheduling is to share the common resources so that some predefined policy will be met.

4. *Queue management:* Queue management algorithms decide which packets to discard in congestion situations and when. The simplest buffer management algorithm is Tail Drop, which simply discards a packet if the queue is full at the arrival time of the packet. Another commonly used algorithm is Random Early Detection (RED) [FJ93], which can drop packets with certain probability even if the queues are not yet full. The dropping probability in RED is based on exponentially averaged queue length.

5. *Admission control:* Admission control has two functions: to determine if a new flow can be accepted to the network and to keep track of the availability of resources. Admission control can be either parameter based or measurement based [Wan01]. In the parameter based admission control traffic flows are described precisely by a few parameters based on which the admission control agent can calculate the required resources. In the measurement-based approach actual traffic loads are measured and this measurement information is used in admission control decisions.

6. *QoS routing:* Traditional IP-routing is based on finding the shortest path between the source and the destination. However, this shortest path approach may not be optimal, since usually it causes traffic to centralize in some parts of the network. In this kind of a situation it could be better for a flow to be routed along a path that may not be the shortest but that is less heavily loaded. The idea in QoS routing is to find a path for a flow or for a traffic aggregate in the network under multiple constraints, such as delay and bandwidth. QoS routing algorithms can be greedy in the sense that they try to optimize the performance of one flow or aggregate without taking into account network wide effects [Wan01]. However, in wider sense one objective of QoS routing is also to achieve high resource utilization in the network.

7. *Network planning:* Network planning is a longer term process that includes deciding what elements and mechanisms will be used in the network and how the network should be dimensioned and provisioned. For example, when planning a radio network questions such as how many base stations are required need to be addressed.

2.5 Different approaches for QoS provisioning

There are many choices related to the provision of QoS. One of the most important choices is whether resources are allocated for individual flows or for traffic aggregates. Another significant choice is the type of guarantees (absolute, relative etc.). In

order to achieve a common agreement on these issues, different service architectures have been proposed. A service architecture is an abstraction of the services and functionality of the network. It does not define at a detailed level either what mechanisms should be used or what services can be offered but it provides a general framework for service provisioning. The most well known service architectures are the Integrated Services and the Differentiated Services architectures.

2.5.1 Intserv

The Integrated Services (IntServ) architecture [BCS94] was developed in early 1990s. At that time multimedia conferencing was considered to be the future killer application and thus the main purpose of IntServ was to provide absolute per-flow QoS for delay sensitive multimedia traffic.

In order to provide absolute guarantees, IntServ relies on per-flow resource reservation. A specific resource reservation protocol, RSVP [Wro97], is used to communicate the user requirements to the network nodes: In the reservation procedure the sender first specifies its traffic characteristics, such as bucket depth and bucket rate, and sends this information to the network. If the requested resources can be guaranteed by the network, the sender may start sending traffic. Besides the resource reservation procedure, in IntServ also classification and forwarding actions, such as scheduling, are made on a per-flow basis. Thus IntServ sets high requirements to both the data and control plane of the router. The reference model of IntServ router is depicted in Figure 2.3.

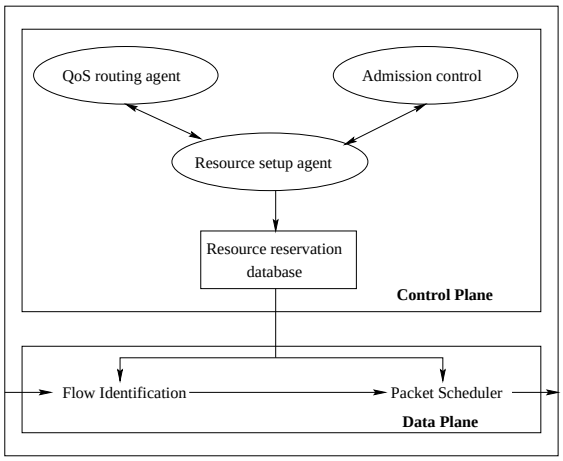

Figure 2.3: Integrated Services reference model

Two service models have been developed for IntServ: guaranteed service and controlled load service. The guaranteed service is meant for non-adaptive applications that require absolute delay and bandwidth guarantees. The controlled load service,

on the other hand, aims to provide service comparable to that of a lightly loaded best-effort network. It is primarily meant for adaptive applications.

IntServ has not been widely deployed for a number of reasons. First, there are major scalability problems in the architecture: If the majority of flows were long lasting multicast streams as was originally thought, per-flow resource reservation and forwarding could be reasonable. However, nowadays the Internet is dominated by web-based applications, meaning that many flows are short-lived. Thus the time to reserve the resources could easily exceed the lifetime of the flow. If there are millions of flows, this leads to very poor cost-efficiency. Second, IntServ requires tight cooperation between different service providers. In order to support guaranteed services over the whole network, bilateral peering agreements between the providers should be extended to Internet wide agreements [Wan01]. Third, it could be difficult for the users to define the required QoS parameters precisely.

2.5.2 DiffServ

In 1997 it was realized that a more scalable architecture would be needed to offer services beyond best-effort. As a result, the Differentiated Services (DiffServ) architecture [BBC+98] was proposed. In DiffServ, the traffic is divided into a limited number of forwarding classes meaning that the resources are allocated to traffic aggregates instead of individual flows. The forwarding class of a packet is encoded in the IP packet header. In DiffServ, no resource reservations are made. Instead, assurances are based on prioritization, provisioning and possibly admission control.

DiffServ architecture addresses the problem of scalability by keeping the state information at the network edges. The task of the edge nodes is to perform packet classification and traffic conditioning: packets are first classified based for example on their source or destination address or application type (port number) and marked with an appropriate Differentiated Services Code Point (DSCP) value. The conditioner then measures how well the traffic of the flow matches its traffic profile. All packets that are in-profile are sent to the network, while the out-profile packets may be remarked, shaped or dropped, as depicted in Figure 2.4. The core nodes, on the other hand, merely forward packets according to the DSCP in the packet header: forwarding actions, such as scheduling, are performed on per-class rather than per-flow basis.

Contrary to IntServ, DiffServ does not define any end-to-end services. Instead, in DiffServ each forwarding class represents a per hop behavior (PHB) that defines how a packet is treated in a single node. The PHB can be implemented by using queue management and scheduling. Even though the PHBs only define forwarding treatments, not services, the services can be constructed from the PHBs by combining them with admission control. Two PHB groups, AF PHB and EF PHB have been standardized by the IETF. The AF PHB [HBWW99] defines four forwarding classes

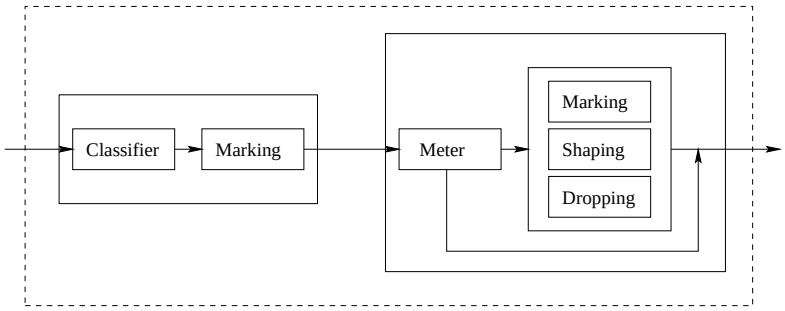

Figure 2.4: Traffic classifier and conditioner

and within each class three drop precedences. For each forwarding class, a minimum amount of buffers and bandwidth is allocated [Wan01]. The drop precedences are used to decide which packets to drop during congestion. The EF PHB [JNP99] aims at providing a treatment with minimal delay and loss. In any DiffServ node, the provisioned capacity for the EF aggregate must equal or exceed the sum of the committed information rates of the flows belonging to the aggregate. The idea is that the traffic belonging to an EF class should experience service similar to that of a virtual leased line.

DiffServ seems to be a scalable and promising architecture. However, resource management in DiffServ is done within a single domain, which can cause problems. Since users are only interested in end-to-end services, this question of interdomain management has to be solved somehow. One solution is to use centralized or distributed bandwidth brokers to handle the resource allocation between multiple domains.

Proportional Differentiated Services

The model for Proportional Differentiated Services was first proposed in [DSR99]. The idea is to quantify the quality spacings between traffic classes so that the ratio of local performance metrics of successive classes, such as delay or packet loss, is kept roughly constant. Consider N classes and denote by Q_i the performance metric of class i and by δ_i the class differentiation parameter of class i. The ratio of performance metrics is then determined by the class differentiation parameters as follows:

$$\frac{Q_i}{Q_j} = \frac{\delta_i}{\delta_j}, 1 \leq i, j \leq N. \tag{2.1}$$

The model provides network operators the tools to tune the quality difference between classes based on their criteria, such as price. For example, if the model is applied to delay differentiation and there are two traffic classes with differentiation parameters $\delta_1 = 1$ and $\delta_2 = 0.25$, the delays experienced by class 2 traffic will be approximately one fourth of the delays experienced by class 1.

2.5.3 Dynamic Packet State

DiffServ provides a solution for scalability issues both in control and data plane. However, the increased scalability does not come without a cost: It is obvious that for example the use of class based aggregate scheduling leads to coarser performance assurances compared with flow based scheduling. Whether absolute performance guarantees are even necessary is an open question. However, there is still interest in the engineering and research community to design a service architecture that would be able to provide per-flow QoS guarantees comparable to IntServ with the scalability of DiffServ. One proposition in this direction is the Dynamic Packet State approach, where control of state information needed in packet scheduling is carried in packet headers [FBTZ01]. The core routers then simply update this per-packet state.

The idea of dynamic packet state was originally proposed by Stoica and Zhang [SZ99]. They showed that the end-to-end delay bounds of the Jitter Virtual Clock (Jitter-VC) scheduling discipline can be achieved without per-flow management by using a core stateless version of this algorithm based on the dynamic packet state technique. Their approach was further generalized to a virtual time reference system (VTRS) which is a general core stateless framework to support guaranteed services in a scalable way [FBTZ01]. In a VTRS system the packets are stamped with packet virtual time stamps. These stamps are updated as the packets traverse from a core router to another. The VTRS system is called core stateless since the packet virtual time stamps can be computed based only on the state carried by the packet and some fixed parameters that are associated with core routers. In other words, no per-flow state is required at core routers [FBTZ01].

Chapter 3

General description of scheduling disciplines

3.1 Introduction

Scheduling discipline is a key component in any resource allocation system: during the last decades a vast amount of scheduling disciplines have been proposed in the literature for different purposes. This chapter outlines at a general level the terminology and some desirable properties of scheduling disciplines and presents possible ways to classify scheduling disciplines.

3.2 Terminology

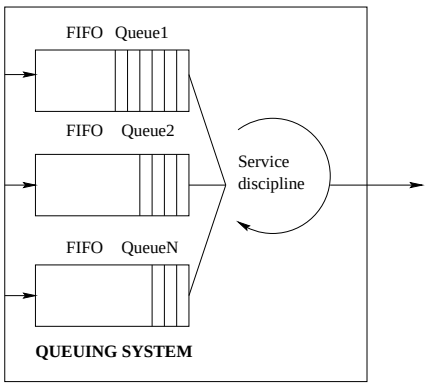

Figure 3.1: A queuing system

The term scheduling discipline is closely related to the concept of a queuing system.

Figure 3.1 illustrates the basic elements of a queueing system.

Queues

Queues are places where arriving packets are stored. The queuing system may contain separate queue for each flow, for each service class or only one queue for all traffic.

Queueing discipline

Queueing discipline defines the order at which packets are served within a queue. Examples of queueing disciplines are First In First Out (FIFO) and Last In First Out (LIFO).

Scheduling discipline

Scheduling discipline, also called scheduling algorithm decides the order at which packets are served among different queues. Scheduling discipline may be either non-hierarchical or hierarchical. In a hierarchical scheduling discipline scheduling decisions are performed at several stages, as depicted in Figure 3.2.

Queue management

Queue management defines which packets are discarded in a congestion situation and when. Packets may optionally be dropped even before congestion occurs.

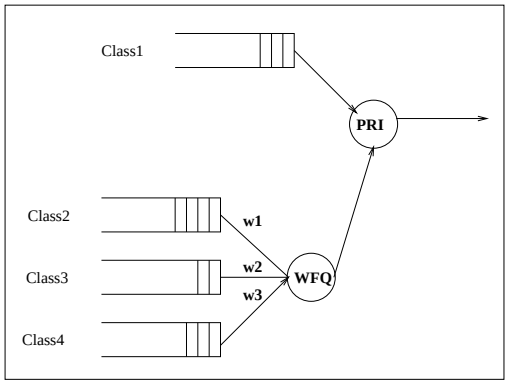

Figure 3.2: A hierarchical scheduler

3.3 Desirable properties of scheduling disciplines

In order to be able to design new scheduling disciplines and to compare the existing ones with each other, it is important to define what are the desirable properties of a scheduling discipline. It is obvious that many of these properties are tightly related to

the guarantees made for the end user. For instance, guaranteeing deterministic delay bounds and bandwidths set different requirements for a scheduling discipline than relative differentiation. However, there are also some general desirable properties common for all scheduling disciplines:

1. *Flexibility:* The scheduling discipline should be able to accommodate applications with varying traffic characteristics and performance requirements rather than just optimize the performance from a certain application's point of view. In future networks several applications with diverse requirements will have to be supported making it necessary for the scheduling discipline to be flexible.

2. *Simplicity:* The scheduling discipline should be simple both conceptually and mechanically [Siv00]. Conceptual simplicity enables tractable analysis of the scheduling discipline such that distributions or worst case bounds for performance parameters, such as delay, can be derived. Mechanical simplicity is required to allow efficient implementation of the scheduling discipline at high speed.

If the performance guarantees are deterministic or statistical, the following desirable properties have further been identified in [Zha95] and [Siv00]:

1. *Protection:* Real network environment is not static. As a consequence, the scheduling discipline should be able to protect the well behaving users from different sources of variability, such as best-effort traffic, ill behaving users and network load fluctuations [Zha95]. Ill behaving users refer for example to users who send more packets than their traffic profile allows. Network load fluctuations, on the other hand, are caused by traffic bursts at a router. These bursts may accumulate even if the users meet their traffic constraints at the entrance of the network. Ideally the scheduling discipline should be able to satisfy the performance requirements of well behaving users even in the presence of these factors.

2. *Efficiency:* When performance guarantees are deterministic or statistical, connection admission control policy is used in order to decide, whether to admit a new flow into the network. Scheduling disciplines providing these kinds of guarantees are associated with a schedulable region, which indicates all possible combinations of flows that can be accepted into the network without violating the end-to-end performance guarantee of any other flow. A scheduling discipline is said to be more efficient than another if it has a larger schedulable region [Siv00]. Thus, efficiency results in a higher network utilization.

Criteria for scheduling disciplines aiming at relative differentiation are somewhat different because the service guarantees are considerably looser compared with deterministic or statistical guarantees. Besides flexibility and simplicity, the most important criteria are [DSR02]:

1. *Predictability:* Relative QoS means that class i receives better (or at least no worse) service than class $i-1$ with regard to a selected performance metric. This differentiation should be consistent, meaning that the scheduling discipline should be able to maintain the right class ordering also in short timescales and in the presence of class load variations.

2. *Controllability:* If the quality spacing between traffic classes is quantified, the scheduling discipline should be able to maintain the specified spacing even in short timescales. It should also be possible to adjust the quality spacing based for example on pricing or some other criteria [LLY01].

3.4 Classification of scheduling disciplines

Scheduling disciplines can be categorized in many ways. Traditionally they have been divided into work-conserving and non-work-conserving disciplines. Another possible classification is based on their internal structure, according to which there are two main architectures: sorted-priority and frame-based [Sti96]. Division can also be made between fluid-based and packet-based scheduling disciplines.

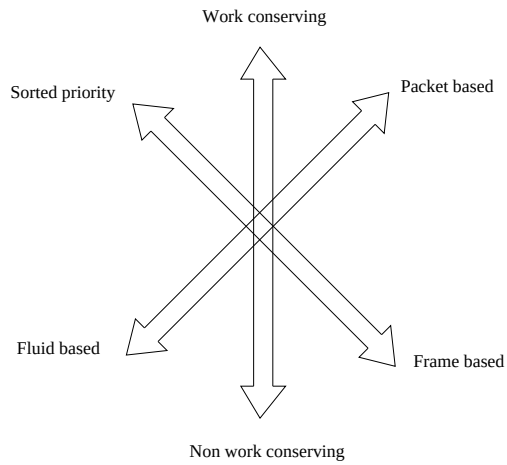

Figure 3.3: Classification of scheduling disciplines

A work-conserving scheduling discipline serves packets as long as there is a non-empty queue in the system. The server is idle only when there are no packets to be sent. A non-work-conserving server, on the other hand, may remain idle even if there are packets waiting in the system. Non-work-conserving servers obviously have larger average delays than work-conserving servers. They also result in lower utilization of network resources [LY99]. However, non-work-conserving scheduling disciplines also have some important advantages. For instance, the server may postpone the transmission of a packet if it expects a more important packet to arrive soon [Sti96].

