Scheduling in CDMA-based wireless packet networks

Stefan Martin Scriba

Submitted in fulfillment of the academic requirements
for the degree of MScEng
in the School of Electrical, Electronic and Computer Engineering
at the University of Natal, Durban, South Africa

July 28, 2003

To my wife Mia

This document was created in LaTeX

Abstract

Modern networks carry a wide range of different data types, each with its own individual requirements. The scheduler plays an important role in enabling a network to meet all these requirements. In wired networks a large amount of research has been performed on various schedulers, most of which belong to the family of *General Processor Sharing* (GPS) schedulers. In this dissertation we briefly discuss the work that has been done on a range of wired schedulers, which all attempt to differentiate between heterogeneous traffic.

In the world of wireless communications the scheduler plays a very important role, since it can take channel conditions into account to further improve the performance of the network. The main focus of this dissertation is to introduce schedulers, which attempt to meet the Quality of Service requirements of various data types in a wireless environment. Examples of schedulers that take channel conditions into account are the *Modified Largest Weighted Delay First* (M-LWDF), as well as a new scheduler introduced in this dissertation, known as the *Wireless Fair Largest Weighted Delay First* (WF-LWDF) algorithm. The two schemes are studied in detail and a comparison of their throughput, delay, power, and packet dropping performance is made through a range of simulations. The results are compared to the performance of four other schedulers. The fairness of M-LWDF and WF-LWDF is determined through simulations. The throughput results are used to establish Chernoff bounds of the fairness of these two algorithms. Finally, a summary is given of the published delay bounds of various schedulers, and the tightness of the resultant bounds is discussed.

Preface

The research work presented in this dissertation was performed by Stefan Martin Scriba, under the supervision of Prof. Fambirai Takawira, at the University of Natal's School of Electrical, Electronic and Computer Engineering, in the Research Centre for Radio Access Technologies. This work was partially sponsored by Telkom S.A. Ltd and Alcatel South Africa as part of the Centre of Excellence programme.

Parts of this dissertation were presented by the author at the SATNAC 2001 conference at the Wild Coast Sun, South Africa, and the SATNAC 2002 conference at the Champagne Sports Resort, South Africa.

The entire dissertation, unless otherwise indicated, is the student's own original work and has not been submitted in part, or in whole, to any other university for degree purposes.

Acknowledgements

I wish to thank my supervisor, Prof. Fambirai Takawira, for the the guidance and the many hours of discussion that have made this dissertation possible. He not only was a dedicated supervisor, but also showed interest and support in my personal life, thereby creating a relaxed and enjoyable work environment.

Many thanks to Telkom S.A. Ltd and Alcatel S.A., who financed this research project. To Baron Peterson and my mentor David Browne, thank you for your guidance and support.

To Bruce Harrison, our LAN manager, and the secretaries, first Ms Brigitte Le Breton, and later Mrs Sharon McGregor and Mrs Marie Wayne, thank you for putting up with the many urgent requests for help. I also wish to thank Prof. Peplow and Dr. Xu for sharing some of their insights.

To my postgraduate colleagues, thank you for the friendship, support, and entertainment that I experienced over the last two years. Neil Pate and Geoffrey Byers need to be highlighted, since they encouraged and helped me to use LaTeX, thereby preventing much frustration.

And last, but definitely not least, a great thank you to my parents, brothers, and close friends. Without their support and encouragement this dissertation would most likely never have been completed. My fiancé deserves a trophy for her constant support, especially when the problem solving process slowed down to a halt. Together we somehow managed to plan a wedding, while I simultaneously finished this dissertation.

Contents

List of Figures

List of Tables

List of Acronyms

3G	Third Generation wireless communication system
3GPP	Third Generation Partnership Project
4G	Fourth Generation wireless communication system
ABR	Available Bit Rate
AMPS	Advanced Mobile Phone Systems
ATM	Asynchronous Transfer Mode
B-EDF	Backward Earliest Due Date First
BER	Bit-Error-Rate
CBQ	Class Based Queueing
CBR	Constant Bit Rate
CC	Compensation Counter
CDMA	Code Division Multiple Access
CIF	Channel-condition Independent Fair
CIF-Q	Channel-Condition Independent Packet Fair Queueing
CSDPS	Channel State Dependent Packet Scheduling
CSDPS+CBQ	Channel State Dependent Packet Scheduling + Class Based Queueing
D-AMPS	Digital Advanced Mobile Phone Systems
DC	Deficit Counter
DRR	Deficit Round Robin
DSP	Digital Signal Processor
EDD	Earliest-Due-Date
EDF	Earliest Deadline First

ELF	Effort-Limited Fairness
ETF	Earliest Timestamp First
ETSI	European Communications Standard Institute
FDM	Frequency Division Multiple
FDMA	Frequency Division Multiple Access
FFQ	Frame-based Fair Queueing
FIFO	First-In-First-Out
FPGA	Field Programmable Gate Arrays
FPLMTS	Future Public Land Mobile Telecommunication Systems
FQ	Fair Queueing
GPS	General Processor Sharing
GR	Guaranteed Rate scheduling
GSM	Global Systems for Mobile communications
HIPERLAN	High Performance Radio LAN
HRR	Hierarchical Round Robin
I-CSDPS	Improved Channel State Dependent Packet Scheduling
IEEE	Institute of Electrical and Electronic Engineers
IMT-2000	International Mobile Telephony 2000
ISO	International Organization for Standardization
ITU	International Telecommunications Union
IWFQ	Idealized Wireless Fair Queueing
LAN	Local Area Network
LLC	Logical Link Control
LQF	Longest Queue First
LR	Latency Rate servers
LTFS	Long-Term Fairness Server
MAC	Medium Access Control
M-LWDF	Modified Largest Weighted Delay First
NMT	Nordic Mobile Telephone
OSI	Open Systems Interconnection model
PCS	Personal Communication Systems

PDA	Personal Digital Assistant
PDC	Japan's Personal Digital Cellular
PGPS	Packetized General Processor Sharing
PQ	Priority Queueing
PRADOS	Prioritized Regulated Allocation Delay Oriented Scheduling
PS-EDF	Permanent Sessions Earliest Deadline First
QoS	Quality of Service
QS	Quantum Size
RPQ$^+$	Rotating-Priority-Queues$^+$
RPPS	Rate Proportional Processor Sharing
RPS	Rate Proportional Servers
RR	Round Robin
RSVP	Resource ReSerVation Protocol
SATNAC	South African Telecommunications and Networking Applications Conference
SBFA	Server-Based Fair Approach
SCFQ	Self-Clocked Fair Queueing
SFQ	Start-Time Fair Queueing
SM-LWDF	Simplified Modified Largest Weighted Delay First
SPFQ	Starting Potential-based Fair Queueing
TACS	Total Access Communication Systems
TDM	Time Division Multiplexing
TDMA	Time Division Multiple Access
TS-EDF	Temporary Sessions Earliest Deadline First
UBR	Unspecified Bit Rate
UMTS	Universal Mobile Telecommunications System
UPT	Universal Personal Telecommunication
VBR	Variable Bit Rate
VC	VirtualClock
VCPB	VirtualClock with Priority Buffer
WARC	World Administrative Radio Conference
WCDMA	Wide-band CDMA

WFFQ	Wireless Fluid Fair Queueing
WFQ	Weighted Fair Queueing
WF^2Q	Worst-case Fair Weighted Fair Queueing
WF-LWDF	Wireless Fair Largest Weighted Delay First
W-LWDF	Wireless Largest Weighted Delay First
WRR	Weighted Round Robin

List of Symbols

α	characteristic value of Gamma distribution
β	characteristic value of Gamma distribution
δ_i	delay violation probability constant of user i
γ_X	"semi-invariant" generating function
ν	TS-EDF requires $\nu > 0$
ϕ	GPS weight (we assume RPPS model)
ρ	average arrival rate
ρ_{norm}	normalized arrival rate
$\Re$	set of possible transmission rates
σ	burstiness parameter
σ_{norm}	normalized burstiness bound
τ	propagation delay
B_c	chipping rate
C	capacity of system
C_i	capacity per channel
c_i	channel attenuation function for user i at transmission start
$c_i(t)$	channel attenuation function for user i
$\overline{c_i}(t)$	short-term average channel attenuation function for user i
$c_{\min}$	minimum channel attenuation
D	delay bound
D_i	steady-state delay of user i
$D_{\max}$	maximum queueing delay

d_p^e	delay deadline of packet p in server e
E_b/I_0	signal-to-interference ratio
$\mathrm{E}[Y_k]$	mean of Y_k
erfc(-)	complementary error function
F	frame size for Round Robin servers
F_i	fairness value of user i
F^S	fairness bound
G_p	processing gain (number of chips per bit)
K	no of nodes
Kb/s	kilo bits per second
L	average packet length
L_c	fixed cell length for Round Robin servers
$L_{\max}$	max packet length
$Load$	network load
$M_X(t)$	moment generating function
N_0	one-sided thermal noise power spectral density
N_T	total number of packets being transmitted
$P_{\text{available}}$	available transmission power not currently being used
P_b	bit-error-probability
P_i	transmission power of user i
$P[Y \geq y]$	Chernoff bound
R	server capacity or effective available bandwidth
R_i	transmission rate of user i
$R_{\max}$	maximum transmission rate
T	simulation measurement time
T_i	delay threshold of user i
t	current system time
V	no of sessions
$\mathrm{Var}[Y_k]$	variance of Y_k
$W_{\max}$	maximum number of bits that can be transmitted
Y_k	decision noise term

$y_{k1}(m)$	desired signal
$y_{k2}(m)$	multiple access interference
$y_{k3}(m)$	Gaussian noise term

Chapter 1

Introduction

1.1 Evolution of the wireless network

Modern wired networks have been interconnected to such a degree, that a web exists, which covers virtually the entire globe. Although mobile communication systems are currently restricted to small cells, so-called global mobile communications are envisaged, which are often referred to as Universal Personal Telecommunication (UPT) or Personal Communication Systems (PCS) and the Third Generation (3G) Wireless Systems or Future Public Land Mobile Telecommunication Systems (FPLMTS) [2].

1.1.1 First Generation Communication Systems

Cellular systems started in the analogue domain. Examples of such systems are Nordic Mobile Telephone (NMT), Advanced Mobile Phone Systems (AMPS) and Total Access Communication Systems (TACS) [2]. These are commonly referred to as the first generation systems [3], which enabled voice communications to go wireless.

1.1.2 Second Generation Communication Systems

Cellular technology advanced into the digital domain with systems such as Global Systems for Mobile communications (GSM), Digital Advanced Mobile Phone Systems (D-AMPS), Japan's Personal Digital Cellular (PDC) and a derivative of GSM operating at 1800MHz, known as DCS1800 [2]. Further examples are cdmaOne (IS-95) and US-TDMA (IS-136), which collectively are referred to as the second generation systems [3]. The current wireless communication system commercially available in South Africa is GSM, which is an international standard, enabling GSM-enabled devices to operate in virtually every country around the world. In GSM, multiple base stations provide wireless coverage. Frequencies are reused in non-adjacent cells. Apart from just wireless voice communications, the digital nature of second generation systems enabled further services such as text messaging and access to data networks [3].

1.1.3 Third Generation Communication Systems

With *third generation* (3G) systems more emphasis has been placed on multimedia communication. High quality images and video capabilities are added to normal person-to-person communication. Higher data rates and new flexible communication capabilities enhance access to information and services on public and private networks.

Wide-band CDMA (WCDMA) has emerged as the most widely adopted air interface for 3G systems. European research on WCDMA was initiated at the start of the 1990's by the European Union research projects CODIT and FRAMES. The *International Telecommunications Union* (ITU) decided at the *World Administrative Radio Conference* (WARC) in 1992 that the available frequencies around 2GHz would be used to implement the 3G systems, which it called the *International Mobile Telephony 2000* (IMT-2000). In January 1998 the European standardization body ETSI decided upon WCDMA as the third generation air interface. The first commercial network was opened in Japan during 2001 for commercial use in key areas, followed by Europe at the beginning of 2002. The specification of the standardization forums was created in 3GPP (the 3rd Generation Partnership

Project), which is a standardization body comprised of Europe, Japan, Korea, the USA and China [3].

An example of a 3G system is the *Universal Mobile Telecommunications System* (UMTS), which promises circuit-switched connections with data rates of 384kbps and packet-switched connections up to 2Mbps. The high data rates make video telephony and quick downloading of data possible. UMTS supports a wide range of applications that possess different *Quality of Service* (QoS), that make it possible for the network to be sensitive to the throughput, transfer delay, and data error rate requirements of the various applications. To achieve such differentiated services, four traffic classes have been identified, that applications can fit into:

- conversational
- streaming
- interactive
- background

The main difference between these classes is how delay-sensitive the traffic is. The conversational class is the most delay-sensitive, while the background class is the least [3].

1.1.4 Fourth Generation Communication Systems

When IMT-2000 was introduced in Europe, the licences cost a very large amount of money. The same is expected to happen in America. Operators and manufacturers are therefore not eager to develop a new generation of mobile communication. However, the need exists to plan for a *Fourth Generation* (4G) mobile communication system now already, since the assignment of spectrum and the commercial deployment are expected to take more than 10 years [4]. Furthermore, planning ahead for 4G allows operators and manufacturers to design current 3G systems in such a manner that future 4G systems will be implementable in the 3G environment.

Although 3G systems are more heavily based on IP and an attempt was made to enable a certain degree of private use, they are still mainly focused on public use. A fourth generation system would make it possible to offer private, unlicensed use as an extension to existing public systems.

In order for 4G to be a truly new generation, it will have to offer some completely new advances. Possible aims would be for it to incorporate and give service to all current systems, at all times and at all places. Enhanced services over the most efficient/preferred networks would be offered, depending on the user profile, the type of data stream and the traffic load of the network. These are possible features, but ultimately the actual shape of 4G is unknown at this time.

1.2 Motivation for research

Advanced scheduling has become an inevitable component of modern Quality-of-Service (QoS)-based data networks, of which 3G is an example. In the past, networks contained only one type of data, making it possible to optimise the architecture of the network according to its specific needs. For example, the telephone network is a rigid structure with good performance guarantees, while packet switched networks are more flexible but only provide marginal performance guarantees [5]. Modern integrated services networks carry a large number of different data types, each with its own requirements.

To provide QoS guarantees, a scheduler is required. The scheduler is usually located inside every switch or router. When the router has decided what the next destination of the various packets should be, the scheduler decides in what order the packets should be transmitted and at what rate, so that delay and throughput guarantees are met.

Wireless networks pose an additional problem since the channel conditions, through which information is transmitted, are constantly changing. A large amount of research on general schedulers has been published, but very little on schedulers in channel-varying wireless networks.

This research focusses specifically on schedulers for wireless networks. A new scheduling algorithm known as *Wireless Fair Largest Weighted Delay First* (WF-LWDF) is proposed, which is based on *Modified Largest Weighted Delay First* (M-LWDF). Both take channel conditions into account and take advantage of *Code Division Multiple Access* (CDMA)'s ability to transmit multiple packets at the same time, which increases the efficiency of the schedulers. The WF-LWDF algorithm was introduced with the primary aim of increasing the fairness of M-LWDF. Several performance aspects of the two algorithms are compared with numerous other schedulers by means of computer simulations. The simulation results are compared with some of the analytical results that have been published.

1.3 Dissertation overview

The dissertation has been divided into six chapters. In Chapter 1 an overview is given of the evolution of wireless networks, followed by the motivation of this work and a list of original contributions.

Chapter 2 starts off with a brief overview of QoS and scheduling requirements, followed by a discussion on a large number of wired and wireless schedulers. The wireless schedulers are further divided into constant bandwidth schedulers and CDMA-based schedulers, which both have fundamental differences.

In Chapter 3 the simulation model and the accompanying results are presented. In total six schedulers were modelled and their throughput, delay performance, power consumption, and their packet dropping rate were measured. The simulation software was programmed in C++. Many problems had to be overcome, one of the most important was how to model schedulers like Earliest Deadline First in a CDMA environment. The data source, buffer and channel models are presented, followed by a discussion on the various measurements performed by the simulator. During the actual simulations, the throughput, power consumption, delay behaviour and the packet loss were measured for six different schedulers, one of which is WF-LWDF, a new scheduler based on M-LWDF, but which is meant to improve its fairness. The simulation results are discussed and conclusions drawn.

Chapter 4 defines and discusses the concept of fairness. Simulation results of the six schedulers are compared. Of particular interest is the performance of the WF-LWDF scheduling algorithm compared to that of M-LWDF. Statistical Chernoff bounds are calculated for the fairness of both M-LWDF and WF-LWDF, based on the simulation results.

In Chapter 5, the analytical delay bounds of a range of different schedulers are compared under various load conditions. Chapter 6 presents the conclusions drawn in this dissertation.

1.4 Original contributions in this dissertation

The original contributions in this dissertation include:

1. The WF-LWDF scheduling algorithm is introduced as a solution to the fairness problem of M-LWDF.
2. SM-LWDF and W-LWDF are introduced to determine how the various factors in the priority equations of M-LWDF and WF-LWDF affect their behaviour and how important simultaneous power distribution is.
3. Statistical fairness bounds of M-LWDF and WF-LWDF are extracted from the simulation results.

Parts of the work presented in this dissertation have been presented by the author at the following conferences:

1. S.M. Scriba and F. Takawira, "Scheduling in Wireless Networks", *Proceedings of the Southern African Telecommunication Networks and Applications Conference (SATNAC 2001)*, Wild Coast Sun, South Africa, 2001.
2. S.M. Scriba and F. Takawira, "A Fairness Comparison of two Channel-aware Wireless Schedulers", *Proceedings of the Southern African Telecommunication Networks and Applications Conference (SATNAC 2002)*, Champagne Castle Sports Resort, South Africa, 2002.

Chapter 2

Introduction to Scheduling

2.1 Introduction

The main aim of this chapter is to present a literature survey on schedulers, or multiplexers as they are also referred to. But instead of just listing a large number of schedulers and explain their functions, we would also like to discuss the context in which schedulers are found and what they try to achieve.

The chapter starts off with a discussion on quality of service, which modern schedulers aim to achieve. This is followed by definitions of some scheduling terminology and the functional categories that schedulers can be divided into.

For the rest of the chapter, the various schedulers are introduced and their operation explained. A large number of schedulers exist, and the ones touched on in this chapter are by no means all of them. However, the scheduling schemes presented here should give the reader a fairly good overview of the different schedulers and how they are incorporated in the world of data networks.

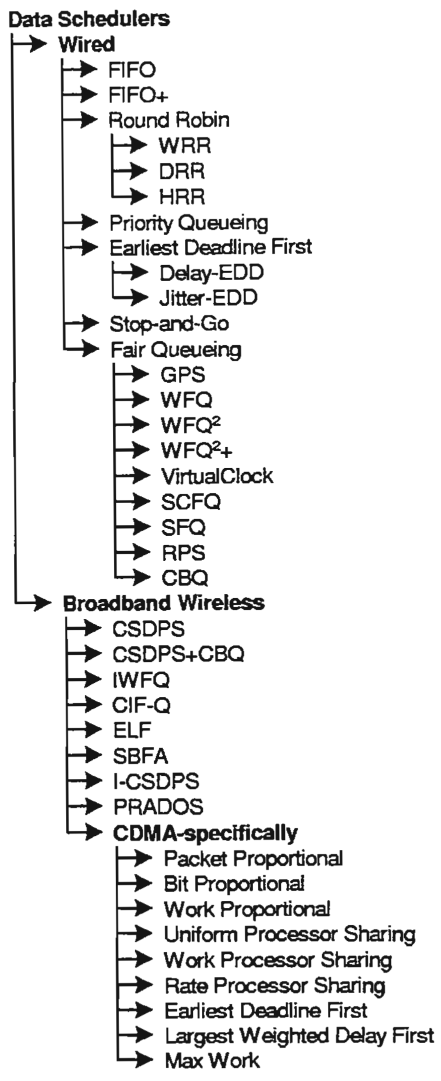

Figure 2.1: Overview of schedulers discussed in this chapter

2.1.1 Scheduling Overview

As was already mentioned, a large variety of schedulers exist, which can be quite overwhelming at first. It is therefore helpful to split the various algorithms into groups. This is not that easy, since some of the schemes have properties that fit into multiple groups. An attempt to achieve such a grouping can be seen in Fig. 2.1, which also represents the structure of how the scheduling schemes will be presented and discussed in the rest of this chapter. Note that many of the schedulers for wired networks are also implementable in a wireless network, but we still chose to leave them in the wired domain, which they were originally intended for.

Referring to Fig. 2.1, one can see that schedulers can in general be split into two major groups. The first are scheduling schemes for wired networks, while the other group are algorithms for wireless networks. As will be discussed in more detail, the difference between the two domains, is that wireless networks experience channel fading, multi-user interference, and general noise.

The wireless domain can be split into another subgroup, which is schedulers for CDMA networks. The problem in CDMA networks is that the available bandwidth depends on the required signal to noise ratio, which constantly changes as the channel quality changes. The result is that the available bandwidth is also varying. Instead of distributing bandwidth, schedulers in CDMA based networks should rather distribute power.

2.2 Scheduling concepts

A network consists of numerous interconnected nodes through which data travel. Each of these nodes contains a router, which decides where the information should be sent to next, in order to reach its destination along the best path. Once the router has made its decision, the information gets queued in a number of buffers, where it waits to be transmitted to the next node. The scheduler then chooses the packet to be sent next and assigns it a transmission rate.

Schedulers are also known as multiplexers, since they are responsible for transmitting data packets of various flows over the same link. This is of course equivalent to many everyday queueing problems that we encounter. Examples are banks and train terminals. Many of the scheduling principles used in modern networks can be used to alleviate everyday problems. An example of such an implementation is Matthew Andrew's PhD thesis [6] and the corresponding journal paper [7], in which he addresses the problems of load balancing in a factory and disk scheduling. In the on-line load balancing problem, jobs arrive on-line and need to be assigned to one of a set of machines, thereby increasing the load on that machine by a certain weight. In the disk scheduling problem, data is stored on a disk. A disk head needs to move to different positions on the disk to read the stored information or write new information. A convex reachability function determines how fast the disk head travels between tracks. The aim is to schedule the head so that it services all the requests in the shortest possible time.

Unlike simple multiplexers found in electronics, modern schedulers can be very complicated. Most modern schedulers, for example, take into account that the data is heterogeneous and that each data type therefore has different requirements to achieve quality of service (QoS).

2.2.1 Quality of Service

In modern networks, resources are getting scarcer. In the wireless domain, carrier frequencies are constantly being increased as bandwidth in the lower frequency domains is becoming scarcer and more expensive. In recent years, bandwidth has not only become a tradable commodity, but an expensive one at that. Strict charging policies have been incorporated in many wireless networks [8] and resource management in the form of buffer management [9], channel establishment [10], frequency distribution in FDMA networks [11], and power distribution in CDMA networks [12] has become an essential research area, in order to reduce the cost of wireless networks.

The cost of networks and the technical expectation have prompted users to expect a performance guarantee from network providers. The main aim of most of the current data communications research is to give networks the ability to guarantee Quality of Service to data sources entering a network. The term *Quality of Service* (QoS) has become very important in the world of telecommunications. Modern networks have changed a great deal from the original configurations, where each data type had a dedicated network. The telephone network, for example, provided good performance guarantees but poor flexibility, while packet switched networks were more flexible but provided limited guarantees [5]. Modern integrated services networks carry a number of different data types with varying characteristics and requirements. An example of such a scenario might be that e-mail, HTTP, and real-time services such as video streaming and telephone conversations might all be on the same network. Users require their telephone conversation to run smoothly without any interruptions, while checking their e-mail, surfing the web, and watching a streaming video. In a network of this kind performance guarantees are typically negotiated between a data source and the network before any data are sent. Once the negotiations have ceased and an agreement has been reached, the network has the responsibility of ensuring that the guarantees are met.

Both Ng [13] and Cao [14] list the first four of the following five points as a requirement for networks to achieve Quality of Service. The fifth point has been added specifically for wireless channels, where bit-error rates become important:

- Delay, jitter, throughput guarantees
- Short term fairness
- Long term fairness
- Graceful degradation
- Bit-Error-Rate (BER) guarantees (wireless channels)

Before a data source begins its transmission, it first negotiates with the network what the acceptable end-to-end transmission delay is. Once a certain percentage of bandwidth has been assigned to each data type, the scheduler is responsible for enforcing the agreement. Jitter is defined as the difference in delay that packets might experience. In most real-time services, jitter has to bė kept to a minimum, in other words, the delay by packets from the same source should be the same.

The scheduler furthermore has the responsibility of ensuring that sessions receive the same amount of attention, even if the bandwidth allocation differs. This is referred to as *short term fairness.* If a channel is blocked and a session cannot receive its share of the bandwidth for a certain amount of time, then *long term fairness* requires that as soon as more bandwidth becomes available, it needs to receive more service than originally agreed upon, in order to make up for the dead time. It is vital though, that during this make-up time, the other sessions still receive a minimum amount of service, which is referred to as *graceful degradation.* Finally, wireless channels are characterized by high error rates. The bit-error-rate can be directly controlled by varying the transmission power.

QoS has become one of the most important research areas in the field of integrated data networks. A large amount of literature has been published on basic mechanisms used in packet networks to support QoS guarantees [15], on QoS management structures and its requirements [16], [17], and specifically on supporting QoS in wireless networks supporting real-time data [18].

2.2.2 Useful scheduling features

To evaluate the performance of a scheduler, it is useful to have a list of desirable features a scheduler should have. Reference [19] lists 'Efficiency','Protection','Flexibility', and 'Simplicity' as the four essential requirements of wired network. Stiliadis [20] has fairly similar requirements:

- Isolation of flows
- Low end-to-end delays
- Efficient utilization of resources
- Fairness

- Simplicity of implementation
- Scalability

Firstly, the various sessions established in a network should be *isolated*, which corresponds to 'Protection' in [19]. Every session should be guaranteed a minimum amount of bandwidth, thereby being protected against any misbehaving sources that threaten to swamp the network with information. A scheduler should furthermore ensure that *queuing times* of packets are kept to a minimum, that bandwidth is used *efficiently*, and *distributed fairly* among the packets. The protocol should be *easy to implement* and *easily upgradable* to a larger network, which corresponds to 'Efficiency' in [19]. Examples of papers discussing implementation issues into a real network are [21] and [22].

2.2.3 Scheduling categories

Schedulers can be divided into various categories [20],[23], [24]:

- Work conserving vs non-work-conserving
- Cut-through device
- Frame-based vs Sorted priority

Firstly, work-conserving schedulers are never idle if any packets are queued in one of the buffers. Non-work-conserving schedulers, on the other hand, might not send a packet queued, but rather wait for a packet of higher priority to arrive, which is transmitted first. The second consideration is whether a scheduler is cut-through, which means that it may start transmitting bits before the entire packet has been received. Most schedulers are non-cut-through, since the transmission rate might be much larger than the rate at which bits arrive. Finally, frame-based schedulers handle only packets of one size, and serve them in a round-robin fashion, while in a sorted-priority scheduler, packets are transmitted according to their priority, with high-priority packets being transmitted first, followed by less important packets.

Dovrolis [25] names another two categories that can be added:

- Lossy vs lossless
- preemptive vs non-preemptive

In a lossless system the utilization of the scheduler has to be less than 100%. In a non-preemtive system, a packet transmission is always carried out in full.

Dovrolis carries on listing two useful requirements of a relative differentiated services architecture:

- Predictable
- Controllable

A predictable architecture is one where the differentiation is consistently independent of the class load variations. With controllable he means that the network operator should be able to adjust the relative QoS differences between classes based on their criteria. For more information on the proportional differentiation model and buffer management, see [25].

Another attempt on ensuring QoS is the Resource ReSerVation Protocol (RSVP). For more information see [26], which is a tutorial on how RSVP can be used by end-applications to ensure that they receive the end-to-end QoS that they require.

2.3 Wired schedulers

Now that we have defined the concept of QoS and listed important features of scheduling algorithms, it is time to discuss some of the schedulers that have appeared in the literature since the beginning of the 90's, when the main stream of literature on the more complicated scheduling algorithms for heterogeneous data networks started appearing. Schedulers for wired networks (or wired schedulers for short) are an important group of algorithms, since they have often been studied and analyzed in great detail and form the basis of the more complicated wireless schedulers. Numerous papers have been published in which a generalized network calculus was proposed for analyzing the delay bounds of schedulers [27], [28], [29], [30]. The results of these papers are often only approximate due to the complexity of many of the scheduling algorithms, which in most cases are still a simplification. Schedulers are usually looked at in isolation, which in real networks is not possible. The effect of routing, for example, should be taken into consideration, and only very few papers have looked into this [31], [32].

2.3.1 FIFO

The *First-In-First-Out* (FIFO) [23] is the simplest scheduling algorithm conceivable. As data arrive at a node, they are queued in a single queue. The packets are picked in the order they have entered the queue. In other words, the data are serviced in the order in which they arrived. The result is an extremely simple scheduler that is highly efficient with a very low complexity. The downside is that no quality differentiation of packets takes place, since all sessions experience the same delay and packet-loss characteristics. Another disadvantage is that FIFO relies completely on traffic control functions in boundary nodes, making the entire system vulnerable to traffic congestion due to misbehaving users.

2.3.2 FIFO+

A disadvantage of FIFO is that the jitter tends to increase dramatically with the number of hops, due to uncorrelated queueing delays at each hop. More hops should give an opportunity for sharing, which reduces the jitter. The trick is to correlate the sharing experience. FIFO+ [23] takes advantage of this by separating the traffic into priority classes. The average delay of packets in each priority class is measured. This result is used to calculate the difference between the particular delay of a packet and the class average. The difference is used at each hop to determine the time of arrival that the packet would have experienced if it had been given the average service. The scheduler then inserts the packet in the queue in the order as if it has arrived at the expected time.

2.3.3 Round Robin

A range of different *Round Robin* (RR) scheduling schemes exist. The basic idea behind RR is very simple. Traffic arrives at a scheduler and is stored in a fixed number of buffers. Each buffer might, for example, contain one data type. The scheduler then moves through all the buffers and serves a certain portion from the head of each buffer. The problem with the basic round robin scheduler is that it does not differentiate between different data types and all data are treated the same. A number of different implementations of the round robin scheduler have therefore been proposed, which attempt to have QoS guarantees.

WRR

The first, and very simple extension of round robin, is the *Weighted Round Robin* (WRR) scheduler [23]. With this scheduler the arriving packets are sorted into a number of priority levels. Each priority level is assigned a weight. A small unit of time referred to as a quantum is defined. The number of sequential time slices that each priority class gets in one round depends on the weight of the service class. Any quanta not used by one priority group when the queue gets empty are passed onto the next group.

DRR

Another extension of the round robin scheduling algorithm is *Deficit Round Robin* (DRR), which was introduced in [33]. The scheme is simple enough to be implemented in hardware and is able to achieve near perfect fairness. The first version of DRR did not use weights to support different priority levels, but can quite easily be extended to do so. The reason why DRR is able to achieve near perfect fairness, is that if a queue was not able to send a packet in the pervious round because its packet size was too large, the remainder of the current quantum is added to the quantum of the next round [23]. Deficits are therefore kept track off, which reduces the quantization problem and explains the name of the scheme.

HRR

A third extension of the round robin discipline is *Hierarchical Round Robin* (HRR) [34], which like WRR has a number of different service levels. The reason for the name is that higher levels receive more bandwidth than the lower levels, which means that the frame time at a higher level is smaller than the frame time at a lower level. Hierarchical round robin is useful in high-speed networks, which carry traffic with a wide range of performance requirements [35]. Gigabit/sec transmission rates are feasible and strict bounds are possible on the buffer space required for rate controlled connections. For more information on scheduling in high speed networks, see [36] and [37].

2.3.4 Priority Queueing

A priority scheduler separates arriving traffic into various priority levels. Each priority group has its own queue, which is served in a FIFO fashion. A priority queue will only be served once all the packets from all queues of higher priority have been served, with the result that the system has a course granularity [23].

2.3.5 Earliest Deadline First

Earliest Deadline First (EDF) works according to the principle that every time a packet p arrives at a server e on its path, server e assigns packet p a deadline d_p^e. Whenever server e has a number of packets waiting it gives all its service to the packet p with the smallest d_p^e, in other words, the packet which is closest to its deadline [38].

A fairly large amount of research on the EDF scheduler has been published. The first delay violation probability bounds of EDF, for example, were published in [38], while [39] used the theory of large deviations and the theory of effective bandwidths to analyze the packet losses of an EDF scheduler. A polynomial delay bound of EDF can be found in [40]. An approximation of EDF introduced in [41] is known as *Rotating-Priority-Queues*$^+$ (RPQ$^+$), which uses a set of prioritized FIFO queues whose priorities are rearranged periodically to increase the priority of waiting packets. Reference [40] contains the first distributed, deterministic EDF scheduling protocol, which has polynomial delay bounds.

Delay-EDD

Another name for *Earliest Deadline First* is *Earliest-Due-Date* (EDD). An extension of this scheduler is the Delay-EDD service discipline [34]. It requires the source to obey a peak and average transmission rate to be able to provide a delay bound. The delay bound is calculated by adding the expected arrival time to the delay bound at the server. The packet is therefore assigned a delay deadline, which is equal to the time at which it should be sent, had it been received according to the contract agreed upon by the source.

Jitter-EDD

The Jitter-EDD [34] scheduler is an extension of Delay-EDD. It is able to provide delay-jitter bounds by stamping each packet that has been served with the difference between its deadline and the actual finishing time. A regulator at the next node holds the packet for this period of time before it is made eligible to be scheduled, thereby providing minimum and maximum delay guarantees, which bounds the jitter.

2.3.6 Stop-and-Go

Stop-and-Go [34] is another scheduler that aims to preserve the smoothness property of traffic as it travels through the network. It achieves this by dividing time into frames. Only packets that arrived in the previous frame are transmitted in the present frame. The result of this scheme is that minimum and maximum delay guarantees are achieved, which bounds the jitter.

2.3.7 Fair Queueing

One of the network requirements is for it to distribute its service in a fair manner. A fair scheduler distributes the available bandwidth proportionally to the amount of service a priority class deserves. Out of this grew a whole class of schedulers, known as the *Fair Queueing* (FQ) family of schedulers [42], [43], [44].

One of the earliest and simplest implementations is the *Fair Queueing* (FQ) scheduler. It works on the simple principle that if N channels share an output trunk, then each should get $1/N$ of the available bandwidth [34]. This scheme is remarkably similar to the classic round robin scheduler. The same problem arises, namely that all data types are treated the same.

To solve this problem, weights had to be assigned to the various data families or sessions. The result is the slightly more advanced scheme *General Processor Sharing* (GPS), which forms the basis of the Fair Queueing family of schedulers [23]. It is an ideal fluid model that cannot be implemented in real networks. The reason for this is that GPS serves all backlogged sessions simultaneously with a minimum rate equal to their reserved rate. The traffic is considered infinitely divisible, which is not possible in a real network containing packets that cannot be split into smaller units.

GPS

To understand the basic idea behind GPS, consider a node with a capacity Cb/s, through which the data of N sessions travel, where ϕ_i represents the weight associated with session i, and $i = 1, \ldots, N$. The minimum service rate guaranteed to each session i is then given by

$$C \cdot \frac{\phi_i}{\sum_{j=1}^{N} \phi_j}, \quad i = 1, \ldots, N. \tag{2.1}$$

In the case of non-backlogged queues, the excess bandwidth is distributed among the set of backlogged queues $B(t)$ in proportion to their respective weights. The real service rate

r_i that session i will receive is [23]

$$r_i(t) = \begin{cases} C \cdot \frac{\phi_i}{\sum_{j \in B(t)} \phi_j}, & i \in B(t) \\ 0, & \text{otherwise.} \end{cases} \tag{2.2}$$

A large amount of literature has been published on the ideal GPS scheme. The first papers that attempted to analyze GPS in detail were [5], which analyzed the delay of GPS in the single node case, while [45] analyzed the end-to-end delay of the multi-node case.

These papers were later followed by [46], which is an extension of Parekh and Gallager's deterministic study of GPS. Another paper that used an exponential characterization to analyze a GPS system was [47]. Here, the stability of virtual packet switching communication networks, that employ processor sharing type service disciplines, was studied in a stochastic setting.

A common way of designing *Generalized Processor Sharing* (GPS) schedulers is by using deterministic QoS guarantees, which are conservative and lead to capacity limitations. Elwalid and Mitra [48] therefore developed a scheduling theory for the design of GPS weights, which is based on statistical QoS guarantees and statistical multiplexing.

A similar paper on the design of GPS weights was [49]. In this paper, the main goal was to achieve statistical multiplexing gains in the presence of multiple traffic and Quality of Service (QoS) classes of connection, that share a common trunk.

Finally, Andrews and Zhang [50] study the difference between temporary sessions and permanent sessions in a network implementing GPS as its scheduler. They separate the two models in terms of stability. This is of interest, since GPS is known to be stable and to have polynomial delay bounds with permanent sessions, but not necessarily with temporary sessions.

The resultant GPS algorithm is ideal, since it is able to guarantee a maximum end-to-end delay to a session whose traffic is leaky bucket shaped. It is furthermore able to ensure fair allocation of bandwidth among all backlogged sessions, regardless of whether or not their traffic is constrained [23]. These qualities make GPS very desirable and a large number of scheduling schemes have therefore been published that attempt to approximate the ideal GPS algorithm as closely as possible. The following sections mention some of the GPS-based schedulers that we have come across.

WFQ

As mentioned already, *General Processor Sharing* (GPS) is a fluid model that cannot be implemented [13]. Instead a *packetized* approximation of GPS (PGPS) is used in real switches and routers, which according to Parekh and Gallager [5] was first proposed by Demers, Shenker, and Keshav [51] under the name *Weighted Fair Queueing* (WFQ).

The problem with this system is its computational complexity. The problem lies in the fact that a virtual GPS fluid system has to be modelled as a reference system, which the WFQ tries to follow as closely as possible. Incoming traffic is modelled as a volume of fluid. A packet is considered to be served in the GPS system, when the fluid volume of the packet has been completely served by the GPS scheduler. WFQ uses the computed departure times of packets in the GPS model to select the packet with the smallest virtual finish time [23].

