Université de Nice - Sophia Antipolis – UFR Sciences
École Doctorale STIC

THÈSE

Présentée pour obtenir le titre de :

Docteur en Sciences de l'Université de Nice - Sophia Antipolis

Spécialité : INFORMATIQUE

par

Natalia OSIPOVA

Équipe d'accueil : MAESTRO – INRIA Sophia Antipolis

IMPROVING RESOURCE SHARING IN COMPUTER NETWORKS WITH STOCHASTIC SCHEDULING

Soutenance à l'INRIA le 27 March 2009 à xxx heures devant le jury composé de :

Président :	Prenom	NOM	Université / Affiliation
Directeur :	Prenom	NOM	Université / Affiliation
Rapporteurs :	Prenom	NOM	Université / Affiliation
	Prenom	NOM	Université / Affiliation
	Prenom	NOM	Université / Affiliation
Examinateurs :	Prenom	NOM	Université / Affiliation
	Prenom	NOM	Université / Affiliation

Thèse

L'amélioration du partage des ressources dans les réseaux de communication par l'ordonnancement stochastique

Improving resource sharing in computer networks with stochastic scheduling

Natalia Osipova

Mars 2009

CONTENTS

FIGURES

TABLES

CHAPTER 1

INTRODUCTION

In the early 1970s, networks that interconnected computers and terminals began to appear. These networks were developed to share computer resources and to interchange data between computers. Since then the task of minimization of the transmission costs and times and maximization of the amount of transmitted data was one of the most important tasks in computer networks. While with the technical progress the capacities of the computers grow, the need of quick, efficient and safe data transmission grows as well.

The Internet is the largest computer network which connects more than one billion of users all over the world. The size of the Internet grows very fast, though the network resources have to be shared between a very large number of users. An incorrect resource allocation may imply server inaccessibility, long delays and other problems in the networks, which lead to the users' dissatisfaction with the provided service. Even though until today a lot of work was done to achieve optimal resource sharing, high performance and low delays, Internet behavior is far from ideal and there are still a lot of problems that have to be solved.

In the present thesis we are dealing with the problem of the resource sharing in computer networks. We consider several scheduling algorithms and their application to flow scheduling in the Internet routers. As the waiting time in the network is one of the most important characteristics for the common users, we concentrate on the problem of mean waiting time minimization. Taking into account the Internet traffic structure we study several size-based differentiation algorithms which give priority to the short flows and can significantly decrease the mean waiting time in the network. We introduce a new flow-aware packet dropping scheme for the Internet routers which improves performance in the network and fairness between the flows.

1.1 The state of the art

1.1.1 Computer networks

A computer network is a set of several computers or terminals, which are interconnected by a communication network. Even if computer networks are widely presented in literature, see [Tan96, Sta03], in this introduction we describe some computer network basics to explain the motivation for the provided analytical results.

Before talking about computer networks in detail, let us first answer the question: "Why are people interested in computer networks, what can they be used for?". Globally we can classify the computer networks users in two groups, companies and common users. The companies use the computer networks mainly to achieve resource sharing (all programs, equipment and data availability to the workers of the company), high reliability (possibility to continue to work in case of hardware failure problems), saving money (high cost of big computers in comparison with several small ones), scalability (possibility to add new working places in the network and to increase system performance by adding new processors without global change of the system structure), communication between workers of the company (reports, work discussion). For the common computer and Internet users the most important aims are: access to remote information (bank accounts check, shopping, newspapers, magazines, journals, on-line digital libraries), personal communication (mail, virtual meetings, videoconferences), entertainment (video, movie, television, radio, music, game playing). So, computer networks take a big part of everyday peoples life and can help us in many different areas.

Now that we pointed why we need the computer networks, let us return to our subject, how computer networks work. The main goal of computer networks is the possibility to interchange data between computers. In its simplest form data communication takes place between two devices that are connected directly. Often, however, it is not practical for two devices to be point-to-point connected. It is the case when the devices are situated very far from each other. An example is the telephone network of the world, or all the computers of a single organization. Then the solution is to connect each device to a communication network. Later in this work we refer to the devices which communicate either as stations or as nodes. The stations may be computers, telephones or other communication devices.

Communication networks may be categorized based on the architecture and techniques used to transfer data. Globally there are broadcast and switched (point-to-point) networks. In the broadcast networks the transmission from one station is broadcast and received by all other stations. In the switched networks data is transferred from source to destination through intermediate nodes. The purpose of every node is to move data from node to node until it reaches its destination. Switched networks are divided into circuit-switched networks and packet-switched networks. In circuit-switched networks, the path between a sender and a destination

is set in advance and then the data is transmitted using this channel.

In packet-switched networks, data is sent in a sequence of small chunks, called packets. Each packet passes through the network from one node to another along some path leading from source to destination. At each intermediate node, called router, the packet is received, stored briefly, and then transmitted to the next node. The router takes decision where to transmit the packet. Packet-switched networks are commonly used for computer-to-computer communications.

The computer networks are also classified according to their size as Local Area Networks (LAN), which cover a campus under a few kilometers in size, Metropolitan Area Networks (MAN), which cover a group of offices or a city, and Wide Area Networks (WAN), which cover a large geographical network. LAN and MAN usually do not use the packet switching, because of their limited sizes. The examples of LAN are Ethernet, IBM Token ring and the most known MAN are Distributed Queue Dual Bus (DQDB), etc. The most famous and the largest WAN is the Internet, which connects more then one billion of users and allows to interchange data between them. The WAN networks usually use the packet switching technology. In particular, the Internet is based on the packet switching technology.

In the present work, we study problems related to packet-switched networks and in particular, to the Internet.

1.1.2 Computer network architecture

Nowadays the communication between computers in the Internet is mostly based on the Open System Interconnection model (OSI) which consists of seven layers, see [Sta94]. The layer structure is used to decompose a complex problem of communication between computers into several smaller problems, the layers are autonomous and do not depend on each other. Every layer is responsible for certain functionalities, it uses the functions of the lower layer and gives functionality to the upper layer. The layers are based on the concept of the protocol, a set of rules which serves to organize data transfer. The OSI layers are: Physical, Data Link, Network, Transport, Session, Presentation and Application.

We do not give a full description of all layers and their functionalities here, but we restrict ourselves to the Transport and Network layers of the OSI system, as they correspond to the data transfer and provide error recovery and flow control.

The Network layer accepts packets from the Transport layer and delivers them from source to destination in the network. The Network layer is based on the Internet Protocol (IP). The IP technology does not check if the packets were delivered, so the error recovery has to be done by the Transport or other higher level protocols. The Network layer provides unreliable delivery service in the networks.

The Transport layer accepts data from the Session layer, splits it into packets and transmits packets to the Network layer. The Transport layer checks the delivery of the packets to the

destination, it provides a reliable transport mechanism, see [Sta94]. The two main Internet protocols of the Transport layer are Transmission Control Protocol (TCP) and User Datagram Protocol (UDP). Both of them use IP protocol of the Network layer, that's why we are usually talking about TCP/IP protocols.

UDP protocol is rarely used in the Internet because of its unreliability. UDP does not provide a reliable delivery of the packets and is used by the applications which provide their own flow control and check packet arrivals. Also UDP can be used when some loss of transferred data can be tolerated, as in the Internet telephony.

The most used protocol in computer networks and in the Internet is TCP. TCP is a reliable connection-oriented protocol which allows to deliver the information from one machine in the network to another without errors. It is designed to provide maximum throughput and reliable transfer over an unreliable and unknown network. Different parts of the WAN can have different topologies, bandwidths, delays, packets sizes and other parameters. Also all these parameters can change. TCP dynamically adapts to properties of the network and is robust in the face of many kinds of failures. TCP provides flow control to make sure that a fast sender does not overflow a slow receiver or intermediate nodes with more information than they can handle. Using the Congestion Control mechanism, TCP reduces its sending rate when a loss occurs in the network, and so adapts its sending rate according to the parameters of the receiver and the network. We give more full description of TCP protocol work in Subsection 1.1.5.

The applications that are nowadays [Sta03] the most used in the Internet and computer networks and which use the TCP/IP protocol are: Telnet (virtual terminal), File Transfer Protocol (FTP) is used for file transfers between systems with different properties and structures, electronic mail protocol (SMTP) is used to transfer the electronic mail messages, Multipurpose Internet Mail Extension (MIME) makes it possible to include pictures and other multimedia in the message, Domain Name System (DNS) is used to find the relation between the host names and their network addresses, Hypertext Transfer Protocol (HTTP) is used to transfer web pages in the Internet, Session Initiation Protocol (SIP) is the application level protocol for sessions control in the networks.

1.1.3 The Internet traffic structure

Let us point out the most important characteristics of the traffic structure in computer networks and in the Internet, which we need in the following analysis.

In the flow-level modelling framework, flow is the basic unit of the data traffic. The flow is defined as an interruptive stream of packets sent from the source to destination. We can model as a flow one TCP connection which opens, sends one or more files and then closes, or we can define every file sent by the application as a separate flow. In the current work we consider that the flow corresponds to the sending of one file. In the current work we use terms

"flow/connection/file/session" interchangeably, in stochastic scheduling we use the term "job". A flow is basically characterized by its duration, size and sending rate.

The Internet traffic structure was widely studied in the literature. In [CB97, TMW97, NMM98, SAM99], authors analyze the real data traffic on the selected web servers during sufficiently long period of time and describe traffic characteristics. In [FML+03], authors propose a monitoring system which is designed to analyze traffic measurement and also provide results they got with the proposed system. In [Wil01], the author describes traffic measurements and characteristics. In [BGG03], authors provide traffic collection and analysis between several important servers in France. In [BFOBR02], authors study the admission control in the Internet in application to elastic and streaming flows. In more recent work [CC08], authors propose a new characteristic, the Quality of Experience, to measure how the users perceive the network performance and provide the real traffic measurements results.