Non-work-conserving scheduling disciplines may also be used to control the delay jitter (maximum difference between the inter arrival times of consecutive packets) of real-time applications: The server delays the packets so that their inter arrival times remain roughly constant. Another possible application of non-work-conserving scheduling disciplines is shaping [LY99].

Sorted-priority scheduling disciplines use a global variable, often called virtual time (to distinguish it from real time), associated with the server. The purpose of this variable is to keep track of the progress of the server and it is usually updated at packet arrival and departure instants. For each packet in the system, a time stamp is computed as a function of this variable. Packets are then sorted based on these time stamps and served in this order. The complexity of a sorted-priority algorithm is determined by the complexity of calculating the time stamp, updating the priority list and selecting the highest priority packet for transmission. The complexity of time stamp calculation is very much dependent on the scheduling discipline. For example, in WFQ the updating of virtual time is considerably more complex than in WRR or Virtual Clock.

Frame-based scheduling disciplines use a frame of fixed or variable length which is divided among different connections/classes based on the reservations of the connections/resources allocated for the class. The more resources are allocated for a connection/class, the larger part of the frame it receives. The frame is split among the connections/classes in a similar way in each service round.

Queueing systems are often analyzed with an idealized fluid-flow model. The model assumes that traffic is infinitely divisible and that different "packets" can be served simultaneously. GPS is an example of a scheduling discipline that is based on the fluid-flow model. Packet-based scheduling disciplines, on the other hand, analyze and serve the packets one at a time: The service of a new packet can begin only after the last bit of the previous packet has been served. The differences between fluid-based and packet-based scheduling disciplines are illustrated in Figure 3.4 and Figure 3.5.

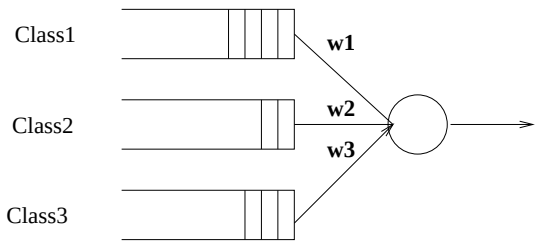

Figure 3.4: A packet based scheduler

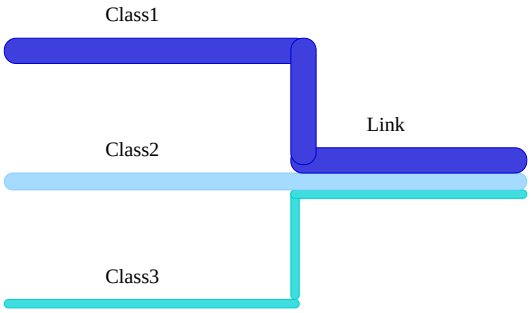

Figure 3.5: A fluid based scheduler

Chapter 4

Scheduling algorithms

4.1 Introduction

This chapter presents several scheduling algorithms that can be used for service differentiation. The algorithms are classified into four groups: basic algorithms, bandwidth sharing algorithms, deadline based algorithms and algorithms for proportional differentiation. All the basic algorithms, bandwidth sharing algorithms and deadline based algorithms have originally been invented at a time when the goal of resource allocation was to guarantee QoS for individual flows. The algorithms for proportional differentiation have in turn been designed during recent years for class based resource allocation.

In the rest of this thesis a basic assumption is that resources will be allocated between traffic classes, such as in the differentiated services model. Let there be N traffic classes, each of which has its own queue. The task of the scheduling algorithm is then to decide the order for transmitting the packets from different queues, as presented in Figure 4.1.

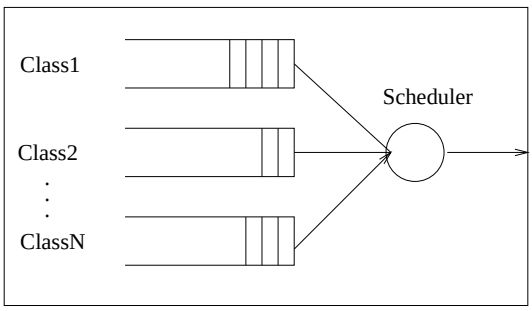

Figure 4.1: Class based resource allocation

Although many of the proposed scheduling algorithms are originally flow based, they can be generalized to apply also for class based resource allocation. In this thesis all formal definitions of different scheduling algorithms are presented in the class based context.

4.2 Basic algorithms

4.2.1 FCFS

First Come First Served (FCFS) is the prevalent scheduling discipline used in best effort routers. In FCFS, packets are served simply in increasing order of their arrival time. However, since the traffic is not classified in any way and scheduling decision is based only on the arrival time, FCFS is not able to provide differentiation. For example, if a flow in some class sends very bursty traffic, this will inevitably increase the delays of the flows in other classes.

4.2.2 Simple Priority

In order to be able to offer better service to more important traffic, the classes can be assigned with different levels of priorities. In a simple priority system, the higher priority has always precedence over the lower priority. This idea has been deployed, for example, in many computer systems. A simple priority scheduler will always serve packets from the queue that is assigned with the highest priority. Queues with lower priority are served only when all the queues with higher priority are empty.

The advantage of a simple priority scheduler is that it can be easily implemented. However, in a priority system the low priority traffic may be starved, unless the amount of high priority traffic is limited somehow. Hence, proper rate control or admission control should be used for the high priority traffic.

4.3 Bandwidth sharing algorithms

4.3.1 Virtual Clock

Virtual Clock, first proposed in [Zha91], is one of the simplest scheduling disciplines that can be used for service differentiation. Originally it was designed to provide guaranteed bandwidth and low delays for flows by emulating a TDM system. Since in a TDM system each connection is allowed to send data only during a certain time slot, the connections are completely isolated from each other and can thus be guaranteed with their reserved bandwidths. However, in a TDM system users are

restricted to send data with a constant bit rate. Furthermore, in case a user has no data to send during its turn, bandwidth can not be allocated to other users but is simply wasted. The idea in Virtual Clock is to preserve the isolation property of the TDM system while allocating "slots" to flows on a demand basis [Zha91]. This can be achieved by using a virtual clock, rather than a real-time clock, to regulate the resource usage.

When the Virtual Clock is applied in a class based context, it can be described formally as follows: In the Virtual Clock algorithm, each class is assigned two control variables, $VirtualClock_i$ and $auxVC_i$ (auxiliary VirtualClock). Each class is also associated with a variable $Vtick_i$, denoting the mean packet inter arrival time in the class. Formally, upon the arrival of the kth packet of class i at the router, the $VirtualClock_i$ and $auxVC_i$ variables are first updated as follows:

$$VirtualClock_i = VirtualClock_i + Vtick_i \tag{4.1}$$

$$auxVC_i = \max\left(auxVC_i, realtime\right) + Vtick_i. \tag{4.2}$$

Then, the packet is stamped with $auxVC_i$ value. At the time when the server becomes free, it selects for transmission a packet with the smallest time stamp.

The roles of the two control variables are following: $VirtualClock_i$ ticks each time a packet arrives in class i queue. The tick step is set to the mean packet inter arrival time in the class, $Vtick_i$, so that $VirtualClock_i$ effectively keeps track of the expected arrival time of the packet that has just arrived. If the packets arrive according to the inter arrival time set for the class, then $VirtualClock_i$ and real time are the same. However, if the packets in the class arrive sooner, $VirtualClock_i$ will be ahead of real time indicating that a packet arrived in the class earlier than was expected. This discrepancy from the expected arrival time is reflected also to the value of $auxVC_i$, and thus a class that tries to consume more resources than it should will be punished with larger queueing delays.

4.3.2 GPS

Generalized Processor Sharing (GPS) is an ideal fair-queuing algorithm that originally aims at providing an exact max-min weighted fair sharing for different flows [Wan01]. GPS is said to be an ideal algorithm since it is based on fluid model and thus assumes that traffic is infinitely divisible and that different flows can be served simultaneously.

When the GPS algorithm is applied to traffic classes instead of individual flows, it can be described as follows: In GPS, each class i is assigned a weight ϕ_i reflecting the amount of resources that should be allocated for the class. If the service rate of the GPS server is r and there are N classes being served, then for any two backlogged classes i and j,

$$\frac{S_i(\tau, t)}{S_j(\tau, t)} = \frac{\phi_i}{\phi_j}, \tag{4.3}$$

where $S_i(\tau, t)$ denotes the amount of traffic serviced for class i in an interval (τ, t) [PG93]. Thus, in any interval (τ, t) class i receives service with a rate

$$g_i \geq \frac{\phi_i}{\sum_{j=1}^{N} \phi_j} r. \tag{4.4}$$

This corresponds to the situation where all classes are backlogged during the interval. However, if some classes are not backlogged, the excess bandwidth will be distributed among the backlogged classes in proportion to their weights. Then, the instantaneous service rate can be expressed as

$$g_i(t) = \frac{\phi_i}{\sum_{j \in B(t)} \phi_j} r, \tag{4.5}$$

where $B(t)$ denotes the set of classes that are currently backlogged.

GPS is considered to be an attractive scheduling discipline, because it has many desirable properties. First, it provides fairness for the classes by servicing each class with a rate equal to or greater than the classes's guaranteed rate. Second, if the incoming traffic is leaky bucket constrained, it can be shown [PG93] that strict bounds for worst-case network queueing delay exist. Third, the classes can be treated in different ways by varying the weights. For instance, if there are two classes with weights $\phi_1 = 1$ and $\phi_2 = 0$, GPS reduces to strict priority scheduling. On the other hand, if all classes are assigned equal weights, GPS behaves as a uniform processor sharing system.

Besides these advantages, GPS has disadvantages as well: As an ideal system it can not be implemented in practice, since in a real IP-router packets are always serviced as entities, one at a time. However, different packet per packet approximations of GPS have been proposed, which try to emulate the GPS system as accurately and simply as possible while still treating packets as entities.

4.3.3 WFQ

Weighted Fair Queueing (WFQ), also known as PGPS, is perhaps the best known scheduling discipline approximating the GPS system. Like GPS, also WFQ was originally designed for allocating resources for flows. However, it can also be applied in the DiffServ context.

The basic idea in WFQ is to emulate the GPS system by stamping each packet p with a virtual finish time F_p that represents the time at which the packet would depart under the reference GPS system. The packets are then served in increasing order of the time stamps. It should be noted, however, that at the time when the WFQ server becomes free, it may be that the next packet to depart under GPS has not yet arrived [Zha95]. For example, suppose that there is only one large packet in the WFQ system at time τ when the server becomes free. In order to be work

conserving, the server selects this packet for transmission. However, just after time τ a very small packet could arrive, such that it would be serviced under the GPS system before the large packet. Obviously, it is not possible for the server to be both work conserving and serve the packets in exact order of F_p. Thus, an additional condition is needed to describe the functioning of WFQ [PG93]: "The server picks the first packet that would complete service in the GPS simulation if no additional packets were to arrive after time τ".

In WFQ, the calculation of virtual finish times is based on the simulation of the reference GPS system in the background. Assume that resources are allocated for classes. Denote by S_i^k and F_i^k the virtual start and finish times of the kth packet of class i in the reference GPS system and by a_i^k and L_i^k its arrival time and length. The server capacity r is assumed to be 1. Assume further that

$$\sum_i \phi_i = 1. \tag{4.6}$$

Now,

$$F_i^k = S_i^k + \frac{L_i^k}{\phi_i}, \tag{4.7}$$

where the second term represents the time required to serve the packet. The calculation of the first term, S_i^k depends on whether queue i is empty at the arrival time of the packet. If the queue is not empty, then we have simply

$$S_i^k = F_i^{k-1}, \tag{4.8}$$

i.e., the virtual start time of the packet corresponds to the virtual finish time of the previous packet. However, if the queue is empty upon arrival of the packet, the progress of the GPS system has to be tracked in some other way. Let $V(t)$ denote the system virtual time, i.e. the total work done by the reference GPS scheduler. Then, the virtual start time of the packet arriving in an empty queue is

$$S_i^k = V(a_i^k). \tag{4.9}$$

Denote by B the set of classes that are backlogged in the interval (t_{j-1}, t_j). The system virtual time function $V(t)$ evolves in the following way during each busy period in the GPS system [PG93]:

$$V(0) = 0, \tag{4.10}$$

$$V(t_{j-i} + \tau) = V(t_{j-i}) + \frac{\tau}{\sum_{i \in B} \phi_i}, \tag{4.11}$$

$$\tau \leq t_j - t_{j-1}, j = 2, 3, \ldots \tag{4.12}$$

$V(t)$ has to be updated each time when there is an event (arrival or departure) in the GPS system.

The calculation of the virtual finish time F_i^k can then be summarized as

$$S_i^k = \max(F_i^{k-1}, V(a_i^k)), \tag{4.13}$$

$$F_i^k = S_i^k + \frac{L_i^k}{\phi_i}. \tag{4.14}$$

4.3.4 WF^{2}Q

Worst Case Weighted Fair Queueing (WF^{2}Q) [BZ96] is a variant of WFQ that aims at emulating more accurately the GPS system. Whereas in WFQ only the finish times of packets in the GPS system are used for making scheduling decisions, in WF^{2}Q also the start times are considered. More precisely, the WF^{2}Q algorithm selects for transmission a packet that would be finished first in the GPS system, from among those packets that have already started receiving service in GPS.

It is shown that the service order of packets under WFQ and WF^{2}Q system can be different for the same traffic arrival pattern [Zha95]. Since WFQ may select a packet for transmission even if the packet has not started receiving service in GPS, WFQ can be far ahead of the GPS system. On the contrary, in WF^{2}Q there is no such problem. According to [Zha95] the service provided by WF^{2}Q and GPS can differ at most by one packet size.

4.3.5 SCFQ

Both WFQ and WF^{2}Q need to emulate the GPS system and frequently update the set of backlogged classes in GPS. This naturally introduces complexity for calculating the virtual start and finish times. The purpose of Self Clocked Fair Queueing (SCFQ) [Gol94] is to simplify the computation by estimating the system virtual time $V(t)$ with the virtual service time of the packet that is currently being served in GPS. Denote by $V_{\text{SCFQ}}(t)$ the approximated system virtual time. Then, the virtual finish time can be calculated as follows

$$F_i^k = \max\left(F_i^{k-1}, V_{\text{SCFQ}}(a_i^k)\right) + \frac{L_i^k}{\phi_i}. \tag{4.15}$$

However, the simplification in computation does not come without a cost: In some situations SCFQ can perform much worse than WFQ and WF^{2}Q.

4.3.6 WRR

Weighted Round Robin (WRR) is a frame based scheduling discipline that provides a simple way to emulate the GPS system. In WRR, the server visits each class in turn

and serves a number of packets based on the class's weight [Wan01]. For instance, if there are two classes with $\phi_1 = 5$ and $\phi_2 = 10$, the server first transmits 5 packets from class 1 and 10 packets from class 2, then again 5 packets from class 1 and 10 from class 2, and so on. If the packet size is fixed, WRR is able to provide fairness. However, in the case of variable size packets, WRR results in an unfair distribution of bandwidth.

4.3.7 DRR

Deficit Round Robin (DRR) [SV95] is an extension of WRR that is able to provide fairness also when variable packet sizes are used. In DRR, a frame of N bits, rather than packets, is divided among the classes in proportion to their weights in each service round. This number of bits reserved for a certain class is called a quantum [Wan01]. In DRR, each class is also associated with a deficit counter that keeps track of the unused quantum for the class from previous rounds.

The DRR algorithm functions as follows: The server visits the classes in turn and serves a fixed number of bits (quantum) from each class. Every time when a packet is transmitted from a class, the quantum is reduced by the packet size. The server continues transmitting packets from the class until there is not enough bits in the quantum left for serving the next packet of the class. The possible unused quantum is recorded in the deficit counter. Then, at the next round, the value of the deficit counter is added to the quantum in order to provide the class with the service that the scheduler owes to it. If the class becomes idle, however, its quantum is simply discarded since the class has wasted its opportunity to send packets [Wan01].