Although [5] proved that in the absence of link-sharing, the WFQ system has to be within one packet transmission time of that provided by GPS, [52] showed that the WFQ scheduler might actually be $N/2$ packets ahead of the GPS reference system, where N represents the number of sessions sharing the link, which can be detrimental for real-time services.

WF^{2}Q

In order to implement GPS, a virtual time function $V(t)$ is usually used, which is a linear function, based on the number of busy sessions and their service rate [23].

$$\frac{\partial V}{\partial t} = \frac{C}{\sum_{i \in B(t)} \phi_i}. \tag{2.3}$$

The *Worst-case Fair Weighted Fair Queueing* (WF^{2}Q) [23] is another approximation of GPS that improves on WFQ. To achieve this it uses $V(t)$ to compute virtual start and finish times as in WFQ and defines a packet to be eligible at time t if its virtual start time is at most t. Finally it schedules eligible packets in increasing order of their virtual finish times. The result is that WF^{2}Q remains closer to the ideal GPS reference system and therefore has tighter delay bounds than WFQ.

WF^{2}Q+

Even though WF^{2}Q is more accurate than WFQ, its complexity remains the same as WFQ. WF^{2}Q+ is a simpler version of WF^{2}Q that attempts to improve the complexity problem. It achieves this by using a new virtual time function [23]

$$V_{WF^2Q}(t+\tau) = \max\left\{V_{WF^2Q+}(t) + W(t,t+\tau), \min\{s_i^{h_i(t)}\}_{i\in B(t)}\right\}, \qquad (2.4)$$

where $W(t,t+\tau)$ is the total amount of service provided by the server during the period $[t,t+\tau]$, $h_i(t)$ is the sequence number of the packet at the head of session i's queue, and $B(t)$ and $s_i^{h_i(t)}$ are the set of sessions backlogged in the WF^{2}Q+ system and the virtual start time of packet $h_i(t)$, respectively.

VC

Virtual Clock (VC) attempts to emulate *Time Division Multiplexing* (TDM) [34]. Each packet is given a virtual transmission time, which is the time at which the packet would have been transmitted were the server actually doing TDM. By sending packets in virtual time order, VC can be shown to emulate TDM.

[53] introduced a slight variation on VC, known as *VirtualClock with Priority Buffer* (VCPB). VCPB combines the VC algorithm with a priority buffering strategy. It is able to provide complete isolation between different traffic classes, which is one of the QoS requirements.

SCFQ

Like WFQ, *Self-Clocked Fair Queueing* (SCFQ) attempts to imitate GPS as closely as possible. It was first introduced by Golestani in [54], in which delay bounds are calculated, based on the leaky-bucket work published by Parekh and Gallager in [5] and [45]. Although SCFQ is simpler than WFQ and WF^{2}Q, its delay bound is larger [23].

The reason why SCFQ is simpler than WFQ and WF^{2}Q is that it does not use a GPS reference system. Instead a virtual time function is calculated, which depends on the progress of work in the actual queueing system, resulting in a $O(\log N)$ complexity [23].

SFQ

Start-Time Fair Queueing (SFQ) was introduced in [55] as a computationally efficient and fair algorithm. The main difference between SFQ and SCFQ is that SFQ picks the packet with the smallest virtual start time for transmission on the link [23].

RPS

Rate Proportional Servers (RPS) is a class of schedulers that behave the same as WFQ. The RPS concept was introduced by Stiliadis in [56], [20], [57], [24], [58], and [59], in which he introduced a new algorithm known as *Frame-based Fair Queueing* (FFQ), which belongs to the RPS class. It aims to provide GPS behaviour, without the complexity of simulating the fluid-model system in the background. Another scheduler that is introduced in [59] is *Starting Potential-based Fair Queueing* (SPFQ). RPS schedulers rely on the system potential function to keep track of the service offered by the system to all sessions sharing the outgoing link [23]. In both FFQ and SPFQ the system potential function is recalibrated. In FFQ this happens at frame boundaries, while for SPFQ the recalibration occurs at packet boundaries.

Classes of schedulers such as RPS are usually introduced to group schedulers that can be analyzed with the same procedure. Other examples of a scheduling class are the Guaranteed Rate class [60][61] and the Rate-Controlled Service Disciplines class of schedulers proposed in [62], which is able to provide end-to-end deterministic and statistical performance guarantees.

CBQ

The modern internet consists of a large number of administrative domains. Link-sharing allows the bandwidth of a link to be distributed between the various protocol families and traffic types. Bandwidth not used from one organization is passed onto the other organizations sharing the link.

To create such a shared link, a hierarchy has to exist, which distributes the bandwidth between the various organizations in some organized manner. Such a scheme is referred to as *Class Based Queueing* (CBQ) [23]. In CBQ each traffic class is assigned a rate, a priority, an average packet size, and a borrow flag, which indicates whether the class is allowed to get a share of the unused bandwidth or not. A CBQ scheduler consists of a general scheduling scheme and a link sharing scheduling scheme. The general scheduler sorts the packets in the absence of congestion. If packets are on time, they are therefore passed on directly by the link-sharing scheduler to the general scheduler, without experiencing any queueing delay.

If congestion does occur, the link-sharing scheduler is activated and takes over. Arriving packets will now be placed into their queues and rate shaped according to the minburst parameter, or they are allowed to borrow bandwidth from other idle traffic classes. Temporary bursts are possible in order to meet demands. Classes of the same priority are served in either a packet-by-packet RR or WRR fashion.

2.4 Broadband wireless schedulers

Modern schedulers in wired networks need to take the varying requirements of heterogeneous traffic into consideration, in order to be commercially viable. It should have become apparent in the discussion up to this point, that some of the schedulers trying to achieve QoS were quite complicated. Schedulers in wireless networks have extra challenges they need to conquer. Some of the difficulties are [14]

- High error rate and bursty errors
- Location-dependent and time-varying wireless link capacity
- Scarce bandwidth
- User mobility
- Power constraint of mobile hosts

In order to overcome these challenges and still be able to guarantee QoS, a scheduler should meet the following requirements [14]:

- Provide long term fairness
- Achieve high wireless channel utilization
- Minimize packet loss
- Provide delay (jitter, if possible) bounds for flows with error-free links or sporadic link errors
- Support multiple classes of traffic with QoS differentiation
- Achieve low power consumption in mobile hosts
- Achieve medium algorithm complexity

Numerous schedulers have been proposed that attempt to achieve these requirements, with varying results. We would like to present some of these here, but need to talk about the media access protocol first.

When radio transmissions started, multi-user interference had to be dealt with. The first idea was to have each transmission on a different frequency band, which is known as

Frequency Division Multiple Access (FDMA). Modern FM and AM radio transmissions still work on this principle. One of the major problems with this system is that it requires a separate frequency channel for every simultaneous transmission. With the advent of cellular telephones this concept was unacceptable. A better approach was to divide a time period into several quanta and assign each quantum to one of the users requiring service. During a transmission period, the base station would rotate through all the time quanta and during each of these communicate with the respective mobile user. The transmission time is therefore divided between the simultaneous users, which is referred to as Time Division Multiple Access (TDMA). If one takes a step back, while investigating cellular phones, one finds that both TDMA and FDMA are being used. A base station will use TDMA to communicate with all its current users, while neighbouring base stations will use different frequencies, in other words *Frequency Division Multiples* (FDM), to ensure that the neighbouring cells to not interfere with one another.

In both FDMA and TDMA, the amount of available bandwidth is effectively determined by the maximum transmission rate of the transmitter. Schedulers try to multiplex various flows onto the same data link. They are therefore required to take the maximum bandwidth available into account when choosing packets for transmission and assigning them data rates. This is quite different in a CDMA network, where the available bandwidth is constantly changing. But let us look at some of the FDMA and TDMA compatible schedulers first.

2.4.1 CSDPS

Channel State Dependent Packet Scheduling (CSDPS) [14] was one of the first schedulers that attempted to take location-dependent and bursty errors into account. It works on the principle that a separate queue is kept for each mobile's packets. The packets in each queue are served in a FIFO fashion. To decide which of the queues to pick packets from, either a Round Robin (RR), Earliest Timestamp First (ETF), or a Longest Queue First (LQF) scheduling scheme have been proposed. What differentiates CSDPS from RR, ETF, and LQF, is that the status of the various links is monitored. If a link is found to be in a bad state, due to bursty errors it might experience, the scheduler does not serve the respective queue until a time-out period has expired. The state of a link is declared bad, if no acknowledgement is received from its destination node.

The problem with CSDPS is that it does not have any mechanism to guarantee bandwidth for a mobile user. When a mobile is believed to be in a bad state, its share of the bandwidth will be distributed to the other mobiles, regardless of how much bandwidth they had already received. No limit is therefore imposed on the amount of service a mobile user may receive.

2.4.2 CSDPS+CBQ

A combination of CSDPS and class-based queueing (CBQ) was proposed to combat the unfair bandwidth distribution of CSDPS [14]. The CBQ component of this scheduler requires users to be grouped into classes and each class to be assigned a certain fraction of the available bandwidth, which in turn assures that the available bandwidth is distributed in a fairer fashion than was the case with CSDPS alone. It achieves this by keeping track of the amount of service received by each class in a certain time interval. The CSDPS component deals with the wireless link variations. Instead of using the longer data packet and acknowledgement pair to determine whether a link is in a bad state, the ready-to-send and clear-to-send pair are used instead, which are usually much shorter. Because of this less transmission bandwidth is wasted on a bad channel. A problem with this algorithm is that channels that were in a bad state do not receive more bandwidth once a link recovers to compensate for the amount of service they lost.

2.4.3 IWFQ

Idealized Wireless Fair Queueing (IWFQ) [14] is an adaptation of WFQ into the wireless domain. The wireless equivalent of the fluid model GPS is *Wireless Fluid Fair Queueing* (WFFQ). While no link-errors are detected, IWFQ behaves exactly the same as WFQ. When a link error is detected, the head-of-queue packet with the smallest finish time will be selected to be transmitted. This process will be repeated, until a packet is picked with a good link-state. A reference WFQ scheduler is run to determine whether the service for a queue is leading, lagging, or in sync, which corresponds to a queue size being smaller than, larger than, or the same as the queue size in the error-free WFQ scheduler. The problem with this system is that a channel that is lagging behind, will receive exclusive service when its link improves. The result of this is the starvation of other error-free links.

2.4.4 CIF-Q

Channel-Condition Independent Packet Fair Queueing (CIF-Q) was first proposed in [13]. Like IWFQ, it uses an error-free fair queueing reference system. *Channel-condition Independent Fair* (CIF) are a set of properties that a Packet Fair Queueing algorithm should have in a wireless environment, which include delay and throughput guarantees, long-term fairness, short-term fairness, and graceful degradation. CIF-Q was designed to meet these requirements.

SFQ was chosen to be the core of CIF-Q, even though other wire-line schedulers can also be used. As was the case with IWFQ, a virtual error-free system is run in parallel to the error-prone system, with the difference that in CIF-Q, the virtual time is only kept in the error-free system and not in the real system. The service order is determined according to the virtual error-free system and then implemented in the real system. If a link error is detected, the affected packet in the real system is kept, while the equivalent packet in the error-free system is assumed to be transmitted. By keeping track of the difference between the amount of service received by the real and the error-free system, the lead or lag of a queue can be determined. A leading flow will still receive $a \cdot r_i$ of the allocated service rate r_i it should have received. a is therefore a parameter that determines the minimal fraction of service to be received by a leading flow, which avoids starvation and enables graceful degradation [14].

2.4.5 ELF

Effort-Limited Fairness (ELF) is a wireless fair scheduler, which was first proposed by [63] and extends WFQ through dynamic weight adjustments. In this paper, two additional goals are attached to a wireless scheduler, which are usually not considered:

- The amount of capacity loss suffered by a flow should not be proportional to its bandwidth or its error rate, but should be configurable through administrative controls.
- It must be possible to administratively bound the amount of capacity that is lost due to location-dependent errors.

These extra goals stem out of the question, how much extra air time should be assigned to high-error flows at the expense of others. A clear distinction between the words "effort" and "outcome" is given, where "effort" is defined as the amount of air time spent on a flow, while "outcome" is the actual useful throughput achieved by the flow. An ELF scheduler will attempt to achieve the outcome desired by the users, which would usually be a specific throughput or a specific fraction of residual link capacity. It does this, taking into account the amount of effort required. The administrator is able to control a power factor, which determines the fairness and efficiency of the algorithm. If, for example, the power factor of a certain flow is 200%, then the proportion of bandwidth assigned to this flow will be doubled, when high error rates are encountered. The power factor of other flows might be 50%, thereby dropping the available bandwidth under high error conditions. The reason for doing this is that an administrator might feel it important for streaming audio and video to keep a constant throughput. In high error conditions, a large amount of effort will be required for a relatively small outcome. More bandwidth will therefore have to be assigned to the audio and video flows, to keep the outcome constant.

2.4.6 SBFA

Server-Based Fair Approach (SBFA) [14] belongs to the group of schedulers using an explicit compensation counter or server. The structure of the SBFA has two queues per session, a packet queue (PQ) and a slot queue (SQ). When a packet of flow i arrives, it is queued into the appropriate packet buffer PQ_i, while a virtual copy, referred to as a slot, is queued in the SQ_i buffer. One of the wireline schedulers is then employed to schedule the slot queues. When the wireline scheduler of choice has chosen a head-of-queue packet from one of the slot queues, the corresponding packet in the packet queue is transmitted, if the corresponding transmission channel is in a good state.

The question remains what happens in the case of a bad link state. A compensation server known as the Long-Term Fairness Server (LTFS) is responsible to keep track of the service that a channel or user has lost, because its link was in a bad state. The way the LTFS achieves this, is as follows: When the wireline scheduler has chosen a packet from the slot queue and finds the link to be in error, the corresponding packet in the packet queue remains in the queue. The chosen slot in the slot queue is then transferred into a separate LTFS buffer. A fraction of the total bandwidth is dedicated to the LTFS, and when the link returns to a good state, the LTFS can use its share of the bandwidth to compensate the flow that did not receive service due to the link error, without disrupting the service patterns of the other queues.

Although this architecture is quite a simple design and is able to provide throughput guarantees, it has no restrictions on the maximum amount of service that a user may receive. If a channel is permanently in a good state, then it might receive far more service than was promised to it. Another problem is that networking resources have to be pre-allocated to each user and the share per user cannot change. Also, the packet sizes of each flow have to be fixed, in order for the algorithm to work effectively.

2.4.7 I-CSDPS

Improved Channel State Dependent Packet Scheduling (I-CSDPS) [14] consists of a Deficit Round Robin (DRR) scheduler plus a compensation counter. In DRR, each flow has its own queue. The queues are served in a round robin fashion. The number of bits served from each queue is determined by the Quantum Size (QS). Since the QS is in most cases not an exact multiple of the sizes of the packet to be served, the remainder of the QS that is too small for another packet to be transmitted is transferred to the Deficit Counter (DC). In the next round, the new QS is added to the DC, which is the total QS available to the queue for that round.

A Compensation Counter (CC) for each flow is added to monitor how much service was lost due to link errors. If a link error is detected, the scheduler decreases the corresponding DC by the QS of the flow, while the CC is increased by the QS. At the beginning of each round $\alpha \cdot \mathrm{CC}$ amount of credit is added to the DC, while CC is decreased by that amount, where $0 < \alpha < 1$.

2.4.8 PRADOS

Prioritized Regulated Allocation Delay Oriented Scheduling (PRADOS) [14] is a scheduler designed specifically for wireless Asynchronous Transfer Mode (ATM) and to work with the MAC protocol MASCARA. Packet scheduling in wireless ATM has some special concerns related to MAC framing structures and heterogeneous traffic classes. To understand and solve these scheduling problems, research on the analysis of queue distributions [64], congestion control algorithms for ATM networks [65], [66], and mechanisms for enforcing ATM contracts by shaping the incoming cell streams [67] needs to be taken into consideration.

PRADOS is based on Backward Earliest Due Date First (B-EDF) with priority and combined with a leaky bucket regulator. Each connection is given a priority number depending on which traffic class it belongs to. Available classes are, in order of priority, Constant Bit Rate (CBR), real-time Variable Bit Rate (VBR), non-real-time VBR, Available Bit Rate

(ABR), and Unspecified Bit Rate (UBR). For further information on the unpredictability of VBR traffic volume, read [68]. A token pool is introduced for each flow, where tokens are generated at a fixed rate equal to the mean cell rate, while the size of the pool is the maximum burst size of a connection.

The packets waiting in a queue are sorted in order to serve them as close to their delay deadline as possible. The head-of-queue packets are chosen according to the priority order of the flow. Every time a packet is sent, a token is removed form that connection's pool. Packets are transmitted until no more tokens are available in any of the queues, or all transmission slots have been used up. If none of the queues have tokens left, yet there are still transmission slots available for the current round, then the scheduler will start filling the remaining slots by choosing any remaining waiting packets from the queues in the order of their priority.

2.4.9 CDMA-specifically

Apart from FDMA and TDMA there exists another air interface known as *Code Division Multiple Access* (CDMA). A large amount of research has been published on this topic, since CDMA has a higher capacity than both FDMA and TDMA. In CDMA networks, multiple users are able to communicate with a base station at the same frequency and at the same time. This is achieved by spreading the transmission onto a large bandwidth. If the spreading code used is chosen to be orthogonal to the codes of other users, then no or very little interference will take place. An example of a system that uses this technology is Bluetooth, which is a popular wireless network protocol for the office environment that is currently available on the market. Reference [69] contains an overview of the scheduling process in a Bluetooth network.

In a FDMA and TDMA environment, the available bandwidth at each node is constant. This is not the case in a CDMA network. The available bandwidth is constantly changing and depends on a range of different factors. It therefore makes more sense in a CDMA environment to distribute the available power instead of bandwidth, since the amount of available power at any point in time is dependent on the power supply and is considered to be constant. The way the available power is distributed depends on the individual CDMA-based schedulers. The following is a list of schedulers that are mentioned in [70]. Other examples of CDMA-based schedulers exist, as for example [71], which makes a comparative study of three scheduling algorithms for the uplink of a Wide-band CDMA (WCDMA) system—restrictive, pre-emptive and random scheduling. But these type of schedulers are not of interest here.

Packet Proportional

With a packet proportional CDMA scheduler, the power is divided in proportion to the number of packets waiting in the queue. This is similar to the Largest Queue First (LQF) scheduler in the wired domain, where the head-of-queue packet from the longest queue gets chosen for transmission. With the packet proportional scheduler the number of packets in a queue are added up and normalized by the total number of packets queued. The share of the available power that a queue will receive is equal to the fraction that is generated in this way. The power that is assigned to each queue can then be translated into a maximum data rate, at which packets are then transmitted. The amount of power assigned to each queue is then re-evaluated at regular intervals.

The problem with the packet proportional scheduler is that it has no isolation. The traffic departures mirror the statistics of the arrivals. In other words, if one of the sources misbehaves and transmits a very large burst of data, then the packet proportional scheduler will detect a large queue forming and will serve this queue almost exclusively, with the result that the other queues will experience starvation. A positive side to the scheduler is that the queue lengths are kept to a minimum.

Bit Proportional

The bit proportional scheduler is very similar to the packet proportional scheduler. With the bit proportional scheduler the power is divided in proportion to the number of bits in the queue, instead of the number of packets. The packet proportional server performs its task well as long as packets are slotted (packets are of constant length). As soon as the packet length varies, then counting the number of packets queued is no longer an indication of how long it will take to empty the queue at some constant bit rate. Instead, a scheduler has to sort the packets on a bit level to minimize the queue length. Apart from this minor change, the algorithms are identical and have the same advantages and disadvantages.

Work Proportional

Counting the number of packets or bits (in the case of variable packet length) gives an indication of the required time that will be needed to empty a queue at a constant bit rate. Another consideration, that is not addressed this way, is the amount of work that will be required, to clear the queues. The amount of work is determined by how much power will be required per bit, which is multiplied by the number of bits in the queue. The work proportional scheduler divides the available power among the queues in proportion to the amount of work that waits in each queue. Once the power has been distributed, the appropriate data rates are calculated and the data is transmitted. The power distribution is recalculated after a certain amount of time has passed. Although this scheme distributes power in a fairer manner, it has the same problem that there is a lack of isolation between the arriving and transmitted data.

Uniform Processor Sharing

The uniform processor sharing scheduler is similar to the round robin scheme in wired networks, which effectively distributes the available bandwidth evenly among the backlogged queues by giving each backlogged queue a turn to use the entire available bandwidth and transmit its packets at the maximum possible baud rate.

The uniform processor sharing scheduler simply divides the power evenly among the users with non-empty queues. The head-of-queue packets of each buffer are then transmitted at the same power. No differentiation is therefore made between the queues, regardless of how

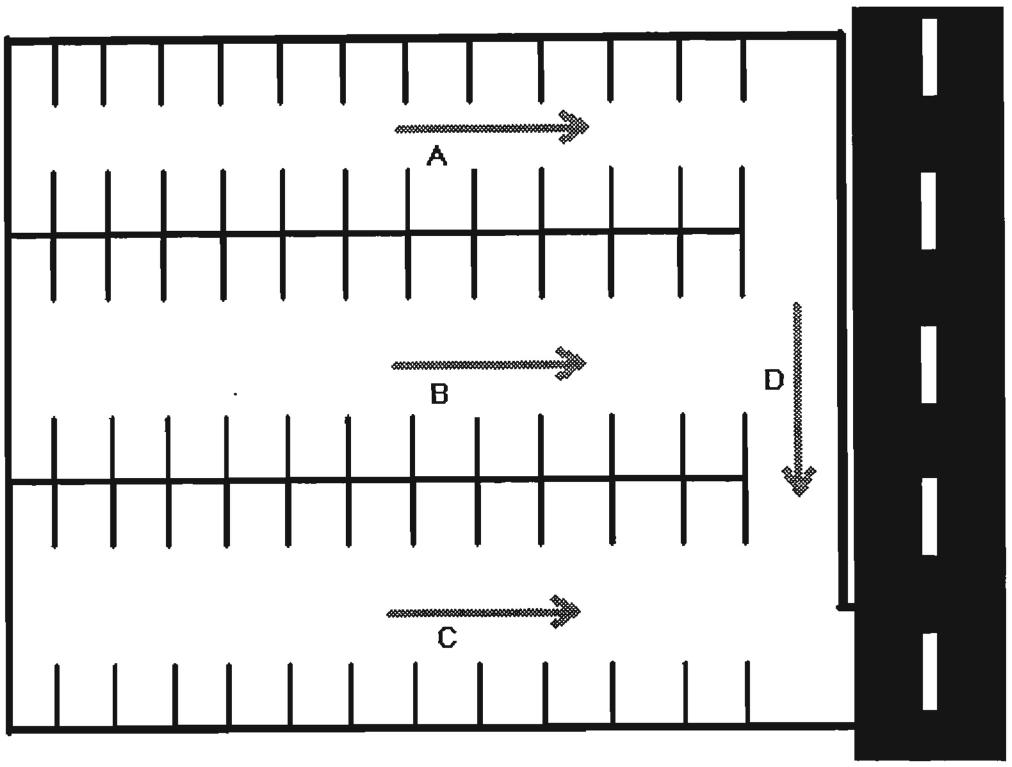

Figure 2.2: Graphical representation of parking lot problem

long the various queues are and for how long packets have been queueing. The problem with this type of approach is the parking lot scenario, depicted in Fig. 2.2. Imagine that this figure represents the parking area of a sport stadium and a particularly popular game is on, with the result that the parking area is full. When the match finishes all cars try to leave at the same time.

The problem consists of three flows, A, B, and C. Two schedulers, which might be in the form of a traffic officer, are required to direct the traffic. The first scheduler would be placed at the point where the flows A and B join up to form flow D. A uniform processor sharing server would work quite well here and would be considered by most drivers as a fair solution. The second scheduler would be placed at the exit point of the parking lot, where flows C and D join up. If this scheduler was a uniform processor sharing server, then flow C and D would receive the same amount of service, which in turn means that flows A and B would receive half the service that flow C is receiving. This is not a fair system. The problem gets even worse if more parking bays are added. For every extra parking bay, the service rate of flows A and B will reduce by a factor of 2.

If instead a packet proportional server was used, then this scenario would not be a problem. The queue length of flow D would be twice as long as that of flow C, with the result that flow D would receive twice as much service. This service can effectively be divided among flows A and B, which in turn means that flows A, B, and C all receive exactly the same amount of service.

The parking lot problem can be extended to a multi-node network scenario with multiple sources. If numerous flows of data are travelling the same path in a network and uniform processor sharing servers are used, then the flows that entered the path a few nodes later than other flows will receive more service, or in the case of a CDMA network, more power.

Work Processor Sharing

The work processor sharing server distributes the available power based on the current attenuation factor of each target channel. In other words, assuming that all the packets in one queue have the same destination, the current attenuation of this channel will be measured to give an estimate of how much power will be required to transmit the queued packets to this destination. The available power is then distributed in proportion to each queue's current channel attenuation function. According to [70], the result is that all queues will be assigned the same data rate R, namely

$$R = \frac{1}{\sum_j c_j}, \tag{2.5}$$

where j denotes all the backlogged queues.

Rate Processor Sharing

Rate Processor Sharing is a type of GPS scheduler. Since we are dealing with CDMA as the air interface, power is distributed using the adjusted GPS algorithm

$$P_i = \frac{\phi_i}{\sum_{j \text{ active}} \phi_j} P, \tag{2.6}$$

where P is the total power available and ϕ_i is the GPS weight. What sets Rate Processor Sharing servers apart from other GPS servers is the way that the weights ϕ_i are found. In the case of Rate Processor Sharing, this is achieved using $\phi_i = c_i \cdot R_i$, where c_i is the current channel attenuation function and R_i is the minimum data rate that still meets the QoS requirements of user i. The minimum rate R_i is computed with high probability using a Chernoff bound.

Out of the CDMA-based schedulers mentioned so far, this is the first one that attempts to meet the QoS requirements of the various data types it might be dealing with.

Earliest Deadline First

In the *Earliest Deadline First* (EDF) scheduler, the total available power is assigned to the user for which the deadline of the packet at the head of the queue is closest. In other words, from the various buffers, the packet at the head of the queue is picked, for which the difference $(T_i - waitingtime)$ is the smallest. This is a fairly simple scheduler, which is nonetheless able to adhere to delay requirements, if the queueing times are of the same order as the delay deadlines.

Largest Weighted Delay First

The *Largest Weighted Delay First* (LWDF) scheduler assigns all available power to the user with the largest $a_i\times$(current delay) product. a_i represents the required minimum rate of decay of user i's delay distribution and is given by $a_i = -\log \delta_i / T_i$, where T_i and δ_i represent the delay threshold of user i and the maximum probability of exceeding it, respectively.

An analysis of LWDF can be found in [72], which shows that the individual flows in the scheduler satisfy a large deviation principle with the rate function given by a finite-dimensional optimization problem. This problem is solved in [73] for each network node, thereby solving the Critical Node Property.

The LWDF algorithm forms the basis for the *Modified Largest Weighted Delay First* (M-LWDF) algorithm, and a new algorithm we present in this dissertation, known as the *Wireless Fair Largest Weighted Delay First* (WF-LWDF) algorithm. The M-LWDF was first introduced in [74] and will be discussed in detail in the rest of this dissertation. Both M-LWDF and WF-LWDF take channel conditions into account when distributing power, which makes them unique. An exception is [75], which presents a solution to the problem of minimum energy scheduling on wireless links. It achieves this by allocating bits at different times instead of allocating them across multiple channels.

Max Work

For Max Work all power is assigned to the user who has the most work waiting, where work is once again defined as the amount of power required to transmit one bit, multiplied by the total number of bits waiting. This algorithm is similar to the Work Proportional server, except that power is now not divided into smaller units and assigned to multiple queues, but instead it is all given to one queue for a fixed period of time. This leads to a more granular distribution, which would be expected to be less effective.

2.5 Summary

This chapter started off introducing some of the scheduling concepts of importance. These included a discussion on the concept of quality of service, which has become a very popular discussion topic, and some useful scheduling features and categories that schedulers fit into. This was followed by an overview of various schedulers. These were divided into two groups, schedulers for wired networks and those for wireless networks. The wireless schedulers were further divided into those suitable for FDMA and TDMA, and the other group suitable for CDMA. Of particular interest in this discussion were the GPS-based schedulers, which form the basis of the fairness concept, which will be discussed in more detail in Chapter 4. The Largest Weighted Delay First scheduler was also mentioned, from which M-LWDF and our invention WF-LWDF were derived, which will be studied in great detail in the next two chapters.

Chapter 3

Simulation of CDMA-based wireless schedulers

3.1 Introduction

The aim of this chapter is to introduce the simulation setup used to compare how various schedulers perform in a *Code Division Multiple Access* (CDMA)-based environment and to present the results of the simulations. The simulation entailed measuring the throughput performance, delay performance, power utilization, and the packet dropping rate of six schedulers. The schedulers we chose are *Modified Largest Weighted Delay First* (M-LWDF), *Wireless Fair Largest Weighted Delay First* (WF-LWDF), *Largest Weighted Delay First* (LWDF) and *Earliest-Deadline-First* (EDF). The final two schedulers known as *Wireless Largest Weighted Delay First* (W-LWDF) and *Simplified Modified Largest Weighted Delay First* (SM-LWDF) were invented by the author with the purpose of comparing how the various factors in the priority function of M-LWDF affect its performance and what difference it makes if multiple packets are assigned different codes and transmitted simultaneously, as opposed to transmitting one packet at a time.

The problem with comparing these six algorithms is that EDF has been designed with a *Time Division Multiple Access* (TDMA) network in mind and has to be adapted for a

Code Division Multiple Access (CDMA) environment.

The rest of the chapter is structured as follows. It starts off with a discussion on the CDMA environment, which is followed by a discussion of the schedulers used in the simulations. The simulation framework is presented next, followed by a discussion of the actual simulation implementation, which includes the measurement techniques used. Finally, the simulation results are presented and discussed.

3.2 CDMA environment

The problem in a CDMA environment is how to distribute the available resources. In *Frequency Division Multiple Access* (FDMA) and *Time Division Multiple Access* (TDMA) networks, a constant amount of bandwidth is available, which can be neatly divided among the various users. In a TDMA environment, a typical scheduler would choose one of the sessions according to some priority scheme and assign it all the available bandwidth for a certain period of time, before choosing a new session.

Although the available bandwidth in CDMA networks is constant, the effective bandwidth per user constantly varies as the processing gain varies. Reference [1], for example, gives a nice summary of how to arrive at the expression for the bit-error-rate of a CDMA network, which can be manipulated to find the available bandwidth per user in a CDMA network:

3.2.1 Bandwidth Allocation [1]

The filter output of a conventional matched filter receiver for simultaneous demodulation of V asynchronous *Direct Sequence Code Division Multiple Access* (DS/CDMA) user signals can be represented as

$$y_k(m) = y_{k1}(m) + y_{k2}(m) + y_{k3}(m), \tag{3.1}$$

where $y_{k1}(m)$ represents the desired signal, $y_{k2}(m)$ is the multiple-access interference, and $y_{k3}(m)$ is the Gaussian noise term. The decision noise term, Y_k, can then be written as

$$Y_k = \sum_{m=1}^{G_p} y_{k1}(m) + y_{k2}(m) + y_{k3}(m). \tag{3.2}$$

Assuming that the number of chips per bit, also known as the processing gain, G_p, is large, the decision variable Y_k can be approximated according to the central limit theorem by a Gaussian random variable. The bit error probability is then given as

$$P_b = \operatorname{erfc}\left(\sqrt{\frac{[\mathrm{E}(Y_k)]^2}{\mathrm{Var}(Y_k)}}\right), \tag{3.3}$$

where erfc(-) is the complementary error function given by

$$\operatorname{erfc}(x) = \frac{1}{\sqrt{2\pi}} \int_0^{\infty} e^{-t^2/2} dt \tag{3.4}$$

and $\mathrm{E}[Y_k]$ is the mean and $\mathrm{Var}[Y_k]$ the variance of the decision variable Y_k. The mean of Y_k is given by

$$\mathrm{E}[Y_k] = \mathrm{E}\left[\sum_{m=1}^{G_p} y_{k1}(m)|b_k = 1\right] + \mathrm{E}\left[\sum_{m=1}^{G_p} y_{k2}(m)\right] + \mathrm{E}\left[\sum_{m=1}^{G_p} y_{k3}(m)\right]. \tag{3.5}$$

But

$$\mathrm{E}\left[\sum_{m=1}^{G_p} y_{k2}(m)\right] = \mathrm{E}\left[\sum_{m=1}^{G_p} y_{k3}(m)\right] = 0 \tag{3.6}$$

and therefore,

$$\mathrm{E}[Y_k] = \mathrm{E}\left[\sum_{m=1}^{G_p} y_{k1}(m)|b_k = 1\right] = G_p\sqrt{E_{ck}}. \tag{3.7}$$

The variance $\mathrm{Var}(Y_k)$ is given by

$$\mathrm{Var}(Y_k) = G_p \mathrm{Var}[y_{k1}(m)] + \mathrm{Var}[y_{k2}(m)] + \mathrm{Var}[y_{k3}(m)]. \tag{3.8}$$

The desired signal variance is

$$\mathrm{Var}[y_{k1}(m)] = 0. \tag{3.9}$$

The variance due to thermal Gaussian noise is

$$\mathrm{Var}[y_{k3}(m)] = \frac{N_0}{2}, \tag{3.10}$$

where N_0 is the one-sided thermal noise power spectral density. The variance of the interfering signals can be computed assuming that the interfering signal is modelled as white noise with the two-sided power spectral density of E_c/T_c.

Taking into account the relative phase differences between the desired signal and interfering signals and averaging over them, the bit error probability is lower bounded by

$$P_b \geq \text{erfc}\left[\sqrt{\frac{2E_{ck}G_p}{N_0 + \sum_{\substack{j=1 \\ j\neq k}}^{V} E_{cj}}}\right], \tag{3.11}$$

where $2E_{ck}G_p = 2E_b$ is the double bit energy and the denominator represents the total power spectral density coming for the thermal noise and multiple-access interference.

If all E_{cj} are equal, then the bit-error probability can be approximated by

$$P_b \approx \text{erfc}\left\{\left[\frac{V-1}{3G_p} + \frac{N_0}{3E_b}\right]^{-1/2}\right\} \tag{3.12}$$

Manipulating this we get

$$\frac{V-1}{3G_p} \approx \left(\frac{1}{\text{erf}^{-1}(P_b)}\right)^2 - \frac{N_0}{3E_b}, \tag{3.13}$$

which results in the expression

$$G_p \approx \frac{(V-1)E_b(\text{erf}(P_b))^2}{3E_b - N_0(\text{erf}(P_b))^2}. \tag{3.14}$$

The same process can be repeated when powers are not equal

$$P_b \geq \text{erfc}\left[\sqrt{\frac{2E_{ck}G_p}{N_0 + \sum_{\substack{j=1 \\ j\neq k}}^{V} E_{cj}}}\right], \tag{3.15}$$

with the result that

$$2E_{ck}G_p \leq \left(N_0 + \sum_{\substack{j=1 \\ j\neq k}}^{V} E_{cj}\right)(\text{erf}^{-1}(P_b))^2. \tag{3.16}$$

We now have the expression for the processing gain, which is dependent on the desired bit-error-rate. From this expression one can see that as the multi-user interference and the fading coefficients change, so the processing gain will change. But the processing gain is the link between the chipping rate B_c and the data rate R

$$B_c = G_p R \tag{3.17}$$

Note that the chipping rate is the total amount of bandwidth that the network requires, which can be held constant. The data rate is therefore constantly changing as the processing gain varies.

3.2.2 Power Allocation

Instead of bandwidth, a more natural resource to distribute in a CDMA network is power. Reference [70] explains how this is done. To start with, the total forward link power available to all data users is normalized to 1. If c_i is the normalized transmit power required per unit data rate (known as the channel attenuation function) and if user i is assigned rate R_i then the transmission power of user i is given by $P_i = c_i \cdot R_i$. Since the total forward link power has been normalized to 1, the following relationship has to hold:

$$\sum_i c_i R_i = \sum_i P_i \leq 1 \tag{3.18}$$

This relationship is useful, since it can be used to distribute the power among the various contending data sources. A scheduler in a TDMA environment would typically choose one packet and assign it all the available bandwidth. Examples are EDF-type schedulers. These schedulers can quite easily be introduced into the CDMA network, using (3.18). The GPS family of schedulers chooses a few packets and distributes the available bandwidth among them. Equation (3.18) is also relevant in this case. The following section explains in more detail how each scheduler uses (3.18) to multiplex the queued arrivals.

3.3 Schedulers Used

The problem with current wireless networks is that virtually none of them can give quality of service guarantees. The aim of most research on schedulers is to find a way to be able to give such guarantees. Throughput guarantees are of particular importance in any network. Emerging networks need to carry a wide range of data including real-time streaming applications, where delay guarantees are of even greater importance. Many schedulers try to meet delay deadlines by prioritizing data according to the queueing delay deadlines negotiated between a network and the various users. A delay-guarantee violation occurs when the packets of user i experience a steady-state delay D_i, which exceeds some delay threshold T_i. The value of T_i is one of the defining characteristics of the different data types and depends on the negotiation process initiated before any data are transmitted. QoS guarantees require that the probability of such violations must be smaller than some pre-fixed threshold δ_i. Mathematically, this can be summarized as:

$$\Pr\{D_i > T_i\} \leq \delta_i \tag{3.19}$$

3.3.1 Earliest-Deadline-First

[70] and [38] explain the basic principles behind the Earliest-Deadline-First scheduler. Every time a packet p arrives at a server e on its path, server e assigns packet p a deadline d_p^e. As soon as a packet completes service, all the power gets assigned to the packet p at the head of the queue with the smallest d_p^e.