In the Internet traffic is divided in elastic and streaming flows. Elastic flows are transferred files, HTTP pages, etc. Streaming flows are created by video and audio applications. Elastic flows are still dominant in the Internet even though audio and video applications are more and more used, see [BFOBR02]. In the current work we study elastic flows simulation, considering that streaming flows take some limited share of the bandwidth.

The traffic transferred with the TCP/IP protocol represents 90% of all Internet traffic, see [CC08, FML+03, BFOBR02].

The interarrival times between the files in the Internet are exponentially distributed and the flow arrivals to the network can be modelled with a Poisson process. The important characteristic of the Poisson process is Poisson Arrivals See Time Averages (PASTA) property [Wol89], which plays an important role in mathematical analysis of the network modelling.

Most flows ($90 - 95\%$) transferred in the Internet are very small, but most of the traffic is created by the long flows, which are not numerous (remaining $5 - 10\%$), see [Wil01, All00, FML+03] and others. According to [CC08], 80% of the traffic is created by the flows larger than 1 MB and 50% by the flows of size larger than 10 MB. This is caused by the fact that the most frequent flows are created by e-mail and web page transfers, which have small sizes and the long flows are generated by file transfers, peer-to-peer applications, etc., and are much rarer. The short flows are then called "mice", long flows "elephants" and this phenomena in the Internet is called "mice-elephant" effect.

It was found that the file size distributions in the Internet are well modelled by long-tailed and heavy-tailed distributions and also have a Decreasing Hazard Rate (DHR). In [NMM98], with the real data analysis, authors confirm that the file size distribution in the Internet can be modelled with heavy-tailed Pareto distributions. In [CB97], authors provide network traffic analysis from a web server and found that the file size distribution is heavy tailed. In [Rob01], the author shows that the streaming flows durations are also heavy-tail distributed.

In contrast to the flow arrivals, the packet arrivals are generally not Poisson. Because of the DropTail router policy, which creates global synchronization in the network, and also because of the TCP algorithm, (we discuss this later in Subsection 1.1.5), the packets have the tendency to arrive in groups, which are called batches. Such arrivals are also called bursty arrivals.

Packet sizes in the Internet vary from Maximum Transmit Unit (MTU), to the acknowledgement (ACK) sizes (40 bytes). According to [Wil01, FML+03], the large packets represent 50% of all packets in the network, ACKs represent 40% and the rest of the packets have sizes which are randomly distributed between these values.

The traffic on the link is usually bidirectional, but not symmetric.

1.1.4 Traffic control in computer networks

One of the main problems associated with the computer networks is traffic control, see [Sta94], which is in regulating the amount of traffic which enters the network so that the network performance is high. Traffic control can be separated on flow control, congestion control and deadlock avoidance.

The deadlock is a situation when the router cannot send a packet because all the buffers are full. The deadlock avoidance techniques are designed to avoid such a situation.

Flow control is needed to prevent the sender from transmitting information with a rate which is higher than the possible receiving rate of the destination. Flow control regulates data transmission rate between two nodes.

Congestion is a situation when the data arrival rate is higher than the network transmission capacity. In this case the router can not serve all the incoming packets, which are then collected in the router buffer and wait in the queue to be served. If the arrival rate does not decrease, the queue size increases dramatically, there is no place for more packets, and the new arriving packets are dropped and later retransmitted. The congestion in the networks is responsible for the most important part of delays. Congestion control techniques try to prevent the congestion situation before it happens, or at least, react on it properly, i.e., decrease data arrival rates. The efficient congestion control algorithm has to avoid buffer overflow and at the same time try to keep the queue not empty, to achieve higher throughput.

To provide efficient traffic control the sender needs to know the situation on the router and in the network, which is not always easy and even more, usually impossible to realize. On the other side, the router neither does not have a direct access to the data senders to control their sending rates.

In the Internet the traffic control is realized with the combination of the DropTail policy on the router and TCP/IP protocols.

The DropTail policy is the simplest and the most commonly used algorithm for the buffer size management in TCP/IP networks. With the DropTail algorithm the router drops the arriving

packets from the tail of the queue if the buffer is full. The enqueued packets are served according to the First Come First Served (FCFS) policy. The buffer size of the router is limited. Even it is technically possible to make the buffer size very large, it is not used, because the large queue size creates large queuing delay. The selection of the router buffer size is an important problem, which is not yet solved. More on this topic one can find in [AANB02, AKM04, BGG+08], etc.

The current TCP implementation provides flow and congestion control. We give more detail description of TCP algorithms in the following Subsection 1.1.5.

1.1.5 TCP/IP protocols

TCP/IP protocols are now widely used in the Internet and play an important role in determining the network performance. The formal TCP description is given in [Pos81]. The idea of the dynamic congestion window size is proposed in [Jac88]. Later changes and extensions are given in [Bra89, JBB92, Ste97, APS99]. Also the description of TCP can be found in such books as [Sta03, Tan96, Wil98] or in reviews, see [Kri00].

TCP is based on the end-to-end argument idea, which is that the sending rate of data flow is controlled by the receiver. To realize data transmission between two nodes, TCP has to be installed on both of them, the sender and the receiver. When TCP sends a data file, it breaks it into packets (or segments) of the given size and sends each of them separately in the data stream. When packets arrive to the destination, they are given to the TCP entity, which reconstructs the original file. As the IP protocol does not give guarantee of packet arrivals, it is up to TCP to find which packets were lost and retransmit them. For that purpose every time the destination TCP receives a packet, it sends back to the sender a packet of small size, which is called acknowledgement (ACK) which contains information about the received packet. The receiver acknowledges the last packet of the received continuous stream of packets. If there is a packet which arrives out of order, TCP sends the ACK for the last packet which was received in order. In this case the sender receives several times the same ACK, which is called duplicate ACK, knows that the packet of the stream was lost and can retransmit it. The time between sending a packet and receiving an ACK for it is round-trip-time (RTT) which is an important notion related to TCP.

The congestion control scheme in TCP is realized with congestion window (cwnd) which controls the amount of data which can be sent without being acknowledged and in fact controls the rate of transmission. The algorithms which are used in TCP congestion control are: Slow Start, Congestion Avoidance, Fast Retransmit and Fast Recovery.

The Slow Start algorithm is used in the beginning of the file transfer to determine the capacity of the network. During Slow Start TCP increments its congestion window by one packet for each received ACK. Slow Start ends when the congestion window reaches some given threshold. After Slow Start algorithm, the Congestion Avoidance algorithm is used. During Congestion

Avoidance, congestion window is incremented by one packet per RTT or by one packet when the data which corresponds to the current size of congestion window is acknowledged. Congestion Avoidance is continued until congestion is detected. The Slow Start algorithm makes it possible for the TCP connection to increase its sending rate quickly in the beginning of the file transfer, while during the Congestion Avoidance the rate increases slowly to avoid the network overload.

To detect a congestion and a packet loss TCP uses a timer. To retransmit the lost packet faster than the timer expires the Fast Retransmit algorithm is used, which is that if TCP receives three duplicate ACK, the packet is considered to be lost. The Fast Recovery mechanism is that the congestion window is reduced in half in the case of packet loss detection. It helps TCP to restore congestion window more quickly than if it was reduced to one packet and so help to achieve higher throughput.

The Tahoe implementation of TCP includes Slow Start, Congestion Avoidance and Fast Recovery. Reno includes Tahoe properties plus Fast Retransmit. NewReno is a slight modification of the Reno version that improves performance during Fast Recovery and Fast Retransmission. In our studies and simulations we consider NewReno version of TCP.

1.2 Computer networks problems and proposed solutions

1.2.1 Traffic control schemes advantages and disadvantages

TCP is the most used data transmission protocol in the Internet as it is flexible and provides a reliable data transfer and traffic control. On the flow level TCP tries to provide a fair share of the bottleneck capacity between all flows currently present in the queue. As the routers generally do not use discriminating or priority policies, the share of the bottleneck capacity depends only on the sending rates of every flow. Then, if the sending rates of every flow are kept equal, the bandwidth share is equal as well.

The advantage of the DropTail policy is its simplicity. There is no need to set various parameters and keep the additional information about the flows and the state of the queue.

However, there are many disadvantages of the currently used combination of DropTail policy and TCP/IP protocols. They are packet retransmissions, global synchronization, unfair bandwidth sharing, absence of Quality of Service.

With the DropTail policy packets are dropped when the buffer is full, TCP reduces its sending rate only after a packet loss is detected, which creates multiple packet retransmissions in the network. DropTail policy does not make differentiation between flows and so there is no Quality of Service.

When several TCP connections share the same bottleneck link, the bottleneck bandwidth is shared unfairly and the flows with small RTTs have an advantage in respect to the flows with large RTTs. This happens because during the congestion moments all connections which

share the bottleneck link decrease their sending rates, but for connection with high RTT it takes longer to restore its sending rate than for connection with small RTT. Then the final transferred amount of data for the slow connection is much smaller than for the fast connection. Also the fact that all connections currently using the bottleneck link reduce their rates nearly at the same time creates global synchronization in the network, which in turn leads to the network underutilization.

There were many proposals to increase performance of the Network and Transport layers of the Internet. Between them are Network Pricing, Explicit Congestion Notification (ECN), Active Queue Management (AQM) algorithms and scheduling algorithms.

Network Pricing is a category of congestion control, where the cost of transmission is used. Making the transmissions of TCP payable may avoid congestion as the senders will be forced to minimize the generated amount of traffic. More on this topic can be found in [Bre96, SCEH96, FR01, FR04, FORR98].

ECN is a flag which is used to warn the TCP sender about the congestion situation in the network, see [Flo95, RF99, RFB01]. When the congestion occurs, the router marks packets with ECN flag instead of dropping them. The packet marked with ECN flag comes to the destination and the receiver sends back the ACK with ECN flag. When the ACK with ECN flag is received by the sender, it reduces in half its congestion window as if a packet loss was detected. So, if in the router instead of dropping the packets, they are marked with ECN flags, the TCP congestion window is reduced, but there is no need to retransmit the packets again.