4.4 Deadline-based algorithms

4.4.1 EDD

Earliest Due Date (EDD) [LL73], also known as EDF (Earliest Deadline First), is a classical example of a deadline-based scheme where packets are scheduled based on the earliest-deadline-first principle [Wan01]. EDD is originally designed for serving individual flows but it can be applied to class based differentiation as well.

Assume that traffic arriving in each class is periodic. Denote by d_i the period for class i. Then, the EDD algorithm works simply as follows: upon the arrival of the kth packet of class i at the router at time a_i^k, the packet is stamped with a deadline

$$D_i^k = a_i^k + d_i, \tag{4.16}$$

i.e. the sum of its arrival time and period. The packets are then served in increasing order of their deadlines. Notice that in reality the arriving traffic is not periodic: the

purpose of the period is only to describe the expected inter arrival time of packets.

4.4.2 Delay-EDD

Delay-EDD [FV90] is an extension to the EDD algorithm. The original idea in delay-EDD was to provide a delay bound for a flow as long as the flow transmits its packets according to a predefined traffic specification, i.e. with a certain peak and average sending rate [Zha95].

When applied to class based differentiation, delay-EDD algorithm functions as follows: The deadline of a packet belonging to class i is set to the sum of its expected arrival time and delay bound at the server as follows:

$$D_i^k = \max\left(a_i^k + d_i, D_i^{k-1} + Xmin_i\right), \tag{4.17}$$

where $Xmin_i$ denotes the minimum packet inter arrival time and d_i the local delay bound for class i.

It should be noted that since the assignment of deadlines in delay-EDD is based on two parameters, $Xmin_i$ and d_i, it is possible to decouple delay requirements from bandwidth requirements at least to some extent. On the contrary, in fair queueing algorithms small delays can not be achieved without reserving large bandwidth. Furthermore, in EDF-based schemes the value of a deadline depends only on the parameters of a single class, not on the state of other classes as, for instance, in GPS.

4.4.3 Jitter-EDD

While delay-EDD bounds only the delay of a flow, jitter-EDD [VZF91] provides bounds also for delay jitter. This extension to the delay-EDD algorithm can be achieved easily: When the service of the packet has been finished at the server, the packet is stamped with a difference between its real finishing time and deadline [Zha95]. Then, a regulator is used before the next server to hold the packet for this time until it becomes eligible. Jitter-EDD may also be applied to individual flows or classes.

Figure 4.2 illustrates a situation where the packet gets served T_{diff} seconds before its deadline at the first server. At the next server, it will be held in the regulator T_{diff} seconds before it is made eligible. In case there is no regulator at the destination server, the end-to-end jitter bound corresponds to the local delay bound of the server just before the destination [Zha95].

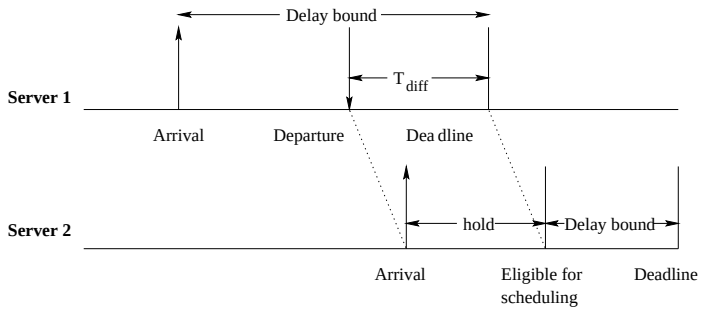

Figure 4.2: Jitter-EDD

4.5 Algorithms for proportional delay differentiation

A model for proportional differentiation was described in chapter 2. In [DSR99], the model has been applied to delay differentiation and is referred to as Proportional Delay Differentiation (PDD) model.

Let there be N FCFS queues, one for each traffic class. Denote by $\bar{d}_i$ the average queueing delay of class i packets and by δ_i the Delay Differentiation Parameter (DDP) of class i. Then, according to the PDD model the ratio of average delays in two classes i and j should equal the ratio of DDPs in these classes

$$\frac{\bar{d}_i}{\bar{d}_j} = \frac{\delta_i}{\delta_j}, \quad 1 \leq i, j \leq N. \tag{4.18}$$

It is further assumed that $\delta_1 > \delta_2 > \ldots > \delta_N$. In [DSR02] the PDD model is interpreted so that the normalized average delays of traffic classes must be equal, i.e.,

$$\tilde{d}_i = \frac{\bar{d}_i}{\delta_i} = \frac{\bar{d}_j}{\delta_j} = \tilde{d}_j, \quad 1 \leq i, j \leq N. \tag{4.19}$$

In [DSR02] three scheduling disciplines, PAD, WTP and HPD for implementing the PDD model are described. All these algorithms are originally designed for the DiffServ environment, i.e. they aim at resource allocation for traffic classes.

4.5.1 PAD

The Proportional Average Delay (PAD) scheduler attempts to keep the normalized average delays of classes equal. It selects for transmission at time t, when the server becomes free, a packet from a backlogged class j with the maximum normalized average delay [DSR02]:

$$j = \arg\max \tilde{d}_i(t). \tag{4.20}$$

The idea is that choosing a packet from class j reduces the differences between normalized average delays of the classes.

In Equation 4.20 the normalized average delay of class i at time t, $\tilde{d}_i(t)$, is needed. Denote by $D_i(t)$ the sequence of class i packets that have been served before time t and by d_i^m the delay of the m'th packet in $D_i(t)$. Then, $\tilde{d}_i(t)$ can be calculated as follows, assuming that at least one packet has departed from class i before t

$$\tilde{d}_i(t) = \frac{1}{\delta_i} \frac{\sum_{m=1}^{|D_i(t)|} d_i^m}{|D_i(t)|} = \frac{1}{\delta_i} \frac{S_i}{P_i}. \tag{4.21}$$

S_i is the sum of the queueing delays of packets that have departed before time t and P_i is the number of these packets. Whenever a new packet is selected for transmission from queue j, S_j and P_j are updated to $S_j = S_j + d_j^{P_j+1}$ and $P_j = P_j + 1$, and a new value for $\tilde{d}_j(t)$ is calculated.

Simulation results in [DSR02] indicate that PAD is able to approximate the PDD model closely, provided that the selected DDPs are feasible. However, in short timescales the differentiation provided by PAD may be highly unpredictable. This is due to the fact that PAD only considers mean delays of the classes and does not take into account the waiting times of the packets that are currently backlogged. Thus, in some situations, the delays experienced by the higher classes can be considerably above the delays of the lower classes.

4.5.2 WTP

The Waiting Time Priority (WTP) scheduling algorithm is based on time dependent priorities originally studied by L. Kleinrock. In WTP, a packet is associated with a priority that varies in proportion to the waiting time of the packet. If a class i packet arrives at time τ, its priority increases with rate $1/\delta_i$

$$p_i(t) = (t - \tau) \times 1/\delta_i. \tag{4.22}$$

In [DSR99] WTP is used for proportional delay differentiation. Denote by $w_i(t)$ the waiting time of the head of line packet of class i at time t. The normalized head waiting time of class i is then defined as

$$\tilde{w}_i(t) = \frac{w_i(t)}{\delta_i}. \tag{4.23}$$

While PAD selects for transmission a packet from class j with the largest normalized average delay, WTP uses the maximum normalized head waiting time as a criterion for choosing class j:

$$j = \arg\max \tilde{w}_i(t). \tag{4.24}$$

Thus, the differences between the normalized head waiting times of classes will be minimized in the long run. If these differences are zero, WTP provides proportional differentiation between the delays of successive packet departures [DSR02].

Contrary to PAD, WTP is able to provide predictable differentiation even in small timescales: It does not violate the right class ordering since it monitors the waiting times of individual packets rather than long term averages. WTP also converges to the PDD model as the utilization of the server approaches 1, such that proportional differentiation can be provided with respect to mean delays [DSR02]:

$$\frac{\bar{d}_i^{WTP}}{\bar{d}_j^{WTP}} \rightarrow \frac{\delta_i}{\delta_j} \qquad \text{as} \quad u \rightarrow 1. \tag{4.25}$$

However, as the utilization decreases, WTP is not able to approximate the PDD model closely. This is because WTP is not originally designed to approximate long term mean delay ratios. Adaptive versions of WTP have been proposed for instance in [EBA01] and [LLY01] that try to achieve more precise approximation of the PDD model when utilization is below 1.

4.5.3 HPD

It is shown in [DSR02] that the WTP scheduler can approximate the PDD model, but only when the utilization is over 0.85. The PAD scheduler, on the other hand, approximates the PDD model well in long time intervals but behaves pathologically in short time scales. To solve these problems, the authors in [DSR02] have proposed a hybrid scheduler, HPD (Hybrid Proportional Delay), that takes into account both the delay history and the current delay by combining the properties of WTP and PAD schedulers. This hybrid scheduler selects for transmission at time t, when the server becomes free, a packet from a backlogged class j with the maximum normalized hybrid delay [DSR02]:

$$j = \arg\max\,(g\tilde{d}_i(t) + (1-g)\tilde{w}_i(t)), \tag{4.26}$$

where $0 \leq g \leq 1$. By simulations it is verified that the HPD scheduler meets the PDD model both in short and long time scales and with different loads with a proper selection for the parameter g.

4.5.4 Adaptive WFQ

Fair queueing schedulers, such as WFQ, have also been considered as an option to provide relative delay differentiation. This is because in fair queueing schedulers the delay of a traffic class is related to the weight of the class. For example the so called Olympic Service model introduced in [HBWW99] could be implemented by using some variant of WFQ. However, it is stated in [Bod00] and [DSR02] that fair queueing schedulers are sensitive to variations in the class load distribution. If static class weights are used, a priority inversion between classes may easily occur when the load to some traffic class increases. One solution for this problem is to

adapt the class weights to the dynamic network conditions [WSS01], [SKLK01]. The adaptation could be based for instance on round trip time, packet loss rate or filtered queue length.

In [SKLK01], an adaptation scheme based on the filtered queue length is proposed. Denote by $q_i(t)$ the filtered queue length of class i at time t. The ratio of WFQ weights $w_i(t)$ and $w_j(t)$ for the interval (τ, t) are set to

$$\frac{w_j(t)}{w_i(t)} = \frac{\delta_i}{\delta_j}\frac{q_j(t)}{q_i(t)}. \tag{4.27}$$

Since $\sum_i w_i(t) = 1$, the adaptive weighting factors are given by

$$w_i(t) = \frac{q_i(t)}{\sum_{k=1}^{n} \frac{\delta_i}{\delta_k} q_k(t)}. \tag{4.28}$$

Obviously selecting an appropriate adaptation interval (τ, t) is problematic. If the interval is too long, adaptation to changing load conditions can not be achieved. On the other hand, if the weights are updated too often, the scheduling algorithm is far from original fair queuing.

4.6 Algorithms for proportional deadline violation probability differentiation

4.6.1 Weighted EDD

The Proportional Differentiation model can be applied also to other performance metrics. In [Bod00], the model is used in differentiating the deadline violation probabilities of traffic classes. This new model corresponds to the PDD model except that the mean delays $\bar{d}_i$ are replaced with deadline violation probabilities p_i such that

$$\frac{p_i}{p_j} = \frac{\delta_i}{\delta_j}. \tag{4.29}$$

Thus higher classes will receive better performance in the sense that the probability of deadline violation is smaller than for lower classes. This kind of differentiation is justified by the fact that for some applications, such as IP telephony the probability that the delay exceeds a certain threshold is more meaningful than the mean delay.

In [Bod00], a new scheduler WEDD is proposed. WEDD is an extension of the classical EDD scheduler in the sense that it provides differentiation both in terms of delay and deadline violation probabilities. WEDD operates in the same way as EDD except when the system is in "congestion mode". In [Bod00] the system is defined to be in congestion mode if there is more than one backlogged class with the first packet having a deadline $t_A + d_i < t_s + \Delta_i$, where t_A denotes the arrival time of the

packet, t_s the current system time and Δ_i is a safety margin. A congestion tag c_i is calculated for each congested class as follows:

$$c_i = \frac{\delta_i}{E_i(t_s)}, \tag{4.30}$$

where $E_i(t_s)$ is an estimate for the real deadline violation ratio in class i based on measurements. The packets are served in ascending order of congestion tags.

Chapter 5

Quality differentiation

5.1 Introduction

In the previous chapters various quality of service mechanisms were introduced. The focus was on scheduling disciplines that can be deployed in the forwarding plane of an Internet router. The rest of this thesis handles the issues related to quality differentiation. The starting point is that if an operator considers to provide quality differentiation at least the following questions have to be answered:

1. What are the possible service approaches for the operator?
2. What kind of differentiation models can the operator use to support these service approaches?
3. What QoS mechanisms are required to implement the differentiation models?
4. How is traffic mapped to different service classes?
5. What are the costs and benefits of implementing the differentiation models?

This chapter handles the first two questions by analyzing the possible service approaches that an operator can adopt and differentiation models for supporting these approaches.

5.2 Service approaches

If an operator decides to provide QoS in the first place, there are basically four reasonable service approaches:

1. Serve applications with highest QoS requirements
2. Serve customers paying the highest price
3. Serve customers with highest QoS requirements
4. Combination of the first three approaches

In [Kil02] different service approaches of an operator are analyzed. It is emphasized that QoS mechanisms should be selected primarily based on operator's business objectives whereas the requirements of applications have only a minor role. It is obvious that any reasonable operator aims at maximization of revenue. At first it might seem clear that this objective is best met by adopting the second approach. However, if the requirements of applications are totally neglected, it is questionable whether the revenue can be maximized in the long run.

Favoring applications with highest QoS requirements

Most users are not at all interested in technical details and want to use telecommunication services with as little extra effort as possible. It can even be argued that quality is something that the user thinks about only when it is missing: as long as the applications work properly, the users are satisfied and do not care much about which quality of service class they belong to. In this sense it would be reasonable to design the QoS system so that the requirements of applications would be met while all the technical details would be hidden from the end users. Naturally this does not mean that the system should satisfy the requirements of every possible application. In practice the applications would be divided into a small amount of groups that would then be mapped to different service classes with a suitable forwarding treatment. If provisioning and mapping of traffic is properly done, more customers could probably be served with the same resources compared with the situation where traffic mapping is based only on money. Also, as stated in [LIP99], if the differentiation is based on the requirements of applications, the treatment of packets will be similar both in upstream and downstream direction, which leads to more consistent differentiation.

Favoring customers paying the highest price

On the other hand, a natural approach for an operator is to provide more resources to the customers that generate more income: there seems to be no reason why the operator should prioritize some users or applications only because they need or require more resources. However, if a QoS system is built solely based on the prices paid by the customers, the traffic mix inside the network can not be properly controlled. If the mapping of traffic to service classes is steered by money, traffic with totally different characteristics could be mixed in the same class. This in turn could lead to a situation where individual applications do not work just because of ineffective mapping of traffic and some customers would be unsatisfied. One solution to alleviate quality deterioration would be to include incentives for the users (such

as price) not to require more resources than they actually need. However, the price differences between service classes should be substantial and still, the mixing of different traffic types could not be totally prevented.

Favoring customers with highest QoS requirements

One basic objective of an operator is to serve the needs of the customers. It was stated earlier that satisfying the requirements of applications is usually what the customer actually needs. However, it is surprising that if the customers are allowed to require freely what their want, the requirements may not have anything to do with the actual quality requirements of the applications. One reason for this is that for some users the main motivator is to receive better quality than the others regardless of whether they need it. Another reason is the fact that ordinary Internet users do not have a vaguest idea for instance about how much bandwidth their application really needs or what is the limit for delay jitter. In this sense it can be argued that the operator knows better what the customer needs than the customer himself. Also if the customer is requiring more resources and is willing to pay for them (as in the second approach) this is a stronger indication of a need than just a pure requirement.

Combination of the first three approaches

It is probable that none of these service approaches alone would be reasonable: the second approach could lead to unreasonable traffic mixes and cause unnecessary deterioration of quality while the first approach would fail to take into account the customers willingness to pay a higher price for a better service. One option is to adopt the second approach but somehow restrict the number of classes into which certain traffic type can be mapped, such that for example VoIP and FTP traffic would not be mixed in any case. Another option is to adopt the first approach and to use for example the access to different services as a basis for charging the customers differently, or to associate the money with admission control decisions. Yet another option would be to build the system hierarchically so that different traffic types would first be separated into individual classes, and resources would then be divided within these classes based on money.