In our simulations we chose the delay deadline T_i from (3.19) as the deadline for packet p, with the result that $d_p^e = T_i + t$, where t is the current system time. Once the packet with the smallest d_p^e has been chosen, a data rate needs to be assigned. To do this let us take another look at (3.18):

$$\sum_i c_i(t)R_i = \sum_i P_i(t) \leq 1. \tag{3.20}$$

Now, assuming that packet k gets picked, we know that it is fully transmitted before the next packet is chosen. This means that all the available power can be assigned to the single packet chosen, in other words, $R_i = 0$ for all $i \neq k$. We now look at the limiting case of (3.18):

$$\sum_i c_i(t)R_i = \sum_i P_i(t) = 1. \tag{3.21}$$

Since $R_i = 0, i \neq k$, we find that

$$R_k = \frac{1}{c_k(t)}. \tag{3.22}$$

For the sake of implementation, it is necessary to assign a packet a fixed data rate. In other words, a scheduler should pick an appropriate data rate for a specific packet. Once the data rate has been assigned to the packet, this information is transmitted to the destination, most probably on a separate dedicated control channel. The destination thereby knows what data rate to expect. While the packet is being transmitted, the transmission rate may not change. The fact that the assigned rate does not change in the transmission period can, of course, be a problem. If the channel attenuation factor increases and the rate is not decreased, then the bit-error-rate will increase. Rapid channel attenuation changes could therefore be a problem for a CDMA system of this kind.

Another aspect worth considering, is to have the scheduler choose data rates from a discrete set. This enables the scheduler to inform the destination of the intended data rate, using only a few bits of information. Another reason why a discrete set of possible data rates would be desirable, as opposed to a continuous one, is that a small set of data rates simplifies the design of the transmitter and receiver. The result of having to choose from some small set $\Re$ is that most packets are transmitted somewhere between 50% and 100% of the maximum rate they could be transmitted at, while still achieving a desirable bit-error-rate. On average, packets are therefore resistant to small increases in channel attenuation. One way to prevent massive re-transmission of packets, is to keep packet lengths as small as possible. This decreases the transmission time of packets, which gives the channels less time to change their attenuation factors $c_i(t)$.

3.3.2 Largest Weighted Delay First

In [70], Andrews explains that a Largest Weighted Delay First scheduler assigns all power to the user for which the value $a_i \times$(current delay) is the largest, where $a_i = -\log \delta_i / T_i$. Roughly speaking, a_i represents the required minimum rate of decay of user i's delay distribution. Thus a larger a_i reflects a more stringent QoS requirement for user i.

In terms of our simulations, the next packet is therefore chosen using

$$\text{Next Packet} = \max_i - \log(\delta_i)\frac{D_i}{T_i}. \tag{3.23}$$

This equation firstly looks for the packet that has experienced the largest steady-state delay D_i. But the packets of various sessions have different delay thresholds T_i. For example, e-mail packets may queue for a much longer period than is acceptable for streaming video packets. The steady-state delay D_i is accordingly normalized by the delay threshold T_i. If the delay ever exceeds the delay threshold, then a delay violation occurs. The rate at which these violations may occur varies between different data types and may be expressed by the violation probability constant δ_i. This constant cannot be used directly as a weight in (3.23) and the authors of [74] accordingly discovered that a logarithmic distribution would achieve the desired effect.

Once again, the chosen packet is assigned all the bandwidth using (3.22).

3.3.3 Modified Largest Weighted Delay First

The Modified Largest Weighted Delay First (M-LWDF) algorithm belongs to the LWDF family. Reference [74] explains that it operates somewhat differently from the three schedulers explained so far. Instead of picking just one packet, all the packets at the head of the queues are sorted in terms of priority. This is done by repeatedly using the following equation:

$$\text{Next Packet} = \max_i \frac{D_i(t)}{T_i}(-\log_2 \delta_i)\frac{\overline{c_i}(t)}{c_i(t)} \tag{3.24}$$

This equation is basically the same as (3.23), with two added factors. In a wireless environment the channel conditions are constantly changing and it is therefore necessary to monitor each channel's attenuation factor $c_i(t)$. The attenuation factor is a measure of how much transmission power will be required for the transmission to arrive at its destination at an optimum power level. To transmit data to users a far distance away requires more power, resulting in a larger $c_i(t)$. The scheduler should try to take advantage of any channels that temporarily experience above-average quality. This is achieved by dividing the resulting product in (3.24) by the attenuation factor, normalized by its short-term average, $\overline{c_i}(t)$.

Once a packet has been chosen, the next step is to assign it a transmission rate. To make the algorithm implementable, a set $\Re$ of possible transmission rates is used. In our simulations we chose $\Re$ to be

$$\Re = \{9600; 19200; 38400; 76800; 153600; 307200; 614400\} \text{ b/s.} \tag{3.25}$$

The M-LWDF algorithm chooses the largest possible data rate from set $\Re$ and assigns it to the chosen packet. If $R_i(t)$ is the assigned data rate then the product $c_i(t) \cdot R_i(t)$ represents the required transmission power. Note that the attenuation factor $c_i(t)$ is normalized by the total transmission power available to the mobile, with the result that if all the transmission power is used then the sum of all the $c_i(t) \cdot R_i(t)$ products is equal to one. As explained earlier, in order not to assign more power to packets, than a wireless node has available, the following relationship has to hold, where N_T are the total number of packets being transmitted:

$$\sum_{i=1}^{N_T} c_i(t) R_i(t) \leq 1 \tag{3.26}$$

If a mobile is only using part of its transmission power to send the chosen packet it may use the rest of the power to transmit a second or maybe even a third packet. The whole process can be written as:

$$\begin{aligned} R_1 &= R(1; c_1(t)), \\ R_i &= R\left(1 - \textstyle\sum_{j=1}^{i-1} R_j c_j; c_i(t)\right), \end{aligned} \tag{3.27}$$

where $R(p; c) = \max\{R \in \Re : c(t)R \leq p\}$. The question here is, what c_j should be.

In a realistic system it is desirable for packets not to change their assigned transmission rates. When the scheduler has decided what the transmission rate for packet i should be, the information can be transmitted to the destination node, before the actual transmission of the packet commences. Another consideration is that the share of the total power that a packet has received does not change during the transmission of the packet. But the relationship between the transmission rate and the transmission power is given by $P_i = c_i R_i$. c_i is therefore the channel attenuation measured at the start of transmission. If, half-way through the transmission of a packet, one would like to calculate the transmission power assigned to a packet, $P_i = c_i R_i$ should be used. This does not mean that the channel attenuation function $c_i(t)$ has not changed in the mean time. $c_i(t)$ changes constantly and if it drops below c_i, then the transmission power is not sufficient to sustain the desirable bit-error rate. It is therefore advisable in an actual implementation to choose the transmission power higher than is necessary. In order to determine how much power is still available after packets have been assigned power in previous rounds, which are still transmitting, one has to use the equation $P_{\text{available}} = 1 - \sum_i c_i R_i$. If $c_i(t)$ was used instead, then the equation $P_{\text{available}} = 1 - \sum_i c_i(t) R_i$ should be interpreted as the amount of power that would be used, if all i packets would start transmission at time t.

3.3.4 Wireless Fair Largest Weighted Delay First

The M-LWDF is a very attractive scheduling protocol for wireless networks, because it varies the transmission rates of packets, depending on the channel conditions. This is a rather elegant scheduler compared to other wireless schemes, which merely block a channel completely if the quality degrades below a certain level. The problem with the M-LWDF algorithm is its rate allocation protocol. The first packet picked will always receive virtually all the bandwidth. Other packets might experience only slightly less delay, require slightly more transmission power, yet have the same delay threshold and violation probability requirement, and still receive only a small share of bandwidth. The rate allocation system is therefore intrinsically not fair.

WF-LWDF—a new scheduler

Our solution to this problem was to replace the rate allocation scheme of M-LWDF with a GPS type approach and give the new algorithm the name *Wireless Fair Largest Weighted Delay First* (WF-LWDF). The packets would still be sorted using the original equation:

$$\text{Next Packet} = \max_i \frac{D_i(t)}{T_i}(-\log_2 \delta_i)\frac{\overline{c_i}(t)}{c_i(t)} \tag{3.28}$$

This equation can also be used to assign a rate, since it results in a value, which indicates the packet's priority. Packets with higher priorities should be transmitted at faster rates than packets with lower priorities. The priority ϕ_i can therefore be found by:

$$\phi_i = \frac{D_i(t)}{T_i}(-\log_2 \delta_i)\frac{\overline{c_i}(t)}{c_i(t)} \tag{3.29}$$

Once the priorities of all the waiting packets have been determined, the scheduler can distribute the available bandwidth B (in bits/s) in a GPS fashion, which should result in a fairer rate distribution:

$$R_i = B\frac{\phi_i}{\sum_j \phi_j} \tag{3.30}$$

Once the data rate R_i has been found, the next smaller value from set $\Re$ can be used. This means that packets always receive a slightly lower data rate than predicted by (3.30). The end result is that a whole lot of bandwidth is still left over, once all the packets have been assigned data rates. A good way to get around this problem is to hand out the bandwidth in reverse, in other words, assign data rates to the low priority packets first. If one recalculates the total available bandwidth, more and more left-over bandwidth becomes available as data rates from set $\Re$ are being assigned, which makes it possible for higher priority packets to receive a slightly larger transmission rate from set $\Re$ than the data rate they should have received according to (3.30).

The bandwidth of a CDMA system varies constantly depending on the chipping rate, which in turn depends on the required signal to noise ratio. We are therefore required to come up with an expression for the available bandwidth at any time. We would furthermore like to take the power limitations of a mobile unit into consideration, as was done with M-LWDF.

Taking the limiting case of (3.26) leaves us with:

$$\sum_{i=1}^{N_T} c_i(t)R_i = 1. \tag{3.31}$$

Now, when a packet has finished transmission, the bandwidth it was using and also the power at which it was being transmitted become available. Within 1ms, the scheduler notices this and distributes the newly available resources to waiting packets. Notice that only head-of-the-queue packets are considered for transmission, with the result that the scheduler will never transmit more packets simultaneously than the number of buffers in the system.

The problem is therefore that usually when the scheduler is distributing resources among waiting packets, a large percentage of the unit's power is being used for packets currently being transmitted. We would like to separate the power used by such packets from the power available for distribution.

$$\sum_{i \text{ inactive}} c_i(t)R_i = 1 - P, \tag{3.32}$$

where i are only head-of-queue packets, which currently have a rate of $R_i = 0$, while

$$P = \sum_{k \text{ active}} c_k R_k, \tag{3.33}$$

where c_k is the channel attenuation function of the destination link of packet k at the start of transmission, for the same reason as was discussed for M-LWDF. Here k are all packets, which are currently being transmitted, in other words $R_k > 0$.

Substituting (3.29) into (3.30), and the resulting (3.30) into (3.32) gives us:

$$\frac{\sum_i c_i(t)\frac{D_i(t)}{T_i}(-\log_2 \delta_i)\frac{\overline{c_i(t)}}{c_i(t)}B}{\sum_j \frac{D_j(t)}{T_j}(-\log_2 \delta_j)\frac{\overline{c_j(t)}}{c_j(t)}} = 1 - P \tag{3.34}$$

Once again, i and j are the head-of-queue packets, which at the beginning of the round are still waiting to be transmitted and have therefore got a current data rate of $R_i = R_j = 0$. Rearranging, we get:

$$B = (1 - P) \times \frac{\sum_j \frac{D_j(t)}{T_j}(-\log_2 \delta_j)\frac{\overline{c_j(t)}}{c_j(t)}}{\sum_i \frac{D_i(t)}{T_i}(-\log_2 \delta_i)\overline{c_i}(t)} \tag{3.35}$$

Substituting (3.35) into (3.30) results in the desired expression, which takes the variation of the available bandwidth and the power of the mobile unit into consideration:

$$R_i = (1 - P) \times \frac{\frac{D_i(t)}{T_i}(-\log_2 \delta_i)\frac{\overline{c_i(t)}}{c_i(t)}}{\sum_j \frac{D_j(t)}{T_j}(-\log_2 \delta_j)\overline{c_j}(t)} \tag{3.36}$$

3.3.5 Wireless Largest Weighted Delay First

Wireless Largest Weighted Delay First (W-LWDF) is an invention of ours, which is a member of the LWDF family. As with LWDF, the priority of packets is determined by comparing the delay experienced by packets, the delay deadline and the violation probability terms of each packet. But unlike LWDF, W-LWDF also takes channel conditions into account, making it similar to M-LWDF. The word *Wireless* in the name of the algorithm is an indication of this. The only difference between M-LWDF and W-LWDF is that instead of serving multiple packets in one scheduling turn, W-LWDF only serves one packet at a time and assigns it as much as possible of the available bandwidth. As with many of the previous algorithms, this is achieved using (3.22).

3.3.6 Simplified Modified Largest Weighted Delay First

We invented *Simplified Modified Largest Weighted Delay First* (SM-LWDF) as an attempt to see what would happen if one does not consider the average attenuation function $\overline{c_i}(t)$ while evaluating the priority of individual packets. The original expression used by M-LWDF to determine the priority of individual packets was

$$\text{Next Packet} = \max_i \frac{D_i(t)}{T_i}(-\log_2 \delta_i)\frac{\overline{c_i}(t)}{c_i(t)}. \tag{3.37}$$

In SM-LWDF we remove the average attenuation function $\overline{c_i}(t)$. The new packet picking algorithm therefore becomes

$$\text{Next Packet} = \max_i \frac{D_i(t)}{T_i}(-\log_2 \delta_i)\frac{1}{c_i(t)}. \tag{3.38}$$

The bandwidth assignment process remains the same as with M-LWDF.

3.4 Simulation Framework

To understand how the various simulation modules work together, let us first of all look at a general overview. The first thing that happens is that data are generated by a source and enter the network. The network usually consists of many nodes, through which the data have to travel. When data arrive at a node, a router or a switch decides where to send the data next. Once the router or switch has made its decision, a scheduler sorts the data into various categories and assigns each packet a data rate. This process is repeated at every node until the data reach their destination. This discussion simplifies the process a great deal, since many aspects like power control, *multiple access control* (MAC) considerations, and connection admission policies, have not been mentioned. But for this discussion this model will be sufficient.

Since the router decides where the various data sessions should be sent to next to reach their destination along the best path, the scheduler does not need to concern itself with multiple hops. The scheduling process is identical at every node and is only concerned with the class of traffic that packets belong to, the current queueing delay packets have experienced, and sometimes with the destination of packets. We therefore decided that our simulations would be sufficiently accurate if we only model the scheduling behaviour of one node in our simulations. One disadvantage with this approach is that in a multiple hop environment, the traffic statistics are changed by any queueing process that took place in previous nodes, before the data arrive at the node of interest.

Every node in a wireless network has three domains that need to be modelled. Firstly, data arrive from other nodes and are processed by the router. Since we look at the node in isolation, we ignore the previous nodes, which the data might have travelled through, and just simply model the process as if the arriving data were transmitted directly from the data sources. Once the router has determined the new destination for the various data streams, the information is queued in buffers. The second domain in our simulations is therefore to model the buffer structure. Finally the third domain are the wireless channels that need to be modelled.

3.4.1 Data Source Model

As already mentioned, the arrival statistics of data were simplified, by ignoring previous nodes that data might have travelled through and assuming that the data arrived via a single hop directly from the source. Transmission delays, channel fading and multi-user interference were also ignored during this hop. The channel between the data source and the node of interest should therefore be treated as a lossless, wired connection. The validity of such assumptions is based on the philosophy that the responsibility of successful data transmission from one node to the other lies solely with the source node. Apart from obvious error checking, the destination node has to assume that the source node successfully performed power control and that the arriving data are therefore error-free.

In our simulation the data source model had the responsibility of simulating the data generation of 15 different sources. Each one of the sources generates data with geometrically distributed random packet lengths, with a mean value of 1000 bits. The geometric distribution was truncated to give a maximum packet length of 10,000 bits and a minimum of 100 bits.

In the simulations it was decided that there would be three different types of data, making the total number of users, buffers, and channels per data type equal to five. The first five sources generate data with a delay deadline D_i of 50 ms, the second with a delay deadline of 25 ms, and the third with a delay deadline of 12 ms. The violation probability of all generated packets is $\delta_i = 0.01$.

In a real network, the arriving data would first be processed by the router, before the scheduler would get its turn. In our simulation model this is simulated by giving each generated packet a destination channel, through which it will be transmitted to its target node. Instead of giving each packet a random target, the concept of sessions was introduced. The packets from each source belonged to a session, which had a specific target channel. Sessions last 10000 cycles, which corresponds to 10s. When a new session starts, a new target channel is chosen for each data source using a flat random distribution.

Finally, the arrival rates have to be determined. We decided that the arrival rates should be Poisson distributed, resulting in exponential inter-arrival times. We therefore had to determine the average arrival rate. To do this the capacity of the system was estimated.

We once again used the limiting case of (3.18),

$$\sum_{i=1}^{N_T} c_i(t) R_i(t) = 1. \tag{3.39}$$

If we assume for a moment that the average arrival rate for all sessions is the same, then the data rate for each user i has to be

$$R_i(t) = \frac{1}{\sum_{i=1}^{N_T} c_i(t)}. \tag{3.40}$$

Now, the attenuation function $c_i(t)$ varies constantly. A slightly more stable capacity estimate can be obtained by using the short-term average $\overline{c_i}(t)$ instead

$$R_i(t) = \frac{1}{\sum_{i=1}^{N_T} \overline{c_i}(t)}. \tag{3.41}$$

This is a rough estimate of the current capacity per channel. We can use this to determine what the average creation rate $\overline{A_i}(t)$ for packets of session i should be:

$$\overline{A_i}(t) = \frac{L}{\sum_{i=1}^{N_T} \overline{c_i}(t)}, \tag{3.42}$$

where L is the percentage of traffic capacity that the network should be loaded with. For example, if L is bigger than 100% , the scheduler will definitely not be able to serve the arriving traffic fast enough and the buffers will overflow. This, of course, might happen at smaller loads as well, if suddenly an above average amount of traffic arrives.

Now that we have an expression for the average arrival rate for each session, this value can be used as the mean value for the Poisson distributed arrival process.

Table 3.1 summarizes the data source model, which can be used as a quick reference.

3.4.2 Buffer Model

As the data packets arrive at the node, they are randomly assigned a destination, which emulates the function of a router. The arrivals are then stored in 15 buffers, one for each data source. Since all the data from one data source belong to the same session, the queued

Table 3.1: Data Source Model for C++ simulator

Variable	Value	Other information
No of sources	15	
Packet length L_i	$\overline{L_i} = 1000$ b	geometrically distributed [100–10000]
Delay D_i, $i = 0 \ldots 4$	50 ms	
Delay D_i, $i = 5 \ldots 9$	25 ms	
Delay D_i, $i = 10 \ldots 14$	12 ms	
Violation probability δ_i	0.01	
Session length	10 s	
Creation rate $A_i(t)$	$\overline{A_i}(t) = \frac{L}{\sum_{i=1}^{N_T} \overline{c_i}(t)}$	Poisson distributed

material in every buffer not only has the same delay deadline and violation probability, but also the same target channel. All the packets in one of the buffers must therefore belong to the same priority level, except that the closer one gets to the head of the queue, the larger the queueing times. For all schedulers the head of queue packet must therefore have the highest priority out of all packets queued. This means that when picking a packet out of the 15 buffers, the scheduler just has to compare the head-of-queue packets from each buffer and individual buffers can be served in a *First-In-First-Out* (FIFO) basis.

Every one of the 15 buffers had a storage capacity of 1000 packets. If all 1000 places are occupied by packets, new arrivals are simply dropped until at least one packet has been served and a space opens up. There is no reason why specifically 1000 packet spaces were chosen, except to be able to accommodate sudden data bursts of reasonable size and compare the three traffic classes under high load conditions, where the buffers should preferably not overflow too fast.

3.4.3 Channel Model

To model radio channels with some precision, one needs to take some factors into account. In [74], for example, Andrews *et al.* take slow and fast fading into consideration. With only these constraints, if we assume that the mobile units do not move relative to one another throughout the simulation period, then only slow fading components will prevail and the attenuation will remain constant as time proceeds. This attenuation behaviour can therefore be expressed as the average attenuation factor of each channel. Table 3.2 contains a list of the various average attenuation factors used for each channel in [74]. The paper takes fast fading into account using a simple three-state Markov chain. In our simulations we found the amount of variation inadequate. We thus upgraded this model to a five-state Markov chain, illustrated in Fig. 3.1, hoping to achieve sufficient variation. The idea behind both of these models is that for most of the time each attenuation factor is equal to its mean value except for brief, random moments, when the attenuation factor either increases or decreases by 3dB. From these states it either returns to the mean value, or it can diverge from the mean value by another 3dB. The various numbers associated with arrows in the figure indicate the probabilities of such excursions taking place. In our simulation the new states of the channels were calculated at the beginning of a new scheduling session.

The appendix of [70] explains how the mean attenuation factors found in Table 3.2 were calculated. To start with, suppose that all base-stations transmit at maximum power at all times, which must be equal to 1. Let I_{ext} be the relative out-of-cell interference, which a user experiences. This is a random variable, which is distributed over all positions in the cell and over log-normal fading. An assumption is made that each base-station has a maximum transmit power of 2, but only half of this power is dedicated to data. The in-cell interference is also simply assumed to be equal to the in-cell received power. The target signal-to-interference ratio E_b/I_0 is assumed to be 7dB, and the system bandwidth

Table 3.2: Mean Attenuation Factors

i	c(t)	i	c(t)	i	c(t)
0	$2.508 \cdot 10^{-6}$	5	$2.924 \cdot 10^{-6}$	10	$5.229 \cdot 10^{-6}$
1	$2.518 \cdot 10^{-6}$	6	$3.623 \cdot 10^{-6}$	11	$6.482 \cdot 10^{-6}$
2	$2.518 \cdot 10^{-6}$	7	$4.142 \cdot 10^{-6}$	12	$6.635 \cdot 10^{-6}$
3	$2.598 \cdot 10^{-6}$	8	$4.307 \cdot 10^{-6}$	13	$7.257 \cdot 10^{-6}$
4	$2.771 \cdot 10^{-6}$	9	$4.533 \cdot 10^{-6}$	14	$7.395 \cdot 10^{-6}$

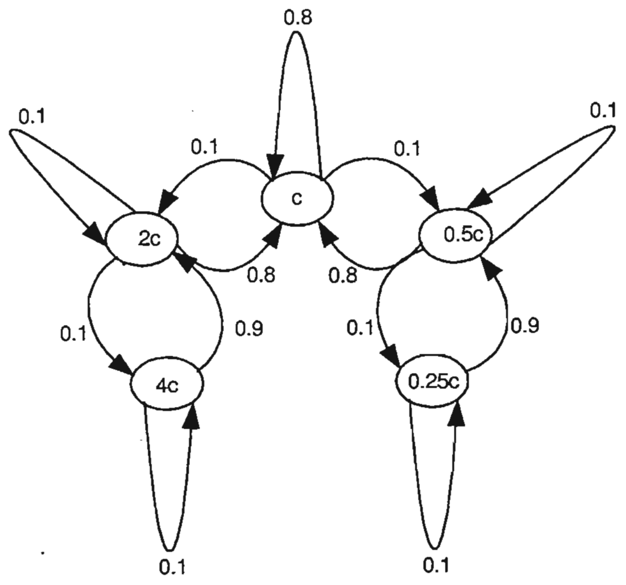

Figure 3.1: 5-state Markov chain

is 4MHz. The equation

$$\frac{E_b}{I_0} = \frac{\text{Bandwidth}}{\text{Transmit Rate}}\text{Signal-Noise Ratio}, \tag{3.43}$$

can then be shuffled to find the required signal-to-noise ratio:

$$c_i = 2 \times \frac{10^{0.7}}{4 \times 10^6}(1 + I_{ext}). \tag{3.44}$$

When we started the simulations, we soon noticed that even when the three-state Markov chain had been replaced by the five-state Markov chain, not enough variation was achieved. In the original three-state model, the channels can only vary between three different states and in most cases are in the centre state, which corresponds to the average channel attenuation factor. The problem that arises is that, for a large proportion of time, the order in which most of the sessions will be served does not change. In most cases, user number 0 will be served first, followed by user number 1, and so on. Only a small proportion of the channels will be in one of the other two states, thereby changing the order in which the sessions are served. The problem with this setup is that user number 0 will receive by far the most service, followed by user number 1. The result is that user number 14 will not receive the fair amount of service it deserves. Even with the five-state model, this problem was not eliviated.

We therefore decided to transform the discrete states of the two Markov models into a continuous range. The idea behind using a Markov model, is that the new output value is dependent on the last state that the chain was in. Important is also to note that the average attenuation factors range somewhere between 10^{-6} and 10^{-5}. The result is the following formula, which also depends on the previous state, yet has a continuous output:

$$c_i(t) = c_i(t-1) + \frac{r \cdot 10^{-6}}{100}, \tag{3.45}$$

where r is a continuous Gaussian random variable, with a mean value of 0 and a variance of 1. The attenuation factors for all sessions were initialized to a value of $c_i(0) = 4.0 \times 10^{-6}$ and were allowed to fluctuate between $c_i(t) = 1.0 \times 10^{-6}$ and $c_i(t) = 10.0 \times 10^{-6}$. The resulting excursions that take place can be seen in Fig. 3.2.

Note that this is a rather simpler version of the more complicated continuous equations used to model channel behaviour in power control research. An example can be found in [76], where the correlation coefficient is represented by the following model based on a Gauss-Markov process .

$$A_{ij}^{(n+1)} = \rho A_{ij}^{(n)} + \sqrt{1-\rho^2} W^{(n)}, \tag{3.46}$$

where $W^{(n)}$ is a Gaussian random variable with mean zero and variance σ^2. The sequence $\{W^{(n)}, n = 1, 2, 3, \ldots\}$ is white. ρ is called the correlation coefficient and is given by $\rho = \exp(-v\tau/D_0)$, where v is the speed at which a mobile travels, τ is the interval between power level adjustments, and D_0 is the correlation distance, which in urban areas is typically around 20m and in suburban areas may be ten times as large.

To calculate short-term average attenuation factors, the following equation was used

$$\overline{c_i}(t) = \sum_{k=0}^{99} c_i(t-k). \tag{3.47}$$

This is equivalent to a low-pass filter with 100 zeroes. The resulting excursions of $\overline{c_i}(t)$ can be seen in Fig. 3.3. When comparing the two Fig. 3.2 and Fig. 3.2 one can see the filtering action, since many of the finer bumps have been ironed out in Fig. 3.3.

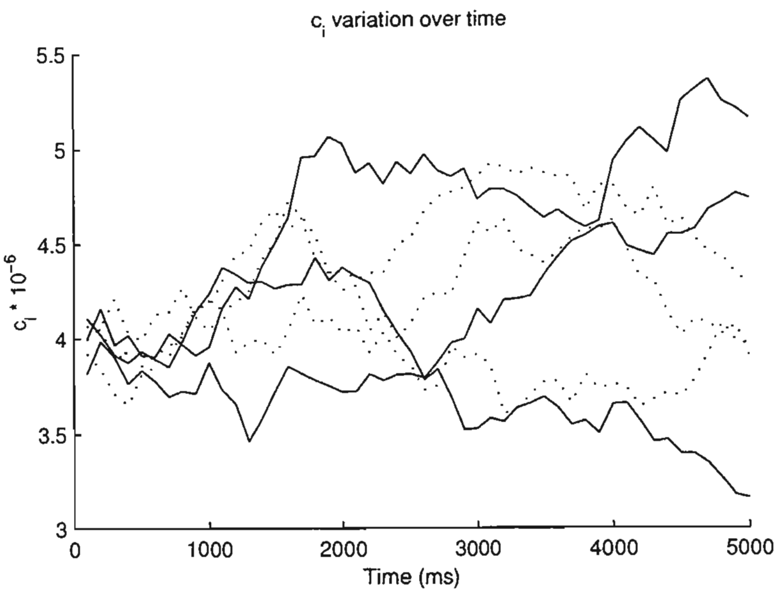

Figure 3.2: Plot of the attenuation functions for 15 different channels

3.5 Simulation Implementation

Now that the various simulation models have been explained, we can shift our focus onto the actual implementation of the algorithms and the extraction of some useful results.

The simulation was designed to run at a 1ms cycle length. This means that once every millisecond, the simulator would change the channel attenuation factors, check whether a packet had finished transmission and accordingly clear its buffer space, whether it was time to generate new packets, service the head-of queue packets if bandwidth became available, and finally update any resulting information that needs to be displayed. One cycle would of course not necessarily really take 1ms to implement, but rather represents 1ms, if the algorithm would be implemented in a real wireless network.

A 1ms cycle length is a realistic choice, since according to [76], the power levels of mobile units in the IS-95 CDMA system are adjusted every 1.25ms. Another consideration is that the maximum transmission rate out of set $\Re$ is 614400b/s. This means if the average packet length is 1000bits long and such a packet is transmitted at the maximum transmission rate, then it will take $t = \frac{1000}{614400} = 1.6$ms, which is longer than one cycle.

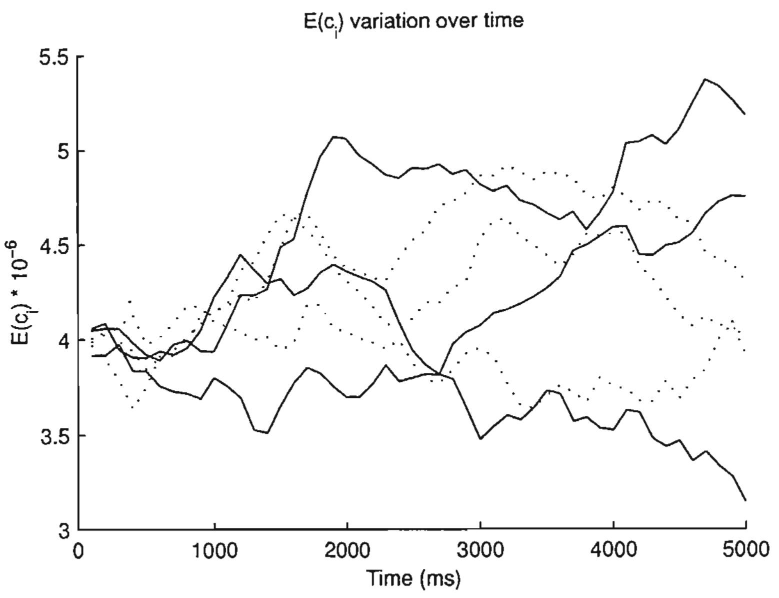

Figure 3.3: Plot of the mean attenuation functions for 15 different channels

During the run-time of the simulation, numerous measurements were made. Every time a packet was serviced, the number of bits that were transmitted, the amount of transmission power the packet required, and the queueing delay the packet experienced were added up. Another measurement that was performed was the number of packets that upon arrival at the node had to be dropped, since the queue was full. The following are more detailed discussions on what each of these measurements entailed.

3.5.1 Throughput

Every 10,000 cycles the bit count would be recorded. This corresponds to the number of bits that were served during a 10s interval. This information is useful when comparing the throughput of the various schedulers. This is especially interesting, since schedulers like M-LWDF and WF-LWDF take channel conditions into account before making a scheduling decision. Both of these algorithms also serve multiple packets at the same time, in contrast to schedulers like EDF. The simulator can therefore be used to determine how these two factors influence the throughput of the schedulers.

The total simulation time was 10,000s. A thousand 10s measurements were taken during this time, which were plotted on a 50 bin histogram. In classical statistics, Sturge's rule is quite often used

$$k = 1 + 3.3 \log_{10} n, \tag{3.48}$$

where n are the number of observations, which in our case are 1000. k are the recommended number of bins, which turn out to be 11. The number of bins we used are therefore quite high compared to Sturge's recommended value, but this large number gave us a good resolution, which upon proper normalization, we were able to use as a *probability density function* (pdf).

The graphs produced by the simulator were therefore histogram plots, where the x-axis consisted of the various bins and the y-axis consisted of the number of samples that fitted into the bin. To demonstrate how such a curve would be generated, imagine that after 10s a measurement was taken of how many bits had been served. If the number was, for example, 10^6b = 1Mb, then the relevant bin would be incremented by 1. This process would be repeated 1000 times for every group.

3.5.2 Delay

One of the most important QoS requirements is that schedulers with deterministic delay bounds ensure that packets do not exceed their delay deadlines, while schedulers with statistical delay bounds do not exceed their delay bound by a certain probability. Comparing the delay measurements of the schedulers under various load conditions gives a clear indication of how well the schedulers adhere to their QoS guarantees.

Ideally, delay measurements should be performed by plotting the delay of every packet on a histogram. The problem with this scenario is that with a simulation time of 10,000 seconds, the delay times of hundreds of thousands or even millions of packets have to be recorded for every queue, before they get plotted. This requires very large arrays for recording just the delay distributions.

In our simulations we have circumvented this problem, by adding up the delay of packets of one queue over a 10s period and afterwards dividing by the number of packets served.

The delay measurements are therefore average delay measurements over 10s. Over 10,000 seconds, 1000 samples were taken per queue and plotted on a 50bin histogram.

3.5.3 Power

The required transmission power of a packet can be found from a derivation of (3.18), which stated that

$$\sum_i c_i R_i = \sum_i P_i \leq 1. \tag{3.49}$$

From this we deduce that

$$P_i = c_i R_i. \tag{3.50}$$

Take note that c_i in this equation has to be the channel attenuation factor of channel i at the time when packet i was assigned its transmission rate. The reason for this is that the power assigned to a packet must be directly proportional to the transmission rate in a CDMA system, in order to achieve a desired bit-error-rate.

Measuring the power consumption of the various groups turned out slightly more complicated than was first envisaged. Initially, the idea was to measure a packet's power consumption as it starts transmission. But ideally one would like to determine how the total power available to the scheduler is distributed between the various queues at a moment in time. The power consumption of individual packets at any time is therefore relatively un-interesting.

A new idea was, therefore, to measure how much power each group with a common delay deadline was consuming at the moment that a packet was being transmitted and record this in the queue which the packet belonged to. To avoid the same problem of overly large arrays as with the delay measurement, the total power consumed per group could then be summed up over the 10s period and finally be divided by the number of packets served.

3.5.4 Loss Rate

Since the buffer length of the various queues is finite, the queues tend to overflow under high network load conditions. The reason for this is that the queues have to overflow if packets arrive faster than they are being served. Buffer overflows are not limited to network loads over 100%, but can happen at much lower loads, if the queues experience an above average burst of data.

As was the case with the other measurements, the number of packets lost over a 10,000 cycle period were recorded. This was repeated 1000 times and the results plotted on a 50 bin histogram.

3.6 Results

Up until now the simulation setup and the implementation have been discussed. We would now like to continue with the presentation of the simulation results and discuss the impact that they have on the field of wireless scheduling. In particular, we would like to compare the throughput, delay, power, and packet loss performance of our newly introduced scheme WF-LWDF against the performance of other schedulers. The most important comparison is between WF-LWDF and its parent algorithm M-LWDF, on which it was based and which it is meant to outperform.

The simulations that were performed, produced a large number of data. The problem was how to evaluate the various results. In our case we solved this problem by including a section in the C++ code of our simulator that would save the resultant histograms as 'm-files', that MATLAB would be able to read. Separate programs were then written in MATLAB to load the data, extract interesting results, and plot these. It was decided that three types of plots would be of particular use:

- Comparison between the three groups of data that are served by each scheduler at an 80% network load.

- Comparison between the *extreme* behaviour of the various schedulers over a range of loads.
- Comparison between the *average* behaviour of the various schedulers over a range of loads.

For each of the measurements performed we have therefore inspected each scheduler in isolation and compared the three data groups that it deals with. Furthermore, the largest histogram value from each scheduler was plotted over a wide range of load conditions. This is a good way to compare the extreme behaviour of the schedulers over a wide range of network loads. But the extreme values alone do not portray the whole picture. To get a feel for the behaviour of the schedulers it is useful also to plot the average values of throughput, delay, power consumption, and packet loss.

3.6.1 Throughput distribution

In this section we would like to compare the throughput of the various schedulers. Note that with throughput we mean the number of bits that a scheduler serves per second. A problem that was experienced was that the throughput was measured over 10s. The results were therefore divided by 10 before they were plotted. This gave the true number of bits served per second.

At this point we would like to remind the reader that the three data types that arrive at each node have different delay deadlines. The data packets stored in buffers 0 to 4 have a delay deadline of 50ms, the packets in buffers 5 to 9 have a delay deadline of 25ms, while the arrivals in buffers 10 to 14 have a deadline of 12ms. In order to guarantee quality of service, one would expect that a quality-of-service aware scheduler would serve twice as many packets from buffers 5 to 9 as from buffers 0 to 4. The packets queued in buffers 10 to 14, on the other hand, have the strictest delay requirements and these buffers would be expected to be served at twice the rate as buffers 5 to 9 and four times as much as buffers 0 to 4. This is a good measure of how well a scheduler adheres to its guarantees.

In order to be able to see how well the various schedulers follow this principle, the legend also includes the average values for each data group.

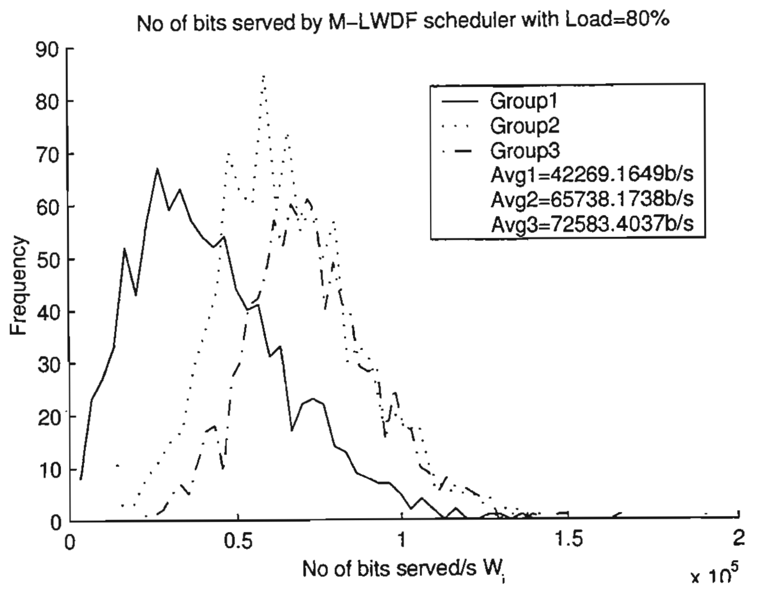

Figure 3.4: Throughput of M-LWDF scheduler

Fig. 3.4 contains the throughput curve for M-LWDF, while Fig. 3.5 contains the throughput curve for WF-LWDF. As was explained earlier in section 3.5.1, these curves were generated by taking 1000 measurements of the rate at which a user was served over 10s and plotting these on a histogram. As can be seen in the figures, the rate at which the data was served, $W_i(t)$, is time-varying. The y-axis therefore represents the frequency of the corresponding throughput value on the x-axis. Note that plotted throughput values have been divided by 10 to give the number of bits served per second and not per 10s.