To avoid unfair resource sharing in the Internet several AQM schemes were proposed. AQM is a family of packet dropping algorithms for FCFS queues which manage the length of packet queues dropping the packets when necessary. AQM algorithms inform the sender about the possibility of the congestion before a buffer overflow happens. Among AQM algorithms are RED [FJ93], GREEN [WZ02], BLUE [FSKS02], MLC(l) [SS07], CHOKe [PPP00], etc. None of them was widely implemented in the networks because of their complexity and nontrivial parameters selection.

From the user point of view the most important characteristic in computer networks is the waiting time, the time which passes between the mouse click and the page appearance on the screen. The delay in the networks consists of the transfer delay, propagation delay, processing delay and queueing delay. In the networks the queueing delay and delay which is caused by the packet drops and retransmissions give the largest part of the waiting time. The queueing delays in the network can be reduced with the efficient scheduling algorithms. While AQM scheme finds the packet which has to be dropped to avoid congestion in the network, scheduling algorithms find the packet which have to be next served and are used to reduce queueing delay and to manage bandwidth share between flows.

To develop an efficient scheduling algorithm one has to take into account the specific problems

of its application domain. In the case of computer networks, these problems are: large number of connections sharing the bottleneck link, the traffic characteristics, the changes of the sending rates, the possible changes of the network topology and properties and so on. Even though there exist a lot of different scheduling algorithms, it is not evident to find one which is efficient, scalable, easy to implement, does not need knowledge of specific system parameters.

In the following subsection we give a short review of the scheduling algorithms which were proposed to be applied in computer networks and in the Internet.

1.2.2 Computer network modelling with stochastic scheduling

From the stochastic scheduling theory, it is known that, applying different scheduling policies to a queue, it is possible to influence the system characteristics a lot. The goal of stochastic scheduling is to find an algorithm which improves system performance and at the same time which is simple to implement.

It is quite difficult to model the network on the packet level, as the packet arrivals are bursty and are not Poisson distributed as flow arrivals, see Subsection 1.1.3. Thus networks are often modelled on the flow level. Every file sent by TCP connection is presented as a job and every router as a queue.

When we talk about a job size, we consider the time that the job is served in the queue if there is no more jobs in the system. Though later in this work we use the terms "job size" and "service time" interchangeably.

As we discussed in Subsection 1.2.1, the bandwidth share on the bottleneck link of the TCP flows in the case when their RTTs are of the same order is well modelled by the Processor Sharing (PS) discipline, see [HLN97, NMM98, MR00, FBP+01, CJ07]. Under the PS policy every job present in the system receives an equal share of the processor capacity. The PS discipline is easy to analyze, Kleinrock in his book [Kle76a, Sec. 4.4] obtained the expression of mean conditional and mean sojourn time in the $M/G/1$ system scheduled with PS discipline. However, PS disciple does not minimize the mean sojourn time in the system.

It is known that the Shortest Remaining Processing Time (SRPT), see [Kle76a, Ch. 3], policy minimizes the mean sojourn time in the system, see also [Sch68]. The SRPT discipline requires knowledge about the job sizes, which is not always possible, as the router does not have information about the size of the file which was send.

Kleinrock in his book [Kle76b, Kle76a] gives an overview of policies, which do not use information about the job sizes and are called non-anticipating. In the last years these policies received a significant attention because of their possible application to resource sharing in computer networks.

It is shown in [Yas87] that the Least Attained Service (LAS) or Foreground-Background (FB) policy, see [Kle76a, Sec. 4.6], minimizes the mean waiting time in the system among all

non-anticipating scheduling policies when the job size distribution function has a decreasing hazard rate (DHR). As this is the case for the job size distribution in the Internet, LAS received a lot of attention, it was studied in [RS89, FM03b, RUKB03, RUKVB04, RBUK05]. The survey on LAS is presented in [NW08]. However, LAS has some disadvantages, for example, it can be very unfair for the long flows in some cases and it increases a lot the service time for the long flows, see [FM03b]. Also the mean waiting time in the system under LAS highly depends on the job size distribution, [RUKB03]. If there is a long flow in the system which is almost finished to be served and there is another long flow which arrives, then the first flow has to wait all the service time of the second flow before quitting the system. The problem of LAS unfairness with the large jobs was studied in [RUKB03, WHB03]. Regarding this problem, in [Bro06] it was shown that when the second moment of the job size distribution is infinite, LAS always has smaller expected conditional sojourn time than PS.

Both, SRPT and LAS policies give priority to the short flows and though minimize mean waiting time in the system. The file size distribution in the Internet is heavy-tailed and most of the flows have short sizes, see Subsection 1.1.3. Then it seems logical to give priority to the short flows in the network. The differentiation between short and long flows in the Internet was widely studied, see [GM01, NT02, GM02a, GM02b, RUKB02, RUKB03, FM03b, WBHB04, AANO04, AABN04].

Among flow differentiating policies is the Multi Level Processor Sharing (MLPS) discipline which was introduced and described by Kleinrock, see [Kle76a, Sec. 4.7]. He shows that the mean sojourn time in the MLPS system can be sufficiently reduced in comparison with the PS system. When the MLPS discipline is applied, the jobs are served according to their attained service up to the given number of thresholds. In [AANO04, AANO05] authors show that when the job size distribution has a DHR, MLPS decreases the mean waiting time in the system with respect to the PS discipline. In [AA06] authors show that with MLPS the mean delay in the system can be very close to optimal when the job size distribution has a DHR.

The particular case of MLPS, Two Level Processor Sharing (TLPS) and its application to resource sharing in computer networks was studied in [AANO04, AABN04]. In [AABN04] based on the TLPS model authors develop the RuN2C algorithm and show that it reduces significantly the mean waiting time in the system in comparison with the standard DropTail policy. The mean waiting time in the TLPS model significantly depends on the threshold selection, which was not yet studied analytically.

The main idea behind LAS and TLPS policies is to give priority to the short jobs, but they do not give possibility to give preference to some selected flows. In contrast, Discriminatory Processor Sharing (DPS) policy allows to introduce the Quality of Service in the network. DPS provides a natural approach to model the resource sharing of the TCP flows with different RTTs or weighted round-robin algorithm, which is used in operating systems. Also the DPS discipline

can be used to model the pricing policies on server, when the different services are provided according to the paid rates. DPS was first introduced by Kleinrock [Kle67]. Under DPS jobs are organized in classes and are served according to the vector of weights, so each class has its priority in the system. The DPS policy was studied in [FMI80, RS94, RS96, GM02b, AJK04, KNQB04, KNQB05, AABNQ05]. Most of the results obtained for the DPS queue were collected together in the survey paper [AAA06]. However, weight vector selection in DPS is not a trivial task because of the system complexity.

The problem of finding an optimal policy between all non-anticipating scheduling policies in the $M/G/1$ queue was solved by Gittins in [Git89]. He showed that in the $M/G/1$ queue the policy which gives service to the job in the system with the highest Gittins index function of the attained service minimizes the mean waiting time in the system between all non-anticipating scheduling policies. The well known results of LAS optimality for the DHR job size distribution can be derived as a corollary of the general optimality of the Gittins policy. However, this optimality result did not receive much attention and so was not fully exploited.

1.3 Thesis contribution and organization

In the current Thesis we study the problem of resource sharing in computer networks. We study several scheduling algorithms from the stochastic scheduling theory and their application to the computer networks. In Chapters 2 - 5 we study the problem of the mean waiting time minimization in the system with various scheduling algorithms. In Chapter 6 we study the congestion control problem in the networks and propose a new flow-aware algorithm to improve the fair resource sharing of the bottleneck capacity.

In Chapter 2 we study the Batch Processor Sharing (BPS) model with hyper-exponential service time distribution. For this distribution we solve Kleinrock's integral equation for the expected conditional response time function and prove the concavity of the solution with respect to the job size. We apply the found result to find the analytical expressions of the mean conditional and unconditional times for the TLPS model in the following Chapter 3. We also use the batch queue analysis in the derivation of the mean conditional sojourn time in Chapter 5. The results of this chapter are published in [Osi08a].

In Chapter 3 we analyze the TLPS scheduling discipline with the hyper-exponential job size distribution and with the Poisson arrival process. In the first part of the chapter we study the case when the job size distribution has two phases. The choice of two-phase job size distribution is motivated with the "mice-elephant" effect of the file size in the Internet, see Subsection 1.1.3. In the case of the hyper-exponential job size distribution with two phases, we find a closed form analytic expression for the expected sojourn time and an approximation for the optimal value of the threshold that minimizes the expected sojourn time. With the numerical results

we show that the mean waiting time in the TLPS system is very close to optimal with the found approximated threshold value. With the simulation results with NS-2 simulator we show that analytically found threshold approximation minimizes the mean waiting time in the TLPS system between other threshold values and gives significant relative gain in comparison with the DropTail policy.

In the second part of Chapter 3 we study the TLPS system when the job size distribution is hyper-exponential with many phases. For this case we derive a tight upper bound for the expected sojourn time conditioned on the job size. We show that when the variance of the job size distribution increases, the gain in system performance increases and the sensitivity to the choice of the threshold near its optimal value decreases. This work is published in [ABO07].

In Chapter 4 we study the comparison of two DPS policies with different weight vectors. We show the monotonicity of the expected sojourn time of the system depending on the weight vector under certain conditions on the system. The restrictions on the system are such that the result is true for systems for which the values of the job size distribution means are very different from each other. The restriction can be overcome by setting the same weights for the classes, which have similar means. The condition on means is a sufficient, but not a necessary condition. It becomes less strict when the system is less loaded. The results of this chapter can be found in [Osi08b].