5.3 Differentiation models

The following differentiation models (or combinations of them) can be used to realize the service goals of the operator:

- Capacity differentiation
- Delay differentiation
- Pricing differentiation

- Predictability differentiation

In addition, importance differentiation is another option but is not considered in this context since the focus of the thesis is on differentiation that is possible with packet scheduling. It should also be noted that if the service approach of the operator is to favor the applications with highest quality requirements, pricing differentiation is not possible as such.

5.3.1 Capacity differentiation

Throughput is one of the most important quality parameters. Especially for elastic applications flow completion time is significant while the delays experienced by individual packets can vary quite freely. Throughput differentiation can be either absolute or proportional: In absolute throughput differentiation each service class is allocated a predefined amount of link capacity. In proportional throughput differentiation the throughput of a class at a given time is proportional for example to the price of the class. It should be noted that the level of service for individual flows depends largely on the class loads.

5.3.2 Delay differentiation

Realtime applications, such as VoIP may not require much throughput but set stringent requirements for delay and delay jitter. During recent years various models for delay differentiation have been proposed, such as the PDD model presented in Chapter 4. Usually these models aim at quantifying the delay spacing between different classes (i.e. the distance between delays) such that the spacing is either proportional or additive. There have also been efforts to integrate absolute delay bounds with the proportionality constraints so that applications with most stringent delay and jitter requirements would function even in the presence of high loads. The possible delay differentiation models include:

- Proportional delay differentiation
- Additive delay differentiation
- Proportional delay differentiation with absolute delay bound
- Deadline violation probability differentiation
- Jitter differentiation

Delay differentiation has previously been studied in [DSR99], [DSR02], [SM02] and [SCF02].

5.3.3 Pricing differentiation

One option is that the differentiation model is based only on price. For example in the Paris Metro model different traffic classes are assigned with different prices with the assumption that the most expensive classes will be only lightly loaded and will receive good quality of service. However, it is very questionable if price alone would be a sufficient mechanism for quality differentiation since no guarantees, not even relative, could be made about the quality levels of different classes at different times.

5.3.4 Predictability differentiation

Predictability of quality is perhaps one of the most important aspects affecting the user experience: It can be argued that most users are more satisfied if the quality is moderate all the time than if the quality is sometimes perfect and sometimes bad, even if the average quality would be better. Predictability is definitely something that the users would be ready to pay for but the implementation of predictability differentiation could be difficult in practice. It would require a mechanism to prioritize ongoing flows over those flows that have just started [Kil99].

5.4 Implementation of differentiation models

In the previous section possible models for differentiation were introduced. In this section the purpose is to analyze how these models could be implemented. The focus will be on capacity and delay differentiation.

5.4.1 Basic implementation options

The operator has basically two options for supporting the requirements of different applications: one option is to forget about all the QoS mechanisms and simply increase the link capacities until the requirements of both realtime and elastic traffic are satisfied. However, this kind of over provisioning of resources is quite inefficient and often also impossible. Furthermore, if there would be enough resources for everyone, there would be no need for capacity or delay differentiation either. Another option is to use packet schedulers for allocating resources between different service classes, which is more efficient than over provisioning. However, adding QoS functionality into network devices requires investments as well. Thus it must be considered whether the benefits that can be achieved by implementing possibly complex mechanisms are larger than the costs.

5.4.2 Selection of packet schedulers

Various packet schedulers have been proposed for capacity and delay differentiation. Based on the findings of the literature survey conducted in this thesis these schedulers can first be compared qualitatively in order to get an idea of which schedulers could be suitable for implementing the desired quality models. After this preliminary selection these schedulers can be implemented and tested in a simulator.

Selecting a packet scheduler for capacity differentiation

Scheduler alternatives for absolute capacity differentiation are mainly the rate based schedulers presented in chapter 4. Among these schedulers DRR seems to be a suitable option because of its simplicity of implementation, ability to emulate the ideal GPS algorithm and to take into account variable packet sizes. Naturally DRR does not emulate GPS as accurately as the traditional WFQ algorithm, but deploying as complex algorithm as WFQ in routers would be overkill. If hierarchical division of capacity is desired, Class Based Queueing (CBQ) is the algorithm of choice. However, also in CBQ the general scheduler could be based on the DRR algorithm.

Selecting a packet scheduler for delay differentiation

If the delay spacing between different service classes is supposed to be controllable (additive or proportional), this calls for a measurement based scheduler. There are basically two main options for implementation: a rate based scheduler where the weights are adapted so that the delay ratios between classes remain the same independent of load conditions or a scheduler that is based on time dependent priorities.

Among the rate based schedulers an adaptive WFQ scheduler that modifies the weights according to the filtered queue length, as proposed in [SKLK01] seems to be a reasonable alternative mostly because of its simple implementation and low overhead. Naturally the updating of weights could also be based on measuring network load or round trip times but this would complicate the algorithm. The actual rate based algorithm used can be some of the algorithms meant for capacity differentiation, such as DRR. In general, there are very few proposals in the literature for adapting the weights in a simple and controllable manner.

Among the time dependent priority schedulers mainly three options exist: WTP, PAD and HPD. According to simulation studies conducted in [DSR02] HPD provides most consistent differentiation both in short and long time scales independent of class loads. Also intuitively HPD seems to be the best of these three algorithms since it takes into account both current delays and the delays in the history.

Proportional delay differentiation as such may not be able to provide sufficient quality of service especially when the load level is high. However, the most delay sensitive class can be assigned with a delay bound (such as in the EDD algorithm) so that whenever this class is about to violate its deadline, the packet is scheduled from

this class immediately instead of checking the proportionality constraints. Thus an appropriate scheduler for integrating proportional delay constraints with absolute constraints could be a combination of adaptive DRR and EDD, or HPD and EDD. Further, it is also possible to assign each class a deadline and differentiate with the probability that this deadline is violated. For this kind of deadline violation probability differentiation the WEDD algorithm proposed in [Bod00] would be suitable.

It should be noted that proportional delay differentiation also leads to proportional differentiation of capacity for elastic traffic (with small propagation delay) since the throughput B and round trip time of a TCP flow are related as follows:

$$B \sim \frac{1}{RTT\sqrt{p}}, \tag{5.1}$$

where p denotes the steady state packet loss.

Selecting a packet scheduler for combined capacity and delay differentiation

A common approach for supporting the requirements of both elastic and realtime traffic is to use some rate based scheduling algorithm for absolute capacity division among elastic traffic and possibly strict priority queueing for the realtime traffic. However, a more efficient option would be to combine for example static DRR with EDD. With this combination the realtime class could be assigned with a deadline and moreover, all the packets violating the deadline could be dropped directly so that they would not waste the capacity.

In Table 5.1 the suitable schedulers for capacity and delay differentiation are presented.

Quality parameter	**Differentiation model**	**Selected packet scheduler**
Capacity	Absolute	DRR CBQ
Delay	Proportional	Adaptive DRR HPD
	Proportional with delay bound	Adaptive DRR with delay bound HPD with delay bound
	Deadline violation probability	WEDD
Capacity and delay	Absolute capacity with delay bound	DRR with delay bound

Table 5.1: Selected packet schedulers

5.5 Mapping of traffic to service classes

5.5.1 Introduction

When the operator has defined its service approach, selected the differentiation models and decided which mechanisms to use in implementing these models, there is still one question remaining: How should the traffic from the users be mapped to different service classes? The answer to this question depends totally on the service approach adopted: if the operator chooses to build the QoS system based on the requirements of key applications, also the decision of traffic mapping is left for the operator. However, if the quality differentiation is steered by the prices paid by customers, mapping is at least to some extent out of the control of the operator. In this chapter the possible options to do traffic classification are briefly described.

5.5.2 Classification based on money

If classification is based on money, this essentially means that the user can decide how his traffic will be mapped provided that he is willing to pay a certain price for it. One extreme of this kind of classification is that the user selects one service class for all of his traffic. Another extreme is that even individual packets of one flow from the user could be mapped to different service classes. However, in this case the mapping would require a mechanism to signal the changing preferences and handle the billing. This kind of operation is not supported in DiffServ.

Classification that is based on money means that the traffic distribution within service classes can not be controlled by the operator: at some moment there could be for example 70% WWW and 30% FTP traffic in some class and in another moment this distribution could again be different.

5.5.3 Automatic classification

Contrary to classification that is based on money, in automatic classification the classification rules could be totally hidden from the end users. This requires automatic classification mechanisms to be deployed in routers. In [IL01] and [LIP99] various methods to do this kind of classification have been proposed. The methods are mainly based on header analysis and traffic analysis. With header analysis it is possible to detect applications by examining for instance the IP addresses, port numbers and protocol identifiers in IP packet headers. Traffic analysis, on the other hand, investigates the characteristics of traffic, such as packet sizes and packet inter arrival times.

Automatic classification leaves many degrees of freedom for the operator to map the

traffic to service classes. Figure 5.1 illustrates possible ways to divide applications to a small number of groups, each group representing a separate traffic class.

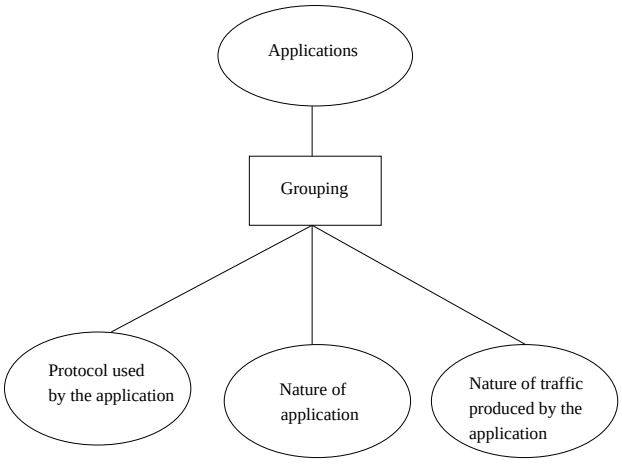

Figure 5.1: Grouping of applications

Chapter 6

Evaluation of different schedulers

6.1 Introduction

In the previous chapters possible scheduling disciplines for supporting various differentiation models were proposed. The mapping of traffic to service classes was also discussed. In this chapter the purpose is to evaluate the proposed scheduling disciplines based on simulations. First, the simulation environment, simulation model, used traffic models and simulation scenarios are described. Finally, the results of the scenarios for the selected scheduling disciplines are presented and analyzed.

6.2 Goals of the simulations

The main goal of the simulations conducted in this thesis is to answer the following questions:

- How well are the proposed scheduling disciplines able to support the quality model that they are designed for?
- What is the effect of the traffic mix on the results?

In many papers scheduling disciplines are evaluated by using simple traffic sources that do not represent realistically the traffic in the Internet. This kind of evaluation gives little information because the traffic processes may have significant impact on the performance of a scheduling discipline. In this simulation study the purpose is to evaluate the scheduling disciplines with traffic patterns that describe the key characteristics of the Internet traffic.

6.3 Simulation environment

6.3.1 CNCL

The simulator used in this thesis has been implemented with CNCL [CNC98], which is a freeware C++ class library package designed by the Communication Networks group in Aachen University of Technology. The class library consists of the functionality required to support event-driven simulation:

- data structures
- random number generation
- event scheduling and handling
- collection of statistics
- simple analysis of statistics

CNCL provides only the basic blocks needed in a simulation; the user has to implement other functionalities in C++. For example, when simulating a communications network, the user has to create the nodes, links, protocols, traffic sources and all the mechanisms in the network. This makes the construction of simulations with CNCL very time consuming. On the other hand, the execution of the simulation is fast, because no high level language is used to support an easy configuration of objects. Another advantage is that when the user has implemented almost all the functionalities by himself, the simulator is easier to control. The user also has more freedom in designing the simulator, since there are only few ready made blocks that restrict the implementation choices.

In CNCL the main simulation objects are implemented as event handlers, which exchange events with each other. The simulation time is advanced each time an event is received by one of the event handlers. In practice, event handlers are implemented as state machines: the state machine executes different methods depending on the type of the event that it received and may also send new events either to itself or to some other event handler.

In Figure 6.1 the basic idea of event handlers is shown in a simple case where there is one traffic source, node, link and sink. The arrows in the figure are indications of incoming events. The event types in the figure are the following:

1. **EV_PKT** : This event is an indication of an incoming packet. The traffic sources send EV_PKT events to the node, and when the link becomes free to serve a new packet, the node sends an EV_PKT event to the link. Finally, when the link has served the packet, it sends EV_PKT event to the sink that deletes the packet.

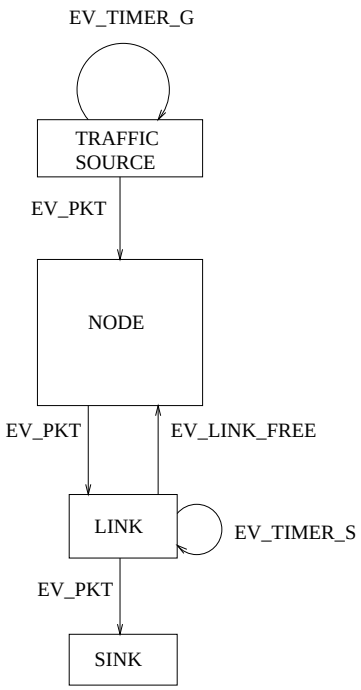

Figure 6.1: The events in the system

2. **EV_TIMER_G** : This is a timer event that triggers the event_handler() method of the traffic generator class. The frequency of this event determines the inter arrival time of the packets generated by the traffic source.

3. **EV_LINK_FREE** : When the link becomes free, it informs the node object.

4. **EV_TIMER_S** : This event indicates that the time required to serve the packet has elapsed and the link is free again.

6.4 Simulation model

The simulation model used for evaluating the scheduling disciplines consists of the following components

- Node and link models
- Traffic generator models
- TCP model

6.4.1 Node models

Three types of nodes are used in the simulations: source nodes, routers and sinks.

- **Source nodes** are the points that connect the traffic generators with the network. Source nodes store incoming packets in a queue with unlimited size and send them in FCFS order to access links.
- **Router** is a node that contains logic to perform more advanced forwarding decisions. It consists of a classifier and a queuing system, as depicted in Figure 6.2.
  - **Classifier** classifies the incoming packets based on their classid.
  - **Queuing system** consists of FCFS queues of finite size, one for each traffic class, and a scheduling algorithm. The packets are enqueued in the right queue based on their classid, provided by the Classifier object. When the link becomes free to serve the next packet, the scheduling algorithm decides from which queue a packet will be transmitted.
- **Sink** is a destination node where all incoming packets are deleted. The sink is also responsible for reassembling arriving TCP segments into packets and for sending acknowledgements to the TCP senders.

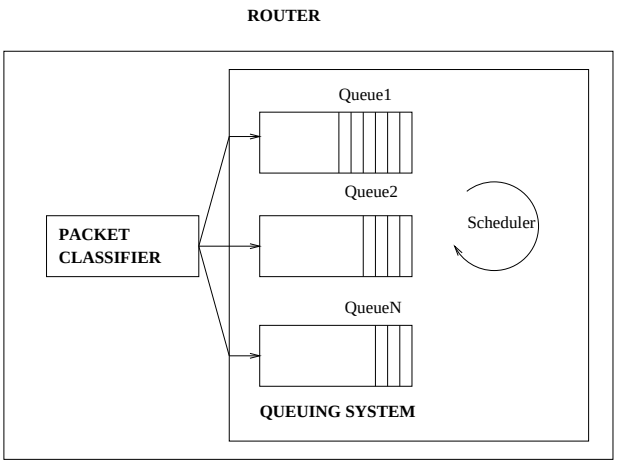

Figure 6.2: Router

6.4.2 Traffic generator models

Traffic generators create flows representing different types of traffic. The traffic types used in the simulation are control traffic, VoIP, Video, HTTP and FTP. HTTP and FTP use TCP as transport protocol while other sources are assumed to use UDP.

In real HTTP and FTP sessions a client sends request messages to the server and the server responds by sending the requested document or file. However, in the simulation model the HTTP and FTP generators are assumed to produce traffic only from the server side. A fictitious client is used that sends request messages to the server at certain intervals but no traffic from the client is actually sent to the network. This simplification is justified since the traffic produced by the client is marginal compared with the traffic produced by the server.