The average throughput values in the legend of both of these figures seem at first somewhat disappointing. The ratio of the throughput of the three groups in each case was expected to be 1:2:4, as explained earlier. In both cases the resultant ratio is closer to 4:6:7.

If one compares the figures of M-LWDF and WF-LWDF with Fig. 3.6, which contains the resulting behaviour of the LWDF algorithm, one finds that LWDF is able to match the required 1:2:4 ratio quite closely, with its 2:4:7 ratio. The question remains why M-LWDF and WF-LWDF are unable to do the same. There are two major differences between LWDF and the other two algorithms. Firstly, both M-LWDF and WF-LWDF are channel-aware schedulers. They take channel conditions into account before making

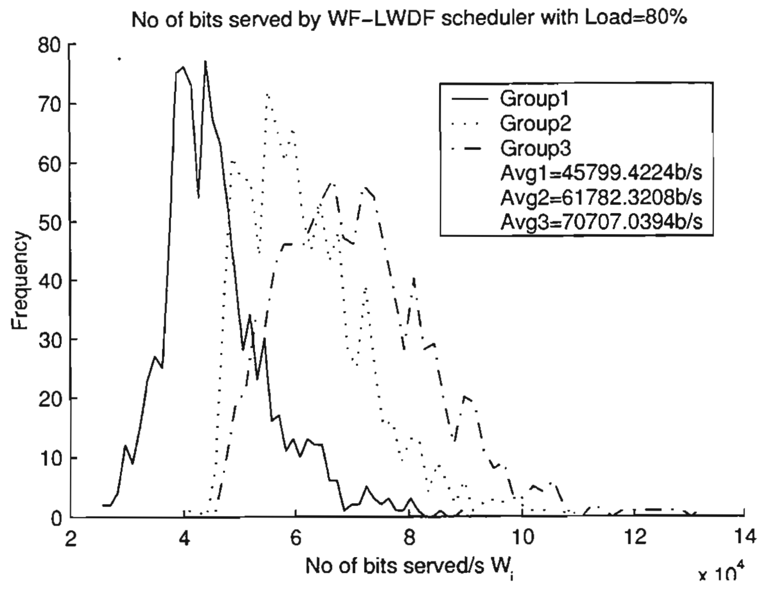

Figure 3.5: Throughput of WF-LWDF scheduler

a decision. The simulation setup allows the channel conditions to vary by a factor of 10. This can therefore alter the service share, which in turn would alter the service ratio. Each data group consists of 5 queues. One might expect that any distortions of this kind would be evened out, as the simulation proceeds. The truth is though that five buffers are not a large enough sample size to average out distortions completely, and we therefore attribute a part of the unexpected results to the channel awareness of these two schedulers.

The other difference between LWDF and the two channel-aware schedulers, is that both M-LWDF and WF-LWDF serve the packets from multiple queues at the same time. LWDF, on the other hand, chooses one packet per time and waits until this packet is finished, before choosing the next packet. Transmitting multiple packets at the same time enables M-LWDF and WF-LWDF to divide the available power resources better between the various packets and thereby utilize more power, which in turn increases the throughput. The problem is off course, how the power is distributed between the multiple packets. M-LWDF assigns most of the power to the highest priority packets, while the other packets that get chosen, receive the left over power. WF-LWDF attempts to distribute power in a fairer manner. In both cases the throughput distributions will be affected by the power distribution algorithms.

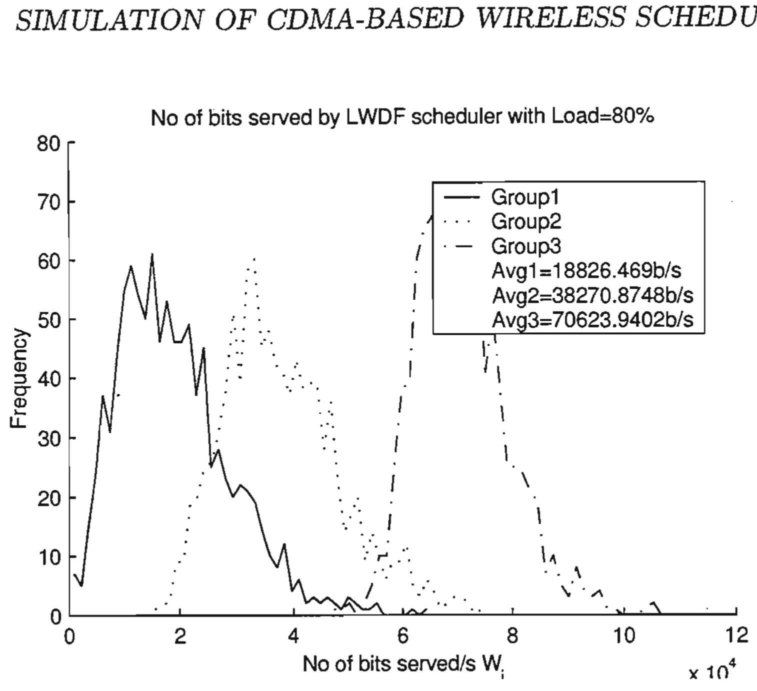

Figure 3.6: Throughput of LWDF scheduler

In general, one can conclude that the ratio 4:6:7 still follows the trend of 1:2:4, even though both M-LWDF and WF-LWDF have both been distorted in such a way, that the lower priority data group has received more service than it should, while the upper priority group is lagging behind in terms of the amount of service it should have received. The effect of channel awareness and serving multiple packets at the same time is therefore to average out the amount. of service received by the three data groups.

Fig. 3.7 contains the curves for SM-LWDF. Once again, SM-LWDF serves multiple packets and is channel aware. What is interesting is that the ratios of SM-LWDF are quite similar to those of WF-LWDF, which is closer to 1:1:1 than the ratio of M-LWDF. This is quite a surprising result, since SM-LWDF is more like M-LWDF than WF-LWDF. In the case of SM-LWDF, the reason for its distortion is that it does not normalize the channel attenuation function by its short-term average. The difference in channel attenuations are therefore more pronounced, which indicates that the channel-awareness property of M-LWDF, WF-LWDF, and SM-LWDF is most likely a major reason for the distortion of their ratios, away from the ideal 1:2:4 ratio.

The ratio of W-LWDF in Fig. 3.8, on the other hand, is around 2.7:4.3:6.9, which corresponds to 1:1.61:2.57, which is closer to LWDF than M-LWDF, WF-LWDF, which re-

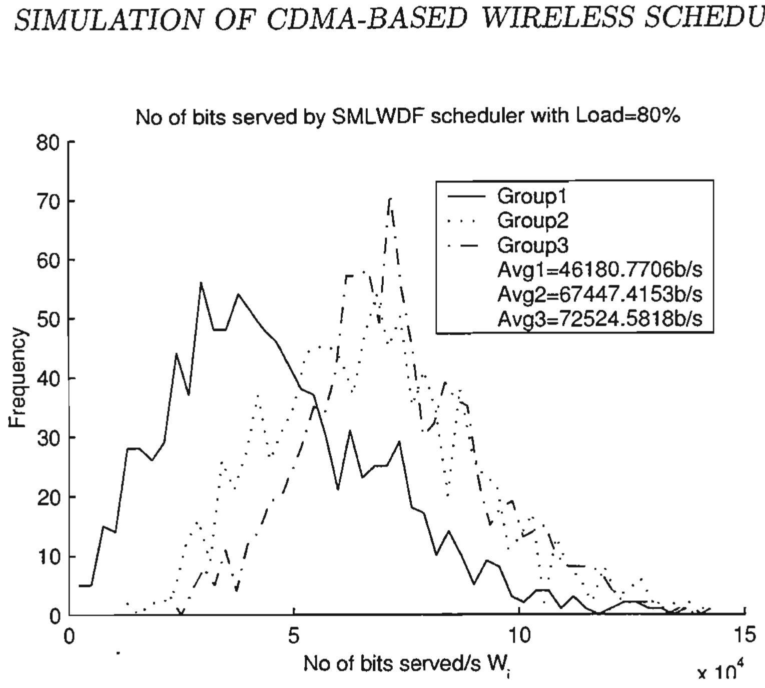

Figure 3.7: Throughput of SM-LWDF scheduler

spectively have corresponding ratios of 1:1.56:1.72 and 1:1.35:1.54, and SM-LWDF with its 1:1.46:1.57 ratio. What makes W-LWDF particularly interesting is that it is channel-aware like M-LWDF and WF-LWDF, but serves only one packet per time, like LWDF. W-LWDF is therefore a good indication, to what degree the channel-awareness of schedulers causes the distortion in their ratios, as opposed to transmitting multiple packets at the same time. The results indicate that, as expected, both factors play a major role, but that channel awareness distorts more than serving multiple channels.

Finally, the behaviour of EDF can be found in Fig. 3.9. This is quite an interesting scheduling scheme. A rather surprising characteristic of this figure is that the three curves of the different groups lie virtually exactly on top of one another. The reason why this result is surprising, is that EDF has been praised for being an optimal scheduler for bounded-delay services, in the sense that it provides the tightest delay guarantees of any scheduler [41]. But if EDF is able to adhere to very tight delay bounds, one would expect the ratio of the average throughput of the data groups to be directly proportional to their delay deadlines. As the figure indicates, this is not the case. Yet the fairness results in the next chapter will indicate that EDF is one of the fairest schedulers simulated in this dissertation. An explanation for this is that the standard deviation of the throughput

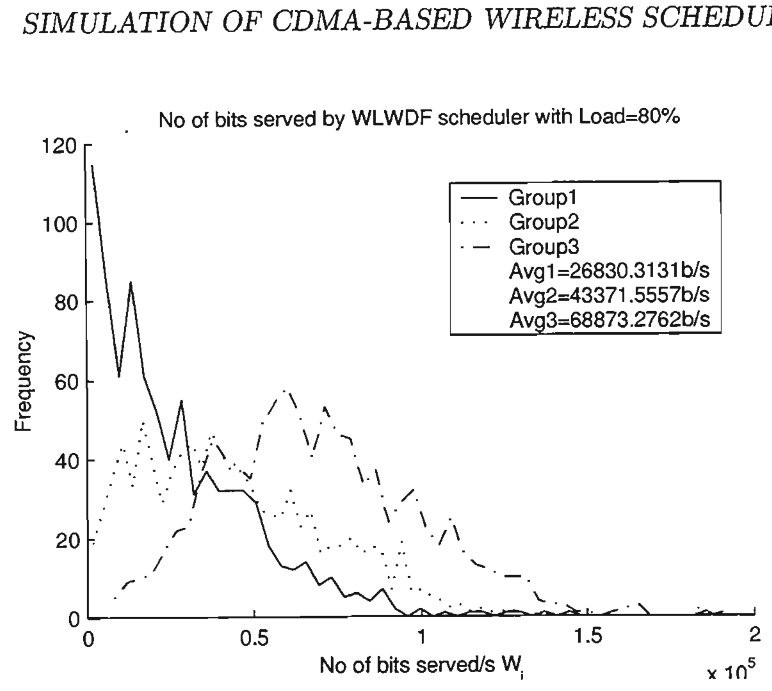

Figure 3.8: Throughput of W-LWDF scheduler

distributions of each group is very narrow. When measuring the fairness of a scheduler, as will be explained in the next chapter, the throughput of each queue is normalized by a factor, which includes the delay deadline. The various data groups are normalized by different amounts, with the result that curves that first lay on top of one another, will be separated during the normalization process. The normalized throughput of each queue of one group is then subtracted from the normalized throughput of each queue of each of the other data groups. Since EDF's distribution has quite a small standard deviation, the maximum difference between the various groups will be much smaller.

But the question remains, why does EDF not adhere to the delay deadlines, as it has been praised to do. The problem lies with the fact that the EDF algorithm treats all three data groups exactly the same. It achieves this by comparing the queueing delay of the head-of-queue packets. As soon as a packet from one of the queues has been served, then the other queues are lagging behind. The fact that the delay deadline of a packet is added to the delay experienced by that packet has no impact on the general behaviour of the traffic, especially over the 10s measurement period. The main reason for this is that at an 80% load, the delay experienced by packets exceeds deadlines of 50ms by such a degree, that the delay deadline component is negligible. The problem here is off course how EDF

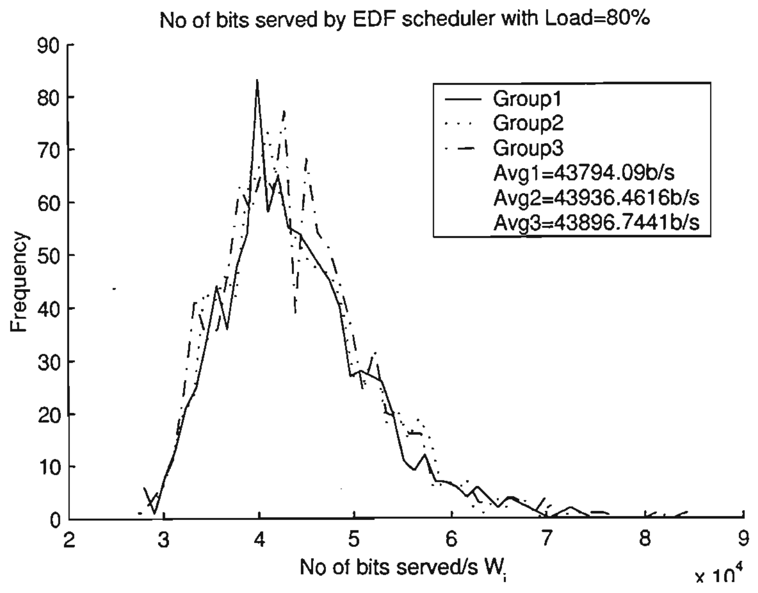

Figure 3.9: Throughput of EDF scheduler

is implemented. To make EDF an acceptable scheduler, the queueing delays should not exceed the delay deadline. The delay deadlines in our simulations would therefore have to be much larger than the current 12ms, 25ms and 50ms. Any packets that exceed their delay would have to be penalized in some way. One way of achieving this would be to simply drop any packets that have exceeded their deadline. Another approach is to penalize the source more and more, as the number of packets increase that do not meet their deadline. This is a form of call admission control and goes beyond the focus of this dissertation.

Fig. 3.10 summarizes the results mentioned above. The graph was plotted by taking the largest throughput of all the schedulers and plotting these for network loads ranging from 10% to 140%. We would like to remind the reader that the arrival rate is a Poisson distribution, where the mean was given by (3.42) in section 3.4.1, which was

$$\overline{A_i}(t) = \frac{L}{\sum_{i=1}^{N_T} \overline{c_i}(t)}. \tag{3.51}$$

By simply varying the value of L we were able to vary the load. This was repeated for values starting from 10% right up to 140%, in increments of 10%. Fig. 3.10 therefore represents a bound on the throughput of the various schedulers under a wide range of loads, where the $x-axis$ shows the various loads, while the y-axis shows the maximum

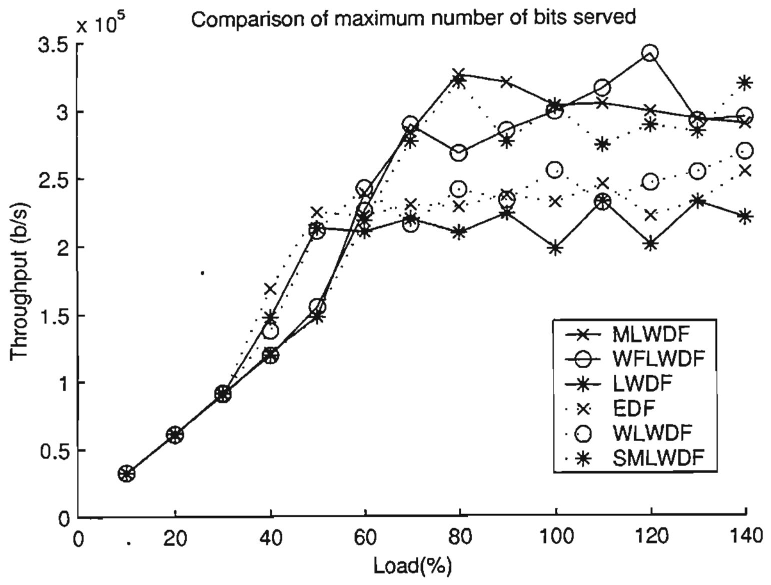

Figure 3.10: Throughput bound for the various schedulers

throughput that the various schedulers achieved at the specific loads.

If one compares the resultant maximum throughput of the various schedulers, one finds that at loads higher than 60% M-LWDF, WF-LWDF, and SM-LWDF have the highest throughput with an average throughput of approximately 300Kb/s. W-LWDF, EDF, and LWDF form a second group, with slightly lower throughput values of around 220Kb/s. This clearly shows the advantage of serving multiple packets in one scheduling round and thereby distributing the power more accurately.

This separation of the two groups of schedulers is even more apparent in Fig. 3.11, which in this case shows the average throughput of each of the schedulers. Here the x-axis represents the various loads that the network had to cope with, while the y-axis shows the average throughput that each of the schedulers achieved for the corresponding load. As can be seen, the multi-packet algorithms have an average throughput of 180Kb/s, while the throughput of the single-packet algorithms is slightly lower at 130Kb/s.

One useful thing to calculate is the maximum number of bits that could be transmitted under the best possible channel conditions. To do this we start with the power limit that

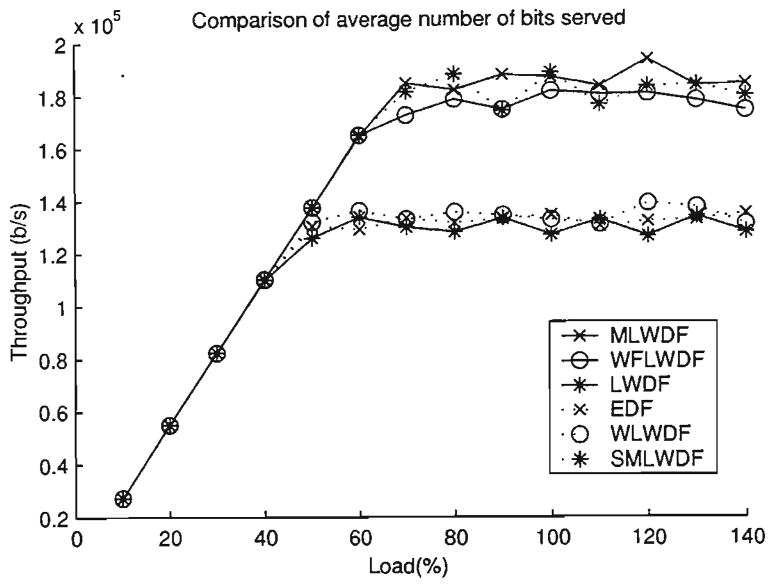

Figure 3.11: Average throughput for the various schedulers

a scheduler has

$$\sum_i R_i c_i \leq 1. \tag{3.52}$$

Many of the schedulers choose only one packet, with the result that the other packets have to wait before they are transmitted, in other words, $R_j = 0$, for all $j \neq i$. Therefore,

$$\sum_i R_i c_i = R_i c_i \leq 1. \tag{3.53}$$

The maximum possible transmission rate $R_{\max}$ occurs, when a packet is transmitted into a channel that experiences the smallest channel attenuation $c_{\min}$, which in our simulations was truncated to 10^{-6}, with the result that

$$R_{\max} \leq \frac{1}{c_{\min}} = 1\text{Mb/s}. \tag{3.54}$$

This expression can then be used to find the maximum number of bits $W_{\max}$ that can be transmitted in a certain period of time T, which in our case we chose to be 1s.

$$W_{\max} = R_{\max} \cdot T \leq \frac{1}{10^{-6}} = 1\text{Mbits} \tag{3.55}$$

It is quite interesting to notice that the schedulers reach about a quarter of this value, as is evident in Fig. 3.10. The first reason for this is that the channel attenuation factors are of course a lot higher than the minimum value $c_{\min}$. If, for example $c_i = 5 \cdot 10^{-6}$ was chosen, which is located half-way between the maximum and minimum values of c_i, then

$$W = R \cdot T \leq \frac{T}{c_i} = \frac{1}{5 \cdot 10^{-6}} = 200\text{Kb/s}. \tag{3.56}$$

This is much closer to the average number of packets served, in Fig. 3.11.

Another important reason why the maximum theoretical throughput is much higher than the corresponding simulation results in Fig. 3.10 is that the schedulers can only transmit packets at a rate chosen from set $\Re$. Especially schedulers that only serve one packet at a time will on average have to assign transmission rates that are much lower than the maximum rate they could have assigned. This factor is very important during an actual hardware implementation of a scheduling algorithm, where the choice of transmission rates is not continuous, but could be quite limited. Transmitting multiple packets at the same time, as M-LWDF, WF-LWDF, and SM-LWDF do, is a way to get around this problem to a certain degree and to utilize the available power more efficiently.

3.6.2 Delay

One of the most important aspects of a scheduler is how long packets have to queue before the scheduler picks them. Modern networks require QoS, which in our case consists of delay deadlines, violation probabilities, and fair resource distribution. The queueing delay that packets experience in a good scheduler should be proportional to the packet's delay deadline. In our simulations this translates to the scenario that packets queued in buffers 5 to 9 should experience half the delay that packets in buffers 0 to 4 should experience, while packets in buffers 10 to 14 should experience a quarter of the delay of packets in buffers 0 to 4. Please note that even though the delay deadlines of packets in buffers 0 to 4 are 50ms, those in buffers 5 to 9 are 25ms, and those in buffers 10 to 14 are 12ms, this does not mean that these deadlines have to be reached. Rather, the ratio of the deadlines experienced by each buffer should be proportional to these delays. The reason for this is that the scheduler itself is not able to control the rate at which data packets arrive. But if a large number of packets arrive and the scheduler is not able to serve them fast enough, then their delay will grow. The scheduler is not able to stop the delay from growing. This would be the task of a call admission policy. Instead the scheduler is only able to ensure that the delay that the packets experience is proportional to their delay deadlines. This is an important concept, since it explains why EDF is not able to meet its QoS guarantees. In order for EDF to be an effective scheduler, the delays packets experience need to be less than the delay deadlines.

Note that in the simulation graphs presented in this section a load of 80% was chosen, which is quite a high load for a network to operate at. The result is that the maximum delays measured were very high, in the order of minutes. In a real network a call admission control algorithm would try to ensure that such high load conditions are not reached and the delays experienced by packets would therefore be much lower. The main reason for choosing such high load conditions is to flood the buffers. An important simulation measurement is the fairness of the schedulers, which will be discussed in the next chapter. In order to perform a fairness measurement, the buffers may never be idle. If the buffers are idle at any stage, than a comparison of the normalized throughput of the various queues becomes meaningless and the fairness measurement is therefore invalid. Much of the literature on fairness mentions this requirement. Examples are references [20], [24], and [57]. Since the buffers have to be flooded for fairness measurements, we decided to do the same for all the simulation measurements. If we had not done this, a comparison between the various measurement metrics of the schedulers would have become more and more difficult and meaningless, as the variation in simulation conditions increases.

Another reason for choosing such high load conditions is that the server is unable to serve data fast enough, and the true characteristics of the scheduler become apparent. Under low load conditions, the resources are not fully utilized, with the result that packets receive more service than they deserve. QoS requirements are therefore not only met, but exceeded. But it is under high load conditions that the scheduler struggles to meet the QoS requirements. It was therefore decided to perform all performance comparisons between the three groups of data at high load conditions. The schedulers performance as a whole would then, in contrast, be considered at a large range of loads.

A further benefit of measuring the throughput and the delay performance of the various schedulers at high load conditions is that the measured results are an indication of how large the maximum throughput and delay values can be. An indication is therefore given what delay values could be expected in a worst-case scenario.

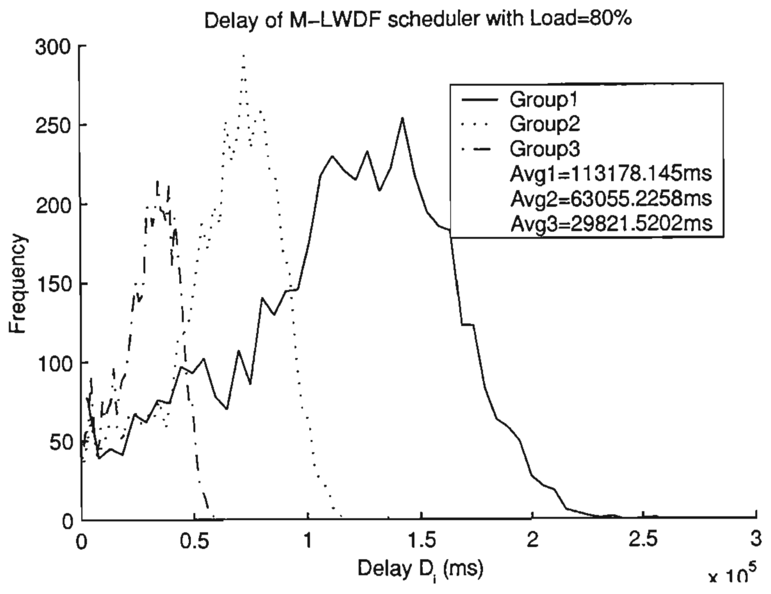

Figure 3.12: Delay of M-LWDF scheduler

Fig. 3.12 and Fig. 3.13 contain the delay distributions of the three data groups in a M-LWDF and a WF-LWDF scheduler, respectively. As mentioned in section 3.5.2, the graphs represent a 50-bin histogram of the delay distribution. They were constructed by incrementing one of the bins for every measurement taken. The x-axis represents the range of delay measurements taken, which consists of 50 bins. The y-axis represents how many times a delay measurement fell into the appropriate bin.

If these schedulers met their QoS requirements perfectly, it would be expected that their delay measurements would be proportional to their delay deadlines. The delay deadline for Group1 is 50ms, for Group2 is 25ms, and for Group3 is 12ms. The expected average delays for each of the three groups would therefore be expected to have a ratio of 4:2:1. For M-LWDF in Fig. 3.12, the average delays experienced by Group1, Group2, and Group3 type packets were respectively 113s, 63s, and 29s. This corresponds to a ratio of 3.9:2.2:1, which is quite close to the required 4:2:1 ratio. Unfortunately, WF-LWDF does not differentiate

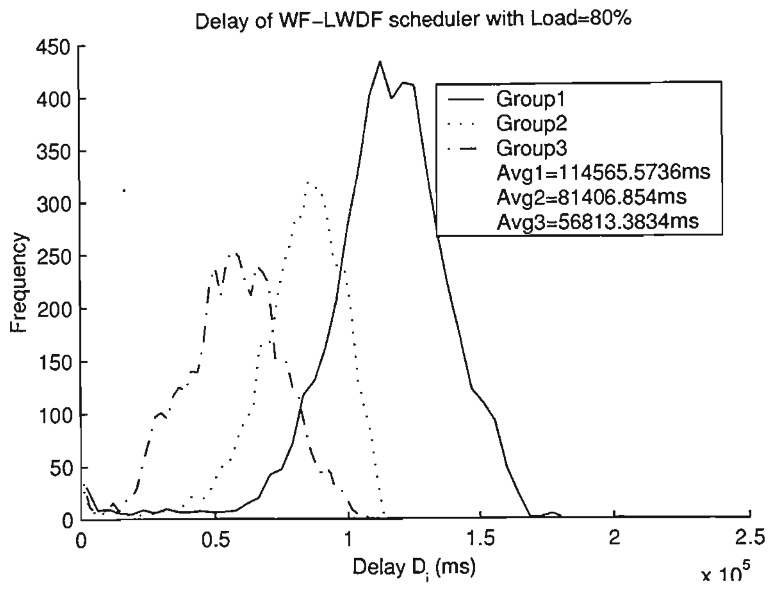

Figure 3.13: Delay of WF-LWDF scheduler

its data quite as effectively. With a delay of 115s for Group1, 81s for Group2, and 57s for Group3, the resulting ratio is 2:1.4:1, which is rather disappointing. On the other hand, the maximum delay experienced by packets in anetwork with a WF-LWDF scheduler is 180s, while in the M-LWDF scheduler this is over 250s, which indicates that for an 80% load, the packets in a WF-LWDF scheduler experience less delay than in a M-LWDF scheduler.

Notice that delays in the order of 100s are very large, but as explained before, originate from the large load conditions. Under normal network loads, where the buffers are not overflowing, the figures would be much smaller. In the case of these results, one furthermore has to take into account the capacity of the buffers, which we chose to be 1000 packets long. The average length of packets was chosen to be 1000 bits long. This is not quite a true reflection, since the packet length was truncated at a minimum of 100 bits and a maximum of 10000 bits. Let us therefore for arguments sake decide that the average packet length is 2000 bits long. This means that when all 15 buffers are overflowing, approximately 30Mb of information is waiting for the scheduler to deal with. Now from the results in the previous section we know that the approximate average throughput of 150Kb/s can be expected. This means that it will take the scheduler 200s to clear the

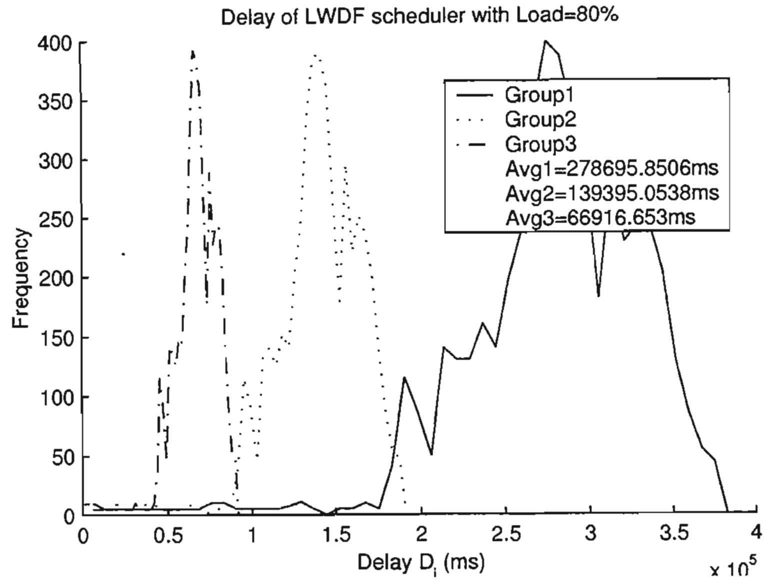

Figure 3.14: Delay of LWDF scheduler

buffers. In order therefore to achieve delays of smaller values, the packet length and the buffer capacity can be decreased. Furthermore, the power capacity of the system can be increased, which would make higher data rates possible. Finally, the network load would have to be held in a check by a good call admission control algorithm.

The queueing delay of packets belong to the three QoS groups in a LWDF scheduler can be found in Fig. 3.14. The average delay of Group1 packets was 279s, of Group2 packets was 139s, while Group3 packets on average queued for 67s. The resulting ratio between the groups is 4.2:2.1:1, which is quite impressive. As expected, the queueing delay of LWDF, which serves one packet at a time, was much higher than that of M-LWDF and WF-LWDF, which serve multiple packets at a time, and are therefore able to utilize the available power more accurately. The maximum queueing delay for LWDF packets turned out to be around 380s, compared to 250s and 180s for M-LWDF and WF-LWDF, respectively.

The queueing delays of SM-LWDF can be found in Fig. 3.15. For Group1, Group2, and Group3 data they were 111s, 58s, and 27s, respectively. This corresponds to a ratio of 4.1:2.1:1, which is similar to M-LWDF and LWDF. Surprisingly, the maximum delay is somewhere around 380s, which is similar to LWDF, eventhough SM-LWDF serves multiple packets at the same time. The average queueing times are similar to those of M-LWDF

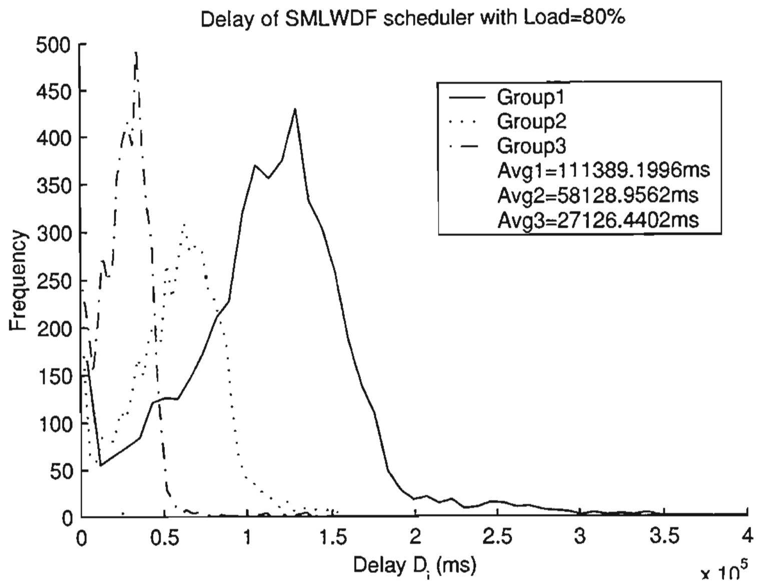

Figure 3.15: Delay of SM-LWDF scheduler

and WF-LWDF though, and not at all like LWDF. The reason for SM-LWDF's strange results are the tails of the three groups, which are caused by SM-LWDF not normalizing its packet-picking algorithm by the short-term average of the attenuation function. The result is that packets with destinations that are geographically close by, receive preferential treatment over those far away. Accordingly, the few packets experiencing the highest channel attenuation, will have uncharacteristically high queueing delays.

The delays of W-LWDF in Fig. 3.16 prove how channel-awareness aids schedulers. The average delays of all three groups are lower than their respective LWDF partners, even though both algorithms serve only one packet at a time. Interestingly enough, the high priority Group3, with the most stringent delay requirements, has a delay value of 72s, which is approximately the same delay value as that of LWDF. But Group2 and especially Group1 have a much better queueing delay than LWDF, with 117s and 152s, respectively. The result is a ratio of 2.1:1.6:1. In other words, the lower priority groups receive more service than they are meant to. In the end, Group1 pays for its preferential service with a very heavy tail, which gives it the same maximum queueing delay as LWDF, even slightly higher. To summarize, the channel-awareness of W-LWDF gives its low-priority queues an advantage over non-channel-aware schedulers. But this extra service does not bound

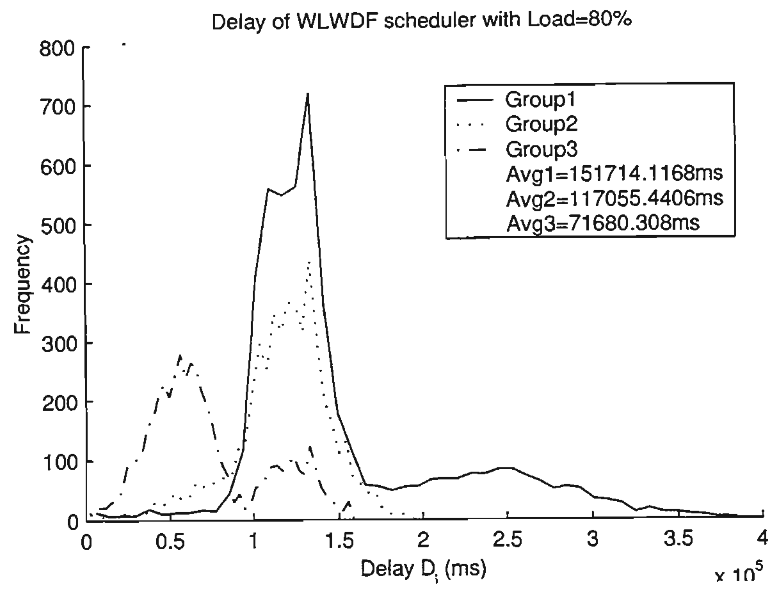

Figure 3.16: Delay of W-LWDF scheduler

the delay, with the result that W-LWDF still has the same maximum queueing delay as LWDF.

The delay distributions of the three data groups of EDF in Fig. 3.17 lie exactly on top of one another. In other words, all three groups are treated exactly the same and EDF is therefore not achieving the desired delay ratio. As was discussed earlier, the reason for this behaviour is that the queueing delays that packets are experience are much larger than the delay deadline. To demonstrate this, the average queueing delay of a packet of either of the groups is 121s. The scheduler chooses the next packet by adding the delay deadline of the packet to the current delay experienced. For Group1 this means adding 50ms to 121s, which makes hardly any difference, yet it is the largest delay deadline. The result is that only the queueing delay is important when picking the next packet and all queues are treated exactly the same, regardless of their delay deadline. The only time the delay deadline will start having an effect, is when the average queueing delay and the delay deadlines are of a.similar magnitude.

The worst-case delay of all the schedulers over a wide range of loads can be seen in Fig. 3.18. What is very encouraging in this diagram is that, apart from EDF, WF-LWDF has the best delay bound, while the worst-case delay of M-LWDF is almost twice as high

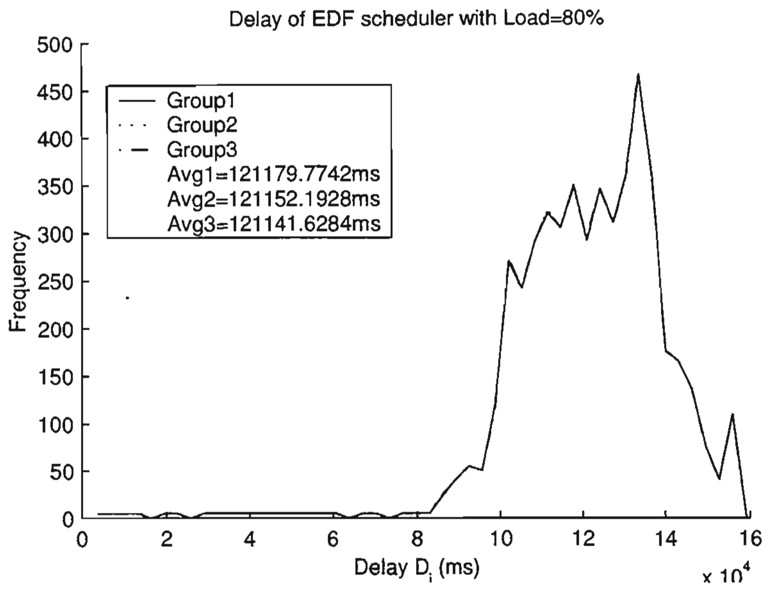

Figure 3.17: Delay of EDF scheduler

for high-load conditions. WF-LWDF and especially EDF are able to have low maximum delays, because their curves are smooth, without any distortions, which would cause long tails. M-LWDF follows with a slightly worse maximum delay, being slightly more distorted than the curves of EDF and WF-LWDF. LWDF, W-LWDF, and SM-LWDF all behave quite similarly, with all of them having either a long tail, or in the case of LWDF low throughput.