In Chapter 5 we obtain the optimal policy for multi-class scheduling in a single server queue. We apply the results of Gittins [Git89], where he founds the optimal policy which minimizes the mean waiting time in the system in a single class $M/G/1$ queue between all non-anticipating policies. In this chapter we show that a straightforward extension of Gittins' results allows us to characterize the optimal scheduling discipline in a multi-class $M/G/1$ queue. We apply the general result to several cases of practical interest where the service time distributions have DHRs, like Pareto or hyper-exponential. We show that in the multi-class case the optimal policy is a priority discipline, where jobs of the various classes depending on their attained service are classified into several priority levels. Using a tagged-job approach we obtain, for every class, the mean conditional sojourn time. This allows us to compare numerically the mean sojourn time in the system between the Gittins optimal and popular policies like PS, FCFS and LAS. As in the Internet the file size is heavy-tailed and has a DHR, see Subsection 1.1.3, the obtained optimal Gittins policy can be applied in the Internet routers, where packets generated by different applications must be served. Typically a router does not have access to the exact required service time (in packets) of the TCP connections, but it may have access to the attained service of each connection. Thus we implement the Gittins' optimal algorithm in NS-2 and we perform numerical experiments to evaluate the achievable performance gain.

In Chapter 6 we introduce MarkMax, a new flow-aware AQM algorithm for Additive Increase Multiplicative Decreases protocols (like TCP). The main idea behind MarkMax is to identify

which connection should reduce its sending rate instead of which packets should be dropped. In contrast with several previously proposed AQM schemes, MarkMax uses the differentiation between flows currently presented in the system and cuts the sending rate of the flows with the biggest sending rate. MarkMax sends a congestion signal to a selected connection whenever the total backlog reaches a given threshold. The selection mechanism is based on the state of large flows. Using a fluid model we derive some bounds that can be used to analyze the behavior of MarkMax and we compute the per-flow backlog. We provide the simulation results, using NS-2, compare MarkMax with Drop Tail and show how MarkMax improves both the fairness and link utilization when connections have significantly different RTTs. We specify the algorithm, perform its theoretical analysis and provide simulation results which illustrate the performance of MarkMax. The work is published in [OBA08].

We give the conclusion and future work in Chapter 7.

Chapter 2

Batch Processor Sharing with Hyper-Exponential Service Time

2.1 Summary

One of the main goals to study BPS is the possibility of its application to age-based scheduling and the possibility to take into account the burstiness of the arrival process. Bursty arrivals often occur in modern systems such as web servers. Age-based scheduling is used in differentiation of short and long flows in the Internet.

We study the BPS model with the hyper-exponential service time distribution. For this distribution we solve Kleinrock's, integral equation for the expected conditional response time function and prove the concavity of the solution with respect to the job size. We note that the concavity of the expected conditional sojourn time for the BPS with the hyper-exponential job size distribution was proven using another method in [KK08].

We apply the obtained results to find the mean conditional sojourn time in the Two Level Processor Sharing (TLPS) system when the job size distribution is hyper-exponential. We prove that in the TLPS system the mean conditional sojourn time is not a concave function.

The results of this chapter are published in [Osi08a].

2.2 Introduction

The Processor Sharing (PS) queueing systems are now often used to model communication and computer systems. The PS systems were first introduced by Kleinrock (see [Kle76a] and references therein). Under the PS policy each job receives an equal share of the processor.

PS with batch arrivals (BPS) is not yet characterized fully. Kleinrock *et al.* [KMR71] first studied BPS. They found that the derivative of the expected response time satisfies an integral equation and found the analytical solution in the case when the job size (service time) distribution function has the form $F(x) = 1 - p(x)e^{-\mu x}$ where $p(x)$ is a polynomial.

Bansal [Ban03], using Kleinrock's integral equation, obtained the solution for the Laplace transform of the expected conditional service time as a solution of the system of linear equations, when the job size distribution is a hyper-exponential distribution. Also he considers distributions with a rational Laplace transform. Rege and Sengupta [RS93] obtained the expression for the response time in condition upon the number of customers in the system. Feng and Mishra [FM03a] provided bounds for the expected conditional response time, the bounds depend on the second moment of the service time distribution. Avrachenkov *et al.* [AAB05] proved existence and uniqueness of the solution of Kleinrock's integral equation and provided asymptotic analysis and bounds on the expected conditional response time.

We study the BPS model with the hyper-exponential service time distribution. For this distribution we solve Kleinrock's integral equation for the expected conditional response time function and prove the concavity of the solution with respect to the job size. We note that the concavity of the expected conditional sojourn time for the BPS with the hyper-exponential job size distribution was proven using another method in [KK08].

One of the main goals to study BPS is the possibility of its application to age-based scheduling and the possibility to take into account the burstiness of the arrival process. Bursty arrivals often occur in modern systems such as web servers. Age-based scheduling is used in differentiation of short and long flows in the Internet. A quite general set of age-based scheduling mechanisms was introduced by Kleinrock and termed as Multi Level PS (MLPS). In MLPS jobs are classified into different classes depending on their attained amount of service. Jobs within the same class are served according to FCFS, PS or FB policy. The classes themselves are served according to the FB policy, so that the priority is given to the jobs with small sizes.

We study the Two Level PS (TLPS) scheduling mechanism, a particular case of age-based scheduling. It is based on the differentiation of jobs according to some threshold and gives priority to jobs with small sizes. The TLPS scheduling mechanism can be used to model size based differentiation in TCP/IP networks and Web server request differentiation, see [AA06, AABN04].

It is known that many probability distributions associated with network traffic and, in

particular, the file size distribution in the Internet are often modelled with heavy-tailed distributions. In [BM06, FW98] it is shown that a heavy-tailed distribution can be approximated with a hyper-exponential distribution with a significant number of phases. We study the TLPS model with the hyper-exponential service time distribution. We apply the results of the BPS queueing model to the TLPS model with the hyper-exponential service time distribution, find an expression for expected conditional sojourn time function and prove that it is not a concave function with respect to the job sizes.

The Chapter is organized as follows. In Section 2.3 the BPS scheduling mechanism with the hyper-exponential service time distribution is considered. In Section 2.4 the results obtained for the BPS model are applied to the TLPS model, where the job size distribution is also hyper-exponential. We put some technical proofs in the Appendix.

This results of this chapter were published in [Osi08a], Chapter 3 of the current Thesis, more detailed proofs can be found in Research Report [Osi07]. The analysis of the queue with batch arrivals is also used in Chapter 5.

2.3 The analysis of the Batch Arrival Processor Sharing model

Let us consider an $M/G/1$ system with batch arrivals and PS queueing discipline. The batches arrive according to a Poisson process with arrival rate λ. Let $\overline{n} > 0$ be the average size of a batch. Let $b > 0$ be the average number of jobs that arrive with (and in addition to) an arbitrary job which is tagged upon arrival. Let $B(x)$ be the required job size (service time) distribution and $\overline{B}(x) = 1 - B(x)$ be its complementary distribution function. The load is given by $\rho = \lambda \overline{n} m$, with $m = \int_0^\infty x dB(x)$. We assume that the system is stable, $\rho < 1$.

It is known that many important probability distributions associated with network traffic are heavy-tailed. In particular, file size distributions observed in the Internet are often heavy-tailed. The heavy-tailed distributions are not only important and prevalent, but also difficult to analyze. In [BM06, FW98] it was shown that it is possible to approximate a heavy-tailed distribution by a hyper-exponential distribution with a significant number of phases. Thus, in our work we use the hyper-exponential function to represent the job size distribution function

$$B(x) = 1 - \sum_i p_i \, e^{-\mu_i x}, \quad 1 < N \leq \infty, \tag{2.1}$$

$p_i > 0$, $\mu_i > 0$, $i = 1, \ldots, N$, $\sum_i p_i = 1$. Without loss of generality, we can assume that

$$0 < \mu_N < \mu_{N-1} < \ldots < \mu_2 < \mu_1 < \infty. \tag{2.2}$$

By $\sum_i$ and $\prod_i$ we mean $\sum_{i=1}^N$ and $\prod_{i=1}^N$. By $\sum_{i \neq j}$ or $\prod_{i \neq j}$ we mean $\sum_{i=1,\ldots,N, i \neq j}$ and $\prod_{i=1,\ldots,N, i \neq j}$.

Let $\alpha(x)$ be the expected conditional response time in the BPS system for a job with service time x and $\alpha'(x)$ be its derivative. Kleinrock showed in [Kle76a, Sec. 4.7] that $\alpha'(x)$ satisfies the following integro-differential equation

$$\alpha'(x) = \lambda\overline{n}\int_0^\infty \alpha'(y)\overline{B}(x+y)dy + \lambda\overline{n}\int_0^x \alpha'(y)\overline{B}(x-y)dy + b\overline{B}(x) + 1. \tag{2.3}$$

Before presenting our main result let us prove auxiliary lemmas. Let us define

$$\Psi(s) = 1 - \lambda\overline{n}\sum_i \frac{p_i}{s+\mu_i}. \tag{2.4}$$

Lemma 2.1 *The zeros b_i, $i = 1, \ldots, N$ of the rational function (2.4) are all real, distinct, positive and satisfy the following inequalities:*

$$0 < b_N < \mu_N, \quad \mu_{i+1} < b_i < \mu_i, \quad i = 1, \ldots, N-1. \tag{2.5}$$

Proof. Following the approach of [FMI80], the equation $\Psi(s) = 0$ has N_1 roots $-b_i$, $i = 1, \ldots, N_1$, where N_1 is the number of distinct elements within μ_i. We have $N_1 = N$ because of (2.2). All $-b_i$, $i = 1, \ldots, N$ are real, distinct, negative and satisfy the following inequalities: $0 > -b_N > -\mu_N$, $-\mu_{i+1} > -b_i > -\mu_i$, $i = 1, \ldots, N-1$. With this we prove the statement of Lemma 2.1. ∎

Lemma 2.2 *The solution of the following system of linear equations:*

$$\sum_j \frac{x_j}{\mu_q^2 - b_j^2} = 1, \quad q = 1, \ldots, N, \tag{2.6}$$

is unique and is given by

$$x_k = \frac{\prod_{q=1,\ldots,N}(\mu_q^2 - b_k^2)}{\prod_{q\neq k}(b_q^2 - b_k^2)}, \quad k = 1, \ldots, N. \tag{2.7}$$

Proof. The proof is given in the appendix. ∎

Corollary 2.1 *The solution of equation (2.6) is positive. Namely, $x_k > 0$ for $k = 1, \ldots, N$.*

Proof. It follows from (2.2) and (2.5). ∎

Now we can prove our main result.