6.4.3 TCP model

The purpose of the TCP model is not to describe the behavior of real TCP at detailed level but rather to capture the effects of TCP's feedback loop on the produced traffic. The following components and mechanisms of TCP are modeled in the simulator:

- **Send buffer**: The send buffer is modeled as a priority queue where new segments are put into the queue with priority 0 and retransmissions with priority 1.
- **List of unacknowledged segments**: All the segments that have been transmitted but not yet acknowledged are stored in this list.
- **Rtt and timeout estimation**: Jacobson/Karels algorithm with exponential backoff is used for estimating the round trip time of the connection and for calculating the current time out value.
- **Retransmission timer**: A timer is associated with each transmitted packet. The value of the timer corresponds to the estimated timeout value. Packet losses are detected only via timeouts.
- **Congestion control**: The congestion control mechanism of the model consists of the slow-start and congestion avoidance phases. The congestion window *cwnd* and slow start threshold *sstresh* are updated as follows: At the beginning of a connection *cwnd* is set to 1 and after each received ACK it is updated to

  $$cwnd = cwnd + 1$$

  When a packet loss is observed, the value of *sstresh* is updated to

  $$sstresh = cwnd/2.$$

  Then the connection starts from slow start again with $cwnd = 1$, and the window is increased exponentially until it equals *sstresh*. At this point, the connection goes to congestion avoidance state where the congestion window is increased linearly with the pattern

  $$cwnd = cwnd + 1/cwnd.$$

In the TCP model *cwnd* is calculated in packets instead of bytes. The effective window of TCP, i.e. the amount of new packets that can be sent to the network corresponds to

$$effectivewindow = cwnd - unack,$$

where *unack* denotes the number of packets that have been transmitted but not yet acknowledged.

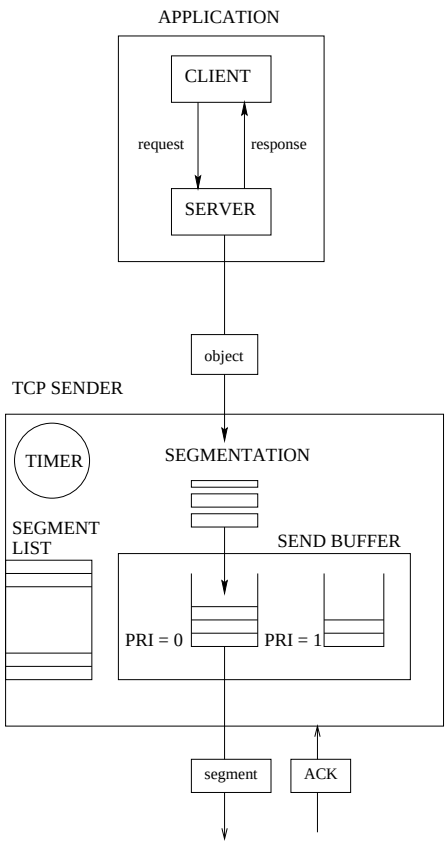

Figure 6.3: TCP model

The following event types are modeled in TCP's state machine in the simulator:

1. **Packet arrival**: When a new packet arrives from the application layer (from HTTP or FTP server), it is first segmented into segments of size MSS bytes. These segments are then put into the send buffer with priority 0 as shown in Figure 6.3. If the packet was the first packet of the session, initialization event is scheduled immediately.

2. **Initialization event**: When this event is received, the sending event is scheduled. Initialization event is used in two occasions: either at the beginning of the connection in order to receive the first acknowledgement or after detection of packet loss in order to send one packet to the network.

3. **Sending event**: The sending event takes a segment from the head of the send buffer and sends it immediately into the network. A retransmission timer is associated with this segment and a copy of the segment is stored into a list of unacknowledged segments.

4. **Acknowledgement**: An acknowledgement causes the TCP sender to update its congestion window, RTT and timeout estimates and to send as many new packets to the network as the effective window allows. The segment that was acknowledged is also removed from the list of unacknowledged segments.

5. **Timer event**: When the timer for a particular segment fires, it is first checked whether this segment has already been acknowledged. If not, the timer event is interpreted as an indication of packet loss. This causes the TCP sender to update its congestion window, schedule the initialization event and put the segment for which the timer fired into the send buffer with priority 1.

In the simulator acknowledgements are sent by the sink. An acknowledgement is sent each time when a new packet is received, regardless of whether all the previous packets have been received or not. The implementation does not contain the actual connection establishment or tear down phases present in real TCP. However, if the last packet of the session has arrived, the send buffer is empty and there are no unacknowledged segments, the TCP sender can be deleted.

6.5 Source models

The idea of the source models used in the simulator is to capture the essential characteristics of different traffic types present in the Internet. The models may not describe the applications at a detailed level but for the purposes of the simulations they model the traffic produced by key applications accurately enough.

6.5.1 HTTP model

The model for HTTP traffic used in the simulations is based on the empirical results of [Mah97]. According to this paper a HTTP session consists of a number of page requests, where the time between between consecutive requests corresponds the time that the user spends on reading the page. Each requested page may in turn contain several objects, such as text and pictures. In the simulations similar distributions and parameters as in [Luo00] were used for the model:

- **Session arrivals:** Poisson process, mean inter arrival time $\bar{\tau}$ is varied to generate a certain load level.

- **Number of page requests in a session:** Geometric distribution with the mean $\bar{p}$ of 5 requests.
- **Reading time:** Geometric distribution with the mean of 12 s.
- **Number of objects in a page:** Geometric distribution with the mean $\bar{o}$ of 3 objects.
- **Object size:** Geometric distribution with the mean $\bar{s}$ of 3300 bytes

The mean load of HTTP traffic in bytes can be calculated as:

$$\bar{\rho} = \frac{1}{\bar{\tau}} \frac{\text{sessions}}{\text{s}} \cdot \bar{p} \frac{\text{pages}}{\text{session}} \cdot \bar{o} \frac{\text{objects}}{\text{page}} \cdot \bar{\text{s}} \frac{\text{bytes}}{\text{object}}$$

6.5.2 FTP model

FTP traffic is modeled in a similar fashion as HTTP traffic with the exception that there is neither reading time nor multiple objects in a file request. Again, similar distributions and parameters as in [Luo00] were used for the model:

- **Session arrivals:** Poisson process, mean inter arrival time $\bar{\tau}$ is varied to generate a certain load level.
- **Number of file requests in a session:** Geometric distribution with the mean $\bar{f}$ of 5 requests.
- **File size:** Geometric distribution with the mean $\bar{s}$ of 500 000 bytes

The reason for a relatively small file size is that in the simulations it should be possible to control the load level of FTP traffic, i.e. FTP is not modeled as a greedy application. The mean load of FTP traffic in bytes can be calculated as:

$$\bar{\rho} = \frac{1}{\bar{\tau}} \frac{\text{sessions}}{\text{s}} \cdot \bar{f} \frac{\text{files}}{\text{session}} \cdot \bar{s} \frac{\text{bytes}}{\text{file}}$$

6.5.3 VoIP model

In the simulator VoIP traffic is modeled both at the call and packet level. The following distributions and parameters were used for VoIP:

Call level

- **Call arrivals:** Poisson process, mean inter arrival time $\bar{\tau}$ is varied to generate a certain load level.
- **Call holding time:** Exponential distribution with the mean $\bar{h}$ of 180 s.

Packet level

- **On period:** Exponential distribution with the mean $\overline{on}$ of 800 ms.
- **Idle period:** Exponential distribution with the mean $\overline{idle}$ of 1 s.
- **Packet inter arrival time:** 20 ms, $\bar{i}$.
- **Packet size:** 160 bytes, $\bar{s}$.

The mean load of VoIP traffic in bytes can be calculated as:

$$\bar{\rho} = \frac{1}{\bar{\tau}} \frac{\text{sessions}}{\text{s}} \cdot \bar{h} \frac{\text{s}}{\text{call}} \cdot \frac{\overline{on}}{\overline{on} + \overline{idle}} \cdot \frac{\bar{s}}{\bar{i}} \frac{\text{bytes}}{\text{s}}$$

6.5.4 Video model

For video traffic, the following parameters were used:

Session level

- **Session arrivals:** Poisson process, mean inter arrival time $\bar{\tau}$ is varied to generate a certain load level.
- **Session length:** Exponential distribution with the mean $\bar{h}$ of 180 s.

Packet level

- **Packet inter arrival time:** 12 ms, $\bar{i}$.
- **Packet size:** Normal distribution with the mean $\bar{s}$ of 750 bytes.

The mean load of video traffic in bytes can be calculated as:

$$\bar{\rho} = \frac{1}{\bar{\tau}} \frac{\text{sessions}}{\text{s}} \cdot \bar{h} \frac{\text{s}}{\text{session}} \cdot \frac{\bar{s}}{\bar{i}} \frac{\text{bytes}}{\text{s}}$$

6.5.5 Control traffic model

A simple way to model network control traffic, such as DNS messages or routing database updates is to use constant size packets sent at constant intervals. In the simulator it is further assumed that DNS traffic consists of small packets that are sent in bursts whereas routing database updates are larger packets, sent one at a time. The following parameters were used in the simulator for DNS and routing messages:

DNS messages

- **Burst inter arrival time:** inter arrival time $\bar{\tau}$ is varied to generate a certain load level.
- **Burst size:** 5 packets, $\bar{b}$.
- **Packet size:** 256 bytes, $\bar{s}$.

The mean load of DNS traffic in bytes can be calculated as:

$$\bar{\rho} = \frac{1}{\bar{\tau}} \frac{\text{bursts}}{\text{s}} \cdot \bar{b} \frac{\text{packets}}{\text{burst}} \cdot \bar{s} \frac{\text{bytes}}{\text{packet}}$$

Routing messages

- **Packet inter arrival time:** inter arrival time $\bar{\tau}$ is varied to generate a certain load level.
- **Packet size:** 1500 bytes, $\bar{s}$.

The mean load of routing messages in bytes can be calculated as:

$$\bar{\rho} = \frac{1}{\bar{\tau}} \frac{\text{packets}}{\text{s}} \cdot \bar{s} \frac{\text{bytes}}{\text{packet}}$$

6.6 Simulation scenarios

6.6.1 Simulated differentiation models and packet schedulers

The following differentiation models were selected for the simulations:

- Absolute capacity differentiation

- Proportional delay differentiation with delay bound
- Deadline violation probability differentiation

Absolute capacity differentiation was selected since it is the most common differentiation model. Proportional delay differentiation model was in turn selected because of its tunability. However, the proportional model was integrated with absolute delay bound because proportional differentiation of delay as such is not able to satisfy the delay and jitter requirements of real time traffic. Finally, deadline violation probability differentiation model was selected because of its tunability and unfamiliarity. It is interesting to see what the performance actually is with this kind of differentiation model.

Table 6.1 shows the packet schedulers that were implemented in the simulator for supporting the selected differentiation models.

Packet scheduler	**Quality parameter**	**Differentiation model**
DRR	Capacity	Absolute
ADRR with delay bound	Delay	Proportional with delay bound
HPD with delay bound	Delay	Proportional with delay bound
WEDD	Delay	Deadline violation probability

Table 6.1: Implemented packet schedulers

6.6.2 Traffic mixes

The relative traffic shares used for different applications in the simulations are depicted in Table 6.2. The total load consists of FTP-traffic (9 %), WWW-traffic (71 %), Video-traffic (9 %), VoIP-traffic (10 %) and Control traffic (1 %). It should be noted that the estimated shares correspond to the situation about six months to one year from now. Thus for instance the fraction of real-time traffic is larger than what it is today. The estimates are based on traffic measurements conducted in Networking Laboratory and predictions about future growth trends.

Application	**Share**
FTP	9 %
WWW	71 %
Video	9 %
VoIP	10 %
Control	1 %

Table 6.2: Relative shares of different applications

Tables 6.3 - 6.11 present the simulation scenarios. In all scenarios the relative shares of applications are the same as in Table 6.2 but the mapping of traffic to service

classes is varied. The percentages in the tables indicate which portion of the total amount of the applications traffic belongs to a certain class. For example, in scenario 1 all FTP and WWW traffic is mapped to class 0 while all video, VoIP and control traffic is mapped to class 1.

Scenario 0 corresponds to a best effort scenario where no traffic classification is performed. In simulation scenarios 1-4 it is assumed that traffic classification is done automatically by the network. Scenarios 5-8 on the other hand correspond to a situation where the classification is based on money. The differences between the simulation scenarios are listed below.

Automatic classification:

- **Scenario 1:** Classify traffic based on the transport protocol.
- **Scenario 2:** Classify traffic based on the transport protocol and further separate long and short TCP flows.
- **Scenario 3:** Classify traffic based on the type of application.
- **Scenario 4:** Classify traffic based on traffic characteristics.

Classification based on money:

- **Scenario 5:** Mix FTP traffic with WWW traffic.
- **Scenario 6:** Mix WWW traffic with video traffic.
- **Scenario 7:** Mix FTP traffic with WWW traffic and WWW traffic with video traffic.
- **Scenario 8:** Mix FTP traffic with WWW traffic, WWW traffic with video traffic, video traffic with VoIP traffic and WWW traffic with VoIP traffic.

	Traffic type				
Class	FTP	WWW	Video	VoIP	Control
0	100 %	100 %	100 %	100 %	100 %

Table 6.3: Scenario 0

	Traffic type				
Class	FTP	WWW	Video	VoIP	Control
0	100 %	100 %	0 %	0 %	0 %
1	0 %	0 %	100 %	100 %	100 %

Table 6.4: Scenario 1

	Traffic type				
Class	FTP	WWW	Video	VoIP	Control
0	100 %	0 %	0 %	0 %	0 %
1	0 %	100 %	0 %	0 %	0 %
2	0 %	0 %	100 %	100 %	100 %

Table 6.5: Scenario 2

	Traffic type				
Class	FTP	WWW	Video	VoIP	Control
0	100 %	0 %	0 %	0 %	0 %
1	0 %	100 %	0 %	0 %	0 %
2	0 %	0 %	100 %	0 %	0 %
3	0 %	0 %	0 %	100 %	100 %

Table 6.6: Scenario 3

	Traffic type				
Class	FTP	WWW	Video	VoIP	Control
0	100 %	0 %	0 %	0 %	0 %
1	0 %	100 %	0 %	0 %	100 %
2	0 %	0 %	100 %	100 %	0 %

Table 6.7: Scenario 4

	Traffic type				
Class	FTP	WWW	Video	VoIP	Control
0	70 %	0 %	0 %	0 %	0 %
1	30 %	100 %	0 %	0 %	0 %
2	0 %	0 %	100 %	0 %	0 %
3	0 %	0 %	0 %	100 %	100 %

Table 6.8: Scenario 5

	Traffic type				
Class	FTP	WWW	Video	VoIP	Control
0	100 %	0 %	0 %	0 %	0 %
1	0 %	70 %	0 %	0 %	0 %
2	0 %	30 %	100 %	0 %	0 %
3	0 %	0 %	0 %	100 %	100 %

Table 6.9: Scenario 6

	Traffic type				
Class	FTP	WWW	Video	VoIP	Control
0	70 %	0 %	0 %	0 %	0 %
1	30 %	70 %	0 %	0 %	0 %
2	0 %	30 %	100 %	0 %	0 %
3	0 %	0 %	0 %	100 %	100 %

Table 6.10: Scenario 7

	Traffic type				
Class	FTP	WWW	Video	VoIP	Control
0	70 %	0 %	0 %	0 %	0 %
1	30 %	70 %	0 %	0 %	0 %
2	0 %	20 %	50 %	0 %	0 %
3	0 %	10 %	50 %	100 %	100 %

Table 6.11: Scenario 8

6.6.3 Provisioning parameters

For each scheduler the buffers are dimensioned so that the total buffer size is 230 packets. 200 packets of the total buffer capacity is allocated for elastic traffic and 30 for real time traffic. If traffic is separated into more than two classes, then for instance the total buffer size for real time traffic is divided evenly among the real time classes.

Scenario	**Weights**
1	(0.6, 0.4)
2	(0.07, 0.53 ,0.4)
3	(0.07, 0.53, 0.18, 0.22)
4	(0.07, 0.53 ,0.4)
5	(0.07, 0.53, 0.18, 0.22)
6	(0.07, 0.53, 0.18, 0.22)
7	(0.07, 0.53, 0.18, 0.22)
8	(0.07, 0.53, 0.18, 0.22)

Table 6.12: Parameters for DRR

In DRR the provisioning principle is that the classes meant for real time traffic are first allocated a weight that is two times their expected load share. This is because in a rate based scheduler small delays can be achieved only by allocating a relatively large weight. The excess weight is then divided between classes meant for elastic traffic in proportion to their expected load shares. This provisioning rule is applied also in scenarios 5-8 although the expected load shares are not valid any more due to the fact that different types of traffic can be mixed in the same class. The reason for

using this kind of static allocation is that also in reality the operator does not know what the momentary load shares of different traffic classes will be: if the selected QoS mechanisms are not self adjusting, the resources have to be provisioned based on the expected load shares.