The average delay values of the six schedulers can be seen in Fig 3.19. The advantage of serving multiple packets simultaneously is clearly visible, with M-LWDF, SM-LWDF, and WF-LWDF all behavingly in a very similar fashion. W-LWDF and EDF follow, with slightly larger average delay values. Packets in a LWDF server experience by far the worst average delay, which is a result of the low throughput that became apparent in the last section.

Another interesting thing to notice about Figs. 3.18 and 3.19 is the slope of the various curves. Note that at loads higher than 60% the buffers of most schedulers are filling up and starting to overflow. This can be seen from the sudden change in slope. The slope of the queueing delay in both figures rises between 40% and 80%. These are the load conditions when the queues of the scheduler start filling up and congestion starts occurring. At smaller loads, virtually no delay takes place, while at loads higher than 80% the curves flatten out, indicating that the queues are completely full, resulting in no further increase in delay, despite any further increases in traffic arrivals.

To work out the maximum possible delay a packet can experience, while the packet ahead of it is being transmitted, we assume that the maximum packet length is $L_{\max}$, which in our case is 10,000bits, and the minimum transmission rate is 9600b/s. We can then find the maximum possible delay

$$D_{max} = \frac{L_{\max}}{R_{\min}} \times (\text{No of buffer spaces}) = \frac{10000}{9600} \times 1000 \approx 1000s. \tag{3.57}$$

Fig. 3.18 shows that none of the schedulers reach even half that value.

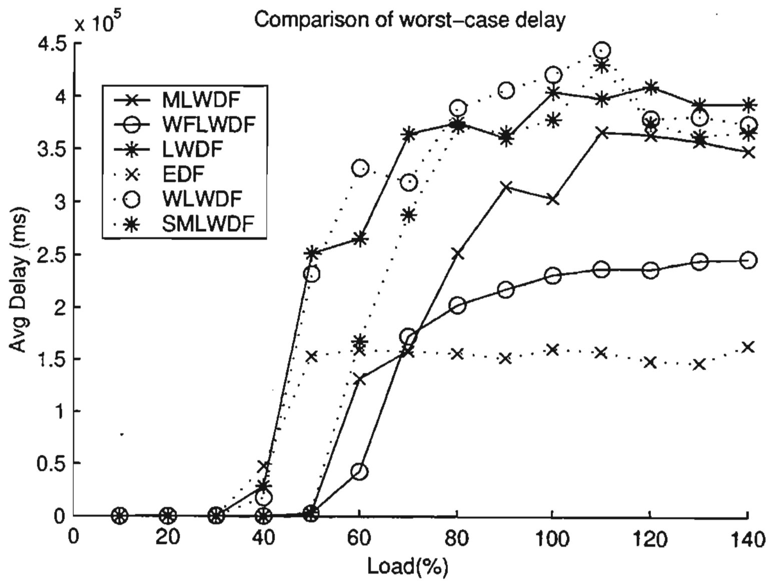

Figure 3.18: Delay bound comparison of various schedulers

3.6.3 Power

In the modern world, mobile communication devices in the form of cellular telephones and wireless computer networks are rapidly becoming a business necessity. All mobile devices have one thing in common—they get their power from batteries, with limited transmission power. The limited battery capacity is often perceived as the major problem with wireless systems, which means that the battery eventually runs out of power. But in

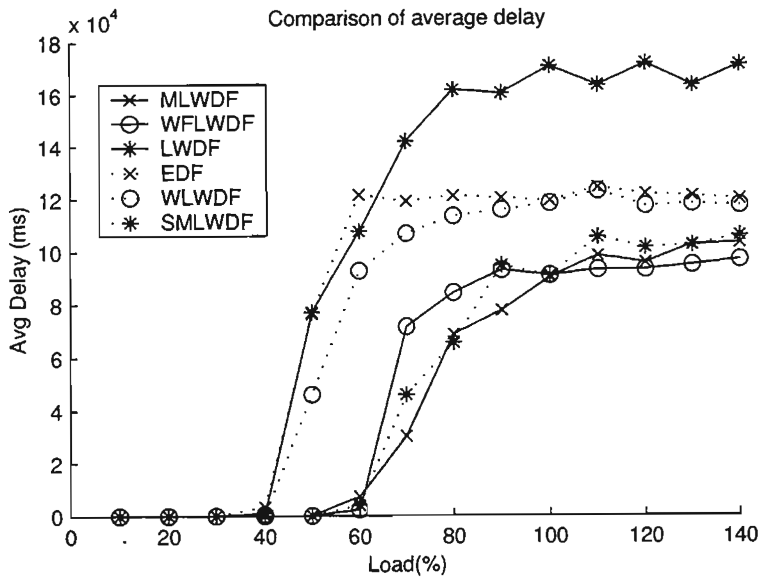

Figure 3.19: Comparison of average delay of various schedulers

reality, these devices can usually be charged at regular intervals. A much bigger problem with the limited power capacity of these devices is how to use the available power to its full capacity. High transmission rates are desirable. The problem is that the higher the transmission rate, the more transmission power is required in a CDMA system. In this section we would therefore like to measure how much of the total available power is being used for transmissions. The best scheduler will use all the power to achieve the highest possible throughput.

The power utilization of the individual groups of each scheduler is not of much interest. Instead we would like to consider the maximum and average utilization curves of the various schedulers. Figs. 3.20 and 3.21 have been included to show how schedulers transmitting multiple packets at the same time are able to utilize the available power much better than schedulers transmitting only one packet at a time. In both cases two schedulers have been chosen from each group to demonstrate this aspect. Notice how in Fig. 3.20, both M-LWDF and WF-LWDF are able to utilize between 96% and 98% of the available power, compared to LWDF and EDF, which use at most around 80%.

The average power consumption in Fig. 3.21 demonstrates the same principle, with M-LWDF and WF-LWDF using around 95% of the available power, while LWDF and EDF

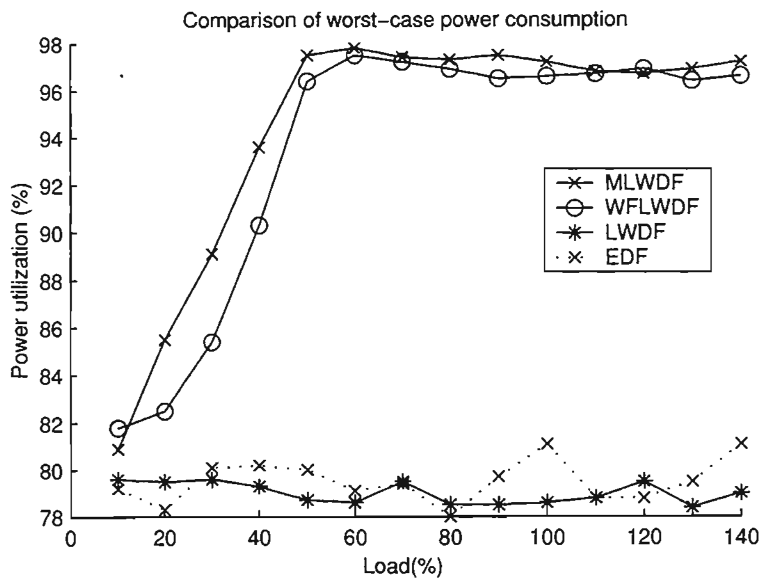

Figure 3.20: Power bound comparison of various schedulers

use only 70%. The behaviour of M-LWDF deserves further discussion. Its power utilization peaks at a load of 60%.. As the load increases further, the power consumption decreases, until it levels off between 85% and 90% at loads over 100%. An explanation of this behaviour is that at a load of 60%, enough data arrives to fully utilize most of the available power. But M-LWDF has enough power available to keep the number of packets queued to a minimum. In other words, at a 60% load, packets queued tend to experience very little delay before they get transmitted. When choosing a packet, M-LWDF will therefore choose a packet that has above-average channel conditions.

As the load increases, so the average queueing delay of packets increase. When choosing a packet, the queueing delay becomes a more dominant feature than the channel conditions, and the M-LWDF scheduler will start making decisions based more on how long packets have been queueing than how good their channel conditions are. But the same thing happens to WF-LWDF, yet it keeps its power utilization at 95%, whereas M-LWDF's power utilization falls by 10% as the load increases beyond 100%. The difference lies in the power distribution algorithms of the two schedulers.

M-LWDF will always choose the largest possible data rate from set $\Re$ that the available power permits, and assign this rate to the first packet it has chosen. The next packet will

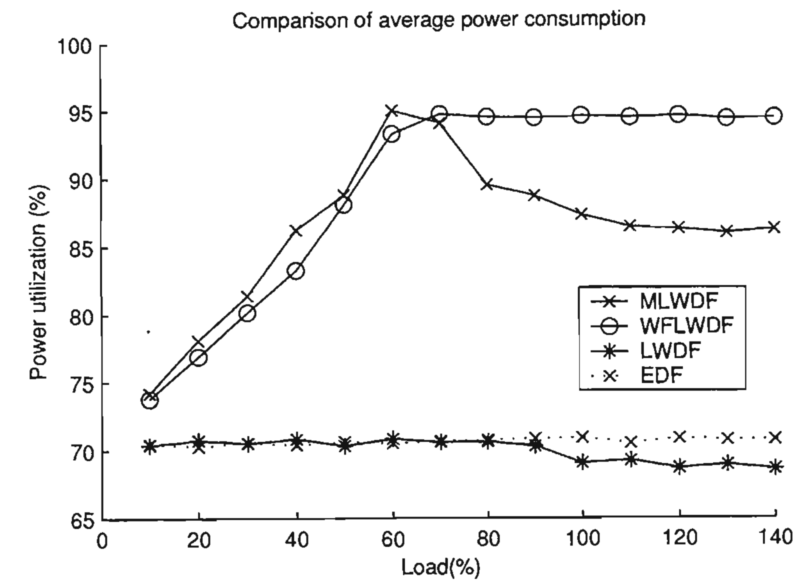

Figure 3.21: Comparison of average power consumption of various schedulers

then receive the biggest data rate that is possible with the remainder of the power. This process continues until the remaining power is too small for any data rate in set $\Re$ to be assigned to the next waiting packet.

Now the transmission power of packet i is given by $P_i = c_i \cdot R_i$. When the load is small, packets with a below average channel attenuation c_i will be chosen. Since c_i is chosen as small as possible and the transmission rate R_i only changes if the available transmission power changes by a large amount, the result is that the fraction of transmission power that will be left over, will on average be maximized. If, on the other hand, the load increases and packets are chosen because of the length of their queueing delay, instead of their channel conditions, then the c_i values will increase on average, with the result that the amount of power left over, will on average decrease. The result is that less of the remaining packets will on average be served, but with each packet using slightly more power. But to utilize as much of the power as possible, it is better to distribute smaller portions among more packets, than bigger portions among less packets. Fig. 3.20 proved this, where schedulers that served multiple packets had a much better power utilization than schedulers that only served one packet at a time.

In the case of WF-LWDF, the described scenario is not a problem. WF-LWDF always tries to share the available power among as many of the head-of-queue packets as possible. This results in smaller shares of power being distributed among more users. The result is that, even as the average channel attenuation increases, the number of users that receive power does not decrease enough to make a remarkable impact on the power utilization of WF-LWDF.

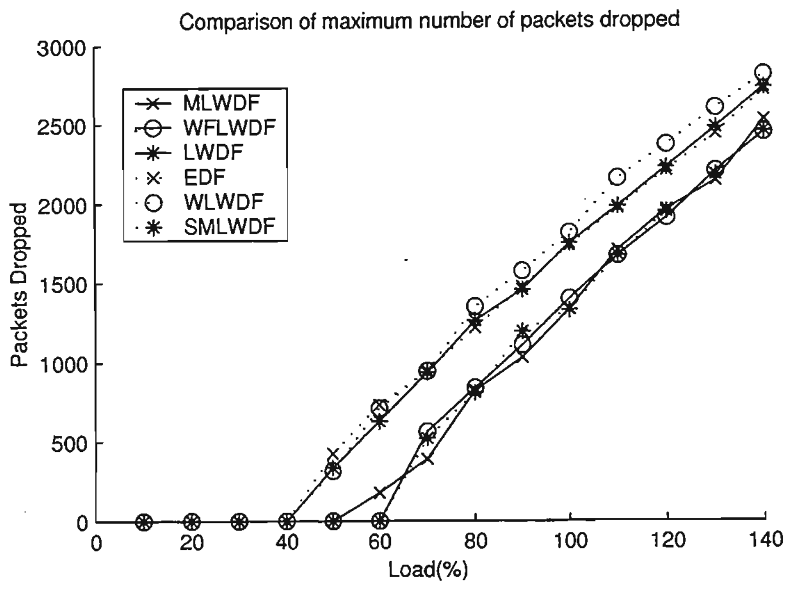

Figure 3.22: Packet loss bound of various schedulers

3.6.4 Packet Loss

This section was added to give some insight into the relationship between the network load and the rate at which the buffers overflow. At low loads a scheduler has no problems sorting the arriving packets into priority groups and transmitting them accordingly, but as the loads increase, one priority group may be of much greater importance than another. The result of this is that the queue containing very important packets does not grow, since it is constantly being served, whereas less important queues start growing uncontrollably. In our simulations we dealt with this problem by giving the buffers a set capacity of 1000 packets. Once all the spaces in the queue were taken up, any further arrivals into that queue were simply dropped.

In this section we once again skip the individual 80%-load distribution curves, since most of them do not contain any visual information of interest. Fig. 3.22, on the other hand, does contain some interesting information. Here the maximum number of packets lost can be found for each scheduler at loads ranging from 10% to 140%. The figure indicates that none of the schedulers lost any packets up to a load of 40%. A clear distinction of the packets dropped can be seen between the schedulers that serve one packet at a time and those serving multiple ones. The explanation for this is that schedulers transmitting multiple packets at the same time are able to utilize the available power better, which increases the throughput. This in turn decreases queueing times, so that packets are served faster.

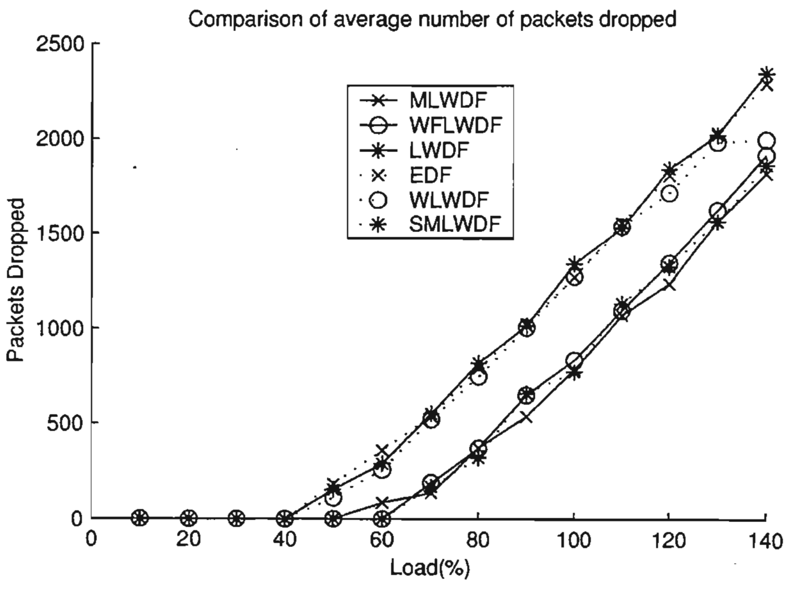

Figure 3.23: Average packet loss of various schedulers

3.7 Summary of Results

In this chapter the simulation model and the accompanying results were discussed. The discussion started with an overview of scheduling in a CDMA environment. This was followed by an overview of the schedulers used, which were EDF, LWDF, M-LWDF, WF-LWDF, SM-LWDF, and W-LWDF. The simulation framework was introduced, which included details on the data source, buffer, and channel models. The aim of the simulations was to measure the throughput, delay, power utilization and the number of packets dropped. Finally the results were presented.

The following observations were made, based on the various measurement results.

- The throughput and delay are proportional to the delay deadlines of data groups, resulting in set ratios.
- The expected ratios are distorted more by the channel awareness of a scheduler, than by serving multiple packets at the same time, although both have marked impact.
- Serving multiple packets at the same time increases the power utilization and the throughput, while accordingly decreasing the number of packets dropped.
- Packets being served by channel-aware schedulers experience less delay.
- Especially at high loads, distributing smaller portions of power to more packets increases the utilization of the available power resources.
- EDF is unable to guarantee QoS unless queueing delays are similar to delay deadlines.

The queueing delays of the various schedulers are summarized in Table 3.3.

Table 3.3: Queueing Delays

Scheduler	Queueing Delay (s)			Delay Ratio	Maximum Delay (s)
	Group1	Group2	Group3		
M-LWDF	113	63	29	3.9:2.2:1	250
WF-LWDF	115	81	57	2:1.4:1	180
LWDF	279	139	67	4.2:2.1:1	380
SM-LWDF	111	58	27	4.1:2.1:1	380
W-LWDF	152	117	72	2.1:1.6:1	390
EDF	121	121	121	1:1:1	16

Chapter 4

Fairness

4.1 Introduction

Fairness is one of the most important qualities in a network. In modern networks there are usually various users contending for bandwidth. If all of them were to pay the same fixed amount of money to use the available resources and the bandwidth is assigned to only one or two of the contenders, then the rest will rightfully feel that the system is not fair and that they deserve better treatment. In networks with only one type of data, it is fairly easy to divide the available bandwidth, especially if everyone is entitled to the same amount of service. The first complication that might arise would be that multiple priority levels exist and users have to pay more to belong to a higher priority level. What would make the system even more complicated, was if not only different priorities would exist for the various users, but if heterogeneous data were introduced, where video needed to be treated differently from e-mail, and the packets of telephone conversations had stricter delay requirements than HTTP. Unfortunately, this scenario is the case in modern networks. A number of different data types are present simultaneously in the same network. A measure of how well the various schedulers adapt to the requirements of the current data flowing through the network is fairness.

The first problem that needs to be addressed is how to define fairness precisely. Many attempts were made at defining fairness verbally. Golestani [77] was the first to define fairness mathematically. In [77], he defines a perfectly fair system to have the quality

$$w_k(t_1,t_2) = w_j(t_1,t_2),\ \ k,j \in B(t_1,t_2). \tag{4.1}$$

Here $B(t_1,t_2)$ is the set of sessions that are backlogged during the interval (t_1,t_2), while

$$w_j(t_1,t_2) = \frac{W_k(t_1,t_2)}{r_k}\ k \in K, \tag{4.2}$$

where K is the set of sessions k set up on a link and $k \in K$ is the service share allocated to session k. $W_k(t_1,t_2)$ are the number of bits of session k transmitted during $(0,t)$, while r_k represents the service share allocated to session k.

Effectively, what this means is that in a perfectly fair server, if the amount of service a session receives is normalized by its share of the available bandwidth it should have received, then the result should be the same for all backlogged sessions. In reality, no packetized implementation of a fluid system can achieve this goal. Stiliadis in [20] and [24] therefore replaced his equivalent version of the perfect fairness definition

$$\left|\frac{W_i^S(\tau,t)}{\rho_i} - \frac{W_j^S(\tau,t)}{\rho_j}\right| = 0, \tag{4.3}$$

with a more realistic version

$$\left|\frac{W_i^S(\tau,t)}{\rho_i} - \frac{W_j^S(\tau,t)}{\rho_j}\right| \leq F^S, \tag{4.4}$$

where ρ_i is the same as r_k in Golestani's work.

The bound F^S is an upper bound and indicates how fair a scheduler is. Note that this measure should maybe rather be called an unfairness measure, since a large F^S indicates an unfair scheduler, while $F^S = 0$ is a perfectly fair scheduler. We will however continue with the convention that other papers in this field have adopted and call it a fairness measure.

To get a better understanding of Stiliadis' fairness equation, we should delve a little deeper into his work [20]. He starts off by introducing the concept of *potential* $P_i(t)$, which represents the state of each connection in a scheduler. This concept is similar to

Golestani's virtual time function [77], which is used to keep track of how much service each session should have received at that moment in time. Stiliadis' potential function is like a global virtual-time function. When connection i is backlogged, its potential will increase exactly by the normalized service it received. Therefore, during any backlogged period $(\tau, t]$,

$$P_i(t) - P_i(\tau) = \frac{W_i(\tau, t)}{\rho_i}. \tag{4.5}$$

Using this definition of potential, one can see that a fair scheduler should increase the potentials of all backlogged connections at the same rate. The potentials should therefore be kept equal. Schedulers like GPS, PGPS, VirtualClock, and SCFQ belong to the class of fair queueing schedulers, which all attempt to achieve this. Even schedulers like M-LWDF and WF-LWDF attempt to do the same, although this is not done in such an obvious manner.

Note that the state information can be reset when the whole system becomes idle and none of the queues are backlogged. The problem is how to update the idle connections, if some of the queues are backlogged. The potential has to increase, otherwise it might be lagging behind by a large amount, when finally new packets arrive. This queue would then receive most of the service for a large amount of time, which is not fair. A session should only receive more service than it deserves, if it has been backlogged and did not receive sufficient service. The way the various schedulers update their potential during idle time periods differs. Ideally, the potential of the functions should increase by the normalized service that the connection could have received, if it were backlogged, which is referred to as *missed service* $S_i(t_1, t_2)$.

One way to update the potential of an idle connection, when it becomes backlogged again is to use a system potential function $P(t)$. This is a non-decreasing function, which is used as a reference system. When an idle session i becomes backlogged at time t, its potential $P_i(t)$ can be set equal to $P(t)$ to account for the service missed. The question remains of course how to maintain the system potential. GPS, for example, will set the potential of a session, which was idle and has become backlogged, equal to the potential of any of the backlogged sessions. In other words, the system potential function is equal to the

Table 4.1: Fairness of various schedulers

Server	Fairness
GPS	0
PGPS	$\max\left(\frac{L_i}{\rho_i}+C_i, \frac{L_j}{\rho_j}+C_j, \frac{L_{\max}}{\rho_j}+\frac{L_i}{\rho_i}, \frac{L_{\max}}{\rho_i}+\frac{L_j}{\rho_j}\right)$, where $C_i = \min\left((V-1)\frac{L_{max}}{\rho_i}, \max_{1\leq n\leq V}(\frac{L_n}{\rho_n})\right)$
SCFQ	$\frac{L_i}{\rho_i}+\frac{L_j}{\rho_j}$
VirtualClock	∞
Deficit Round Robin	$\frac{3F}{r}$
Weighted Round Robin	$\frac{F}{r}$

potential of one of the backlogged queues, that is

$$P(t) = P_i(t), \text{ for any } i \in B(t), \tag{4.6}$$

where $B(t)$ is the set of backlogged connections at time t. Another example is VirtualClock, where the potential of a queue is initialized to the real time when it becomes backlogged,

$$P(t_1, t_2) = t_2 - t_1. \tag{4.7}$$

Returning to (4.5), we saw that

$$P_i(t) - P_i(\tau) = \frac{W_i(\tau, t)}{\rho_i}. \tag{4.8}$$

But, as was seen in (4.4), the fairness of a scheduler is bounded by

$$\left|\frac{W_i^S(\tau,t)}{\rho_i} - \frac{W_j^S(\tau,t)}{\rho_j}\right| \leq F^S. \tag{4.9}$$

Table 4.1 gives a summary of the fairness bounds of many well-known schedulers, as found in [20]. Here, L_i is the maximum packet size of session i and $L_{\max}$ the maximum packet size among all sessions. C_i is the maximum normalized service that a session may receive in a PGPS server in excess of that in the GPS server. In weighted round-robin and deficit round-robin, F is the frame size and ϕ_i is the amount of traffic in the frame allocated to session i. L_c is the size of the fixed packet (cell) in weighted round-robin.

The reason for including fairness measurements in this simulation, is that we introduced the WF-LWDF algorithm as a fairer substitute for M-LWDF. The problem with M-LWDF is the way it distributes data rates—the first packet gets the biggest chunk of power, the next packet gets a smaller chunk, and so the process continues, until no more power is available. We decided to distribute power based on priority. To see whether this process is fairer than the original M-LWDF, we therefore need to measure $W_i^S(\tau,t)/\rho_i - W_j^S(\tau,t)/\rho_j$.

A problem we experienced was to determine what ρ_i should be interpreted to be. In Golestani's and Stiliadis' papers, ρ_i represented the amount of service guaranteed to a session. Both the M-LWDF and the WF-LWDF schedulers do not have any direct service guarantee built into their algorithm. Instead, the effective service guarantee can be calculated by taking into account the delay guarantee of each type. The guaranteed rate per session can therefore be estimated to be

$$\rho_i = C\frac{D_i}{\sum_j D_j}. \tag{4.10}$$

Another consideration that needs to be taken into consideration is that in our simulations, W_i represents the number of bits that session i has been served over 10s. This figure should be normalized by the simulation period over which the sample was taken, in other words, $T = 10s$.

Finally the capacity C of the system can be found using the same techniques as in the simulation setup in section 3.4.1. The resultant capacity per channel was found to be

$$C_i = \frac{1}{\sum_{i=1}^{N} \overline{c_i}}. \tag{4.11}$$

The total capacity of the system can be found by simply adding up the capacity of each of the channels.

$$C = \frac{N}{\sum_{i=1}^{N} \overline{c_i}}. \tag{4.12}$$

Since this figure is used to normalize the fairness value, its exact amount is not critical. We have therefore chosen $C = 1Mb/s$, which is the maximum capacity the system will have, under the constraints of the channel attenuation functions.

To perform a fairness measurement on the M-LWDF and WF-LWDF scheduling schemes, the following bound should therefore be found

$$\frac{W_i(\tau,t)}{\rho_i} - \frac{W_j(\tau,t)}{\rho_j} = \frac{1}{C}\left[\frac{W_i(\tau,\tau+T)}{T}\frac{\sum_k D_k}{D_i} - \frac{W_j(\tau,\tau+T)}{T}\frac{\sum_k D_k}{D_j}\right] \leq F^S. \tag{4.13}$$

Note that we got rid of the absolute value sign. By doing this we enable negative results, which are not realizable. The problem with the absolute value is that the resulting pdf actually consists of a negative and a positive domain, but the negative part is mapped into the positive region, which results in a distorted, non-negative graph. For analytical purposes we therefore found it better to include negative measurements, which separates the two domains and clarifies the trends of the curves. This is acceptable, as long as one remembers that a negative fairness measure is conceptually meaningless, and when comparing the fairness curves, one should always consider the absolute values of measurements.

The measurements performed by the simulator were very similar to those performed for the throughput, delay, power, and packet dropping performance of the various schedulers, as presented in the previous chapter. In the case of the fairness measurements, the throughput measure of each queue was normalized by the required minimum guaranteed rate expression once every 10 seconds. This process was repeated 1000 times. Once all the measurements had been performed, the measured results of every queue at one particular time were subtracted from the results of all the other queues measured at the same time. Every second therefore produced 105 results. These were then plotted on a 50-bin histogram.

A very important thing to realize is that after normalizing the throughput values, they are quite small. In order to plot them on a 50-bin histogram, the results were multiplied by 1000. In order to find the real fairness values and compare them to the analysis results that will be presented in section 4.3, one first has to divide the simulation results by 1000.

4.2 Fairness results

The resultant fairness curves can be seen in Figs. 4.1, 4.2, 4.3, 4.4, 4.5, and 4.6. Notice that the curves that lie on top of one another and are narrowest around zero are the most fair. The reader should be careful to note the scale on the x-axis, which varies among the different schedulers. Of major interest are the extreme x-values, in the positive and the negative domain, since these represent the largest fairness values measured. The reader should also not forget that these results have been multiplied by 1000. In order to compare them to the analysis results in the next section, they have to be normalized by 1000 first.

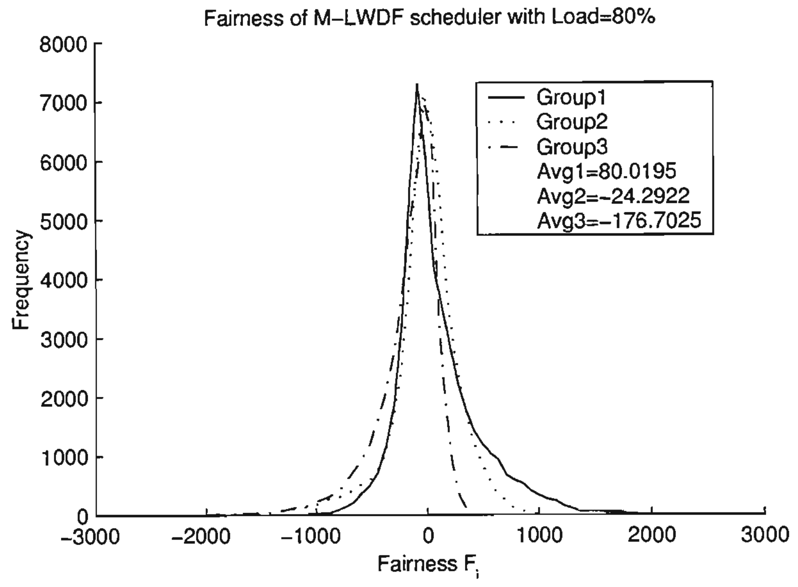

Figure 4.1: Fairness of M-LWDF scheduler

Comparing M-LWDF and WF-LWDF in Figs. 4.1 and 4.2, one can see, with great relief, that WF-LWDF turns out to be a much fairer scheduler than M-LWDF. The x-axis represents the fairness, which was measured by finding the difference between the normalized throughput of the various queues, as discussed in the previous section. The x-axis, once again, consists of 50 bins, with the y-axis indicating how many of the samples fitted into each of the bins.

Whereas M-LWDF has a maximum F_i value of around 2000, the equivalent F_i value for WF-LWDF is approximately 700. The equivalent value for LWDF in Fig. 4.3 is even lower, at less than 300. This can be explained by the fact that LWDF only takes the queueing delay into account when making scheduling decisions, while M-LWDF and WF-LWDF also have to consider channel conditions. In contrast, M-LWDF and WF-LWDF utilize the available power better than LWDF and therefore have better throughput and delay behaviour than LWDF, as became apparent in the previous chapter.

In terms of fairness, SM-LWDF in Fig. 4.4 once again performs quite similar to M-LWDF.

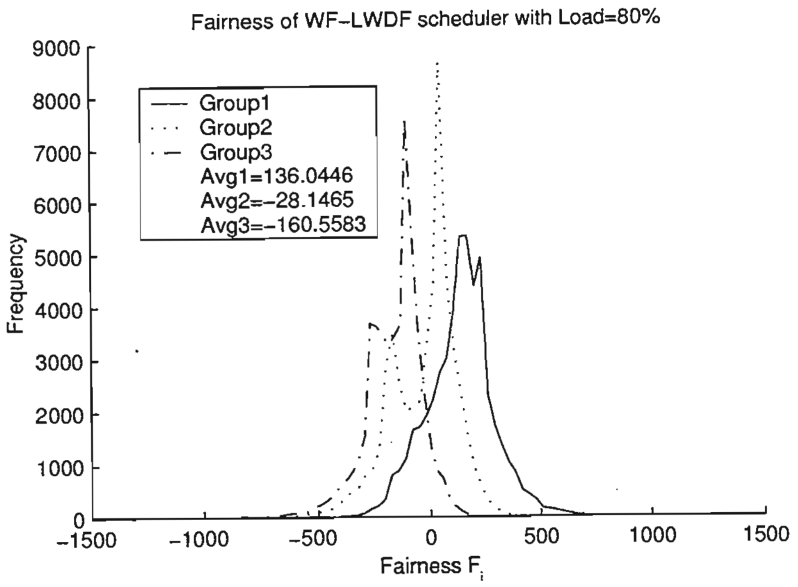

Figure 4.2: Fairness of WF-LWDF scheduler

The performance of W-LWDF in Fig. 4.5 too is almost identical to that of M-LWDF, in contrast to WF-LWDF. What this shows is that the way packets are chosen, and whether multiple packets are transmitted simultaneously or only one packet is transmitted makes no difference to the fairness of a scheduler. What does matter is how power is distributed between the various users. Another conclusion is that channel-awareness decreases the fairness of a scheduler, as can be seen when comparing W-LWDF to LWDF, the only difference being the channel-awareness of W-LWDF.

Surprisingly, EDF in Fig. 4.6 turns out to be fairer than most of the other schedulers mentioned, except for LWDF. One would expect this scheduler to be intrinsically unfair, when inspecting the throughput distributions of EDF, which was done in section 3.6.1. The throughput curves for the various data groups lay on top of one another, which should result in great unfairness, once the normalization is applied. The curves in Fig. 4.6 actually show that the three curves were indeed separated by the normalization process, but that the maximum F_i value is still lower than that of most other schedulers. The reason for this impressive fairness behaviour, is that EDF has a relatively low throughput and the throughput variation is very small compared to other schedulers. The reason for this small throughput variation is that EDF does not have to concern itself with the channel

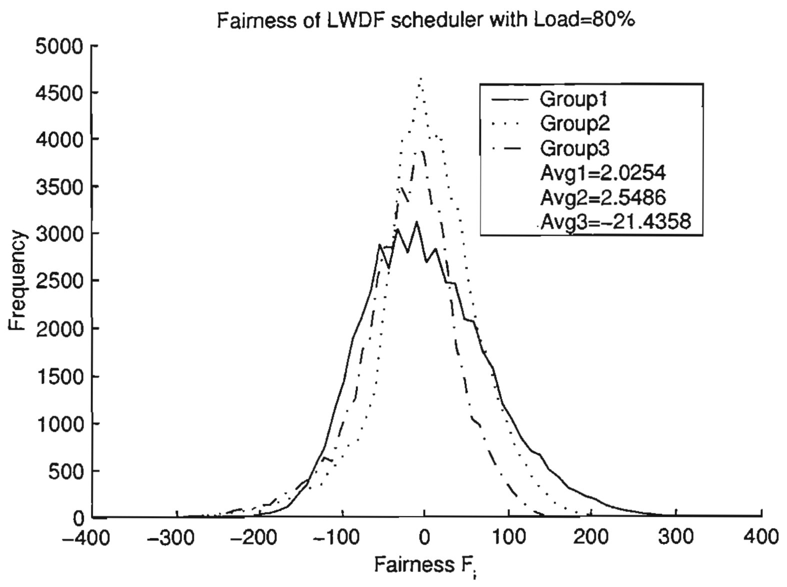

Figure 4.3: Fairness of LWDF scheduler

conditions and is able to serve packets one at a time, thereby distributing the available power very precisely between the backlogged queues. The result is that the throughput distribution has a very small variance and, even if the normalization separates the three group distributions, the largest fairness difference between the three groups is still small compared to the other schedulers.

A summary of the fairness bounds of all the schedulers can be seen in Fig. 4.7. The fairest schedulers are LWDF and EDF with quite similar performance, followed closely by our newly introduced WF-LWDF algorithm. At loads larger than 60%, M-LWDF and SM-LWDF perform quite a bit worse. Finally, the most unfair scheduler is W-LWDF.

The average fairness values of the various schedulers in Fig. 4.8 are quite similar. LWDF is still the fairest scheduler, followed by a tie between EDF and WF-LWDF. M-LWDF and SM-LWDF behave quite similarly to W-LWDF, except under very high load conditions, where W-LWDF is once again slightly less fair.

In the next section we would like to find the fairness bounds of M-LWDF and WF-LWDF, using the throughput simulation results of section 3.6.1. To compare these with the fairness measurements of this section, the values in Fig. 4.7 have to be normalized by 1000.