Theorem 2.1 *The expected conditional response time in the BPS queue with the hyper-exponential job size distribution function as in (2.1) is given by:*

$$\alpha(x) = c_0 x - \sum_k \frac{c_k}{b_k} e^{-b_k x} + \sum_k \frac{c_k}{b_k}, \quad \alpha(0) = 0, \tag{2.8}$$

$$c_0 = \frac{1}{1-\rho}, \tag{2.9}$$

$$c_k = \frac{b}{2\lambda\overline{n}} \left(\frac{\prod_q (\mu_q^2 - b_k^2)}{b_k \prod_{q \neq k} (b_q^2 - b_k^2)} \right), \quad k = 1, \ldots, N, \tag{2.10}$$

where b_k, $k = 1, \ldots, N$ are the solutions of the equation $\Psi(s) = 0$ and are all positive, distinct, real and satisfy inequalities (2.5).

Proof. Let us denote by $L_{\alpha'}(s)$ the Laplace transform of $\alpha'(x)$ and $L_i = L_{\alpha'}(\mu_i)$, $i = 1, \ldots, N$. From (2.1), (2.3):

$$\alpha'(x) = \lambda\overline{n} \sum_i p_i L_i e^{-\mu_i x} + \lambda\overline{n} \int_0^x \alpha'(y) \overline{B}(x-y) dy + b\overline{B}(x) + 1.$$

Taking the Laplace transform of the above equation and using the convolution property, we have

$$L_{\alpha'}(s)\Psi(s) = \lambda\overline{n} \sum_i \frac{p_i L_i}{s + \mu_i} + b \sum_i \frac{p_i}{s + \mu_i} + \frac{1}{s}.$$

Using the results of Lemma 2.1 we get:

$$L_{\alpha'}(s) = \lambda\overline{n} \sum_i p_i L_i \frac{\prod_{k \neq i}(s + \mu_k)}{\prod_k (s + b_k)} + b \sum_i p_i \frac{\prod_{k \neq i}(s + \mu_k)}{\prod_k (s + b_k)} + \frac{1}{s} \frac{\prod_k (s + \mu_k)}{\prod_k (s + b_k)}. \tag{2.11}$$

Hence there exist c_0 and c_k, $k = 1, \ldots, N$ such that:

$$L_{\alpha'}(s) = \frac{c_0}{s} + \sum_k \frac{c_k}{s + b_k}. \tag{2.12}$$

Then, taking the inversion of the Laplace transform and using $\alpha(0) = 0$, we get (2.8). From (2.11) and (2.12)

$$c_0 = L_{\alpha'}(s) s|_{s=0} = \frac{\prod_i \mu_i}{\prod_i b_i}. \tag{2.13}$$

From (2.4) we have

$$\frac{\prod_i b_i}{\prod_i \mu_i} = \Psi(s)|_{s=0} = 1 - \lambda\overline{n} \sum_i \frac{p_i}{\mu_i} = 1 - \rho.$$

So, then for c_0 we have (2.9). Let us find c_k, $k = 1, \ldots, N$. We denote:

$$L^*_{\alpha'}(s) = \sum_i \frac{c_i}{s + b_i}, \tag{2.14}$$

and $L^*_j = L^*_{\alpha'}(\mu_j)$, $j = 1, \ldots, N$. Using (2.11), (2.12) and (2.14), we can write

$$L^*_{\alpha'}(s)\frac{\prod_i(s + b_i)}{\prod_i(s + \mu_i)} = \lambda\overline{n}\sum_i \frac{p_i L^*_i}{s + \mu_i} + b\sum_i \frac{p_i}{s + \mu_i}.$$

Multiplying the above equation by $(s + \mu_q)$, setting $s = -\mu_q$, $q = 1, \ldots, N$ and using (2.14) we get

$$\sum_j \frac{c_j}{b_j - \mu_q}\frac{\prod_i(b_i - \mu_q)}{\prod_{i\neq q}(\mu_i - \mu_q)} = \lambda\overline{n}p_q\sum_j \frac{c_j}{b_j + \mu_q} + bp_q, \quad q = 1, \ldots, N. \tag{2.15}$$

Let us notice that from (2.4) we have the following

$$\frac{\prod_i(b_i - \mu_q)}{\prod_{i\neq q}(\mu_i - \mu_q)} = \Psi(s)(s + \mu_q)|_{s=-\mu_q} = -\lambda\overline{n}\,p_q, \quad q = 1, \ldots, N.$$

Then, using (2.15), we get

$$\sum_j \frac{c_j b_j}{\mu_q^2 - b_j^2} = \frac{b}{2\lambda\overline{n}}, \quad q = 1, \ldots, N. \tag{2.16}$$

So, c_k, $k = 1, \ldots, N$ are solutions of the linear system (2.16). If we denote

$$x_k = \frac{c_k b_k}{b/(2\lambda\overline{n})}, \quad k = 1, \ldots, N,$$

then the system (2.16) will take the form (2.6) and by Lemma 2.2 for c_k we have the statement (2.10). This completes the proof of Theorem 2.1. ∎

Corollary 2.2 *The expected conditional sojourn time function in the BPS system with the hyper-exponential job size distribution as in (2.1) is a strictly concave function.*

Proof. The function (2.8) is a strictly concave function if $\alpha''(x) = -\sum_k c_k b_k e^{-b_k x} < 0$. This is true, as $c_k > 0$, $b_k > 0$, $k = 1, \ldots, N$, which follows from $b > 0$, $\overline{n} > 0$, Corollary 2.1 and Lemma 2.1. ∎

The result of Corollary 2.2 was also proven using another method in [KK08].

Remark 2.1 *Let us denote by $n(x)$ an average density of jobs still in the system which have received an amount of service equal to x . Then $n(x)dx$ is the average number of jobs still in*

the system which received an amount of service between x and $x + dx$. From [Kle76a, Ch. 4], we have

$$n(x)dx = \lambda \overline{B}(x)\alpha'(x)dx.$$

As $\alpha'(x)$ and $\overline{B}(x)$ are positive decreasing functions, $n(x)dx$ is also a positive decreasing function. Then the average number of jobs which are still in the system and received an amount of service around x is decreasing with respect to the received amount of service. This property is not true for all queuing systems. In particular, as we will see later, it is not true for the TLPS system with the hyper-exponential job size distribution.

Let us denote the expected sojourn time in the BPS system as $\overline{T}^{BPS} = \int_0^\infty \alpha'(x)\overline{B}(x)dx$. Let us prove the following theorem.

Theorem 2.2 *The expected sojourn time $\overline{T}^{BPS}$ in the BPS system with the hyper-exponential job size distribution as in (2.1) is given by*

$$\overline{T}^{BPS} = \frac{m}{1-\rho} + \sum_{i,j} \frac{p_i c_j}{\mu_i + b_j}.$$

Proof. As the expected sojourn time $\overline{T}^{BPS}$ is given by

$$\overline{T}^{BPS} = \int_0^\infty \alpha'(x)\overline{B}(x)dx,$$

then using (2.8) we receive the statement of the Corollary. ∎

2.4 The analysis of the Two Level Processor Sharing model

Let us study the TLPS scheduling discipline with the hyper-exponential job size distribution $F(x)$. Let $\overline{F}(x) = 1 - F(x)$. The jobs arrive to the system according to a Poisson process with rate λ. We give a detailed TLPS model description in the Section 3.3 of the later Chapter 3. Let $\theta > 0$ be a given threshold. There are two queues in the system, low and high priority queues. Both queues are served with the PS discipline. In the high priority queue jobs are served until they receive θ of service, if after the job received θ amount of service it is still in the system, it waits in the low priority queue to be served. The low priority queue is served only when the high priority queue is empty, thus, we can consider the low priority queue as a queue with batch arrivals, see also [Kle76a, Sec. 4.7].

Let us denote by $\overline{T}^{TLPS}(x)$ the expected conditional sojourn time in the TLPS system for a job of size x and by $\overline{T}(\theta)$ the expected sojourn time of the system. According to [Kle76a, Sec.

4.7] the expected conditional sojourn time of the system is given by:

$$\overline{T}^{TLPS}(x) = \begin{cases} \dfrac{x}{1-\rho_\theta}, & x \in [0,\theta], \\ \dfrac{\overline{W}(\theta)+\theta+\alpha(x-\theta)}{1-\rho_\theta}, & x \in (\theta,\infty). \end{cases}$$

Here we use the following notations. Let us denote by $\overline{X_\theta^n} = \int_0^\theta n y^{n-1}\overline{F}(y)dy$ the n-th moment for the distribution truncated at θ and $\rho_\theta = \lambda\overline{X_\theta^1}$ the utilization factor. According to [Kle76a, Sec. 4.7] the average batch size is $\overline{n} = \overline{F}(\theta)/(1-\rho_\theta)$, the average number of jobs that arrive to the low priority queue in addition to the tagged job is $b = 2\lambda\overline{F}(\theta)(\overline{W}(\theta)+\theta)/(1-\rho_\theta)$ and $\alpha(x-\theta)/(1-\rho_\theta)$ is the time spent by the job in the low priority queue. Here $\overline{W}(\theta) = \lambda\overline{X_\theta^2}/(2(1-\rho_\theta))$.

Using the result of Theorem 2.1 Section 2.3 we obtain the following result, which is used in [ABO07] and in Chapter 3.