For adaptive DRR and HPD with delay bound the target ratio for delays between consecutive classes is set to 4 and the delay bound for the highest class is set to 5 ms. Safety margin is set to be 1/10 of the delay bound. In HPD, the parameter g is set to be 0.875. It should be noted that the provisioning is now the same for each scenario (except for the buffer sizes), since the schedulers are measurement based and should be able to adjust to changing load conditions.

In WEDD the deadlines for the classes are set according to the requirements of the traffic type for which the class is primarily meant. These deadlines are set as shown in Table 6.13. The safety margin is set to be 1/10 of the deadline. The weights

Application	**Deadline**
FTP	200 ms
WWW	50 ms
Video	15 ms
VoIP	5 ms
Control	5 ms

Table 6.13: Deadlines in WEDD

are set according to the principle that the deadline violation probability for WWW traffic can be 10 times as large as the violation probability for real time traffic while the violation probability for FTP traffic can be 50 times the violation probability of real time traffic.

6.6.4 Topology and general parameters

The simulated network is depicted in Figure 6.4. It consists of 10 access links and one bottleneck link. It is assumed that a session from a certain traffic generator is destined with probability 0.1 to a certain access node, i.e. the load distribution between different access links should be approximately even. The reason for using such a small network for simulations is to first see how the selected scheduling algorithms perform in a single bottle neck link. Later the performance of the most promising algorithms can be tested also in larger networks. The following general parameters were used in the simulations:

1. Simulation

    - **Simulation time of one replication:** 3600 s (including warm-up period and freezing time).

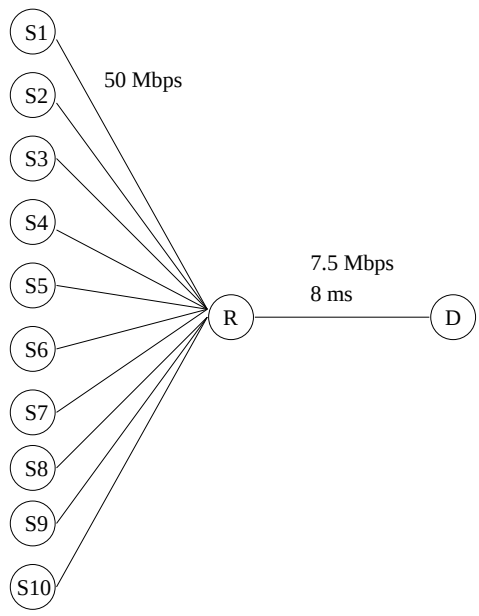

Figure 6.4: Topology. S = source node, R = router, D = destination

- **Warm-up period:** 300 s. Statistics are collected only for flows that have started after the warm-up period.
- **Freezing time:** 3300 s. Statistics are collected only for flows that have started before the freezing time.
- **Number of replications:** 5.

2. Links

   - **Access link capacity:** 50 Mbps.
   - **Bottleneck link capacity:** 7.5 Mbps.
   - **Bottleneck link delay:** 8 ms.

3. TCP

   - **Maximum value for cwnd:** 65 000 bytes
   - **Maximum segment size:** 1500 bytes

4. Buffers

   - **Total buffer size:** 230 packets.
   - **Queue management:** TailDrop

5. Load

   - **Mean offered load (theoretical):** 0.95.

6.6.5 Collected performance statistics

Performance statistics are collected from the simulations both at aggregate and session level. Since all the evaluated scheduling disciplines are class based, aggregate level statistics indicate how fairly the scheduling discipline is able to share resources among different classes. In general, aggregate level statistics may also be used by the operators for provisioning purposes. End users, on the other hand, are only interested on the session/flow level performance. Thus results such as delay, jitter, throughput and packet loss probability are collected at session level in order to see whether these class based scheduling disciplines are able to support the selected differentiation model also for individual sessions.

The performance metrics of interest are measured from the simulations in the following way:

Aggregate level performance metrics

The following performance metrics are measured for each traffic class:

- **Bandwidth:** Bandwidth allocated for different classes is recorded at 2 seconds intervals.
- **Load:** Load in different classes is recorded at 2 seconds intervals.
- **Queue sizes:** Queue sizes of different classes are recorded at 2 seconds intervals.
- **Queueing delay:** Average queueing delay is measured for each class in the core router.
- **Loss:** Average loss is measured for each class in the core router.

Session level performance metrics

The following performance metrics are measured for each session:

- **Throughput:** Throughput at network level is measured in the sink. Only packets that have been received once are included in the measurement.
- **Queueing delay:** Queueing delay is measured in the core router.
- **Jitter:** Difference between the queuing delay of consecutive packets is measured in the core router
- **Loss:** Packet loss in the core router.

Chapter 7

Simulation results

This chapter shows the simulation results in various scenarios. The result tables depict the performance of individual sessions while the figures show the performance at aggregate level. It should be noted that the throughput of WWW sessions is surprisingly large. This is due to the fact that the throughput was measured over the whole session (including page retrievals and reading times). Thus sessions with only one page request with few objects may attain considerably larger throughput than sessions with many page requests, since the effect of reading time is not included in the measurement for the short sessions. In order to be fully comparable, the throughputs should be measured for individual pages, not for individual sessions.

The target total load for all simulations is 0.95. The theoretical offered load is calculated for each traffic class and the measured, real load is also shown for the classes. The measured loads are in some scenarios clearly smaller than the theoretical loads. This is partly due to the backoffing of TCP connections, partly due to the fact that the mean load was measured only from the first iteration. Also the values given by the random number generators may not always converge exactly to the desired mean, especially if the number of sessions in some class is very small.

7.1 Best effort scenario

The best effort scenario (scenario 0) provides a baseline for which the selected differentiation models and scheduling disciplines can be compared. In Tables 7.2 and 7.3 the results for this scenario are shown. It can be observed that the queuing delays for real time traffic are intolerable (40 ms - 50 ms). Also the packet losses are quite high, at least for FTP. The throughput of WWW and FTP seems to be relatively large. The effect of increasing the capacity of the bottleneck link was also tested. It seems that when the capacity is doubled, the queuing delays of real time traffic become almost acceptable and also the packet losses of all traffic types become small.

Class	Offered load (theoretical)	Measured load (1st iter.)
0	0.95	0.82

Table 7.1: Mean load in scenario 0

		Queueing delay		Jitter	
Traffic	**Class**	**Mean**	**Stdev**	**Mean**	**Stdev**
FTP	0	134 ms	19 ms	3 ms	2 ms
WWW	0	39 ms	39 ms	11 ms	14 ms
Video	0	46 ms	19 ms	5 ms	0 ms
VoIP	0	35 ms	16 ms	7 ms	1 ms
Control	0	33 ms	0 ms	4 ms	0 ms

Table 7.2: Delay statistics in scenario 0

		Throughput		Loss	
Traffic	**Class**	**Mean**	**Stdev**	**Mean**	**Stdev**
FTP	0	1847690 bps	776860 bps	3.6 %	4.8 %
WWW	0	149720 bps	354360 bps	0.8 %	3.6 %
Video	0	496780 bps	5600 bps	0.6 %	0.7 %
VoIP	0	30080 bps	6240 bps	0.4 %	0.5 %
Control	0	70790 bps	0 bps	0.9 %	0 %

Table 7.3: Throughput and loss statistics in scenario 0

When the capacity is further doubled, the quality is sufficient for all traffic types.

7.2 General observations

For all scheduling algorithms the same general observations can be made in scenario 1 where classification is done between real time and non-real time traffic. When compared with the best effort scenario, the delays and jitters of real time traffic have decreased considerably and the packet losses are smaller especially for FTP. However, the throughputs for FTP and WWW have decreased since more resources are now allocated to real time traffic. The mean link utilization is also somewhat higher in scenario 1 compared with the best effort scenario. This is due to the fact that in the best effort scenario the TCP connections back off more frequently. The results for scenario 1 are shown for the DRR algorithm in Tables 7.5 and 7.6.

Class	**Offered load (theoretical)**	**Measured load (1st iter.)**
0	0.76	0.67
1	0.19	0.23
total	0.95	0.9

Table 7.4: Mean load for DRR in scenario 1

		Queueing delay		**Jitter**	
Traffic	**Class**	**Mean**	**Stdev**	**Mean**	**Stdev**
FTP	0	226 ms	54 ms	4 ms	4 ms
WWW	0	91 ms	86 ms	21 ms	28 ms
Video	1	3 ms	2 ms	1 ms	0 ms
VoIP	1	2 ms	1 ms	1 ms	0 ms
Control	1	2 ms	0 ms	1 ms	0 ms

Table 7.5: Delay statistics for DRR in scenario 1

		Throughput		**Loss**	
Traffic	**Class**	**Mean**	**Stdev**	**Mean**	**Stdev**
FTP	0	1459620 bps	712590 bps	2.3 %	5.0 %
WWW	0	119670 bps	307040 bps	1.0 %	4 %
Video	1	499620 bps	800 bps	0 %	0.2 %
VoIP	1	30210 bps	6370 bps	0 %	0.3 %
Control	1	71150 bps	0 bps	0.2 %	0 %

Table 7.6: Throughput and loss statistics for DRR in scenario 1

Also in scenario4 where control traffic is mixed with WWW traffic a general obser-

vation can be made for each algorithm: the delays and jitters experienced by control traffic are clearly too high (queuing delay more than 30 ms). This was expected since WWW and control traffic have totally different delay requirements.

7.3 Differentiation with DRR

With the selected provisioning parameters for DRR the mean delays and jitters of real time traffic are very small in each scenario. This is due to the fact that real time classes are allocated with relatively large weights in order to guarantee small delays in the presence of momentary load fluctuations. However, this kind of provisioning leaves little capacity for FTP traffic: queuing delays are high in each scenario where FTP traffic is separated into its own class. Furthermore, the throughput is small.

In scenario 2 where short and long TCP flows are separated the packet losses of both WWW and FTP are decreased. For FTP this decrease is about 50%. Also more bandwidth is allocated for WWW traffic and the delays and jitters are approximately halved compared with scenario1, which is a clear improvement. On the other hand, the throughput received by FTP is now smaller. The results for scenario 2 are shown in Tables 7.8 and 7.9.

Class	**Offered load (theoretical)**	**Measured load (1st iter.)**
0	0.09	0.07
1	0.67	0.63
2	0.19	0.23
total	0.95	0.93

Table 7.7: Mean load for DRR in scenario 2

		Queueing delay		**Jitter**	
Traffic	**Class**	**Mean**	**Stdev**	**Mean**	**Stdev**
FTP	0	328 ms	245 ms	13 ms	9 ms
WWW	1	44 ms	38 ms	13 ms	14 ms
Video	2	2 ms	2 ms	1 ms	0 ms
VoIP	2	2 ms	1 ms	1 ms	0 ms
Control	2	2 ms	0 ms	1 ms	0 ms

Table 7.8: Delay statistics for DRR in scenario 2

Figure 7.1 and Figure 7.2 depict the queue lengths and bandwidths of different classes as a function of time in scenario 2. It seems that the bandwidth allocation follows quite well the development of the queue lengths.

		Throughput		Loss	
Traffic	**Class**	**Mean**	**Stdev**	**Mean**	**Stdev**
FTP	0	866390 bps	462240 bps	1.1 %	4.1 %
WWW	1	131460 bps	313550 bps	0.9%	3.9%
Video	2	499750 bps	780 bps	0 %	0.1 %
VoIP	2	30210 bps	6230 bps	0 %	0.2 %
Control	2	71200 bps	0 bps	0.1 %	0 %

Table 7.9: Throughput and loss statistics for DRR in scenario 2

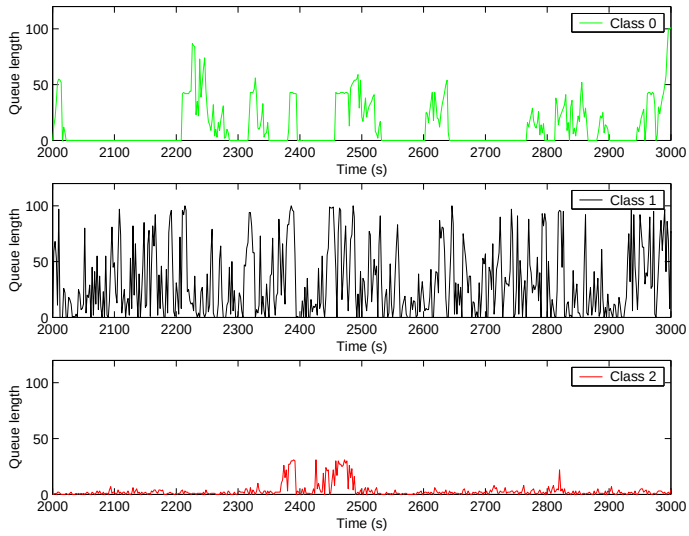

Figure 7.1: Queue lengths for DRR in scenario 2

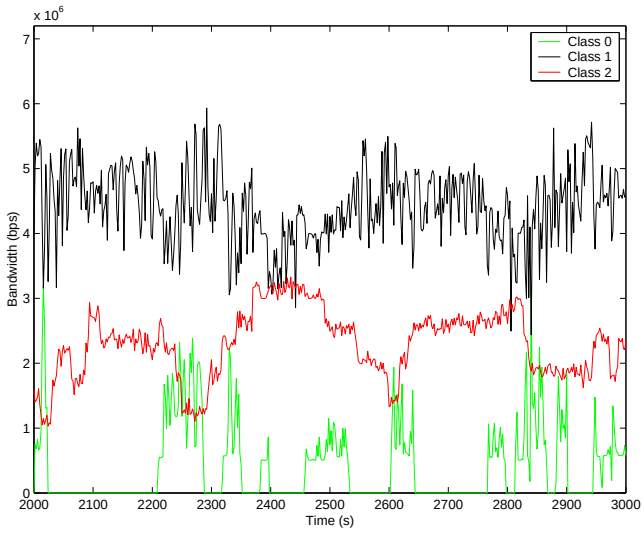

Figure 7.2: Bandwidths for DRR in scenario 2

In scenario 3 there seems to be no benefit when video traffic is further separated since the packet loss for video increases from 0% to 1.4% and the bandwidth is also smaller. However, the problems related to packet loss could probably be solved at least to some extent by using active queue management algorithms instead of simple tail drop.

In scenario 5 where 30% of FTP traffic is mixed with WWW traffic, the packet loss of the mixed FTP increases significantly to 3.6%. However, the throughput of the mixed FTP is still clearly larger than the throughput of the FTP in the lowest class, indicating that differentiation is possible. On the contrary, in scenario 6 where 30% of WWW traffic is mixed with video traffic it is surprising that there is hardly any difference between the throughputs and queueing delays of WWW sessions that belong to different classes. The only notable difference is that the packet loss for the WWW sessions in the higher class is much worse (over 7%) than the packet loss in the lower class. In scenario 8 where WWW is allowed to mix even with VoIP and control traffic, the packet loss in the highest class becomes intolerable since the allocated buffer size of 15 packets can not accommodate all load fluctuations.

7.4 Differentiation with ADRR and delay bound

With adaptive DRR integrated with a delay bound the mean delays and jitters of real time traffic are larger in each scenario compared with DRR, but the mean delays are still within the limits of the deadline (5 ms) that is set for realtime traffic. Packet losses are in general larger when the weights are adapted. The reason for this is that the algorithm updates the weights at 1 s intervals. Thus it is not possible to follow every change in the load distribution and sometimes the adaptation of weights based on history may even be harmful if load conditions change rapidly. Also the feedback mechanism of TCP causes some problems when the weights are adapted: if the weight for elastic traffic is increased in a congestion situation, the TCP connections will send even more traffic into the network which further increases the congestion level.

In scenario 3 video traffic experiences too large packet losses (6.6%) and queuing delays (30 ms). This is because only the highest class is assigned with a deadline and the resources for other classes are divided in a proportional manner. In scenarios 5-8 better differentiation is achieved compared with static DRR: for example in scenario 6 and 7 there are differences between the throughputs of WWW traffic mapped to lower and higher class, which was not the case when static weight setting was used. However, as with DRR, the packet losses seem to be a huge problem when different traffic types are mixed. The results for scenario 7 are shown in Tables 7.11 and 7.12.

Figure 7.3 and Figure 7.4 depict the queue lengths and bandwidths of different classes as a function of time in scenario 7.