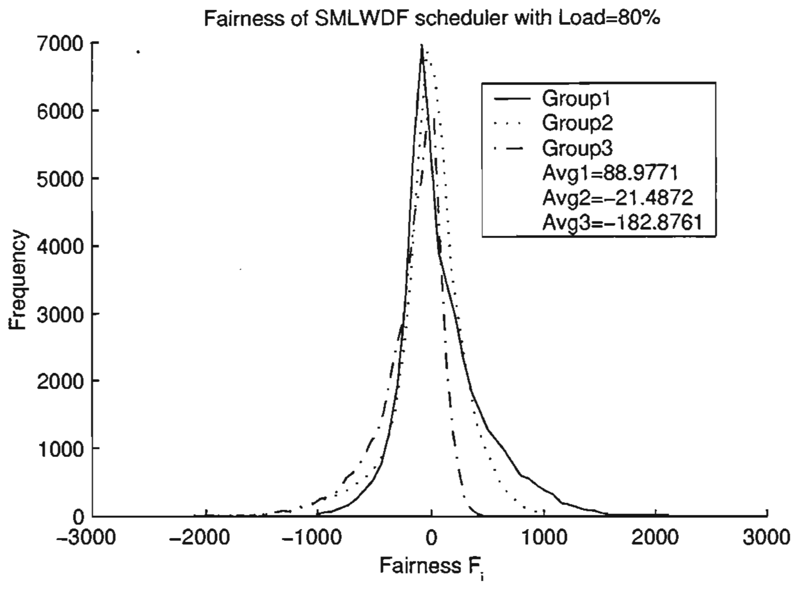

Figure 4.4: Fairness of SM-LWDF scheduler

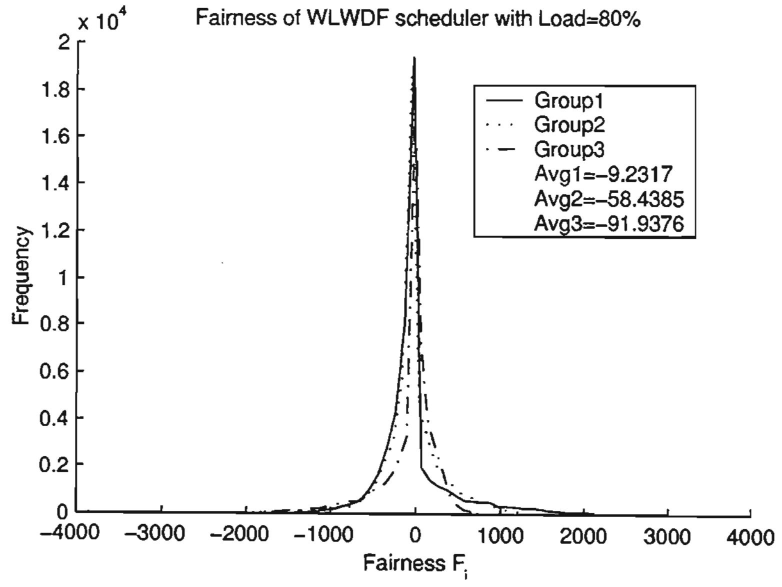

Figure 4.5: Fairness of W-LWDF scheduler

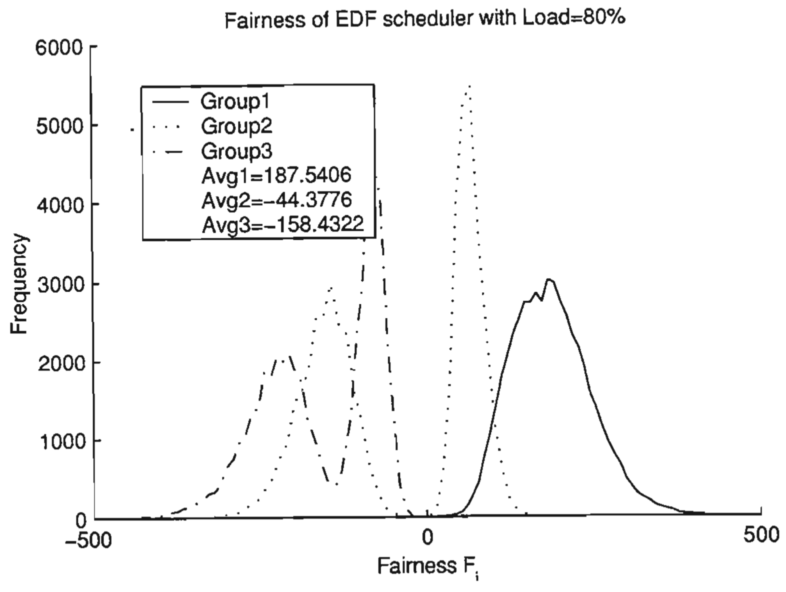

Figure 4.6: Fairness of EDF scheduler

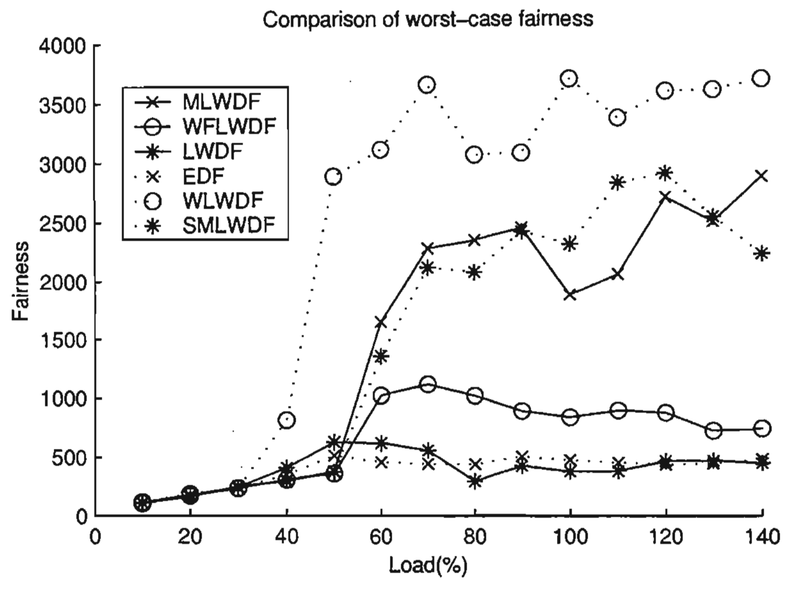

Figure 4.7: Fairness bound comparison of various schedulers

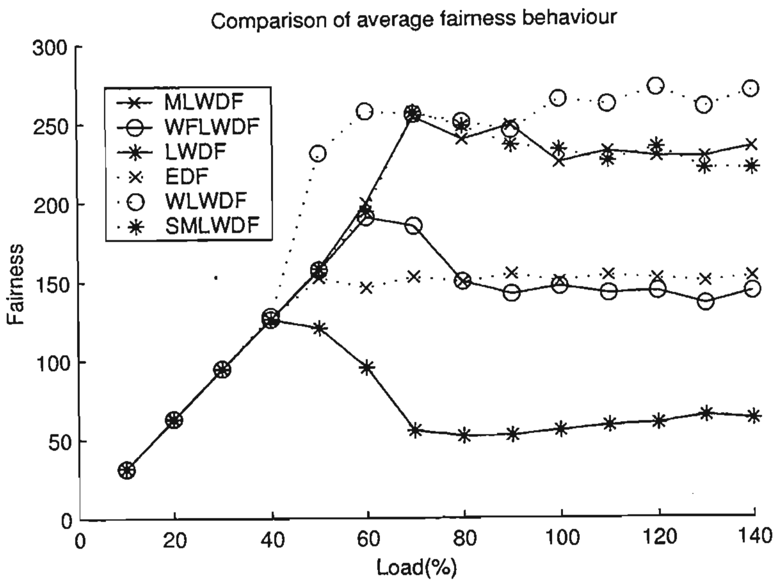

Figure 4.8: Comparison of average fairness of various schedulers

4.3 Fairness analysis

The fairness bounds of many schedulers were found and published by Stiliadis in papers such as [20], [24], and [58]. All the schedulers studied in this paper belong to the GPS family of schedulers. We have not come across any literature that attempts to find fairness bounds for LWDF-based schedulers, like M-LWDF.

In this section we would therefore like to attempt to find fairness bounds of the M-LWDF and WF-LWDF schedulers, using the throughput simulation results presented in section 3.6.1. It is important to realize that this work has not been published before and that the technique we used to find this bound has not been implemented before in any other paper we have come across, to find a fairness bound. Finding a fairness bound, especially a tight one, is of great benefit to the network designer, since he or she can compare the various bounds to decide which scheduler to implement, while being guaranteed that these bounds will not be exceeded.

In general, two types of bounds exist, deterministic and stochastic. A deterministic bound states that a measured value x will never be larger than some constant y, which can be written as

$$\Pr(\mathrm{x} > \mathrm{y}) = 0. \tag{4.14}$$

The problem with a bound like this is that under no circumstances may this constant be exceeded. These bounds can therefore be much larger than the average value, even if the bound is tight. In some cases, deterministic bounds might not even exist. For both these reasons it is often better to find a stochastic bound instead. A stochastic bound states that the probability of a measured value x ever being equal to or exceeding some constant y, will definitely be smaller than some value δ, which mathematically becomes

$$\Pr(\mathrm{x} \geq \mathrm{y}) < \delta. \tag{4.15}$$

A range of different bounds exist, with varying tightness. One of the tightest bounds is the Chernoff bound, which we chose. To find the fairness bounds of M-LWDF and WF-LWDF we used the throughput simulation results from the previous chapter, which summarize the output characteristics of these two schedulers. We decided to perform the necessary normalization on the throughput curves, before analyzing them further. The resultant curves of the various data groups of M-LWDF and WF-LWDF can be seen in Figs. 4.9, 4.10, 4.11, and in Figs. 4.12, 4.13, 4.14, respectively. In order to find a Chernoff bound, the moment generating function of a distribution needs to be found. The problem with the available data was that it consisted of hundreds of measurement results. To find a moment generating function, a distribution therefore had to be found that would fit on top of the curve created by the simulation results. To fit a curve onto such a distribution of data points, it is useful to find the mean and the variance of the distribution. These are relatively easy to find, using the equations

$$\mathrm{Mean}(x) = \overline{x} = \frac{1}{n}\sum_{i=1}^{k} f_i x_i, \tag{4.16}$$

where x_i is the midpoint of the ith class interval, f_i is the frequency of the ith class interval, k is the number of class intervals, and n = number of observations = $\sum_{i=1}^{k} f_i$.

$$\text{Sample variance}(x) = s^2 = \frac{1}{n-1}\sum_{i=1}^{k} f_i(x_i - \overline{x})^2. \tag{4.17}$$

Once these were found, attempts were then made to fit Gaussian, Poisson, Gamma, Erlang-k, Chi-square, and Beta distributions onto the curves, using the acquired mean and variance values in each case. In Figs. 4.9 to 4.14 it can be clearly seen that the Gamma distribution fits neatly on top of the simulation results. This was by far the best fit, compared to attempts using the other distributions. The probability density function of a Gamma curve is given by

$$f(x) = \frac{1}{\Gamma(\alpha)\beta^{\alpha}} x^{\alpha-1} e^{-x/\beta}, \tag{4.18}$$

where $\alpha, \beta > 0$ are the two parameters that determine the Gamma distribution. They are related to the mean and the variance as follows

$$\begin{aligned} \text{Mean } E[X] &= \alpha\beta, & (4.19) \\ \text{Variance } E\{X - E[X]\}^2 &= \alpha\beta^2. & (4.20) \end{aligned}$$

The moment generating function for a Gamma distribution is

$$M_X(t) = \left(\frac{1/\beta_1}{1/\beta_1 - t}\right)^{\alpha_1} \left(\frac{1/\beta_2}{1/\beta_2 - t}\right)^{\alpha_2}, \tag{4.21}$$

where $t > 0$ and n represents the number of random distributions that are considered. In our case we always only consider two distributions at any time, with the result that $n = 2$.

From this we can define the "semi-invariant" generating function

$$\gamma_X(t) = \log[M_X(t)]. \tag{4.22}$$

For the Gamma distribution this turns out to be

$$\gamma_X(t) = \log\left[\left(\frac{1/\beta_1}{1/\beta_1 - t}\right)^{\alpha_1} \left(\frac{1/\beta_2}{1/\beta_2 - t}\right)^{\alpha_2}\right] = \log[f(t)g(t)]. \tag{4.23}$$

The first order derivative of this function is

$$\gamma_X^{(1)}(t) = \frac{1}{f(t)g(t)} \times [f(t)g(t)]', \tag{4.24}$$

where

$$\begin{aligned} f'(t) &= \frac{d}{dt}\left\{\left[\frac{1/\beta_1}{1/\beta_1 - t}\right]^{\alpha_1}\right\} = \alpha_1 \left[\frac{1/\beta_1}{1/\beta_1 - t}\right]^{\alpha_1 - 1} \left[\frac{1/\beta_1}{(1/\beta_1 - t)^2}\right] \\ &= \frac{\alpha_1}{1/\beta_1 - t} \left[\frac{1/\beta_1}{1/\beta_1 - t}\right]^{\alpha_1}. \end{aligned} \tag{4.25}$$

The result of this calculation gives us y

$$\begin{aligned} y &= n \cdot \gamma_X^{(1)}(t) = n \cdot \frac{\frac{\alpha_2}{1/\beta_2 - t}\left[\frac{1/\beta_1}{1/\beta_1 - t}\right]^{\alpha_1}\left[\frac{1/\beta_2}{1/\beta_2 - t}\right]^{\alpha_2} + \frac{\alpha_1}{1/\beta_1 - t}\left[\frac{1/\beta_1}{1/\beta_1 - t}\right]^{\alpha_1}\left[\frac{1/\beta_2}{1/\beta_2 - t}\right]^{\alpha_2}}{\left[\frac{1/\beta_1}{1/\beta_1 - t}\right]^{\alpha_1}\left[\frac{1/\beta_2}{1/\beta_2 - t}\right]^{\alpha_2}} \\ &= n\left[\frac{\alpha_1}{1/\beta_1 - t} + \frac{\alpha_2}{1/\beta_2 - t}\right]. \end{aligned} \tag{4.26}$$

Rearranging, we get

$$y = \frac{n\alpha_1(1/\beta_2 - t) + n\alpha_2(1/\beta_1 - t)}{(1/\beta_1 - t)(1/\beta_2 - t)}. \tag{4.27}$$

Now, the Chernoff bound is given by

$$P[Y \geq y] < e^{n\gamma_X(t) - ty} = \delta. \tag{4.28}$$

Rearranging this, we get

$$y = \frac{1}{t}n\gamma_X(t) - \frac{1}{t}\log\delta = \frac{1}{t}n\log\left[\left(\frac{1/\beta_1}{1/\beta_1 - t}\right)^{\alpha_1}\left(\frac{1/\beta_2}{1/\beta_2 - t}\right)^{\alpha_2}\right] - \frac{1}{t}\log\delta. \tag{4.29}$$

(4.27) and (4.29) can now be solved simultaneously to give a solution for y, for a specific δ. The results can be seen in Table 4.2, which were calculated using MATLAB.

Table 4.2: Resulting bounds for various group combinations.

δ	M-LWDF			WF-LWDF		
	Groups 1,2	Groups 1,3	Groups 2,3	Groups 1,2	Groups 1,3	Groups 2,3
10^{-3}	-6.597881	-9.226422	3.932648	5.214364	5.491106	9.081910
10^{-6}	-13.706851	-20.335167	4.760328	6.676904	6.956942	11.767444
10^{-9}	-23.056544	-34.072929	5.225987	7.602784	7.868499	13.514805
10^{-12}	-35.551307	-50.928027	5.536719	8.274073	8.522002	14.809432

In each case the worst bound is chosen from each of the three group combinations. This represents the bound for that scheduler, which can be found in Table 4.3.

These results clearly show that WF-LWDF is a much fairer scheduler than M-LWDF. Note, as explained earlier in this chapter, only the absolute values of the two schedulers should be compared, since negative fairness values are meaningless. If one compares the analytical

Table 4.3: Resulting bounds for specific δ values.

δ	M-LWDF	WF-LWDF
10^{-3}	-9.226422	9.081810
10^{-6}	-20.335167	11.767444
10^{-9}	-34.072929	13.514805
10^{-12}	-50.928027	14.809432

bounds with the simulation results in section 4.2, one has to divide the simulation results by 1000, as explained before. The Chernoff bound for a violation probability of 10^{-3} for M-LWDF is approximately 4.2 times larger than the corresponding simulation result. For WF-LWDF the Chernoff bound is 13 times larger than the simulation result predicts for a violation probability of 10^{-3}. In both cases the Chernoff bounds are therefore relatively tight.

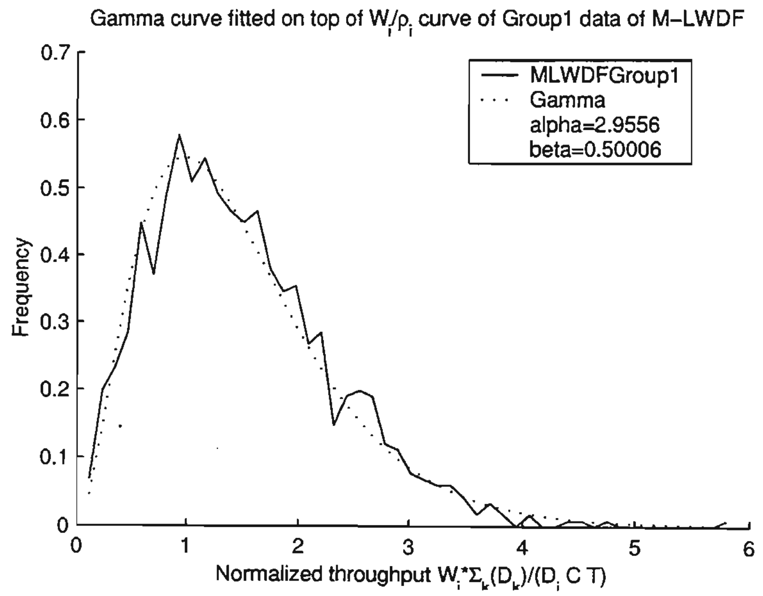

Figure 4.9: Gamma curve fitted onto Group1 data of M-LWDF scheduler

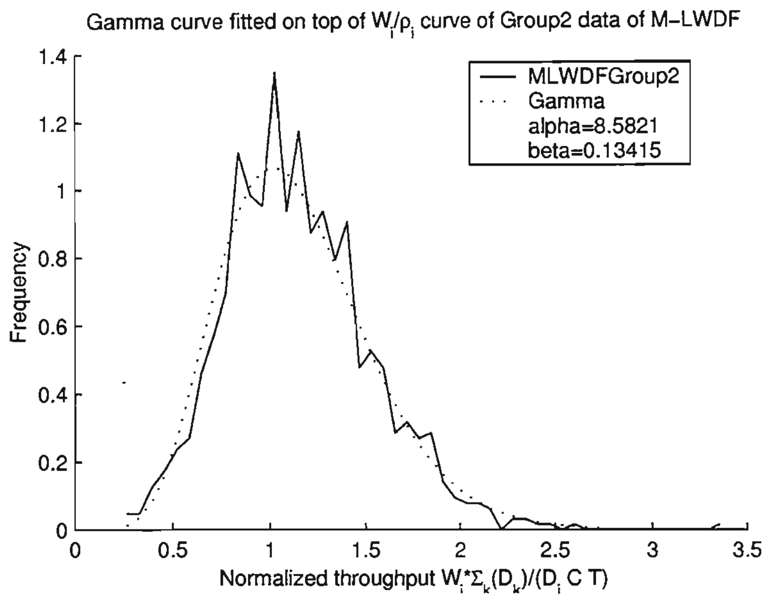

Figure 4.10: Gamma curve fitted onto Group2 data of M-LWDF scheduler

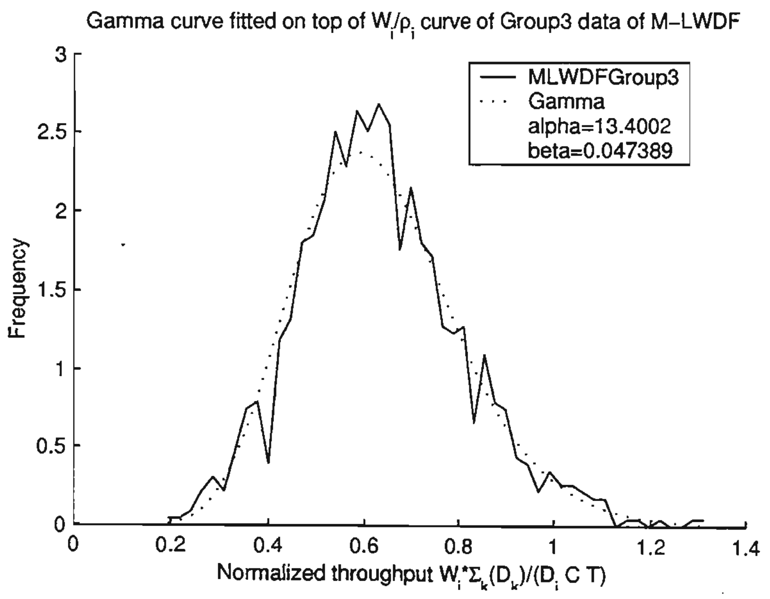

Figure 4.11: Gamma curve fitted onto Group3 data of M-LWDF scheduler

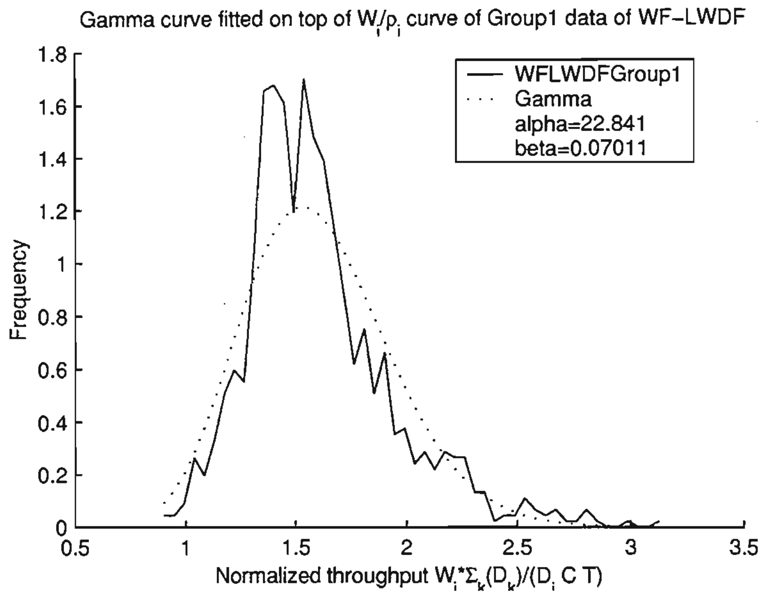

Figure 4.12: Gamma curve fitted onto Group1 data of WF-LWDF scheduler

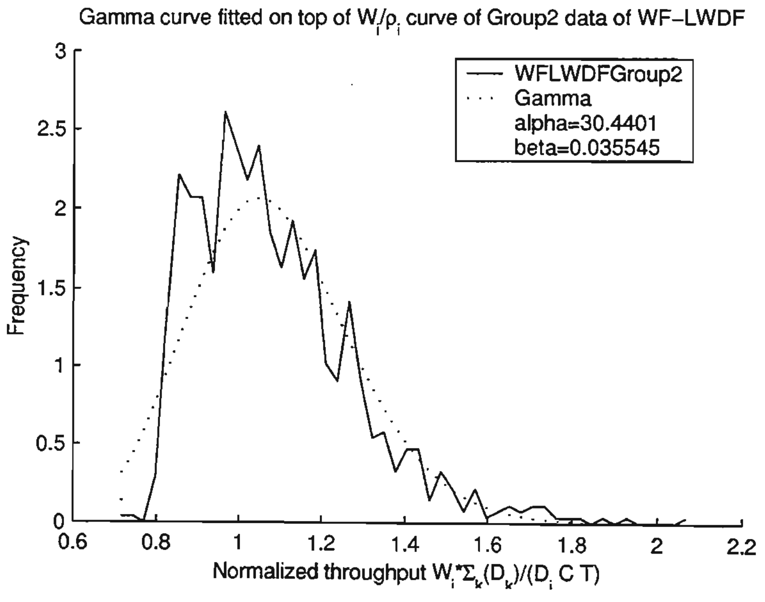

Figure 4.13: Gamma curve fitted onto Group2 data of WF-LWDF scheduler

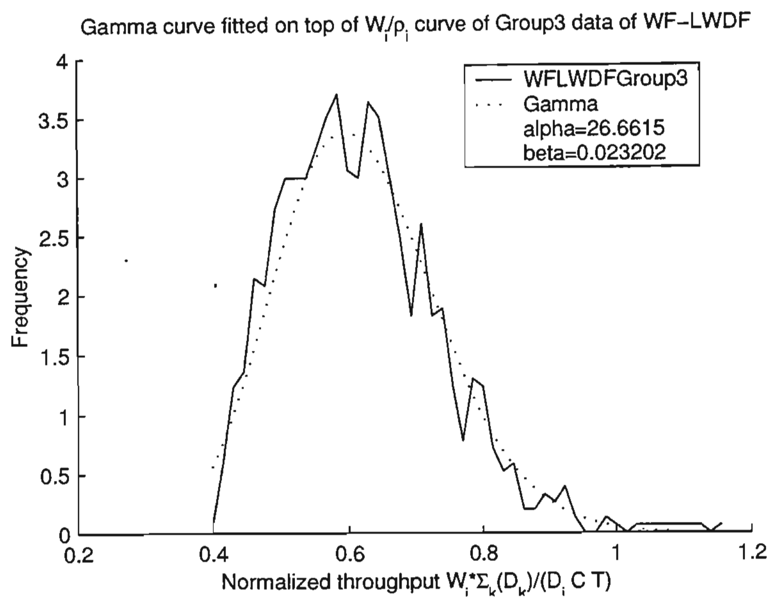

Figure 4.14: Gamma curve fitted onto Group3 data of WF-LWDF scheduler

4.4 Summary of results

The aim of this chapter was to compare the fairness of various schedulers, as an indication of how well they are able to adhere to their QoS guarantees. Of particular importance was the comparison of the fairness of M-LWDF and WF-LWDF, since WF-LWDF was proposed as an improvement of M-LWDF. The following are the simulation results obtained:

- For both the maximum fairness measure and the average fairness measure, LWDF was the fairest scheduler.
- Despite EDF's unexpected throughput distributions, it turned out to be the second fairest scheduler. The reason for this was attributed to the small variance of the EDF distributions, due to very precise power distribution.
- After LWDF and EDF, WF-LWDF turned out to be the fairest scheduler, which was a fortunate result, since WF-LWDF was meant to be an improvement of the existent M-LWDF algorithm.

- M-LWDF, SM-LWDF, and W-LWDF followed, proving together with the WF-LWDF results, that channel-awareness decreases the fairness of a scheduler.

A fairness analysis of M-LWDF and WF-LWDF , using the throughput results, resulted in the following points:

- The stochastic bounds calculated, inform a network operator that the probability of the fairness of M-LWDF and WF-LWDF exceeding a value y, will definitely be less than some constant δ. This makes this analytical approach of great value to a network operator.
- The Chernoff bound of the fairness of WF-LWDF is 58% that of M-LWDF for $\delta_i = 10^{-6}$, which gets even larger as δ_i decreases. For $\delta_i = 10^{-12}$, for example, the fairness of WF-LWDF is only 29% that of M-LWDF. This means that WF-LWDF is fairer than M-LWDF.

Chapter 5

Analytical Delay Results

5.1 Introduction

Research relies on three techniques for the evaluation of new protocols. The first method is an implementation. This is usually done last, since it is expensive and usually also takes more time to implement than software implementations. A much cheaper method is a computer simulation. The accuracy of simulations is based on the accuracy of the underlying models used. Simulations are inexpensive, easily implementable, and quickly changeable. They are usually employed to get a rough estimate of the performance of an algorithm.

Finally, the third evaluation technique is a mathematical analysis, which usually tries to find bounds on the behaviour of an algorithm. The analytical approach is therefore concerned with the extreme behaviour of the protocol of interest.

In the world of scheduling a large amount of literature has been published, which attempts to determine bounds on the behaviour of many different schedulers. Some of the metrics, which have been analyzed are the stability of the scheduler, its fairness, and the delay that packets experience, either at a particular node or end-to-end. The first metric of concern is the worst-case end-to-end delay bound of a scheduler. In this chapter we would like to look at some delay analyses that have been published and compare the resulting bounds.

In the case of each scheduler only a brief overview of the system model and the resulting bound with an explanation of the meaning of the variables will be mentioned. The entire derivation of the results will not be reproduced, since many of the derivations span several pages. The reader might notice that most of the schedulers that were modelled in the previous two chapters are not mentioned in this chapter. The reason for this is that schedulers like M-LWDF and WF-LWDF are very complicated to analyze and we have not come across any analysis, which attempts to bound the end-to-end delay of these schedulers. This might be a good topic for future research.

5.2 Delay Analysis

5.2.1 System Model

In order to compare the various analytical results, we need to develop a common system model. Every published.analysis unfortunately uses its own variable and constant symbols to represent the various components and processes of a network that need to be considered, when determining the worst-case end-to-end delay that packets experience. This was the first problem that needed to be addressed. The variables and constants had to be standardized to one set of symbols. The resulting set can be found in Table 5.1. Using this set, the various delay bounds can thus be translated.

5.2.2 First Come First Serve

To be able to understand and improve networks, they need to be mathematically modelled and analyzed. This can be very complicated and attempts have been made to develop a networking calculus, which can be used to find the bounds and limitations of a network.

Table 5.1: System Model

Symbol	Meaning
V	no of sessions
K	no of nodes
L	average packet length of all sessions
L_{max}	maximum packet length of all sessions
L_v	average packet length of session v
L_v^{max}	max packet length of session v
$Load$	network load
C	server capacity
R	serving rate
ρ	average arrival rate of all sessions
ρ_{norm}	normalized arrival rate of all sessions
ρ_v	average arrival rate of session v
ρ_v^{norm}	normalized arrival rate of session v
σ	burstiness parameter of all sessions
σ_{norm}	normalized burstiness parameter of all sessions
σ_v	burstiness parameter of session v
σ_v^{norm}	normalized burstiness bound of session v
ν	constant, TS-EDF requires $\nu > 0$
ϕ	GPS weight of all sessions
ϕ_v	GPS weight of session v
τ	propagation delay
F	frame size for Round Robin servers
L_c	fixed cell length for Round Robin servers

One of the first attempts was [27] and its sequel [28], which obtain deterministic bounds on delay and buffering requirements in a packet switched network under a fixed routing strategy. In the first paper [27] a single node network element is analyzed. In particular a *First Come First Serve* (FCFS) multiplexer is analyzed, which, for the sake of simplicity, has only two inputs. A FCFS multiplexer is the simplest scheduling design. As its name implies, at each node the arriving packets are queued in the order that they arrived. The packets are therefore all given the same priority, with the result that a FCFS multiplexer cannot guarantee any form of Quality-of-Service.

Reference [27] starts off introducing a FCFS multiplexer that has two inputs, the first having a maximum arrival rate C_1, the second C_2. The output link has a maximum transmission rate C_{out}. The maximum packet size is assumed to be $L_{\max}$. In this paper the traffic is considered to be leaky bucket shaped. If R is the rate function of a traffic stream flowing on a given link, then over a time period from t_1 to t_2 it must therefore conform to

$$\int_{t_1}^{t_2} R \leq \sigma + \rho(t_2 - t_1). \tag{5.1}$$

The result of the analysis is that the delay of any data bit entering an FCFS multiplexer from stream 1 is upper bounded by

$$D_{FCFSMUX,1} = \frac{1}{C_{out}} \max_{u \geq 0}[b_1(u) + b_2(u + (L_{\max}/C_2)) - C_{out} \cdot u], \tag{5.2}$$

where $u = t_2 - t_1$. $W_\rho(R)(t)$ is the argument of the maximum backlog of a queue

$$W_\rho(R)(t) = \max_{t_1 \leq t_2} \left[\int_{t_1}^{t_2} R - \rho(t_2 - t_2)\, dt \right]. \tag{5.3}$$

For the full derivation of this result, please see the original document [27].

We are actually interested in the result for a multi-stage network, since we would like to study the affect of multiple nodes on the end-to-end delay that packets experience. Reference [28] considers networks with $N = 2^n$ inputs and N outputs, which use 2x2 switching nodes. In other words, each node has 2 inputs and two destination nodes. A total of K stages are assumed to exist. V sessions are assumed to be set up. Packets are not allowed to be longer than $L_{\max}$ bits, where

$$L_{\max} \leq \frac{\sigma_{norm}}{1-\rho_{norm}}. \tag{5.4}$$

V are the maximum number of sessions that are allowed to enter a given input link of a switch and exit a given output link of the switch. The delay of any packet through a switch in stage v has been derived in [28] to be not more than

$$D_v = V(V\rho_{norm})^{-v}[\rho_{norm}L_{\max}(V\rho_{norm}) + \sigma_{norm}]. \tag{5.5}$$

Once translated into our set of symbols, the total delay of each session has then been shown in [28] to be

$$D = \sum_{s=1}^{H} D_s = [L_{\max}(V\rho_{norm}) + \frac{\sigma_{norm}}{\rho_{norm}}][(1 - V\rho_{norm})^{-K} - 1]. \tag{5.6}$$

5.2.3 EDF

TS-EDF

In [40] the first distributed deterministic scheduling protocol is presented that has a polynomial delay bound for the temporary sessions model. In a permanent sessions model, there are a fixed number of sessions in the network, each of which travels along a fixed path through the network. In a temporary sessions model, or adversarial queueing model, as it is also known, an adversary is allowed to change the path on which it is injecting packets.

In the analysis of FCFS in [28] an exponential bound was given, which we evaluate in section 5.3.1 and turns out to have an incredibly large bound for large network loads and a large number of nodes. What makes Andrews' analysis of EDF special, is that it predicts polynomial delay bounds. Even-though they might still be much bigger than the real end-to-end delay that packets might experience, the function will at least not grow as fast as it would if it were exponential.

In [40] a network of V servers is considered. σ_{norm} is the network burst parameter and $1-\rho_{norm}$ is the network load parameter such that for all servers v, the total number of packets, across all sessions that are injected into the network during a time interval of length W and that have server v on their path is bounded by $(\rho_{norm})\sigma_{norm}$. K denotes the maximum session path length. Now, time is divided into blocks of length M, where each block represents the end-to-end delay bound. Reference [40] shows for any $\nu \geq 0$ it is possible to choose the systems operating parameters such that

$$M = \mathbf{O}\left(\left(\frac{KV\sigma_{norm}}{(1-Load)^4}\right)^{(1+\nu)}\right). \tag{5.7}$$

To clarify the **O** notation, we refer to [78], which contains the following definition:

Definition Let $f(x)$ and $g(x)$ be positive for x sufficiently large. Then f is of at most the order of g as $x \to \infty$, if there is a positive integer M for which

$$\frac{f(x)}{g(x)} \leq M, \tag{5.8}$$

for x sufficiently large. We indicate this by writing **f=O(g)**.

In other words, what this definition is saying, is that (5.7) is the end-to-end delay bound of TS-EDF.

PS-EDF

Reference [40] continues to introduce the same protocol, but this time for permanent sessions. A deadline for each packet is defined for each server on the packet's path. Each server always gives priority to the packet with the smallest deadline.

In the PS-EDF network, let each session i have a burst size σ_{norm}, an injection rate ρ_{norm} and a path length K. The maximum load is given by *Load*, which is equal to the sum of the injection rates. In the initial server v_0, let S_{v_0} be the set of sessions that have v_0 as their initial server and pass through server v.

Reference [40] concludes with the following expression, which has been translated into our standard set of symbols

$$\mathbf{O}\left(\frac{\sigma_i^{norm}+1/(1-Load)}{\rho_i^{norm}}+\frac{KV}{(1-Load)^3}\log(\frac{V}{\rho_i^{norm}})\right), \tag{5.9}$$

(5.9) represents the bound of the PS-EDF scheduler in the system model outlined.

5.2.4 GPS

The original *Generalized Processor Sharing* (GPS) paper is [5], in which Parekh and Gallager analytically compare GPS and its packetized version PGPS for the single node case. Let t_k be the time that packet p_k departs under PGPS and u_k be the time that p_k departs under GPS. If L_{max} is the maximum packet length and R is the rate of the server, then

$$t_k \leq u_k + \frac{L_{\max}}{R}. \tag{5.10}$$

The overall result of this analysis is that, for all packets p, the departure time difference in a PGPS and GPS server can never exceed $\frac{L_{\max}}{R}$, or symbolically

$$\hat{t}_p - t_p \leq \frac{L_{\max}}{R}, \tag{5.11}$$

where $\hat{t}_p$ represents the time at which packet p departs under PGPS and t_p represents the time at which packet p departs under GPS.

In reference [45] the analysis is extended to the multiple node case. Parekh and Gallager start off by introducing the network model, which is a directed graph in which nodes represent switches, and arcs represent links. P_i represents the route taken by session i, a path in the graph. $P(i,k)$ is the k^{th} node in $P(i)$ and K_i the maximum number of nodes in P_i. The rate of server m is R.

$A_i(\tau,t)$ is a measure of the amount of session i traffic that enters the network in the interval $[\tau,t]$. $S_i^{(k)}(\tau,t), k=1,\ldots,K_i$ represents the amount of session i traffic served by node $P(i,k)$ in the interval $[\tau,t]$. The amount of traffic leaving the network is therefore given by $S_i^{(K_i)}$. In order to simplify the analysis of the network, the service function is characterized by leaky bucket parameters σ_i and ρ_i, which as usual represent the burst size and the average guaranteed rate, respectively. The result is that

$$S_i^{(k)}(\tau, t) \leq \sigma_i + \rho_i(t - \tau), \forall t \geq \tau \geq 0. \tag{5.12}$$

Using the leaky bucket constraint, Parekh and Gallager continue with the analysis of GPS. The maximum delay for session i is defined as

$$D_i = \max_{(A_i, \ldots, A_N)} \max_{\tau \geq 0} D_i(\tau), \tag{5.13}$$

and the maximum backlog for session i is defined as

$$Q_i = \max_{(A_i, \ldots, A_N)} \max_{\tau \geq 0} Q_i(\tau) \tag{5.14}$$

The surprising result is that for locally stable sessions, the maximum backlog and maximum delay are always bounded by

$$\begin{aligned} Q_i &\leq \sigma_i, & (5.15) \\ D_i &\leq \frac{\sigma_i}{g_i}, & (5.16) \end{aligned}$$

where g_i is the minimum session i clearing rate, given by

$$g_i = \min_{m \in P(i)} g_i^m. \tag{5.17}$$

g_i^m in turn is the minimum guaranteed service rate and is given by

$$g_i^m = \frac{\phi_i^m}{\sum_{j=1}^N \phi_j^m} R, \tag{5.18}$$

where R is the rate of the link represented by node m, whereas ϕ_i^m is the weighting factor of node m for session i.

Parekh and Gallager continue with the analysis of stable systems with known internal burstiness, in which the derivation of the session i universal service curve is developed. The universal service curve is a graphical representation of the arrival and serving statistics, which can be used to determine the largest backlog and delay. A result of this analysis is that for non-cut-through GPS, the backlog and delay bounds become

$$\begin{aligned} Q_i &\leq \max_{\tau \geq 0}\{\hat{A}_i(0, \tau) - G_i^K(\tau)\} + K L_i^{\max}, & (5.19) \\ D_i &\leq \max_{\tau \geq 0}\{\min\{t : G_i(t) = A_i(0, \tau) + (K-1)L_i^{\max} - \tau\}\}, & (5.20) \end{aligned}$$

where the curve $G_i^K(t)$ is a continuous curve constructed from the elements of E_i^k, which is a collection of all the pairs $(s_j^m, d_j^m$ for $m = 1, 2, \dots, k-1)$, which can be mathematically represented by

$$E_i^k \bigcup_{m=1}^{k} \bigcup_{j=1}^{n_m} \{(s_j^m, d_j^m)\}. \tag{5.21}$$

Analytically $G_i^k(t)$ has been determined to be

$$G_i^k(t) = \begin{cases} \hat{S}_i^1(0,t), & \text{for } k = 1 \\ \min_{\tau \in [0,t]}\{G_i^{k-1}(\tau) + \hat{S}_i^k(0, t-\tau)\}, t - \tau \leq t_k^B, & \text{for } k \geq 2. \end{cases} \tag{5.22}$$

For more information on the graphing techniques, please see [45].