Theorem 2.3 *In the TLPS priority queue with the hyper-exponential job size distribution:*

$$\alpha(x) = c_0(\theta)x - \sum_k \frac{c_k(\theta)}{b_k(\theta)}e^{-b_k(\theta)x} + \sum_k \frac{c_k(\theta)}{b_k(\theta)},$$

$$\alpha(0) = 0,$$

$$c_0(\theta) = \frac{1-\rho_\theta}{1-\rho},$$

$$c_k(\theta) = \frac{b}{2\lambda\overline{n}}\left(\frac{\prod_{q=1,\dots,N}(\mu_q^2 - b_k^2(\theta))}{b_k(\theta)\prod_{q\neq k}(b_q^2(\theta) - b_k^2(\theta))}\right), \quad k = 1,\dots,N,$$

where $b_i(\theta)$, $i = 1,\dots,N$ are the roots of the rational function $1 - \frac{\lambda}{1-\rho_\theta}\sum_i \frac{\overline{F_\theta^i}}{s+\mu_i} = 0$, and satisfy the following inequalities: $0 < b_N(\theta) < \mu_N$, $\mu_{i+1} < b_i(\theta) < \mu_i$, $i = 1,\dots,N-1$. Here $\overline{F_\theta^i} = p_i e^{-\mu_i\theta}$, $i = 1,\dots,N$. The coefficients $c_k(\theta)$, $k = 1,\dots,N$ are strictly positive for positive θ. The function $\alpha(x)$ is a strictly concave function with respect to the job size for positive θ.

Corollary 2.3 *The expected conditional sojourn time $\overline{T}^{TLPS}(x)$ in the TLPS queue with the hyper-exponential job size distribution is a strictly concave function for $x > \theta$, linear for $x < \theta$ and is not a concave function on the interval $(0,\infty)$ with respect to the job sizes for positive values of θ.*

Proof. The concavity of $\overline{T}^{TLPS}(x)$ for $x > \theta$ follows from Theorem 2.3. The function $\overline{T}^{TLPS}(x)$ is linear for $x < \theta$, this follows from the standard PS model. As $\overline{T}^{TLPS}(x)$ is not continuous at the point $x = \theta$, it is also not concave on the interval $(0,\infty)$. ∎

From Theorem 2.3 it follows that $1 < \alpha'(0)$ and then $\overline{T}^{TLPS\prime}(x)|_{x=\theta-0} < \overline{T}^{TLPS\prime}(x)|_{x=\theta+0}$. Then for the TLPS system the average number of jobs which are still in the system and received

an amount of service around x is not a decreasing and not even monotone function with respect to the received amount of service.

Now let us give the expression of the mean sojourn time in the TLPS system, which we use in Chapter 3 of the current Thesis.

Theorem 2.4 *The expected sojourn time $\overline{T}(\theta)$ in the TLPS system with the hyper-exponential distribution function is given by the following equation:*

$$\begin{aligned}\overline{T}(\theta) &= \frac{\overline{X_\theta^1} + \overline{W}(\theta)\overline{F}(\theta)}{1-\rho_\theta} + \frac{(m - \overline{X_\theta^1})}{1-\rho} \\ &+ \frac{(\overline{W}(\theta)+\theta)}{1-\rho_\theta}\sum_{i,j}\frac{\overline{F_\theta^i}}{b_j(\theta)(\mu_i + b_j(\theta))}\frac{\prod_q(\mu_q^2 - b_j^2(\theta))}{\prod_{q\neq j}(b_q^2(\theta) - b_j^2(\theta))},\end{aligned} \tag{2.17}$$

where $b_i(\theta)$, $i = 1, \ldots, N$ are defined as in Theorem 2.3.

Proof. According to [Kle76a, Sec. 4.7]

$$\overline{T}(\theta) = \frac{\overline{X_\theta^1} + \overline{W}(\theta)\overline{F}(\theta)}{1-\rho_\theta} + \frac{1}{1-\rho_\theta}\overline{T}^{BPS}(\theta).$$

Then using the result of Theorem 2.3 we get the statement of the current Theorem. ∎

2.5 Conclusion

We study the BPS queueing model, when the job size distribution is hyper-exponential, and we find an analytical expression of the expected conditional response time and for the expected sojourn time. We show that the function of the expected conditional sojourn time in the BPS system with hyper-exponential job size distribution is a concave function with respect to job sizes. We apply the results obtained for the BPS model to the TLPS scheduling mechanism with the hyper-exponential job size distribution and we find the expressions of the expected conditional response time and expected response time for the TLPS model.

2.6 Appendix

Lemma 2.2: The solution of the system of linear equations (2.6) is unique and is given by (2.7).

Proof. Let x, $\underline{1}$ be two vectors of size N and D be the matrix of size $N \times N$.

$$x = [x_1, x_2, \ldots, x_N]^T, \quad \underline{1} = [1, 1, \ldots, 1]^T|_{1\times N}$$

$$D = \left[\frac{1}{\mu_i^2 - b_j^2}\right]_{i,j=1,\ldots,N}$$

Then, system (2.6) can be rewritten as

$$Dx = \underline{1}.$$

Applying Cramer's rule [Kur72] we obtain:

$$x_k = \frac{\det D_k}{\det D}, \quad k = 1, \ldots, N, \tag{2.18}$$
$$D_k = [D_{[1]}, \ldots D_{[k-1]}, \underline{1}, D_{[k+1]}, \ldots D_{[N]}].$$

As D is a Cauchy matrix, its determinant is known [Kur72]:

What is that?

$$\det D = \frac{\prod_{1 \le j < k \le N} ((\mu_j^2 - \mu_k^2)(b_k^2 - b_j^2)))}{\prod_{j,k=1,\ldots,N} (\mu_j^2 - b_k^2)}. \tag{2.19}$$

Under the product sign by $1 \le j < k \le N$ we mean that we take all the combinations (j, k) such as $1 \le j < N$, $1 < k \le N$ and $j < k$. By $j, k = 1, \ldots, N$ we mean that we take all the pairs (j, k) such as $1 \le j \le N$, $1 \le k \le N$.

Due to (2.5), $\det D > 0$ and we can use Cramer's rule to calculate x_k. Let us find $\det D_k$.

$$\begin{aligned} \det D_k &= \det [D_{[1]}, \ldots D_{[k-1]}, \underline{1}, D_{[k+1]}, \ldots D_{[N]}] \\ &= (-1)^{k-1} \det [\underline{1}, D_{[1]}, \ldots D_{[k-1]}, D_{[k+1]}, \ldots D_{[N]}]. \end{aligned}$$

To simplify the ensuing computations let us introduce the following notations:

$$\beta_i = -b_{i-1}^2, \quad i = 2, \ldots, k, \quad \beta_i = -b_i^2, \quad i = k+1, \ldots, N.$$

Let us notice that here β_i, $i = 1, \ldots, N$ depend on the index k. As at this point the index k is fixed we do not represent this dependency in the notation of β_i.

Then, we have

$$\det D_k = (-1)^{k-1} \begin{vmatrix} 1 & \frac{1}{\mu_1^2+\beta_2} & \cdots & \frac{1}{\mu_1^2+\beta_N} \\ \cdots & \cdots & \cdots & \cdots \\ 1 & \frac{1}{\mu_N^2+\beta_2} & \cdots & \frac{1}{\mu_N^2+\beta_N} \end{vmatrix}_{N \times N}$$

Under the sign of determinant we subtract the first line from all the other lines.

$$\det D_k = (-1)^{k-1} \begin{vmatrix} 1 & \frac{1}{\mu_1^2+\beta_2} & \cdots & \frac{1}{\mu_1^2+\beta_k} & \cdots & \frac{1}{\mu_1^2+\beta_N} \\ 0 & \frac{\mu_1^2-\mu_2^2}{(\mu_2^2+\beta_2)(\mu_1^2+\beta_2)} & \cdots & \frac{\mu_1^2-\mu_2^2}{(\mu_2^2+\beta_k)(\mu_1^2+\beta_k)} & \cdots & \cdots \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & \frac{\mu_1^2-\mu_N^2}{(\mu_N^2+\beta_2)(\mu_1^2+\beta_2)} & \cdots & \frac{\mu_1^2-\mu_N^2}{(\mu_N^2+\beta_k)(\mu_1^2+\beta_k)} & \cdots & \cdots \end{vmatrix}_{N \times N}$$

$$\det D_k = \frac{(-1)^{k-1} \prod_{k=2,\ldots,N} (\mu_1^2 - \mu_k^2)}{\prod_{k=2,\ldots,N} (\mu_1^2 + \beta_k)} \begin{vmatrix} \cdots & \frac{1}{\mu_2^2+\beta_k} & \cdots \\ \cdots & \cdots & \cdots \\ \cdots & \frac{1}{\mu_N^2+\beta_k} & \cdots \end{vmatrix}_{(N-1)\times(N-1)}$$

So, as the above matrix under the sign of determinant is a Cauchy matrix of size $N-1$, the following equation holds:

$$\det D_k = \frac{(-1)^{k-1}\prod\limits_{q=2,\dots,N}(\mu_1^2-\mu_q^2)\prod\limits_{2\le j<q\le N}\left((\mu_j^2-\mu_q^2)(\beta_j-\beta_q)\right)}{\prod\limits_{q=2,\dots,N}(\mu_1^2+\beta_q)}\prod_{j,q=2,\dots,N}(\mu_j^2+\beta_q)$$

Let us recall that $\beta_i=-b_{i-1}^2$, $i=2,\dots,k$ and $\beta_i=-b_i^2$, $i=k+1,\dots,N$, then

$$\begin{aligned}\det D_k &= \frac{(-1)^{k-1}\prod\limits_{1\le j<q\le N}(\mu_j^2-\mu_q^2)\prod\limits_{1\le j<q\le N,\, j,q\neq k}(b_q^2-b_j^2)}{\prod\limits_{j,q=1,\dots,N,q\neq k}(\mu_j^2-b_q^2)}\\ &= \frac{\prod\limits_{1\le j<q\le N}\left((\mu_j^2-\mu_q^2)(b_q^2-b_j^2)\right)\prod\limits_{j=1,\dots,N}(\mu_j^2-b_k^2)}{\prod\limits_{j,q=1,\dots,N}(\mu_j^2-b_q^2)\prod\limits_{q=1,\dots,N,\,q\neq k}(b_q^2-b_k^2)}.\end{aligned}$$

Finally, from (2.18) and (2.19), we have expression (2.7) for x_k, which proves Lemma 2.2. ∎

Nice proof.