Class	Offered load (theoretical)	Measured load (1st iter.)
0	0.06	0.05
1	0.50	0.46
2	0.29	0.20
3	0.10	0.11
total	0.95	0.82

Table 7.10: Mean load for adaptive DRR in scenario 7

		Queueing delay		Jitter	
Traffic	**Class**	**Mean**	**Stdev**	**Mean**	**Stdev**
FTP	0	284 ms	122 ms	6 ms	3 ms
FTP	1	167 ms	48 ms	5 ms	3 ms
WWW	1	50 ms	55 ms	14 ms	19 ms
WWW	2	22 ms	18 ms	9 ms	8 ms
Video	2	17 ms	7 ms	4 ms	1 ms
VoIP	3	4 ms	1 ms	1 ms	0 ms
Control	3	4 ms	0 ms	1 ms	0 ms

Table 7.11: Delay statistics for adaptive DRR in scenario 7

		Throughput		Loss	
Traffic	**Class**	**Mean**	**Stdev**	**Mean**	**Stdev**
FTP	0	1626380 bps	583450 bps	1.0 %	3.7 %
FTP	1	1440270 bps	842350 bps	5.2 %	7.2 %
WWW	1	146700 bps	351600 bps	0.9%	3.9%
WWW	2	192130 bps	378980 bps	8.8%	13.3%
Video	2	479790 bps	16090 bps	4.0 %	3.2 %
VoIP	3	30160 bps	5550 bps	0 %	0 %
Control	3	71250 bps	0 bps	0 %	0 %

Table 7.12: Throughput and loss statistics for adaptive DRR in scenario 7

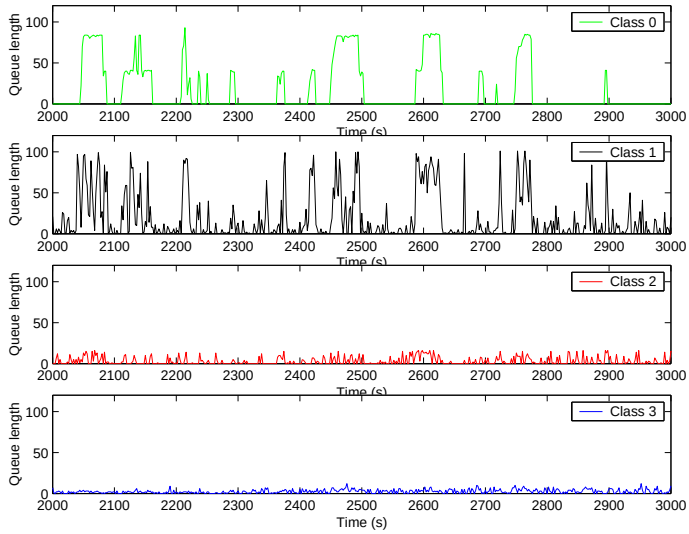

Figure 7.3: Queue lengths for adaptive DRR in scenario 7

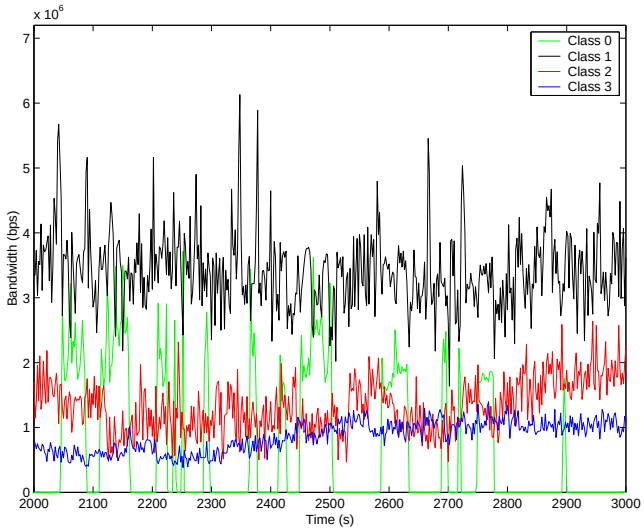

Figure 7.4: Bandwidths for adaptive DRR in scenario 7

However, although better differentiation can be achieved with adaptive weight setting, the ratios of delays between consecutive classes do not correspond to the target ratios. The explanation for this is that the load levels in some classes may vary a lot. For instance in a class meant for FTP traffic the queue may be empty for tens of seconds and then suddenly it starts to grow rapidly. If the traffic in every class was relatively smooth, the target ratios of the adaptive DRR algorithm could easily be achieved. However, if the variation in load levels is high, not only the filtered queue length but also the arrival rates in the classes should be taken into account when adapting the weights. It may still be that even though the ratios are not achieved in this particular simulation setting, in larger networks the algorithm would perform better because the traffic would be smoother inside the network than in the edges.

7.5 Differentiation with HPD and delay bound

With HPD algorithm integrated with a delay bound the mean delays for the highest class are within the delay bound and mean jitters range from 1 ms to 2 ms. Contrary to adaptive DRR, also the target delay ratios, which were set to 4, are met quite well both at class and session level. However, the ratios can not be achieved exactly since the absolute constraint for the delay of the highest class requires that in some occasions the proportionality constraints have to be relaxed.

In scenario 2 where WWW and FTP traffic are separated the packet loss for FTP is halved as in DRR but the packet loss for WWW is roughly doubled. The explanation for the increased packet loss is most likely the same as with adaptive DRR. The queueing delays for FTP are in turn clearly smaller with HPD than with DRR and also the throughput is larger especially at flow level.

In all scenarios consistent differentiation is achieved in terms of delays and throughputs: for example the delays of the WWW sessions mapped to the lower class are approximately four times the delays of the WWW flows mapped to the higher class, as depicted in Table 7.14.

Class	**Offered load (theoretical)**	**Measured load (1st iter.)**
0	0.09	0.09
1	0.47	0.46
2	0.29	0.24
3	0.10	0.11
total	0.95	0.90

Table 7.13: Mean load for HPD with delay bound in scenario 6

In Tables 7.17 and 7.18 the results are shown also in scenario 8 where several traffic

		Queueing delay		Jitter	
Traffic	**Class**	**Mean**	**Stdev**	**Mean**	**Stdev**
FTP	0	291 ms	160 ms	9 ms	4 ms
WWW	1	62 ms	49 ms	15 ms	17 ms
WWW	2	17 ms	13 ms	4 ms	4 ms
Video	2	17 ms	8 ms	2 ms	1 ms
VoIP	3	4 ms	1 ms	1 ms	0 ms
Control	3	5 ms	0 ms	1 ms	0 ms

Table 7.14: Delay statistics for HPD with delay bound in scenario 6

		Throughput		Loss	
Traffic	**Class**	**Mean**	**Stdev**	**Mean**	**Stdev**
FTP	0	1292860 bps	575100 bps	1.0 %	3.6 %
WWW	1	118720 bps	300210 bps	0.6%	3.4%
WWW	2	188260 bps	391110 bps	4.8%	9.7%
Video	2	490970 bps	8980 bps	1.8 %	1.7 %
VoIP	3	30360 bps	5960 bps	0 %	0 %
Control	3	71250 bps	0 bps	0 %	0 %

Table 7.15: Throughput and loss statistics for HPD with delay bound in scenario 6

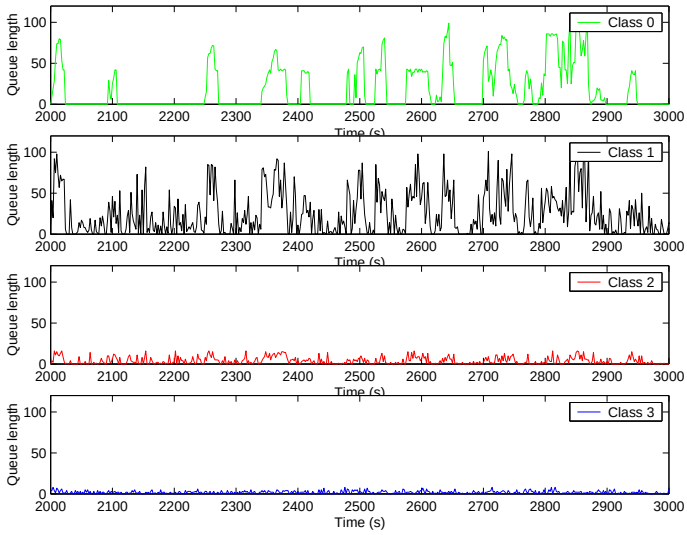

Figure 7.5: Queue lengths for HPD with delay bound in scenario 6

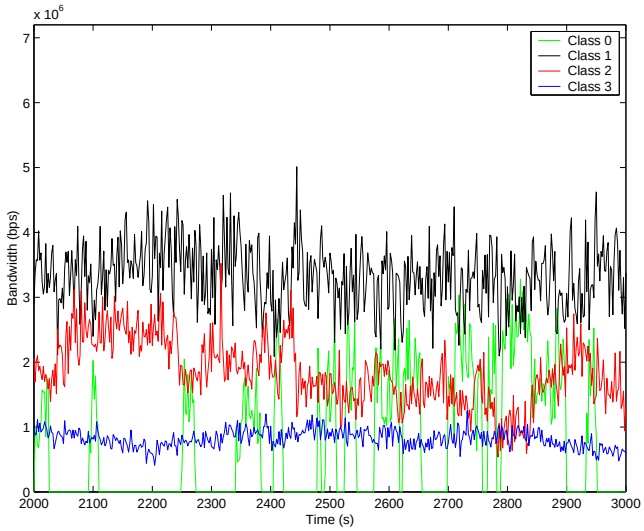

Figure 7.6: Bandwidths for HPD with delay bound in scenario 6

types are mixed. The target delay ratios are met quite well at class level and to some extent at session level but at least for FTP it seems difficult to achieve the desired delay ratio at session level.

Class	**Offered load (theoretical)**	**Measured load (1st iter.)**
0	0.06	0.05
1	0.50	0.46
2	0.18	0.14
3	0.21	0.22
total	0.95	0.87

Table 7.16: Mean load for HPD with delay bound in scenario 8

		Queueing delay		Jitter	
Traffic	**Class**	**Mean**	**Stdev**	**Mean**	**Stdev**
FTP	0	276 ms	175 ms	10 ms	4 ms
FTP	1	156 ms	43 ms	5 ms	1 ms
WWW	1	63 ms	54 ms	16 ms	19 ms
WWW	2	17 ms	14 ms	5 ms	4 ms
WWW	3	6 ms	2 ms	2 ms	1 ms
Video	2	17 ms	9 ms	3 ms	0 ms
Video	3	5 ms	0 ms	1 ms	0 ms
VoIP	3	4 ms	1 ms	1 ms	0 ms
Control	3	5 ms	0 ms	1 ms	0 ms

Table 7.17: Delay statistics for HPD with delay bound in scenario 8

		Throughput		Loss	
Traffic	**Class**	**Mean**	**Stdev**	**Mean**	**Stdev**
FTP	0	1211400 bps	547050 bps	0.7 %	3.0 %
FTP	1	1557670 bps	676790 bps	4.0 %	3.9 %
WWW	1	120880 bps	303790 bps	0.8%	4.0%
WWW	2	182010 bps	393090 bps	2.9%	7.3%
WWW	3	220470 bps	451800 bps	1.0%	3.2%
Video	2	493700 bps	5190 bps	1.3 %	1.0 %
Video	3	499320 bps	350 bps	1.4 %	0 %
VoIP	3	30290 bps	6250 bps	0.1 %	0.1 %
Control	3	71030 bps	0 bps	0.5 %	0 %

Table 7.18: Throughput and loss statistics for HPD with delay bound in scenario 8

7.6 Differentiation with WEDD

The parameter selection for the WEDD algorithm seems to be much harder task than the parameter selection for the other algorithms. With the tested parameters the mean queueing delay for real time traffic is smaller than would be required while the queueing delay for video and FTP is too large. In addition, the target ratios for deadline violation probabilities are not achieved in any scenario. It should also be noticed that although the mean delays for real time traffic are very small, the deadline violation probabilities are intolerable especially in the highest class (almost 30% in the worst case).

The algorithm seems to work with this parameter setting in the sense that there are clear differences between the throughputs and delays of classes. However, the differentiation is not even close to the actual target. One major reason for this is that the parameters are not well selected. For instance, the target ratio of deadline

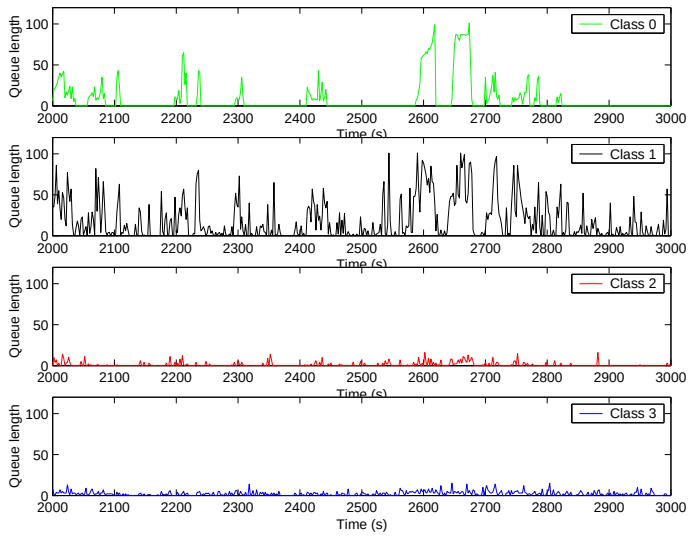

Figure 7.7: Queue lengths for HPD with delay bound in scenario 8

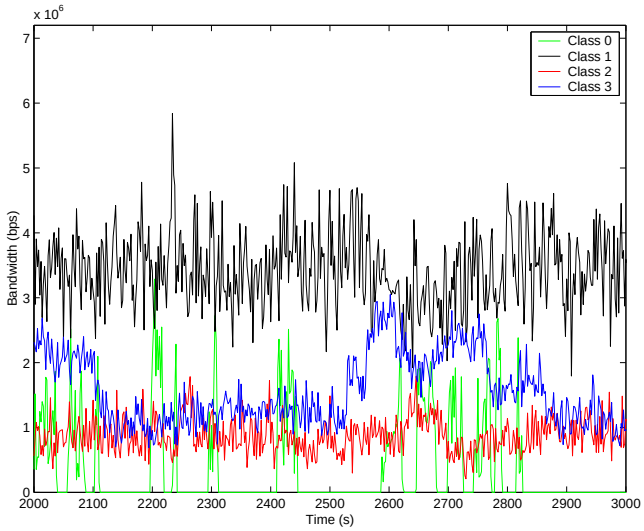

Figure 7.8: Bandwidths for HPD with delay bound in scenario 8

violation probabilities between VoIP and FTP was set to 50, but obviously it is not even theoretically possible to achieve this kind of differentiation with the used traffic mixes. If the deadlines and target ratios are relaxed, desired differentiation may be achieved. However, one can ask whether there is any meaning in this kind of differentiation where some proportionality constraints are met without any connection to real requirements. Tables 7.20 and 7.21 show the results in scenario 5 where there is FTP traffic in two different classes. It can be seen that the queueing delays for FTP belonging to the lowest class are enormous.

Class	**Offered load (theoretical)**	**Measured load (1st iter.)**
0	0.06	0.06
1	0.70	0.66
2	0.09	0.07
3	0.10	0.11
total	0.95	0.90

Table 7.19: Mean load for WEDD in scenario 5

		Queueing delay		**Jitter**	
Traffic	**Class**	**Mean**	**Stdev**	**Mean**	**Stdev**
FTP	0	614 ms	1.433 s	87 ms	290 ms
FTP	1	117 ms	25 ms	3 ms	2 ms
WWW	1	42 ms	38 ms	12 ms	14 ms
Video	2	17 ms	8 ms	4 ms	2 ms
VoIP	3	2 ms	1 ms	1 ms	0 ms
Control	3	3 ms	0 ms	0 ms	0 ms

Table 7.20: Delay statistics for WEDD in scenario 5

		Throughput		**Loss**	
Traffic	**Class**	**Mean**	**Stdev**	**Mean**	**Stdev**
FTP	0	926030 bps	488940 bps	1.8 %	4.8 %
FTP	1	1900330 bps	679260 bps	3.3 %	3.2 %
WWW	1	131530 bps	312290 bps	0.8%	3.6%
Video	2	468456 bps	22720 bps	6.3 %	4.5 %
VoIP	3	30220 bps	6160 bps	0 %	0 %
Control	3	71250 bps	0 bps	0 %	0 %

Table 7.21: Throughput and loss statistics for WEDD in scenario 5

Figure 7.9 and Figure 7.10 depict the queue lengths and bandwidths of different classes as a function of time in scenario 5. Although the queue lengths in class 0 grow large, the algorithm allocates only small amount of bandwidth for this class.