For non-preemptive service GPS, which is the packetized version known as PGPS, the worst case delay bound becomes

$$D_i^{\text{PGPS}} \leq \max_{\tau \leq 0}\{\min\{t : G_i^K(t) = A_i(0,\tau) + (K-1)L_i^{\max}\} - \tau\} + \sum_{m=1}^{K} \frac{L_{max}}{r^m}, \tag{5.23}$$

which once again is in terms of the $G_i^K(t)$ function, which has to be determined for each specific service type, using linear methods.

An example given in [45], where this problem has been solved is the Rate Proportional Processor Sharing Network. As mentioned before, a GPS scheduler evaluates the share of the total available bandwidth, using the following equation

$$g_i^m = \frac{\phi_i^m}{\sum_{j=1}^{N} \phi_j^m} R. \tag{5.24}$$

Now, the ϕ_i^m are weighting factors. A GPS server is known as a Rate Proportional Processor Sharing (RPPS) Network, if $\phi_i^m = \rho$ for every session i and $m \in P(i)$. According to [45], after translating the symbols into our model, the delay bound for such a network would become

$$D^{\text{RPPS}} \leq \frac{\sigma_i + 2(K-1)L_i^{\max}}{\rho_i} + K \cdot \frac{L_{\max}}{R}. \tag{5.25}$$

5.2.5 Fair queueing algorithms

GPS servers not only have the lowest end-to-end delay deadline, but are also perfectly fair. The problem is how to implement the idealized GPS in a real network. The first attempt analyzed by Parekh and Gallager was the packetized version PGPS, as discussed in the previous section. This made it possible to approximate GPS as closely as possible in a real packet network. The problem with this algorithm was that during the bandwidth allocation process, the PGPS server would run a virtual GPS server in the background as a reference system, with the result that the server has to run two scheduling algorithms, the reference system and the multiplexer itself. This requires too much processing overhead.

A solution to the problem was VirtualClock, first proposed by Zhang [79], which ran a reference timer in the background, instead of a whole virtual GPS scheduling algorithm. The timer could then be used to estimate the service share was meant to receive. The analysis of Stiliadis in [20] is one of the examples that proves that the resulting end-to-end delay bounds are identical to PGPS. But Stiliadis continues to explain that one of the problems with VirtualClock is that it is intrinsically unfair, since a backlogged session can be starved for an arbitrary period of time as a result of excess bandwidth it received from the server when other sessions were idle.

A better implementation, which has a bigger delay bound but is fairer, is Self-Clocked Fair Queueing, first proposed by Golestani [77]. In [54], Golestani analyzes this algorithm. This analysis is based on Parekh and Gallager's analysis of GPS schedulers in [5] and [45]. Golestani's approach is firstly to build on Cruz's network analysis in [27] and [28].

The main concept of importance in the rest of the paper is the potential backlog of a session. Let $W_v(t)$ be the total number of bits from session v transmitted up to time t. If $A_v(t_1, t_2)$ is the arriving traffic of session v with a leaky-bucket characterization of (σ_v, ρ_v) then $a_v(t)$ is the corresponding permit function, so that

$$\begin{aligned} 0 &\leq a_v(t) \leq \sigma_v, \text{ for all } t, & (5.26)\\ A_v(t_1, t_2) &\leq \rho_v(t_2 - t_1) - a_v(t_1, t_2), \text{ for all } t_1 \text{ and } t_2,\ t_1 < t_2. & (5.27)\end{aligned}$$

Using this definition of $a_v(t)$ the potential backlog of session v at time t is defined as

$$w_v(t) \equiv a_v(t) + Q_v(t), \tag{5.28}$$

where $Q_v(t)$ is the backlog of session v at time t, including the residue of partially transmitted packets

$$Q_v(t) \equiv A_v(t) - W_v(t). \tag{5.29}$$

Golestani continues with an introduction of a class of fair queueing algorithms, which he analyzes graphically along the same lines as in [45]. With the results of this analysis he was able to find the bounded backlog $Q_v^{\max}$ and end-to-end delay $D_v^{\max}$ of the traffic of session v travelling through K links to be

$$\begin{aligned} Q_v^{\max} &\leq \sigma_v + \rho_v \cdot \sum_{l=1}^{K} d_v^l & (5.30) \\ D_v^{\max} &\leq \frac{\sigma_v}{r_v} + \sum_{l=1}^{K} d_v^l, & (5.31) \end{aligned}$$

where

$$d_v^l \equiv \frac{\Gamma_v^l}{r_v^l}. \tag{5.32}$$

In this analysis the sessions are assumed to be locally stable. As Golestani mentions, a session v is locally stable in the network N, if it is locally stable at each link along its route, in other words

$$r_v^l = \phi_v^l R > \rho, \tag{5.33}$$

for each link l traversed by v, where r_v^l is the service rate of server v on link l. This is used to define the minimum expected rate of session v as

$$r_v \equiv \min r_v^l, \tag{5.34}$$

which is the bottleneck along server v's path. This should therefore be its fair share of service.

The terms d_v^l each correspond to one link of the route of session v. The term Γ_v is defined to be

$$\Gamma_v \equiv \Delta_v + \gamma_v, \tag{5.35}$$

where

$$\gamma_v \equiv \begin{cases} L_v^{\max} & \text{in general,} \\ \phi_v^l \cdot L_v^{\max}, & \text{if } S \text{ is based on the SCFQ scheme.} \end{cases} \tag{5.36}$$

Δ_v represents the maximum possible increase that the service lag of a session v can undergo, beginning with a time at which v becomes backlogged, which in general turns out to be

$$\Delta_v \equiv \phi_v(\delta_v^{\max} - \delta_v^{(\text{init})\min}), \; v \in V. \tag{5.37}$$

For SCFQ, Golestani showed in [77] that, for our system model,

$$\Delta_v = (1-\phi)L_{\max} + \phi_v \sum_{j \in K, j \neq k} L_j^{\max}. \tag{5.38}$$

It is important to realize that in this equation the session service shares have to be normalized in such a way to add up to less than or equal to one, in other words

$$\sum \phi \leq 1. \tag{5.39}$$

If we substitute the various expressions for SCFQ, the following bound for the end-to-end delay of the server results

$$D_k^{\max} \leq \frac{\sigma_v}{\min_l(\phi_v^l R)} + \sum_{l=1}^{K} \frac{(1-\phi_v^l)L_{\max} + \phi_v^l \sum_{j \in K, j \neq k} L_j^{\max} + \phi_v^l L_v^{\max}}{\phi_v^l R} \tag{5.40}$$

5.2.6 Guaranteed Rate Scheduling

Modern heterogeneous networks carry a wide range of different data types, ranging from real time services like streaming audio and video to data services like FTP and e-mail. Furthermore, every switch or router inside a network might have a different scheduling algorithm. The problem is that most techniques used to determine the end-to-end delay consider only a specific traffic specification and scheduling algorithm, which is therefore not really suitable for a real network. To combat this [60] defined a class of scheduling algorithms known as the Guaranteed Rate scheduling algorithms, examples of which are VirtualClock, Packet-by-packet GPS, and Self-Clocked Fair Queueing.

The defining characteristic of a Guaranteed Rate scheduling algorithm is the delay deadline that these algorithms can guarantee. The delay guarantees are based on guaranteed rate clock values, which is the same concept as the virtual clock idea employed in the

VirtualClock algorithm and the Self-Clocked Fair Queueing algorithm. If L_f^j is the length of the jth packet of flow f, $GRC^i(p_f^j)$ is the guaranteed rate clock value and $A^i(p_f^j)$ the arrival time of packet p_f^j at server i, then

$$GRC^i(p_f^0) = 0 \tag{5.41}$$
$$GRC^i(p_f^j) = \max\{A^i(p_f^j), GRC^i(p_f^{j-1})\} + \frac{L_f^j}{R}, j \geq 1. \tag{5.42}$$

To check whether a scheduler belongs to the class of GR scheduling algorithms, the following definition can be used:

Definition A scheduling algorithm at server i belongs to class GR for flow f if it guarantees that packet p_f^j will be transmitted by $GRC^i(p_f^j) + \beta^i$, where β^i is a constant which depends on the scheduling algorithm and the server.

Goyal, Lam, and Vin proved that VirtualClock, PGPS, and SCFQ all satisfy this condition. For VirtualClock, for example, the virtual clock value for packet p_f^j of flow f at server i is denoted by $VC^i(p_f^0)$, which was computed in reference [60] to be

$$VC^i(p_f^0) = 0 \tag{5.43}$$
$$VC^i(p_f^j) = \max\{A^i(p_f^j), VC^i(p_f^{j-1})\} + \frac{L_f^j}{R}, j \geq 1. \tag{5.44}$$

Using this, a delay guarantee can be found. Let the active flows at server i at time t be denoted by $a^i(t)$. Server i has a capacity of C^i and exceeds its capacity if $\sum_{n \in a^i(t)} R_n > C^i$. If $L_i^{\max}$ is the maximum length of the packet served by server i and $L_{VC}^i(p_f^j)$ is the time at which the transmission of packet p_f^j is completed, then

$$L_{VC}^i(p_f^j) \leq VC^i(p_f^j) + \frac{L_i^{\max}}{C^i}, \tag{5.45}$$

where flow f is assigned rate R. Note that this fits the definition of a GR server, where $\beta^i = \frac{L_i^{\max}}{C^i}$.

Reference [60] repeated the same process for PGPS. The delay guarantee turns out to be

$$L_{PGPS}^i(p_f^j) \leq GRC^i(p_f^j) + \frac{L_i^{\max}}{C_i}, j \geq 1. \tag{5.46}$$

Table 5.2: β-values of GR servers

Server	β^i
VirtualClock	$\frac{L_i^{\max}}{C^i}$
PGPS	$\frac{L_i^{\max}}{C^i}$
SCFQ	$\sum_{m\in c^i \wedge m\neq f} \frac{L_m^{\max}}{C^i}$

This also corresponds to the definition of a GR server with $\beta^i = \frac{l_i^{\max}}{C^i}$.

For SCFQ, the analysis in [60] is more in depth and the departure time of packet p_f^j, the jth packet of flow f, turns out to be

$$L^i_{SCFQ}(p_f^j) \leq GRC^i(p_f^j) + \sum_{m\in c^i \wedge m\neq f} \frac{L_m^{\max}}{C^i}, \tag{5.47}$$

where c^i is the set of flows serviced by server i employing SCFQ. The result is that in GR's definition, $\beta^i = \sum_{m\in c^i \wedge m\neq f} \frac{L_m^{\max}}{C^i}$. Note that here the ratio $\frac{L_m^{\max}}{C^i}$ is summed for all flows m that employ SCFQ, except for flow f. The β^i values for the three schedulers have been neatly summarized in Table 5.2.

Having found the delay guarantees for each scheduler, [60] continues finding the end-to-end delay bound. The first approach would be to find the summation of the maximum delay at each server along the path from the source to the destination. But as mentioned in [45], [54], and [60], this gives bounds which are too loose. Instead the network has to be analyzed as a whole to determine end-to-end delays.

The result of the analysis is that if the scheduling algorithm at each of the servers on the path belongs to GR for flow f, then the end-to-end delay of packet p_f^j is given by

$$d_f^j \leq GRC^l(p_f^j) - A^l(p_f^j) + (K-1) \max_{n\in[1\ldots j]} \frac{L_f^n}{R} + \sum_{n=1}^{n=K} \alpha^n, \tag{5.48}$$

where $\alpha^n = \beta^n + \tau^{n,n+1}$ and K is the number of servers on the path of the flow. $\tau^{n,n+1}$ is the propagation delay between servers n and $n+1$.

The problem with (5.48) is that the arrival statistics $A^l(p_f^j)$ are not known. To find this the source traffic has to be specified. The first example is leaky bucket traffic. The result is that if flow f conforms to a Leaky Bucket with parameters (σ_f, ρ_f) and the scheduling algorithm at each of the servers on the path of a flow belongs to GR for the flow, then

$$D_v^j \leq \frac{\sigma_f + (K-1)L_v^{\max}}{\rho_f} + \sum_{n=1}^{K} \alpha^n, \tag{5.49}$$

where again $\alpha^n = \beta^n + \tau^{n,n+1}$ and K is the number of servers on the path of the flow.

A second example is an Exponentially Bounded Burstiness (EBB) process. A flow f conforms to an EBB process with parameters $(R_f, \Lambda_f, \gamma_f)$, if

$$\Pr(AP_f^i(t_1, t_2) \geq R_f(t_2 - t_1) + x) \leq \Lambda_f e^{-\gamma_f x}, x \geq 0, t_1 \leq t_2. \tag{5.50}$$

The result is that, if flow f conforms to EBB with parameters $(R_f, \Delta_f, \gamma_f)$ and the scheduling algorithm at each of the servers on the path of a flow belong to GR, then

$$\Pr\left(d_f^j \geq x + (K-1)\frac{L_f^{\max}}{R^f} + \sum_{n=1}^{K} \alpha^n\right) \leq \Lambda_f e^{-\gamma_f x r_f}, x \geq 0, \tag{5.51}$$

with $\alpha^n = \beta^n + \tau^{n,n+1}$ and K being the number of servers on the path of the flow. In the simulation comparison in this chapter, we only look at the leaky bucket bound, in other words, in the context of this chapter we focus on (5.49) and ignore (5.51). For more information on EBB processes and an explanation of the meaning of the various parameters, please see [60].

5.2.7 Latency Rate Servers

In [20], Stiliadis introduces *Latency-Rate Servers* or LR Servers for short. He starts off by introducing the model. A packet switch is considered where a set of V connections share a common output link. ρ_i denotes the rate allocated to connection i. The servers are assumed to be non-cut-through devices, where $A_i(\tau, t)$ are the arrivals from session i during the interval (τ, t) and $W_i(\tau, t)$ the amount of service received by session i during the same interval. The following terms are then defined:

System busy period This is a maximal interval of time during which the server is never idle.

Backlogged period for session i This is any period of time during which packets belonging to that session are continuously queued in the system. Therefore, if $Q_i(t) = A_i(0,t) - W_i(0,t)$ represents the amount of queued traffic in the server at time t, a connection is backlogged at time t if $Q_i(t) > 0$.

Session i busy period This is a maximal interval of time $(\tau_1, \tau_2]$ such that for any time $t \in (\tau_1, \tau_2]$, packets of connection i arrive with a rate greater than or equal to ρ_i, or,

$$A_i(\tau_1, t) \geq \rho_i(t - \tau_1). \tag{5.52}$$

Stiliadis carries on with an explanation of the difference between a session backlogged period and a session busy period. A session busy period is defined with respect to a hypothetical system, where a backlogged connection i is serviced at a constant rate ρ_i. A busy period may therefore contain intervals during which the actual backlog is zero.

We are trying to find the worst-case behaviour of a system, which here refers to the case where traffic is served at the minimum guaranteed rate. The session busy period therefore provides a convenient means to bound the queueing delays within the system.

Reference [20] continuous with the introduction of the Latency-Rate Server. The servers in this class are characterized by the latency Θ_i and the minimum allocated rate ρ_i. The following definition sets LR servers apart from other servers:

Latency Rate Servers A server S belongs to the class LR if and only if for all times t after time τ that the j-th busy period started and until the packets that arrived during this period are serviced,

$$W_{i,j}^S(\tau, t) \geq \max(0, \rho_i(t - \tau - \Theta_i^S)). \tag{5.53}$$

Θ_i^S is the minimum non-negative number that can satisfy the above inequality.

This definition bounds the minimum service offered to session i during a busy period. The latency Θ_i^S represents the worst-case delay that a packet can experience, when it arrives at an empty queue. Reference [20] assumes that traffic from session i is leaky-bucket shaped

Table 5.3: Latencies of six schedulers

Server	Latency
GPS	0
PGPS	$\frac{L_i}{\rho_i} + \frac{L_{max}}{r}$
SCFQ	$\frac{L_i}{\rho_i} + \frac{L_{max}}{r}(V-1)$
VirtualClock	$\frac{L_i}{\rho_i} + \frac{L_{max}}{r}$
Deficit Round Robin	$\frac{3F-\phi_i}{r}$
Weighted Round Robin	$\frac{F-\phi_i+L_c}{r}$

at the source and therefore

$$A_i(\tau, t) \leq \sigma_i + \rho_i(t-\tau) \tag{5.54}$$

during any time interval $(\tau, t]$. Assuming that session i is allocated a minimum rate ρ_i in the network, the following result holds:

Theorem 1 The maximum delay D_i^K and the maximum backlog Q_i^K of session i after the Kth node in an arbitrary network of LR-servers are bounded as

$$D_i^K = \frac{\sigma_i}{\rho_i} + \sum_{j=1}^{K} \Theta_i^{(S_j)}; \tag{5.55}$$

$$Q_i^K = \sigma_i + \rho_i \sum_{j=1}^{K} \Theta_i^{(S_j)}; \tag{5.56}$$

where $\Theta_i^{(S_j)}$ is the latency of the jth server on the path of the session.

The values in Table 5.3 can be substituted into (5.55) to find the delay bounds for six different schedulers. In Table 5.3, L_i is the maximum packet size of session i and L_{max} the maximum packet size among all the sessions. C_i is the maximum normalized service that a session may receive in a PGPS server in excess of that in the GPS server. F is the maximum frame size in a weighted round-robin and deficit round-robin scheduler, while ϕ_i is the amount of traffic in the frame allocated to session i. In weighted round-robin, L_c is the size of the fixed packet.

5.3 Comparison of analytical bounds

In the previous section we summarized a wide range of analytical delay bounds that were published over the last fifteen years. We would like to compare the various bounds to one another in order to determine their tightness. To do this we introduced a common system model, which we applied to each analytical result to get some numerical results. It was quite difficult to determine realistic values for the various constants and variables. We decided to keep as many of the variables, like the maximum packet length for example, the same as in the system model for the simulations. Many variables allowed for different values between the various sessions. An example is the burst size parameter σ_v, which allows for a different burst size for each session. These were standardised to one value for all sessions.

To summarise, the maximum number of sessions in the network coinciding at one node V was kept the same as the simulation model at $V = 15$ nodes. The maximum path length K, which translates to the maximum number of nodes that a packet has to hop along, to travel from its source to its destination, was varied between 1 and 10, to show how the end-to-end delay varies as the path length changes. The maximum packet length was kept at $L_{max} = 10$Kb, while the average packet length was once again $L = 1$Kb. The network load, denoted by *Load*, was varied between the values of $Load = 0.1$ to $Load = 0.9$. R denotes the rate at which each node serves packets, which corresponds to that node's bandwidth. In this model the rate was chosen to be $R = 1$Mbps.

All of the analytical models mentioned in this chapter rely on Leaky Bucket traffic shaping to analyze the end-to-end queueing delay of packets. The arrival statistics of traffic are therefore characterized by $A \sim (\sigma, \rho)$. In other words, an arriving burst of data will not exceed σ bits, while the average arrival rate of traffic will not exceed ρ bits per second. This effectively smooths out the traffic bursts. In our model we fixed the maximum burst size to $\sigma = 20$Kb. In most of the analytical models a normalized version of the burst limit, $\sigma_{norm} = \frac{\sigma}{R}$ is used instead, since this makes the analysis independent of the actual serving capacity of each node. In [28], Cruz mentions that $\sigma_{norm} \geq L_{max}(1 - \rho_{norm})$. This is useful when determining a possible value for σ_{norm}. The average arrival rate is given

Table 5.4: System Model with Substitution Values

Symbol	Value	Meaning
V	15	no of sessions
K	$1 \rightarrow 10$	no of nodes
L_{max}	10Kb	max packet length
L	1Kb	average packet length
$Load$	$0.6 \rightarrow 0.9$	network load
R	1Mb/s	server capacity
ρ	$\frac{R\times(1-Load)}{V}$	average arrival rate
ρ_{norm}	ρ/R	normalized arrival rate
σ	20Kb	burstiness parameter
σ_{norm}	σ/R	normalized burstiness bound
ν	0.1	TS-EDF requires $\nu > 0$
ϕ	$\frac{1}{V}$	GPS weight (we assume RPPS model)
τ	$1\mu s$	propagation delay
F	$2L_{\text{max}}$	frame size for Round Robin servers
L_c	L	fixed cell length for Round Robin servers

by $\rho = \frac{R\times(1-Load)}{V}$. Again, many of the analytical models prefer to use the normalized version $\rho_{norm} = \frac{\rho}{R} = \frac{1-Load}{V}$.

Another factor that has been included by [60] is the transmission delay τ of packets. In wireless networks the distances between nodes are usually in the order of tens of meters. For a transmission distance of 300m, the propagation delay is given by $\tau = \frac{300\text{m}}{3\times10^8\text{ms}^{-1}} = 1\mu s$. At packet lengths of over 1000bits this delay really only becomes significant at transmission rates in the order of hundreds of megabits or even gigabits per second. For current wireless networks these transmission rates are not feasible. In the case of wireless networks the transmission delays are therefore negligible compared to other delays experienced by packets along their journey. In a wired network, the situation is different. The transmission speed is slower in mediums like copper and the transmission distances can be

Table 5.5: Delay Bounds

Scheduler	Delay bound
FCFS	$\left[L_{max} V \rho_{norm} + \frac{\sigma_{norm}}{1-\rho_{norm}}\right] \cdot \left[(V \rho_{norm})^{-K} - 1\right]$
TS-EDF	$\left[\frac{K V \sigma_{norm}}{(1-Load)^4}\right]^{1+\nu}$, where $\nu > 0$
PS-EDF	$\frac{\sigma_{norm}+1/(1-Load)}{\rho_{norm}} + \frac{KV}{(1-Load)^3} \cdot \log_2\left(\frac{V}{\rho_{norm}}\right)$
RPPS	$\frac{\sigma+2(K-1)L_{max}}{\rho} + K \cdot \frac{L_{\max}}{R}$
SCFQ	$\frac{\sigma}{\phi R} + \frac{K L_{max}}{\phi R} \cdot (1 + V\phi)$
GR-VirtualClock	$\frac{\sigma+(K-1)L_{max}}{\rho} + K \cdot \frac{L_{\max}}{R} + \tau$
GR-PGPS	$\frac{\sigma+(K-1)L_{max}}{\rho} + K \cdot \frac{L_{\max}}{R} + \tau$
GR-SCFQ	$\frac{\sigma+(K-1)L_{max}}{\rho} + K(V-1) \cdot \frac{L_{\max}}{R} + \tau$
LR-GPS	$\frac{\sigma}{\rho}$
LR-PGPS	$\frac{\sigma}{\rho} + K \cdot \left[\frac{L_{\max}}{\rho} + \frac{L_{max}}{R}\right]$
LR-SCFQ	$\frac{\sigma}{\rho} + K \cdot \left[\frac{L_{\max}}{\rho} + (V-1) \cdot \frac{L_{max}}{R}\right]$
LR-VirtualClock	$\frac{\sigma}{\rho} + K \cdot \left[\frac{L_{\max}}{\rho} + \frac{L_{max}}{R}\right]$
LR-DRR	$\frac{\sigma}{\rho} + K \cdot \left[\frac{3F-\phi}{R}\right]$
LR-WRR	$\frac{\sigma}{\rho} + K \cdot \left[\frac{F-\phi+L_c}{R}\right]$

significantly larger. The transmission rates are also often in the gigabit per second range, sometimes even terrabits per second are sent.

In our model we have included the propagation delay as $\tau = 1\mu$s. This was done more as a sentimental notion to show that this factor might be of importance in a different system model. In our case this factor makes no significant difference to the resulting bounds of our system model. For the convenience of the reader, we have summarized the simpler model with all the values of the variables in Table 5.4.

Using the set of symbols from Table 5.1, we can now proceed to translate the delay bounds presented, to our system model. The result can be found in Table 5.5.

5.3.1 Results

To compare the end-to-end delay bounds of the various schedulers, it is easiest to substitute the values of the constants and variables in Table 5.1 into the delay bound expressions in Table 5.5 and plot the results on a graph. The following sections contain the graphs for each scheduling family.

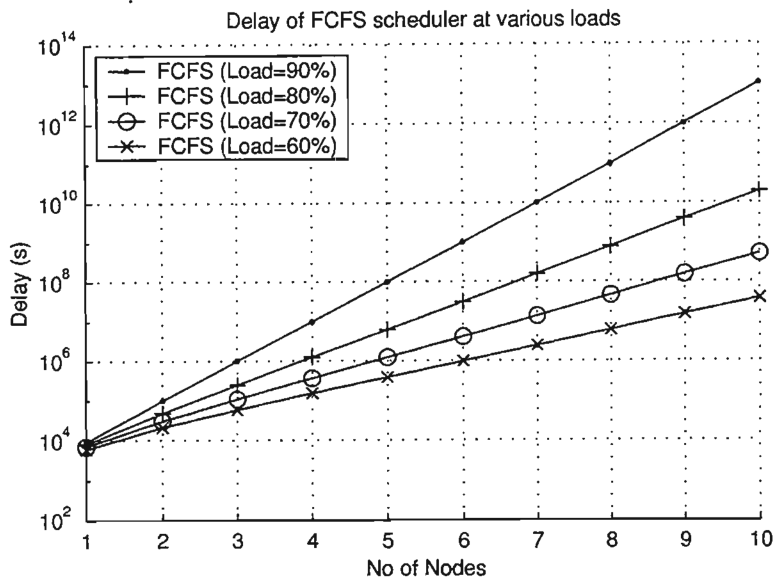

Figure 5.1: FCFS end-to-end delay

FCFS

The First-Come-First-Serve scheduler is one of the simplest schedulers. The resulting delay bounds in Fig. 5.1 are therefore quite surprising. Even for a single node the worst-case delays to be expected are 10^4 seconds. As the number of nodes increase and the network load increases, so the worst-case end-to-end delays shoot up to incredible numbers. For example, for a relatively small 60% load, the worst-case end-to-end delay to be expected for 10 nodes lies between 10^7 and 10^8 seconds. This corresponds to over 115 days, which is unacceptable for any commercial network. For a 90% load, the worst-case delay lies somewhere around 10^{13} seconds, which is more than three hundred thousand years. The explanation for these unreasonable delays is that this analysis was performed by Cruz

[28] in 1991 and was one of the first delay analyses performed in this field. The resulting bound is exponential and very loose, which was touched on already at the beginning of this chapter in section 5.2.3. Even though the resultant delay guarantees do not correspond to real network delay measurements, they still guarantee a maximum limit which will never be exceeded. The importance of this result should not be underestimated, since it was one of the first papers that proved that there is a maximum limit and the delays cannot simply grow to infinity.

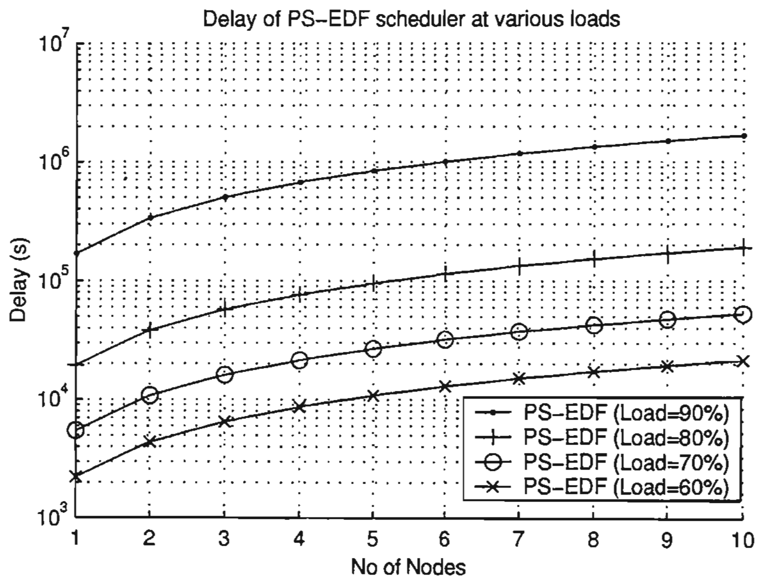

Figure 5.2: PS-EDF end-to-end delay

EDF

Earliest-Deadline-First schedulers can be divided into two groups, one with temporary sessions and one with permanent sessions. Although the resulting delay bounds for PS-EDF in Fig. 5.2 are by far not as incredible as those for FCFS, the fact that for a 60% load a network with only one node can still be expected to experience a delay of over 2000 seconds is unacceptable. For ten nodes, the delay increases to 20,000 seconds or five-and-a-half hours. For a 90% load, the delay increases to just under two million seconds, which corresponds to 23 days. Once again, these figures do not correspond to real network delays, but only bound the end-to-end delay of the system. What makes this analysis valuable is

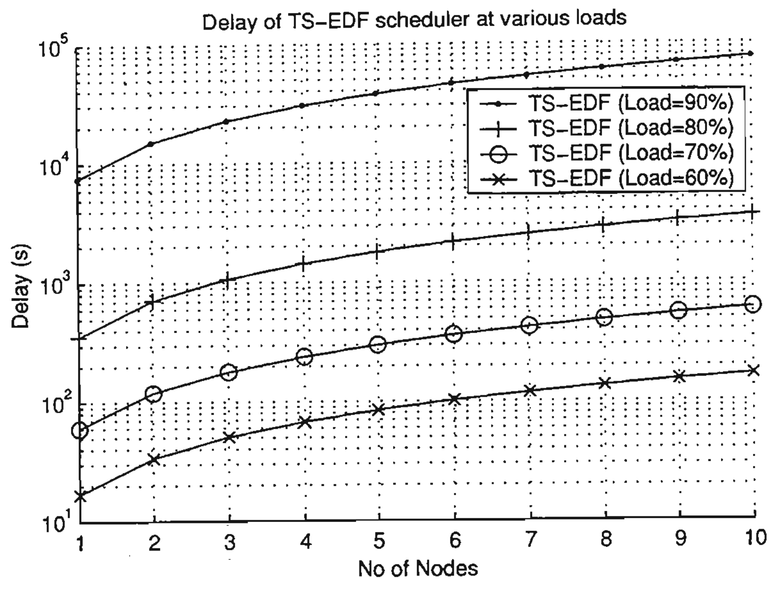

Figure 5.3: TS-EDF end-to-end delay

that the exponential bounds of FCFS have been replaced with polynomial bounds, which turn out to be much tighter.

A network employing a temporary sessions EDF scheduler has even tighter delay bounds. As can be seen in Fig. 5.3, the worst-case delay for a single node network with a 60% load will not be higher than 20 seconds, whereas for ten nodes, the delay will be less than 200 seconds. In general the resulting bounds seem to be approximately a hundred times larger for PS-EDF than what they are for TS-EDF.

PGPS

Analyses for the GPS family of schedulers, with all its variations and packetized implementations, have produced much tighter end-to-end delay bounds than those for FCFS and EDF, as is evident in Fig. 5.4, for example. Here, the end-to-end delay of three different analyses of PGPS are compared with that of GPS. GPS is the ideal, non-implementable scheduler, which at an 80% load has a delay bound of under 2 seconds, regardless of the number of nodes it travels through. The reason for this is that sessions are perfectly isolated from one another. They therefore only experience a queueing delay when they meet

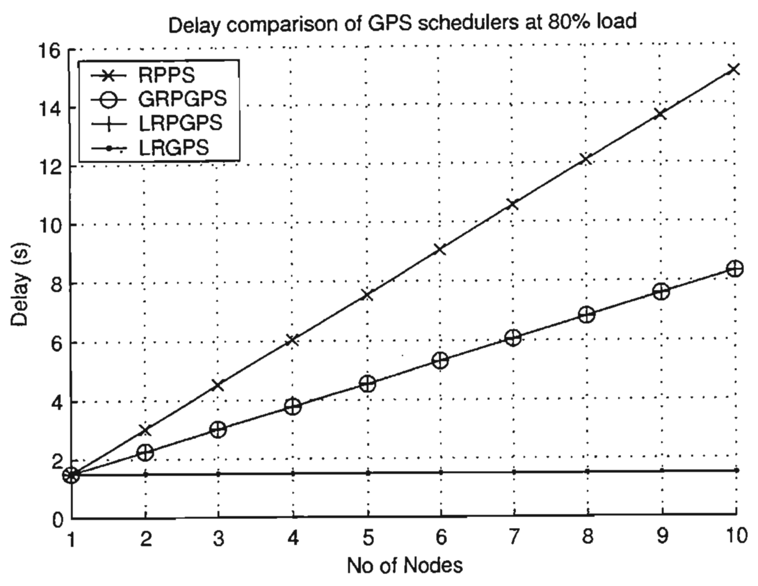

Figure 5.4: PGPS end-to-end delay at 80% load

the Leaky Bucket traffic shaper, which smooths out the bursts. Once the traffic shaping has taken place, the data travel from node to node without experiencing any further queueing delay. Note that this analysis only takes queueing delays into consideration and not transmission delays.

In a real implementation packets experience further queueing delay as they travel through the network. Streams are not perfectly isolated, since they now travel through the network in non-divisible packets. In Fig. 5.4 three end-to-end delay bounds of PGPS can be seen. The GRPGPS bound, published in [60], and LRPGPS, published in [20], lie on top of one another. In [20] Stiliadis proves that this is the tightest bound for PGPS, by giving an example where the bound is reached. In other words, the bound cannot be smaller, or else there is an example where the delay exceeded the bound, and the bound seizes to exist.

The third bound, which we refer to as RPPS, was published in [45], a year before the other two. As can be seen in Fig. 5.4, it is looser than the other two. In Fig. 5.5 we look at RPPS in isolation and how its delay bound changes over various loads. For a 60% load, the end-to-end delay bound lies around 1 second for 1 node, while for 10 nodes the worst-case delay increases to seven and a half seconds. As the load increases, this figure becomes worse. For a load of 90%, the delay is 3 seconds for a single node and linearly

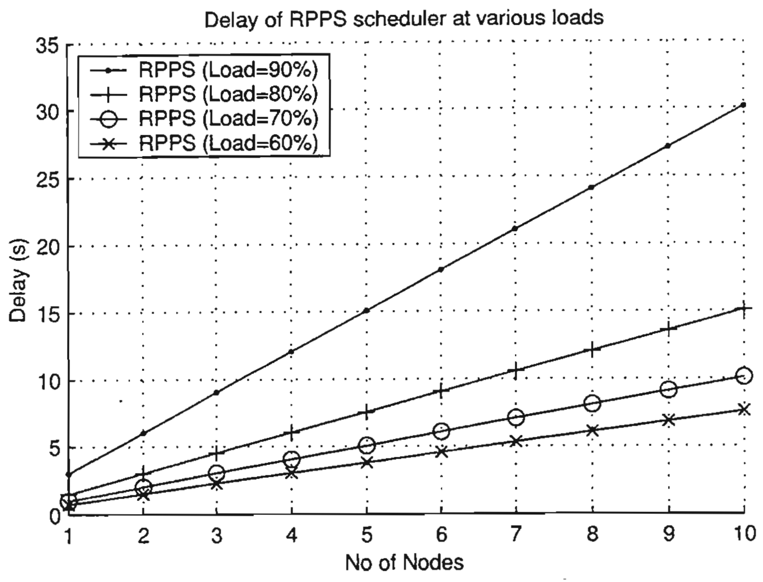

Figure 5.5: RPPS end-to-end delay at various loads

increases to 30 seconds for 10 nodes.

The same process was repeated for GRPGPS and LRPGPS. Since they have the same delay bounds, it was only necessary to look at LRPGPS. Fig. 5.6 shows that if one only looks at a single node, the worst-case delay is the same as for RPPS. But as the number of nodes increase, so the delay bound for RPPS outgrows those of GRPGPS and LRPGPS, until they are finally almost twice as large at 10 nodes.

VirtualClock

When VirtualClock was introduced into the scheduling world, researchers were greatly excited by the fact that this was a scheduler which was far less complex than PGPS, yet still had the same delay bounds. That this is the case can be seen in Fig. 5.7. Here two delay bounds of VirtualClock, published in [60] and [20], are compared with LRPGPS. As can be seen, both GRVC and LRVC both perform exactly the same as LRPGPS.