Chapter 3

Optimal choice of threshold in Two Level Processor Sharing

3.1 Summary

We analyze the TLPS scheduling discipline with the hyper-exponential job size distribution and with the Poisson arrival process. TLPS is a convenient model to study the benefit of the file size based differentiation in TCP/IP networks. In the case of the hyper-exponential job size distribution with two phases, we find a closed form analytic expression for the expected sojourn time and an approximation for the optimal value of the threshold that minimizes the expected sojourn time. Using NS-2 simulator we implement the TLPS algorithm in the router queue and provide simulation results for the case of two phase hyper-exponential job size distribution. We show that the found optimal threshold approximation value minimizes mean waiting time in the TLPS system between other threshold values. We show that with the TLPS policy the relative gain in mean waiting time in comparison with the DropTail policy is very near to the relative gain which can be reached using the optimal LAS policy and goes up to 36%.

In the case of the hyper-exponential job size distribution with more than two phases, we derive a tight upper bound for the expected sojourn time conditioned on the job size. We show that when the variance of the job size distribution increases, the gain in system performance increases and the sensitivity to the choice of the threshold near its optimal value decreases.

The results of this chapter are published in [ABO07].

3.2 Introduction

The Two Level Processor Sharing (TLPS) scheduling discipline was first introduced by Kleinrock, see [Kle76a, Sec. 4.7]. It uses the differentiation of jobs according to a threshold on the attained service and gives priority to the jobs with small sizes. The TLPS scheduling mechanism can be applied in file size based differentiation in TCP/IP networks [AANO04, AABN04, FM03b] and Web server request differentiation [GM02a, HBSBA03]. A detail description of the TLPS discipline is presented in the ensuing chapter. Of course, TLPS provides a sub-optimal mechanism in comparison with SRPT, which minimizes the expected sojourn time, see [Sch68]. Nevertheless, as was shown in [AA06], when the job size distribution has a decreasing hazard rate, the performance of TLPS with appropriate choice of threshold is very close to optimal.

In the present chapter we characterize the optimal value of the threshold when the service time is hyper-exponential. The motivation to study TLPS with the hyper-exponential service time is as follows. The distribution of file sizes in the Internet often can be modelled with a heavy-tailed distribution. It is known that heavy-tailed distributions can be approximated with hyper-exponential distributions with a significant number of phases [BM06, FW98]. Also in [KSH03], it was shown that a hyper-exponential distribution models well the file size distribution in the Internet. In [KSH03] authors propose an efficient algorithm to approximate heavy-tailed distributions with hyper-exponential distributions with many phases.

The chapter organization and main results are as follows. In Section 3.3 we provide the model formulation, main definitions and equations. In Section 3.4 we study the TLPS discipline in the case of the hyper-exponential job size distribution with two phases. It is known that the Internet connections belong to two distinct classes with very different sizes of transfer. The first class is composed of short HTTP connections and P2P signaling connections. The second class corresponds to downloads (PDF files, MP3 files, MPEG files, etc.), see Subsection 1.1.3. This fact provides motivation to consider first the hyper-exponential job size distribution with two phases.

We find an analytic expression for the expected sojourn time in the TLPS system. Then, we present the approximation of the optimal threshold which minimizes the expected sojourn time. We show that the approximated value of the threshold tends to the optimal threshold when the second moment of the job size distribution function goes to infinity.

We show that the ratio between the expected sojourn time of the TLPS system and the expected sojourn time of the standard PS system can be arbitrary small for very high loads. For realistic loads this ratio can reach 1/2. We also show that the system performance is not too sensitive to the choice of the threshold around its optimal value.

In [AABN04] authors provide the scheduling algorithm, RuN2C, which is based on the TLPS policy and uses packets sequence numbers to schedule packets. Using NS-2 simulator

we implement the TLPS algorithm in the router queue, which schedules packets according to the attained service of every connection presented in the system. For that we keep the trace of the connection's attained service until there are no more packets from the connection in the queue. We provide the simulation results for the different values of the threshold and show that analytically found threshold approximation minimizes mean waiting time in the TLPS system. We compare the mean waiting time in the system when the bottleneck queue is scheduled with TLPS, LAS and DropTail policies. We found that the relative gain of the TLPS policy with the approximated value of the optimal threshold can achieve up to 36% in comparison with the DropTail policy and is very close to the relative gain achieved with the optimal LAS policy in comparison with the DropTail policy.

In Section 3.5 we analyze the TLPS discipline when the job size distribution is hyper-exponential with many phases. We provide an expression of the expected conditional sojourn time as the solution of a system of linear equations. Also we apply an iteration method to find the expression of the expected conditional sojourn time and using the resulting expression obtain an explicit and tight upper bound for the expected sojourn time function. In the experimental results we show that the relative error of the latter upper bound with respect to the expected sojourn time function is 6-7%.

We study the properties of the expected sojourn time function when the parameters of the job size distribution function are selected in such a way that with the increasing number of phases the variance increases. We show numerically that with the increasing number of phases the relative error of the found upper bound decreases. We also show that when the variance of the job size distribution increases the gain in system performance increases and the sensitivity of the system to the selection of the approximate optimal threshold value decreases.

We put some technical proofs in the Appendix.

3.3 Model description

3.3.1 Main definitions

We study the TLPS scheduling discipline with the hyper-exponential job size distribution. The jobs arrive to the system according to a Poisson process with rate λ. We measure the job size in time units. Specifically, as the job size we define the time which would be spent by the server to treat the job if there were no other jobs in the system.

Let $\theta > 0$ be a given threshold. When a new job arrives to the system, it goes to the high priority queue, where it is served until it receives the amount of service θ. If the job is still in the system and needs more service than θ, the rest of the job, which is not yet served, goes to the low priority queue. So, the jobs which attain an amount of service more then θ are accumulated in the low priority queue. The low priority queue is served when the high priority queue is empty.

Both queues are served according to the PS discipline, namely, the server equally divides its capacity among all jobs present in the queue. When the high priority queue is empty, the jobs which are accumulated in the low priority queue arrive to the server in a batch. Thus, we can consider the low priority queue as a queue with batch arrivals, see also [Kle76a, Sec. 4.7].

Let us denote the job size distribution by $F(x)$. By $\overline{F}(x) = 1 - F(x)$ we denote the complementary distribution function. The mean job size is given by $m = \int_0^\infty x dF(x)$ and the system load is $\rho = \lambda m$. We assume that the system is stable ($\rho < 1$) and is in steady state.

It is known that many important probability distributions associated with network traffic are heavy-tailed. In particular, the file size distribution in the Internet is heavy-tailed. A distribution function has a heavy tail if $e^{\epsilon x}(1 - F(x)) \to \infty$ as $x \to \infty$, $\forall \epsilon > 0$. The heavy-tailed distributions are not only important and prevalent, but also difficult to analyze. Often it is helpful to have the Laplace transform of the job size distribution. However, there is evidently no convenient analytic expression for the Laplace transforms of the Pareto and Weibull distributions, the most common examples of heavy-tailed distributions. In [BM06, FW98], [FW98], it was shown that it is possible to approximate heavy-tailed distributions by hyper-exponential distributions with a significant number of phases. A hyper-exponential distribution $F_N(x)$ is a convex combination of N exponents, $1 \le N \le \infty$, namely,

$$F_N(x) = 1 - \sum_{i=1}^{N} p_i e^{-\mu_i x}, \quad \mu_i > 0, \quad p_i \ge 0, \quad i = 1, \dots, N, \quad \text{and} \quad \sum_{i=1}^{N} p_i = 1. \tag{3.1}$$

In particular, we can construct a sequence of hyper-exponential distributions such that it converges to a heavy-tailed distribution [BM06]. For instance, if we select

$$p_i = \frac{\nu}{i^{\gamma_1}}, \quad \mu_i = \frac{\eta}{i^{\gamma_2}}, \quad i = 1, \dots, N,$$
$$\gamma_1 > 1, \quad \frac{\gamma_1 - 1}{2} < \gamma_2 < \gamma_1 - 1,$$

where $\nu = 1/\sum_{i=1,\dots,N} i^{-\gamma_1}$, $\eta = \nu/m \sum_{i=1,\dots,N} i^{\gamma_2 - \gamma_1}$, then the first moment of the job size distribution is finite, but the second moment goes to infinity when $N \to \infty$. The first and the second moments m and d for the hyper-exponential distribution are given by:

$$m = \int_0^\infty x\, dF(x) = \sum_{i=1}^{N} \frac{p_i}{\mu_i}, \quad d = \int_0^\infty x^2 dF(x) = 2 \sum_{i=1}^{N} \frac{p_i}{\mu_i^2}. \tag{3.2}$$

Let us denote

$$\overline{F_\theta^i} = p_i e^{-\mu_i \theta}, \quad i = 1, \dots, N. \tag{3.3}$$

We note that $\sum_{i=1}^{N} \overline{F_\theta^i} = \overline{F}(\theta)$. The hyper-exponential distribution has a simple Laplace transform:

$$L_{\overline{F}(x)}(s) = \sum_{i=1}^{N} \frac{p_i \mu_i}{s + \mu_i}.$$

We would like to note that the hyper-exponential distribution has a decreasing hazard rate. In [AA06] it was shown that when a job size distribution has a decreasing hazard rate, then with an appropriate selection of the threshold the expected sojourn time of the TLPS system can be made close to optimal. Thus, in our work we use hyper-exponential distributions to represent job size distribution functions. In the first part of the current chapter we look at the case of the hyper-exponential job size distribution with two phases and in the second part of the chapter we study the case of more than two phases.