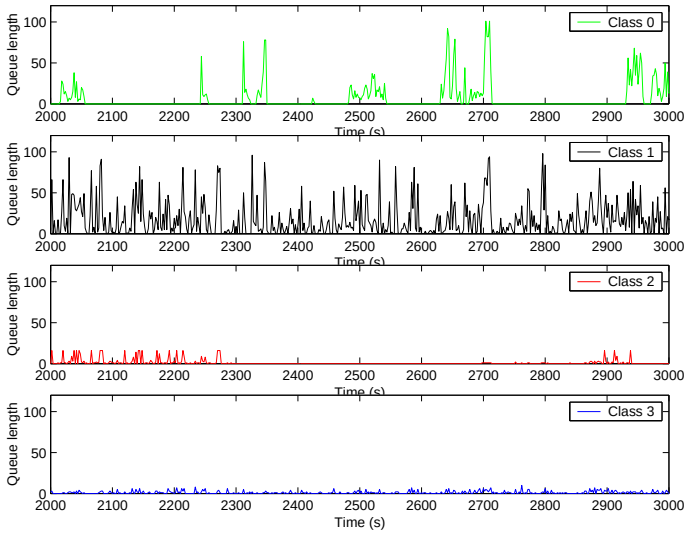

Figure 7.9: Queue lengths for WEDD in scenario 5

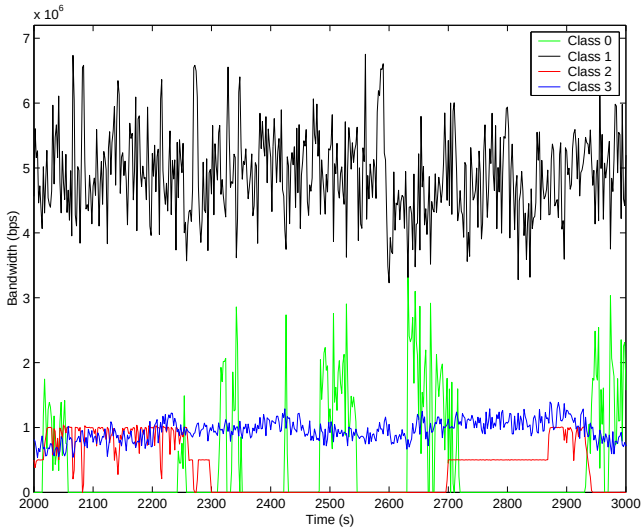

Figure 7.10: Bandwidths for WEDD in scenario 5

Chapter 8

Conclusions

8.1 Summary and discussion

In this thesis issues related to quality differentiation and scheduling in Differentiated Services were studied. First, possible service approaches and differentiation models for an operator were analyzed at qualitative level. In addition, alternative scheduling disciplines for implementing these models were proposed based on a literature survey, with the focus on capacity and delay differentiation. Finally, the proposed scheduling disciplines were evaluated with simulations. The objective of the simulations was to see how well the scheduling disciplines support the differentiation model that they are designed for in various scenarios, both at aggregate and session/flow level. Another important objective was to study how traffic should be mapped to different service classes.

The following differentiation models were evaluated in the simulations: absolute capacity differentiation, proportional delay differentiation with absolute delay bound and deadline violation probability differentiation. For the implementation of absolute capacity differentiation, a rate based DRR scheduler was selected. For proportional delay differentiation with delay bound two different schedulers were chosen: adaptive DRR and HPD with delay bound. The scheduler used for deadline violation probability differentiation was WEDD. Furthermore, a best effort scenario with simple FCFS scheduler was simulated to have a baseline for which the differentiation models could be compared. The traffic mix used in the simulations consisted of FTP, WWW, VoIP, video and control traffic.

Based on the results it can be concluded that with best effort technology significant over provisioning is required in order to satisfy the requirements of both elastic and real time traffic. Since this kind of over provisioning is not always possible, differentiation is certainly needed. However, the crucial question is how this differentiation should be performed. The answer depends largely on the service approach adopted

by the operator.

If the operator wishes to serve applications as well as possible, the primary criteria for the selection of differentiation models and schedulers is their ability to support the quality requirements of the applications. Based on the simulation results, an important conclusion can be made: from the applications point of view the service is often intolerable if different traffic types are allowed to be mixed freely in the same class. Especially the packet losses are so high in some scenarios that there is hardly any mechanism that could solve this problem. This phenomenon is independent of the differentiation model or scheduler that is used. Thus, if the goal is to satisfy the requirements of the applications it is beneficial that traffic is separated in some meaningful way before the resources are allocated, for example based on the transport protocol or characteristics of the traffic.

The simulation results indicate that when separation is performed between UDP and TCP traffic, the delay requirements of real time traffic are met with each scheduler. Furthermore, the overall service is more predictable compared with the best effort case, indicating that clear improvements can be achieved even with the simplest form of differentiation. However, elastic traffic still experiences some problems: the packet loss for long TCP flows (FTP) and the delays for short TCP flows (WWW) are quite high. This is due to the commonly known battle between the mice and the elephants: the mice (short TCP flows) and the elephants (long TCP flows) operate at different phases of TCP's congestion control and may thus interfere with each other.

When differentiation is added between short TCP flows (WWW) and long TCP flows (FTP), WWW traffic experiences smaller delays and also the packet loss for FTP traffic is smaller with each scheduler. However, some important differences can be observed in the degree of separation of WWW and FTP traffic between different schedulers. For example, in the static DRR scheduler the delays of WWW traffic are very small but only little capacity is left for FTP traffic causing excessive delays. On the contrary, in the adaptive DRR and HPD schedulers with delay bound the resources are allocated more evenly so that WWW experiences smaller delays but FTP traffic is not starved either. The packet loss for WWW is increased in these adaptive schedulers but this problem could probably be solved at least to some extent with active queue management.

It seems that from applications point of view the best solution would be to have three service classes: one for real time traffic, one for short TCP flows and one for long TCP flows. One could argue that it is not necessary to have two separate classes for elastic traffic. However, it should be kept in mind that for example file transfer and Web surfing are very different in nature: web surfing is interactive and the user expects to receive constant feedback on the proceeding of a document retrieval. Thus, for WWW delay is considerably more relevant than for file transfer. For resource allocation between the classes an adaptive scheduler with delay bound could be used. The delay bound ensures that delay requirements of real time traffic are met. The

adaptive allocation of resources should be used instead of static allocation since exact prediction of the loads of different classes is impossible. In this thesis the adaptive schedulers modified the resource allocation so that certain delay constraints were met, but some other adaptation criteria could be used as well.

It is often criticized that the operators should not choose QoS functionality based on the requirements of applications but rather based on the prices paid by the customers. If the operator chooses to favor the customers that are paying the highest price, an important criteria for the selection of differentiation models and schedulers is the level of differentiation that they are able to provide.

According to the results the differentiation with the static DRR scheduler was problematic. For example, hardly any separation could be achieved in the delays and throughputs of WWW sessions belonging to different classes. In adaptive DRR algorithm with delay bound better differentiation was received but the target delay ratios between classes could not be achieved. On the contrary, in the HPD scheduler with delay bound the target delay ratios could be met reasonably well both at class and session level. The differentiation with the WEDD scheduler was a difficult task. This scheduler uses many parameters that have to be set correctly in order to receive the desired result. Originally the algorithm was suggested to be used only for real time traffic [Bod00]. In this case it could probably be possible to set the deadlines, weights etc. so that the target of differentiation is met. However, it can be asked whether there is a need to have a separate scheduler for real time traffic since a reasonable number of real time classes is probably one or two.

It seems that tunable differentiation can be achieved with adaptive schedulers. However, if the differentiation is steered by money, this means that traffic mixes within service classes are uncontrollable. As was stated earlier, this may result in so poor service that individual applications do not function. Thus one can ask whether there is any sense in developing various, possibly complex differentiation models that meet some proportionality constraints but are not able to support the applications. Are the customers really satisfied when they pay double price for a service that may be less than best effort? Another important question is that of predictability. Many of the proportional models promise to provide predictable and controllable differentiation. This is true in the sense that the higher classes receive better (or no worse) service than the lower classes regardless of load distribution. However, the meaning of predictability for a customer is something totally different. Predictability essentially means that the level of service is constant over time, i.e. the customer knows what to expect. However, this is certainly not the case with the proportional models. In fact, the simulation results indicate that the lack of predictability is a problem for all simulated differentiation models.

Even though systems where money steers the differentiation and traffic mixes within service classes can not be controlled are undesirable, this does not mean that the business aspect should be totally forgotten. For example, one option would be to perform differentiation hierarchically so that real time traffic, short TCP flows

and long TCP flows would first be separated into different classes. For real time class, a deadline could be set and further, packets exceeding this deadline could be dropped. Resources within the other classes could be divided based on money by using some adaptive scheduler with several priority levels. However, it should be somehow ensured that the acknowledgements sent by the receiving end of the TCP connection would be mapped to the same service class as the original TCP packets in order to achieve consistent differentiation.

8.2 Further work

Qualitative analysis and simulations conducted in this thesis provided insight into the problems related to quality differentiation, scheduling and traffic mapping. It seems that even though QoS provisioning in the Internet has been a popular research topic for a long time many important issues still remain to be solved.

In this thesis the differentiation models and mechanisms were primarily compared based on the service that they are able to provide. However, as was emphasized earlier, from the operator's point of view the most relevant question is which models and mechanisms would be able to maximize the revenue. Unfortunately, forwarding plane simulations are unable to provide an answer to this question. One direction for future work would be to combine forwarding plane simulations with higher level economic analysis in order to find policies that result in revenue maximization.

Furthermore, in the future the simulations could be performed with more accurate models for TCP protocol and traffic sources and in more realistic network topologies in order to gain insight into the end-to-end performance issues. Also the effect of active queue management mechanisms on the results should be studied, since packet loss seemed to be the worst problem in the simulations.

Many of the schedulers studied in this thesis were measurement based. Thus, one possible direction for future work would be to examine different measurement techniques and algorithms that can be utilized by schedulers. However, measurement based schemes are not restricted only to scheduling; they could be applied in any form of resource allocation or network management.

Bibliography

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

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

[Bod00] S. Bodamer. A New Scheduling Mechanism to Provide Relative Differentiation for Real-Time IP Traffic. In *Proceedings of the IEEE Global Telecommunications Conference*, pages 646–650, November 2000.

[BZ96] J.C.R. Bennett and H. Zhang. WF^2Q: Worst-case Fair Weighted Fair Queueing. In *Proceedings of IEEE Infocom*, pages 120–127, March 1996.

[CNC98] CNCL. CNCL Reference Manual, 1998.

[DSR99] C. Dovrolis, D. Stiliadis, and P. Ramanathan. Proportional Differentiated Services: Delay Differentiation and Packet Scheduling. In *Proceedings of ACM SIGCOMM*, pages 109–119, August 1999.

[DSR02] C. Dovrolis, D. Stiliadis, and P. Ramanathan. Proportional Differentiated Services: Delay Differentiation and Packet Scheduling. *IEEE/ACM Transactions on Networking*, 10(2):12–26, February 2002.

[EBA01] L. Essafi, G. Bolch, and A. Andres. An Adaptive Waiting Time Priority Scheduler for the Proportional Differentiation Model. In *Proceedings of the High Performance Computing Symposium*, April 2001.

[FBTZ01] V. Firoiu, J-Y. Le Boudec, D. Towsley, and Z-L. Zhang. Advances in Internet Quality of Service. Technical report, National Science Foundation, 2001.

[FJ93] S. Floyd and V. Jacobson. Random Early Detection Gateways for Congestion Avoidance. *IEEE/ACM Transactions on Networking*, 3:397–413, August 1993.

[FV90] D. Ferrari and D. Verma. A scheme for real-time channel establishment in wide-area networks. *IEEE Journal of Selected Areas in Communications*, 8:368–379, April 1990.

[Gol94] S.J. Golestani. A Self-Clocked Fair Queueing Scheme for High Speed Applications. In *Proceedings of IEEE Infocom*, 1994.

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

[IL01] M. Ilvesmäki and M. Luoma. On the capabilities of application level traffic measurements to differentiate and classify Internet traffic. In *Proceedings of SPIE*, volume 4523, pages 233–242, August 2001.

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

[Kil99] K. Kilkki. *Differentiated Services for the Internet*. Macmillan Technical Publishing, 1999.

[Kil02] K. Kilkki. Selection of QoS Mechanisms Based on Operator's Business Objective. In *Proceedings of Sixteenth Nordic Teletraffic Seminar*, pages 313–324, August 2002.

[LIP99] M. Luoma, M. Ilvesmäki, and M. Peuhkuri. Source characteristics for traffic classification in Differentiated Services type of networks. In *Proceedings of SPIE*, September 1999.

[LL73] C. Liu and J. Layland. Scheduling algorithms for multiprogramming in a hard real-time environment. *Journal of ACM*, 20:46–61, January 1973.

[LLY01] M. Leung, J. Lui, and D. Yau. Adaptive Proportional Delay Differentiated Services: Characterization and Performance Evaluation. *IEEE/ACM Transactions on Networking*, 9(6):801–817, December 2001.

[Luo00] M. Luoma. Simulation Studies of Differentiated Services Networks. Licenciate Thesis, Networking Laboratory, Helsinki University of Technology, 2000.

[LY99] J. Liebeherr and E. Yilmaz. Workconserving vs. Non-workconserving Packet Scheduling: An Issue Revisited. In *Proceedings of IEEE/IFIP International Workshop on Quality of Service (IWQOS)*, volume 3, pages 1484–1492, May 1999.

[Mah97] B. Mah. An empirical model of http network traffic. In *Proceedings of IEEE Infocom*, volume 2, pages 592–600, April 1997.

[PG93] A.K. Parekh and R.G. Gallager. A Generalized Processor Sharing Approach to Flow Control in Integrated Services Networks: The Single-Node Case. *IEEE/ACM Transactions on Networking*, 1(3):344–357, June 1993.

[SCF02] P. Sousa, P. Carvalho, and V. Freitas. Tuning Delay Differentiation in IP Networks Using Priority Queueing Models. In *NETWORKING 2002*, pages 709–720, May 2002.

[Siv00] V. Sivaraman. *End-to-End Delay Service in High Speed Packet Networks using Earliest Deadline First Scheduling*. PhD thesis, University of California, Los Angeles, 2000.

[SKLK01] J. Shin, J. Kim, D. Lee, and C. Jay Kuo. Adaptive Packet Forwarding for Relative Differentiated Services and Categorized Packet Video. In *Proceedings of the IEEE International Conference on Communications*, June 2001.

[SM02] A. Striegel and G. Manimaran. Packet Scheduling with Delay and Loss Differentiation. *Computer Communications*, 25(1), January 2002.

[Sti96] D. Stiliadis. *Traffic Scheduling in Packet-Switched Networks: Analysis, Design, and Implementation*. PhD thesis, University of California, Santa Cruz, 1996.

[SV95] M. Shreedhar and G. Varghese. Efficient Fair Queueing using Deficit Round Robin. In *Proceedings of ACM SIGCOMM*, 1995.

[SZ99] I. Stoica and H. Zhang. Providing Guaranteed Services Without Per Flow Management. In *Proceedings of ACM SIGCOMM*, September 1999.

[VZF91] D. Verma, H. Zhang, and D. Ferrari. Guaranteeing delay jitter bounds in packet switching networks. In *Proceedings of Tricomm*, pages 35–46, April 1991.

[Wan01] Z. Wang. *Internet QoS: Architecture and Mechanisms for Quality of Service*. Morgan Kaufmann Publishers, 2001.

[Wro97] J. Wroclavski. The Use of RSVP with IETF Integrated Services. IETF RFC 2210, September 1997.

[WSS01] H. Wang, C. Shen, and K.G. Shin. Adaptive-Weighted Packet Scheduling for Premium Service. In *Proceedings of the IEEE International Conference on Communications*, June 2001.

[Zha91] L. Zhang. VirtualClock: A New Traffic Control Algorithm for Packet-Switched Networks. *ACM Transactions on Computer Systems*, 9(2):101–124, May 1991.

[Zha95] H. Zhang. Service Disciplines for Guaranteed Performance Service in Packet-Switching Networks. *Proceedings of the IEEE*, 83(10):1374–1396, October 1995.