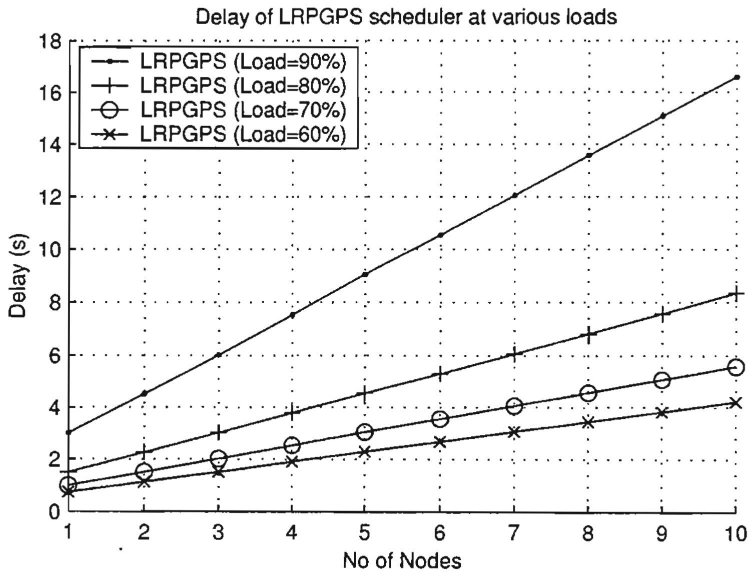

Figure 5.6: LRPGPS end-to-end delay at various loads

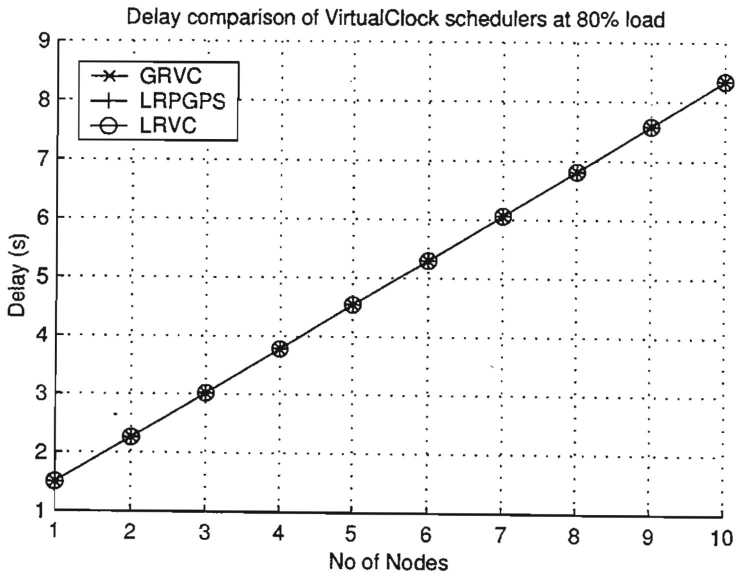

Figure 5.7: VirtualClock end-to-end delay

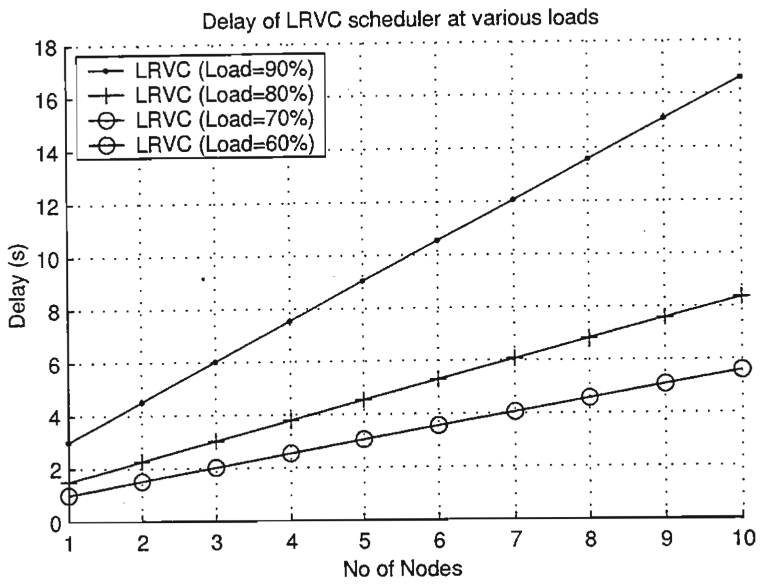

Figure 5.8: VirtualClock end-to-end delay

To further confirm this, LRVC was plotted for various loads in Fig. 5.8. If one compares this with Fig. 5.6 one can see that even under various load conditions, VirtualClock has exactly the same worst-case end-to-end delay bound as PGPS. This is to be expected, since they have the same equation in Table 5.5.

SCFQ

The problem with VirtualClock, as explained before, is that it is not fair and under the right circumstances the traffic flow can get blocked indefinitely for one or more channels. A better implementation is SCFQ. Unfortunately, its delay bounds are worse than those of PGPS. In Fig. 5.1, the delay bounds of three SCFQ analyses are compared with the one for LRPGPS. As expected, the resulting delay bounds of GRSCFQ and LRSCFQ are greater than for LRPGPS. Once again, the analyses of Guaranteed Rate servers [60] and Latency Rate servers [20] result in exactly the same bounds. The SCFQ analysis performed by Golestani in [54], on the other hand, has rather surprising results. The bounds seem to be much tighter than those of the other two analyses. Even PGPS has a greater delay bound than Golestani's SCFQ analysis. If one compares this result with that of LRGPS in Fig. 5.4, then Golestani's result is still by far the best.

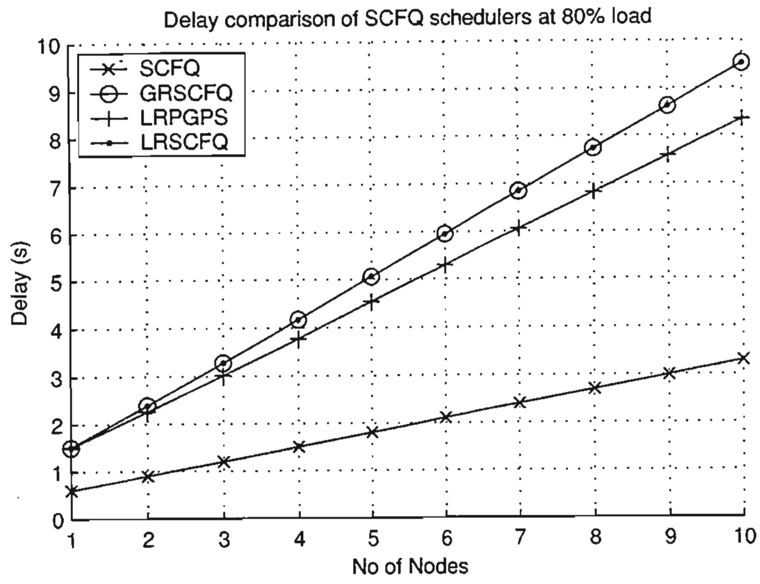

Figure 5.9: SCFQ end-to-end delay

The reason for this strange result is that Golestani's analysis results in a similar expression for the worst-case end-to-end delay bounds as Stiliadis and Goyal. For Stiliadis and Goyal, the delay expression is:

$$D = \frac{\sigma}{\rho} + K \cdot \left[\frac{L_{\max}}{\rho} + (V-1) \cdot \frac{L_{\max}}{R}\right]. \tag{5.57}$$

Now, Golestani's version looks quite similar, especially if one regroups the terms slightly:

$$D = \frac{\sigma}{\phi R} + K \cdot \left[\frac{L_{\max}}{\phi R} + V \cdot \frac{L_{\max}}{R}\right]. \tag{5.58}$$

If we were to set $\rho = \phi R$, the two equations would be identical, except for the V term, which in Stiliadis and Goyal's papers is $(V - 1)$. The problem is that the average arrival rate ρ is not the same as $r = \phi R$, which is the average service rate. As a matter of fact, in order for the server to be stable, in other words not to overflow, the arrival rate has to be less than the service rate, in other words $\rho < \phi R$.

The reason why Golestani's delay bound is much better than those of Stiliadis and Goyal is that the service rate $r = \phi R$ is independent of the network load. The SCFQ curve in Fig. 5.9 does not change as the load changes, while all the other curves do. In [54], Golestani compares his expression for SCFQ's worst-case end-to-end delay bound to that

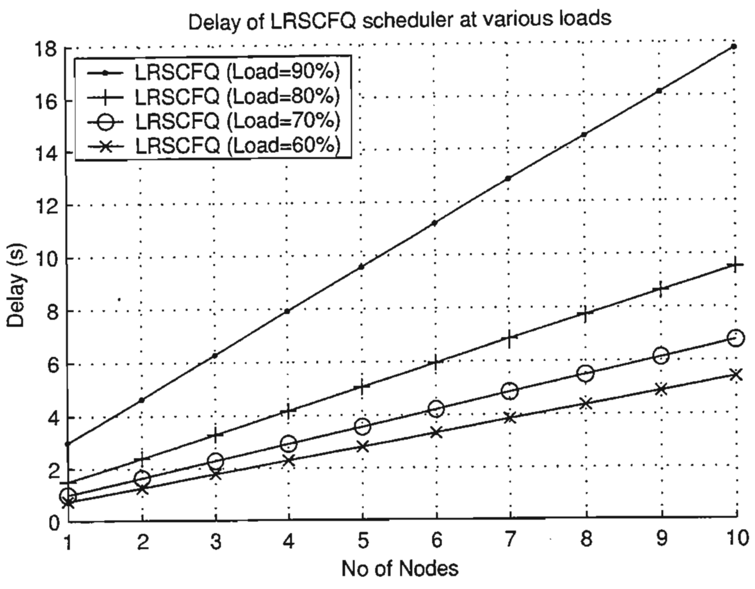

Figure 5.10: LRSCFQ end-to-end delay

of PGPS. If the symbols of Golestani's expression for PGPS are translated to our system model, it becomes

$$D = \frac{\sigma}{\phi R} + (K-1)\cdot\left[\frac{L_{\max}}{\phi R} + \frac{L_{\max}}{R}\right]. \tag{5.59}$$

This is exactly the same expression as that of LRPGPS, except that again Golestani lets $\phi R = \rho$. From this expression we can deduce that Golestani's version of the worst-case delay of a GPS server would be

$$D = \frac{\sigma}{\phi R}. \tag{5.60}$$

This expression is again independent of the network load, which is unrealistic. When traffic arrives at a Leaky Bucket shaper, the bursts get smoothed out, in other words, the average transmission rate is reduced. The result is that between bursts, a traffic backlog is created at the traffic shaper. Now according to Golestani's expression, the average arrival rate of traffic will not have an impact on the queue length, with the result that the worst-case delay stays the same regardless of the arrival rate of traffic. This is not true. A simple everyday example that proves this, is traffic on a highway. During rush-hour a large number of cars are travelling along the highway. As long as the cars maintain a constant speed, the traffic flows smoothly. But as soon as a construction sight appears and the traffic has to slow down even slightly, it starts backing up and the highway is suddenly congested. At mid-night, on the other hand, the arrival rate of cars is very low and accordingly no congestion takes place.

If we were to correct Golestani's expression for SCFQ delay, and replace all the ϕR expressions with ρ, then the resulting bound is slightly worse than that of Stiliadis and Goyal, because of Golestani's V term, which should be $V - 1$ according to Goyal and Stiliadis.

Fig. 5.10 shows the worst-case delay behaviour of LRSCFQ under various load conditions. For a 60% load the delay varies between one and five seconds, as the number of nodes vary between one and ten. For a 90% load, the delay varies between five and eighteen seconds. If one compares these results with those of LRPGPS in Fig. 5.6, then one finds that the difference between SCFQ and PGPS is relatively small. For the single node case, the two algorithms perform the same. For the ten node case, the worst-case delay of SCFQ is one second larger than that of PGPS. The difference between VirtualClock and SCFQ is therefore minor in terms of end-to-end delay, but SCFQ is a fair algorithm, which is not the case with VirtualClock.

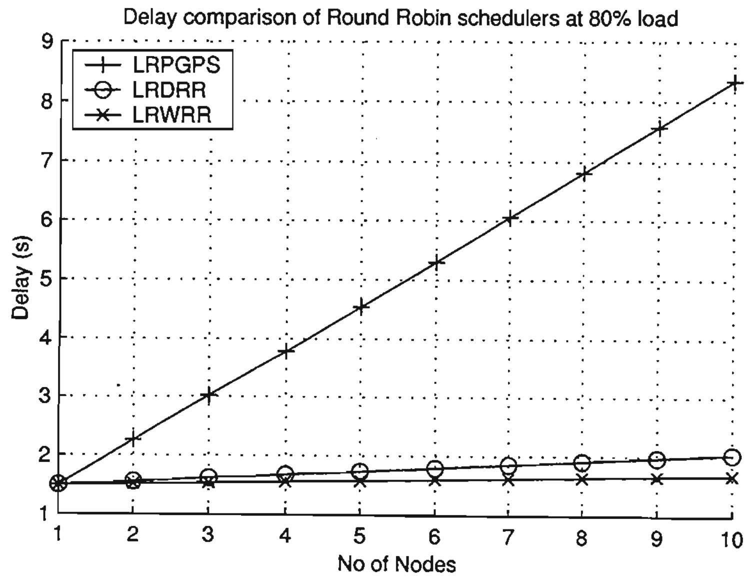

Figure 5.11: Round Robin end-to-end delay

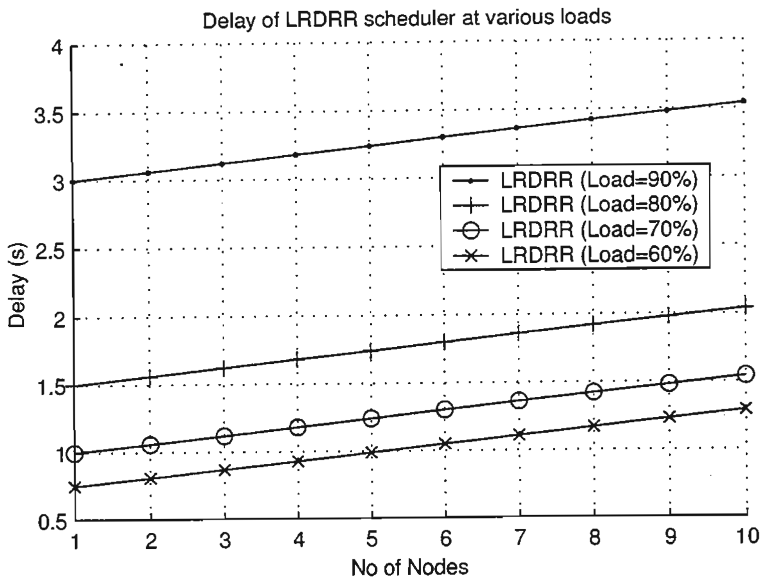

Figure 5.12: Deficit Round Robin end-to-end delay

Round Robin

Fig. 5.11 compares Deficit Round Robin and Weighted Round Robin with PGPS. Strictly speaking, the Round Robin schedulers should not be compared with PGPS, since the Round Robin analysis involves fixed cell lengths and frames, which affect the delay behaviour of the two schedulers. PGPS, on the other hand, is only concerned with packets. The two families of protocols would usually operate on completely different network layers, with Round Robin being employed in a switch in the data link layer, while PGPS is employed in a router in the network layer. But we have still included the two Round Robin schedulers, for interest sake. For the particular frame size and cell size we have chosen, DRR and WRR have very low end-to-end delay bounds, which almost approach those of the ideal GPS. In Fig. 5.12 and Fig. 5.13, the delay behaviour of each individual scheduler can be seen under various load conditions. Whereas the delay of DRR varies by just over half a second from that of GPS, as its travels through ten nodes, WRR varies by less than a quarter of a second. What is also interesting is that, apart from the initial delay caused by the Leaky Bucket traffic shaper, the delay does not increase as the network load increases. Although this behaviour is excellent compared to the other scheduling schemes presented, the downside is that DRR is not able to achieve traffic differentiation, while

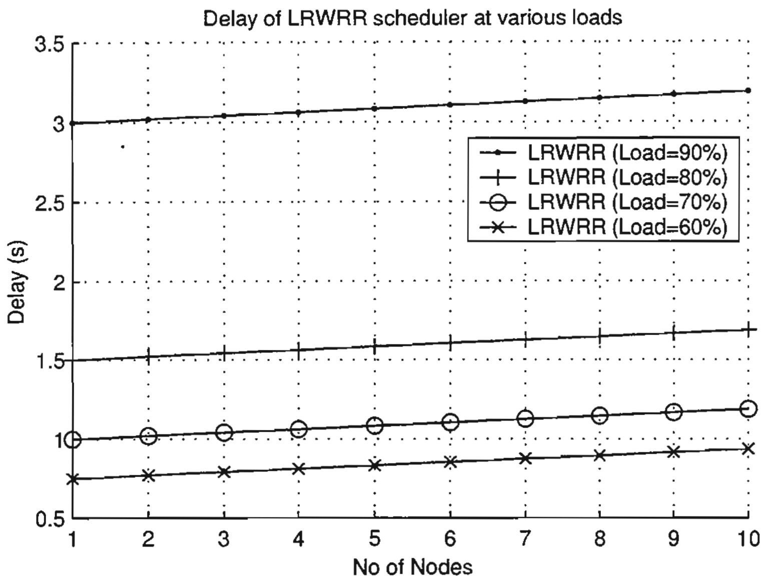

Figure 5.13: Weighted Round Robin end-to-end delay

WRR uses weights to achieve some crude form of differentiation.

5.4 Summary of Results

We would like to conclude this chapter with a brief overview of the results obtained. This chapter was started with an overview of the end-to-end delay bound analyses performed on

- FIFO in [27] (single-node case), [28](multi-node case)
- TS-EDF and PS-EDF in [40]
- GPS in [5] (single-node case), [45] (multi-node case), [20](LR-server)
- PGPS in [5] (single-node case), [45] (multi-node case), [60] (GR-server), [20] (LR-server)
- VirtualClock in [60] (GR-server), [20](LR-server)
- SCFQ in [54], [60] (GR-server), [20] (LR-server)

- DRR and WRR in [20]

In general the results can be summarized as follows:

- FIFO is one of the simplest schedulers. The delay bound acquired from the literature is very loose and turns out to be exponential. Large loads and a large number of nodes to be traversed make the resultant delay limit in the order of many years.
- TS-EDF and PS-EDF are two EDF implementations, one with temporary sessions, one with permanent sessions, respectively. The delay bound presented for both the schedulers was polynomial, which results in a much tighter bound, than was the case for FIFO. The delay bound of TS-EDF is tighter than that of PS-EDF.
- GPS traffic has the lowest end-to-end delay of all schedulers. It only experiences a queueing delay while it is being shaped. Once the traffic shaping is complete, it flows through the various links and nodes without any further queueing delay.
- PGPS, the packet-by-packet implementation of GPS, achieves the smallest delay-bounds of a real implementation, which is able to obey QoS constraints. The various delay bounds that were presented were found to be tight.
- VirtualClock is a less complex version of PGPS, but with the same delay bounds as PGPS. The delay bounds were found to be tight.
- SCFQ is a fairer scheduler than VirtualClock, but with a larger delay bound. The bound was tight.
- DRR and WRR both achieve better delay bounds than PGPS, but are unable to meet any QoS constraints. Out of the two, DRR has the smaller delay bound.

Chapter 6

Conclusion

Modern networks require strict QoS adherence. One of the most important components employed in such networks to achieve QoS constraints are advanced schedulers. In the world of wireless networks, CDMA has become a focus of research, due to its ability to fit more users into a bandwidth at a desired signal-to-interference ratio than both FDMA and TDMA. Schedulers designed for wireless networks are able to monitor the channel conditions and take advantage of below-average channel attenuations.

6.1 Dissertation Summary

The introductory chapter gives a brief overview of the evolution of the wireless network, including third generation and fourth generation wireless communication systems. A motivation for the research that was conducted and an overview of this dissertation are given. Finally, the original contributions in this dissertation are listed.

Chapter 2 contains a literature survey of wired and wireless schedulers. It starts off with an overview of important scheduling concepts, which include QoS and useful scheduling features. This is followed by an overview of wired schedulers. The schedulers that were discussed include FIFO, FIFO+, Round Robin, Priority Queueing, Earliest Deadline First, Stop-and-Go, and finally the large family of Fair Queueing schedulers, which are based on the ideal General Processor Sharing fluid model. Many packetized implementations of GPS exist. Some of them are WFQ or PGPS, VirtualClock, SCFQ, SFQ, RPS, and CBQ. The next section introduces the wireless schedulers, which include CSDPS, IWFQ, CIF-Q, ELF, and SBFA. For CDMA specifically, numerous power distribution algorithms are listed, which include Packet Proportional, Bit Proportional, Work Proportional, Uniform Processor Sharing, Work Processor Sharing, Rate Processor Sharing, Earliest Deadline First, and Largest Weighted Delay First.

In Chapter 3, a new wireless scheduling algorithm, known as Wireless Fair Largest Weighted Delay First (WF-LWDF), is introduced, which was built on the Modified Largest Weighted Delay First (M-LWDF) foundation. M-LWDF is capable of serving multiple packets simultaneously, while taking channel conditions into consideration during the packet picking process. Both these attributes make M-LWDF an advanced CDMA-based scheduler, capable of taking advantage of temporary channel quality improvements. The problem with M-LWDF is that its rate-allocation algorithm is unfair. WF-LWDF uses the same packet picking algorithm as M-LWDF, but distributes the available power in a GPS fashion. Through a set of simulations, this was shown to make WF-LWDF a superior scheduler, compared to M-LWDF. The aim of the simulations was to measure the throughput, delay, power utilization and the number of packets dropped. The schedulers that were simulated in this chapter were EDF, LWDF, M-LWDF, WF-LWDF, SM-LWDF, and W-LWDF. The simulation results showed that throughput and delay are proportional to the delay deadlines of data groups. What was of great importance was the result that schedulers that serve multiple packets at the same time are able to increase their power utilization and their throughput, which decreases the number of packets that are dropped because of the queues overflowing. Another important result was that distributing smaller portions of power to more packets increases the utilization of the available resources at a high traffic load.

Chapter 4 served to compare the fairness of M-LWDF and WF-LWDF, to prove that WF-LWDF improves the fairness of M-LWDF, as it was intended to. This was achieved both through simulations, and by finding a Chernoff bound of the fairness of M-LWDF and WF-LWDF. Finding a statistical fairness bound by using Chernoff's work is a new approach that, as far as we are aware, has never been published before.

The fairness of a scheduler is a measure of how well it is able to comply with its QoS guarantees. The chapter starts off with an overview of the fairness concept and a brief history of how it was defined mathematically. In the case of M-LWDF and WF-LWDF, no guaranteed service rates are given. Instead a delay deadline is available. The fairness definition therefore had to be altered slightly, to define the minimum guaranteed rate in terms of delay deadlines in a CDMA environment. The resultant fairness measurements showed that WF-LWDF, which we introduced in this dissertation, is a fairer scheduling algorithm than M-LWDF, as intended. Chernoff bounds were calculated from the throughput results of Chapter 3, using the knowledge that a Gamma distribution best fits the throughput curves. The resultant Chernoff bound of M-LWDF's fairness measure was 3.7 times larger than the worst fairness measure in the simulation results. The bound of WF-LWDF's fairness measure was 6 times larger than the simulation results, and about 66% of the size of the M-LWDF bound.

Chapter 5 consisted of an overview of delay bounds of various schedulers published in the literature over the last 15 years. Scheduling categories that appeared were FIFO, EDF, GPS, PGPS, VirtualClock, SCFQ, and two round robin schemes, DRR and WRR. Once the analytical results had been listed, a common set of symbols and constants had to be defined. Typical values for each of these symbols were substituted to find typical delay bounds of the various schedulers. FIFO had a very loose, exponential bound. The temporary and permanent session versions of EDF both had polynomial bounds, which were also quite loose, but much tighter than the bound for FIFO. GPS has the tightest end-to-end delay bound of any conceivable scheduler. According to our results, both WRR and DRR have very small delays, but are unable to adhere to QoS guarantess. SCFQ has a slightly larger end-to-end delay than PGPS and VC, but with the advantage of having less complexity than PGPS and being fairer than VC.

6.2 Future Directions

The following are potential research projects that could build on the work presented in this dissertation:

- The effect of packet loss due to low signal-to-interference ratios and the resultant retransmission of packets could be explored.
- Suitable congestion control in the form of traffic shaping and call admission algorithms could be explored.
- Fairness and delay bounds of the M-LWDF and WF-LWDF scheduling algorithms that do not rely on any simulation results could be developed, which correspond suitably with the simulation results presented in this dissertation.

Bibliography

[1] S. Glisic and B. Vucetic, *Spread Spectrum CDMA Systems for Wireless Communications.* Artech House Publishers Inc., ISBN 0-89006-858-5, 1997.

[2] C.-K. Toh, *Wireless ATM and ad-hoc networks.* Kluwer Academic Publishers, ISBN 0-7923-9822-X, 1997.

[3] H. Holma and A. Toskala, *WCDMA for UMTS, second edition.* Jon Wiley and Sons Ltd, ISBN 0-407-84467-1, 2002.

[4] J. M. Pereira, "Balancing public and private in fourth generation," *PIMRC*, 2001.

[5] A. Parekh and R. Gallager, "A generalized processor sharing approach to flow control in integrated services networks: the single-node case," *IEEE/ACM Transactions on Networking*, vol. 1, no. 3, pp. 344–357, 1993.

[6] M. Andrews, *Scheduling Techniques for Packet Routing, Load Balancing and Disk Scheduling.* PhD thesis, Massachusetts Institute of Technology, June, 1997.

[7] M. Andrews, M. Bender, and L. Zhang, "New algorithms for the disk scheduling problem," *Proceedings of IEEE FOCS*, pp. 550–559, 1996.

[8] D. Ferrari and L. Delgrossi, "Charging for QoS," *IEEE/IFIP IWQOS '98 keynote paper*, 1998.

[9] J. Liebeherr and N. Christin, "Buffer management and scheduling for enhanced differentiated services," Tech. Rep. CS-2000-24, University of Virginia, Aug 2000.

[10] D. Ferrari and D. C. Verma, "A scheme for real-time channel establishment in wide-area networks," *IEEE Journal on Selected Areas in Communications*, vol. 8, no. 3, pp. 368–379, 1990.

[11] M. Andrews, S. Borst, F. Dominique, P. Jelenkovic, K. Kumaran, K. Ramakrishnan, and P. Whiting, "Dynamic bandwidth allocation algorithms for high-speed data wireless networks," *Bell Labs Technical Journal 3*, pp. 30–49, 1998.

[12] N. Joshi, S. R. Kadaba, S. Patel, and G. S. Sundaram, "Downlink scheduling in CDMA data networks," *Proc. ACM MobiCom*, pp. 179–190, 2000.

[13] T. S. E. Ng, I. Stoica, and H. Zhang, "Packet fair queueing algorithms for wireless networks with location-dependent errors," in *INFOCOM (3)*, pp. 1103–1111, 1998.

[14] Y. Cao and V. Li, "Scheduling algorithms in broadband wireless networks," *Proceedings of the IEEE*, vol. 89, no. 1, pp. 76–87, 2001.

[15] R. Guerin and V. Peris, "Quality-of-service in packet networks: Basic mechanisms and directions," *Computer Networks*, vol. 31, no. 3, pp. 169–189, February 1999.

[16] O. Lataoui, T. Rachidi, L. G. Samuel, S. Gruhl, and R. Yan, "A QoS management architecture for packet switched 3rd generation mobile systems," Tech. Rep. CS-2001-19, NetWorld+Inerop2000 - Engineers Conference on Broadband Internet Access Technologies Systems and Services, May 2000.

[17] C.-S. Chang and Y. H. Lin, "A general framework for deterministic service guarantees in telecommunication networks with variable length packets," *IEEE Trans. on Automatic Control*, vol. 46, pp. 210–221, 2001.

[18] M. Andrews, K. Kumaran, K. Ramanan, A. Stolyar, and P. Whiting, "Providing quality of service over a shared wireless link," *IEEE Communications Magazine*, vol. 39, no. 2, pp. 150–154, Feb. 2001.

[19] H. Zhang, "Service disciplines for guaranteed performance service in packet-switching networks," *Proceedings of the IEEE*, vol. 83, no. 10, pp. 1374–1396, October 1995.

[20] D. Stiliadis and A. Varma, "Frame-based fair queueing: A new traffic scheduling algorithm for packet-switched networks," Tech. Rep. UCSC-CRL-95-39, 1995.

[21] M. Katevenis, S. Sidiropoulos, and C. Courcoubetis, "Weighted round-robin cell multiplexing in a general-purpose ATM switch chip," *IEEE Journal on Selected Areas in Communications*, vol. SAC-9, no. 8, pp. 1265–1279, 1991.

[22] J. Moorman and J. Lockwood, "Implementation of the multiclass priority fair queuing (MPFQ) algorithm for extending quality of service in existing backbones to wireless endpoints," *Globecom '99*, pp. 2752–2757, 1999.

[23] M. Markaki, E. Nikolouzou, and I. Venieris, "Performance evaluation of scheduling algorithms for the internet," in *ATM and IP 2000*, IFIP Workshop, July 2000.

[24] D. Stiliadis and A. Varma, "Latency-rate servers: a general model for analysis of traffic scheduling algorithms," *IEEE/ACM Transactions on Networking*, vol. 6, no. 5, pp. 611–624, 1998.

[25] C. Dovrolis, D. Stiliadis, and P. Ramanathan, "Proportional differentiated services: Delay differentiation and packet scheduling," in *SIGCOMM*, pp. 109–120, 1999.

[26] P. White, "RSVP and integrated services in the internet: A tutorial," *IEEE Commun. Mag.*, pp. 100–106, May 1997.

[27] R. Cruz, "A calculus for network delay, part i: Network elements in isolation," *IEEE Transaction on Information Theory*, vol. 37, no. 1, pp. 114–131, 1991.

[28] R. Cruz, "A calculus for network delay, part ii: Network analysis," *IEEE Transaction on Information Theory*, vol. 37, no. 1, pp. 132–141, 1991.

[29] J. Liebeherr, S. Patek, and A. Burchard, "A calculus for end-to-end statistical service guarantees," Tech. Rep. CS-2001-19, University of Virginia, June 2001.

[30] S. Rajagopal, M. Reisslein, and K. W. Ross, "Packet multiplexers with adversarial regulated traffic," in *INFOCOM (1)*, pp. 347–355, 1998.

[31] Andrews and Zhang, "Packet routing with arbitrary end-to-end delay requirements," in *STOC: ACM Symposium on Theory of Computing (STOC)*, pp. 557–565, 1999.

[32] M. Andrews, A. Fernandez, A. Goel, and L. Zhang, "Source routing and scheduling in packet networks," *IEEE Symposium on Foundations of Computer Science*, pp. 168–177, Oct 2001.

[33] M. Shreedhar and G. Varghese, "Efficient fair queueing using deficit round robin," in *SIGCOMM*, pp. 231–242, 1995.

[34] H. Zhang and S. Keshav, "Comparison of rate-based service disciplines," in *SIGCOMM*, pp. 113–121, 1991.

[35] C. R. Kalmanek and H. Kanakia, "Rate controlled servers for very high-speed networks," *Proceedings of the Conference on Global Communications (GLOBECOM)*, pp. 12–20, 1990.

[36] J. Rexford, A. G. Greenberg, and F. Bonomi, "Hardware-efficient fair queueing architectures for high-speed networks," in *INFOCOM (2)*, pp. 638–646, 1996.

[37] C. Chang, "Stability, queue length, and delay of deterministic and stochastic queueing networks," *IEEE Transactions on Automatic Control*, vol. 39, pp. 913–931, May 1994.

[38] M. Andrews, "Probabilistic end-to-end delay bounds for earliest deadline first scheduling," *Proc. IEEE Infocom 2000*, pp. 603–612, 2000.

[39] V. Sivaraman and F. M. Chiussi, "Statistical analysis of delay bound violations at an earliest deadline first (EDF) scheduler," *Performance Evaluation*, vol. 36-37, no. 1-4, pp. 457–470, 1999.

[40] M. Andrews, "Distributed, deterministic scheduling protocols with polynomial delays." unpublished, obtainable from: citeseer.nj.nec.com/412422.html, 2000.

[41] D. E. Wrege and J. Liebeherr, "A near-optimal packet scheduler for QoS networks," *Proceedings of the IEEE*, vol. 2, pp. 576–583, 1997.

[42] S. Lu, T. Nandagopal, and V. Bharghavan, "A wireless fair service algorithm for packet cellular networks," in *Mobile Computing and Networking*, pp. 10–20, 1998.

[43] S. Lu, V. Bharghavan, and R. Srikant, "Fair scheduling in wireless packet networks," *IEEE/ACM Transactions on Networking*, vol. 7, no. 4, pp. 473–489, 1999.

[44] N. Pekergin, "Stochastic bounds on delays of fair queueing algorithms," in *INFOCOM (3)*, pp. 1212–1219, 1999.

[45] A. Parekh and R. Gallager, "A generalized processor sharing approach to flow control in integrated services networks: the multiple node case," *IEEE/ACM Transactions on Networking*, vol. 1, no. 3, pp. 521–530, 1993.

[46] Z.-L. Zhang, D. F. Towsley, and J. F. Kurose, "Statistical analysis of generalized processor sharing scheduling discipline," *IEEE Journal on Selected Areas in Communications*, vol. 13, no. 6, pp. 1071–1080, 1995.

[47] O. Yaron and M. Sidi, "Generalized processor sharing networks with exponentially bounded burstiness arrivals," in *INFOCOM (2)*, pp. 628–634, 1994.

[48] A. Elwalid and D. Mitra, "Design of generalized processor sharing schedulers which statistically multiplex heterogeneous QoS classes," in *INFOCOM (3)*, pp. 1220–1230, 1999.

[49] K. Kumaran, G. Margrave, D. Mitra, and K. Stanley, "Novel techniques for the design and control of generalized processor sharing schedulers for multiple QoS classes," *Proceedings of IEEE INFOCOM 2000*, 2000.

[50] M. Andrews and L. Zhang, "The effects of temporary sessions on network performance," *Proceedings of the 11th Annual ACM-SIAM Symposium on Discrete Algorithms*, pp. 448 - 457, January 2000.

[51] A. Demers, S. Keshav, and S. Shenkar, "Analysis and simulation of a fair queueing algorithm," *SIGCOMM '89*, vol. 19, no. 4, pp. 3–12, September 1989.

[52] J. Bennett and H. Zhang, "Why WFQ is not good enough for integrated services networks," *Proc. of the 6th International Workshop on Network and Operating Systems*, April 1996.

[53] J. Chen, C. Huang, M. Devetsikiotis, and I. Lambadaris, "Virtual clock with priority buffer: A resource sharing algorithm," *Proceedings of IEEE GLOBECOM '98*, 1998.

[54] S. J. Golestani, "Network delay analysis of a class of fair queueing algorithms," *IEEE Journal on Selected Areas in Communications*, vol. 13, no. 6, pp. 1057–1070, 1995.

[55] P. Goyal, H. M. Vin, and H. Cheng, "Start-time fair queuing: A scheduling algorithm for integrated servicespacket switching networks," *Proceedings of the ACM SIGCOMM '96*, pp. 157–168, August 1996.

[56] D. Stiliadis, *Traffic Scheduling in Packet-Switched Networks: Analysis.* PhD thesis, Department of Computer Science and Engineering, University of California at Santa Cruz, 1996.

[57] D. Stiliadis and A. Varma, "Rate-proportional servers: a design methodology for fair queueing algorithms," *IEEE/ACM Transactions on Networking*, vol. 6, no. 2, pp. 164–174, 1998.

[58] D. Stiliadis and A. Varma, "Design and analysis of frame-based fair queuing: A new traffic scheduling algorithm for packet switched networks," in *Measurement and Modeling of Computer Systems*, pp. 104–115, 1996.

[59] D. Stiliadis and A. Varma, "Efficient fair queueing algorithms for packet-switched networks," *IEEE/ACM Transactions on Networking*, vol. 6, no. 2, pp. 175–185, 1998.

[60] P. Goyal, S. S. Lam, and H. M. Vin, "Determining end-to-end delay bounds in heterogeneous networks," in *Network and Operating System Support for Digital Audio and Video*, pp. 273–284, 1995.

[61] P. Goyal and H. M. Vin, "Generalized guaranteed rate scheduling algorithms: a framework," *IEEE/ACM Transactions on Networking*, vol. 5, no. 4, pp. 561–571, 1997.

[62] H. Zhang and D. Ferrari, "Rate-controlled service disciplines," *Journal of High-Speed Networks*, vol. 3, no. 4, pp. 389–412, 1994.

[63] D. Eckhardt and P. Steenkiste, "Effort-limited fair (ELF) scheduling for wireless networks," *IEEE INFOCOM 2000*, pp. 1097–1106, 2000.

[64] A. Elwalid and D. Mitra, "Analysis, approximations and admission control of a multiservice multiplexing system with priorities," *Proc. IEEE INFOCOM '95*, pp. 463–472, 1995.

[65] L. Kalampoukas, *Congestion Management in High Speed Networks.* PhD thesis, University of California Santa Cruz, 1997.

[66] L. Kalampoukas, "An efficient rate allocation algorithm for atm networks," Master's thesis, University of California Santa Cruz, 1995.

[67] J. L. Rexford, A. G. Greenberg, and F. G. Bonomi, "A fair leaky-bucket traffic shaper for ATM networks." unpublished report, obtainable from: citeseer.nj.nec.com/48841.html, 1995.

[68] A. Elwalid, D. Mitra, and R. H. Wentworth, "A new approach for allocating buffers and bandwidth to heterogeneous regulated traffic in an ATM node," *IEEE Journal of Selected Areas in Communications*, vol. 13, no. 6, pp. 1115–1127, 1995.

[69] M. Kalia, D. Bansal, and R. Shorey, "Data scheduling and SAR for Bluetooth MAC," *Proc. IEEE Vehicular Technology Conference (VTC)*, vol. 2, pp. 716–720, May 2000.

[70] M. Andrews, K. Kumaran, K. Ramanan, A. Stolyar, and P. Whiting, "Data rate scheduling algorithms and capacity estimates for CDMA forward link," Tech. Rep. BL0112120-990922-32TM, Bell Labs, 1999.

[71] M. Kubik, "Uplink packet scheduling in WCDMA systems," Tech. Rep. EX061/1999, Chalmers University of Technology Technical Report, Dec 1999.

[72] A. L. Stolyar and K. Ramanan, "Largest weighted delay first scheduling: Large deviations and optimality," *Annals of Applied Probability*, no. 1, 2000.

[73] A. L. Stolyar, "Control of end-to-end delay tails in a multiclass network: LWDF discipline optimality," *Annals of Applied Probability*, 2000.

[74] M. Andrews, K. Kumaran, K. Ramanan, A. Stolyar, R. Vijayakumar, and P. Whiting, "CDMA data QoS scheduling on the forward link with variable channel conditions." Bell Labs Technical Memorandum, 2000.

[75] C. Schurgers and M. B. Srivastava, "Energy efficient wireless scheduling: Adaptive loading in time," *IEEE Wireless Communications and Networking Conference (WCNC'02), Orlando, FL*, 2002.

[76] C. W. Sung and W. S. Wong, "Performance of a cooperative algorithm for power control cellular systems with a time-varying link gain matrix," *ACM Wireless Networks*, vol. 6, no. 6, pp. 429–439, 2000.

[77] S. Golestani, "A self-clocked fair queueing scheme for broadband applications," *Proceedings of INFOCOM '94*, pp. 636–646, 1994.

[78] G. B. Thomas and R. L. Finney, *Calculus 9th edition.* Addison-Wesley Publishing Company, ISBN 0-201-40015-4, 1996.

[79] L. Zhang, "Virtual Clock: A new traffic control algorithm for packet switching," *ACM Trans. Comput. Syst.*, vol. 9, no. 2, pp. 101–124, 1991.