3.3.2 The expected sojourn time in the TLPS system

Let us denote by $\overline{T}^{TLPS}(x)$ the expected conditional sojourn time in the TLPS system for a job of size x. Of course, $\overline{T}^{TLPS}(x)$ also depends on θ, but for expected conditional sojourn time we only emphasize the dependence on the job size. On the other hand, we denote by $\overline{T}(\theta)$ the overall expected sojourn time in the TLPS system. Here we emphasize the dependence on θ as later we shall optimize the overall expected sojourn time with respect to the threshold value.

To calculate the expected sojourn time in the TLPS system we need to calculate the time spent by a job of size x in the high priority queue and in the low priority queue. For the jobs with size $x \leq \theta$ the system will behave as the standard PS system where the service time distribution is truncated at θ. Let us denote by

$$\overline{X_\theta^n} = \int_0^\theta y^n dF(y) + \theta^n \overline{F}(\theta) = \int_0^\theta n y^{n-1} \overline{F}(y) dy$$

the n-th moment of the distribution truncated at θ. The distribution truncated at θ equals to to $F(x)$ for $x \leq \theta$ and equals to 1 when $x > \theta$. In the following sections we will need

$$\overline{X_\theta^1} = m - \sum_{i=1}^N \frac{\overline{F_\theta^i}}{\mu_i}, \quad \overline{X_\theta^2} = 2\sum_{i=1}^N \frac{p_i}{\mu_i^2} - 2\theta \sum_{i=1}^N \frac{\overline{F_\theta^i}}{\mu_i} - 2\sum_{i=1}^N \frac{\overline{F_\theta^i}}{\mu_i^2}. \tag{3.4}$$

The utilization factor for the truncated distribution is

$$\rho_\theta = \lambda \overline{X_\theta^1} = \rho - \lambda \sum_{i=1}^N \frac{\overline{F_\theta^i}}{\mu_i}. \tag{3.5}$$

Then, the expected conditional response time is given by

$$\overline{T}^{TLPS}(x) = \begin{cases} \dfrac{x}{1-\rho_\theta}, & x \in [0, \theta], \\ \dfrac{\overline{W}(\theta) + \theta + \alpha(x-\theta)}{1-\rho_\theta}, & x \in (\theta, \infty). \end{cases}$$

Here $\overline{W}(\theta)$ is the mean workload in the system for the jobs of size less then θ and according to the Pollaczek-Khinchin formula equals to

$$\overline{W}(\theta) = \frac{\lambda \overline{X_\theta^2}}{2(1-\rho_\theta)}.$$

According to [Kle76a, Sec.4.7], $\theta/(1-\rho_\theta)$ expresses the time spent in the high priority queue, where the flow is served up to the threshold θ and $\overline{W}(\theta)/(1-\rho_\theta)$ is the time spent waiting for the high priority queue to empty. The remaining term $\alpha(x-\theta)/(1-\rho_\theta)$ is the time spent in the low priority queue. According to Kleinrock [Kle76a, Sec.4.7] the low priority queue can be interpreted as an interrupted PS queue with batch arrivals. Then, $\alpha'(x) = d\alpha/dx$ is the solution of the following integral equation

$$\alpha'(x) = \lambda\overline{n}\int_0^\infty \alpha'(y)\overline{B}(x+y)dy + \lambda\overline{n}\int_0^x \alpha'(y)\overline{B}(x-y)dy + b\overline{B}(x) + 1. \tag{3.6}$$

Here $\overline{n}$ is the average batch size, $\overline{B}(x)$ is the complementary truncated distribution and $b = b(\theta)$ is the average number of jobs that arrive to the low priority queue in addition to the tagged job. The expressions for parameters $\overline{n}$, $b(\theta)$ are explicitly explained in [Kle76a, Sec.4.7] and equal to

$$\overline{B}(x) = \frac{\overline{F}(\theta+x)}{\overline{F}(\theta)},$$
$$\overline{n} = \frac{\overline{F}(\theta)}{(1-\rho_\theta)},$$
$$b(\theta) = \frac{2\lambda\overline{F}(\theta)(\overline{W}(\theta)+\theta)}{(1-\rho_\theta)}.$$

The expected sojourn time in the system is given by the following equations:

$$\overline{T}(\theta) = \int_0^\infty \overline{T}^{TLPS}(x)dF(x),$$
$$\overline{T}(\theta) = \frac{\overline{X_\theta^1} + \overline{W}(\theta)\overline{F}(\theta)}{1-\rho_\theta} + \frac{1}{1-\rho_\theta}\overline{T}^{BPS}(\theta), \tag{3.7}$$
$$\overline{T}^{BPS}(\theta) = \int_\theta^\infty \alpha(x-\theta)dF(x) = \int_0^\infty \alpha'(x)\overline{F}(x+\theta)dx. \tag{3.8}$$

3.4 Hyper-exponential job size distribution with two phases

3.4.1 Notation and motivation

In the first part of our work we consider the hyper-exponential job size distribution with two phases. In particular, the application of the hyper-exponential job size distribution with two phases is motivated by the fact that in the Internet TCP connections belong to two distinct classes with very different sizes of transfer. The first class is composed of short HTTP connections and P2P signaling connections. The second class corresponds to downloads (PDF files, MP3 files, MPEG files, etc.). We discuss this problem more in the Introduction of the present Thesis, see Section 1.1.3.

According to (3.1) the cumulative distribution function $F(x)$ for $N = 2$ is given by

$$F(x) = 1 - p_1 e^{-\mu_1 x} - p_2 e^{-\mu_2 x},$$

where $p_1 + p_2 = 1$ and $p_1, p_2 > 0$. The mean job size m, the second moment d, the parameters $\overline{F_\theta^i}$, $\overline{X_\theta^1}$, $\overline{X_\theta^2}$ and ρ_θ are defined as in Section 3.3.1 and Section 3.3.2 by formulas (3.2), (3.3), (3.4), (3.5) with $N = 2$.

Let us define

$$\epsilon = \frac{\mu_2}{\mu_1}.$$

We note that the system has four free parameters. In particular, if we fix μ_1, ϵ, m, and ρ, the other parameters μ_2, p_1, p_2 and λ will be functions of the former parameters.

3.4.2 Explicit form for the expected sojourn time

To find the expression of $\overline{T}(\theta)$ we use the result we obtained in the previous Chapter 2, Section 2.4, Theorem 2.4 and so prove the following Theorem.

Theorem 3.1 *The expected sojourn time in the TLPS system with the hyper-exponential job size distribution with two phases is given by*

$$\overline{T}(\theta) = \frac{\overline{X_\theta^1} + \overline{W}(\theta)\overline{F}(\theta)}{1-\rho_\theta} + \frac{m - \overline{X_\theta^1}}{1-\rho} + \frac{b(\theta)\left(\mu_1\mu_2(m-\overline{X_\theta^1})^2 + \delta_\rho(\theta)\overline{F}^2(\theta)\right)}{2(1-\rho)\overline{F}(\theta)\left(\mu_1+\mu_2-\gamma(\theta)\overline{F}(\theta)\right)}, \tag{3.9}$$

where $\delta_\rho(\theta) = 1 - \gamma(\theta)(m - \overline{X_\theta^1}) = (1-\rho)/(1-\rho_\theta)$ *and* $\gamma(\theta) = \lambda/(1-\rho_\theta)$.

Proof. As we found in the previous Chapter 2, Section 2.4, Theorem 2.4,

$$\begin{aligned}\overline{T}(\theta) &= \frac{\overline{X_\theta^1} + \overline{W}(\theta)\overline{F}(\theta)}{1-\rho_\theta} + \frac{(m-\overline{X_\theta^1})}{1-\rho} \\ &+ \frac{(\overline{W}(\theta)+\theta)}{1-\rho_\theta}\sum_{i,j}\frac{\overline{F_\theta^i}}{b_j(\theta)(\mu_i+b_j(\theta))}\frac{\prod_q(\mu_q^2-b_j^2(\theta))}{\prod_{q\neq j}(b_q^2(\theta)-b_j^2(\theta))},\end{aligned} \tag{3.10}$$

where $b_i(\theta)$ are the roots of the rational function $\Psi(s) = 1 - \frac{\lambda}{1-\rho_\theta}\sum_i \frac{\overline{F_\theta^i}}{s+\mu_i} = 0$. Let us define $\delta_\rho(\theta) = (1-\rho)/(1-\rho_\theta)$ and $\gamma(\theta) = \lambda/(1-\rho_\theta)$. Then for the case of two phase job size distribution function $\Psi(s)$ equals to

$$\Psi(s) = \frac{s^2 + s(\mu_1+\mu_2-\gamma(\theta)\overline{F}(\theta)) + \mu_1\mu_2\delta_\rho(\theta)}{(s+\mu_1)(s+\mu_2)}$$

and has two roots, $-b_1(\theta)$ and $-b_2(\theta)$, which are the solutions of the square equation $s^2 + s(\mu_1 + \mu_2 - \gamma(\theta)\overline{F}(\theta)) + \mu_1\mu_2\delta_\rho(\theta) = 0$. Then we know that

$$b_1(\theta) + b_2(\theta) = \mu_1 + \mu_2 - \gamma(\theta)\overline{F}(\theta), \tag{3.11}$$

$$b_1(\theta)b_2(\theta) = \mu_1\mu_2\delta_\rho(\theta). \tag{3.12}$$

Simplifying expression (3.10) and using (3.11) and (3.12) we get expression (3.9) and so prove the statement of the Theorem.

The same result can be obtained using the Laplace transform based method described in [Ban03]. ∎

3.4.3 Optimal threshold approximation

We are interested in the minimization of the expected sojourn time function $\overline{T}(\theta)$ with respect to θ. Of course, one can differentiate the exact analytic expression provided in Theorem 3.1 and set the result of the differentiation to zero. However, this will give a transcendental equation for the optimal value of the threshold. In order to find an approximate solution of $\overline{T}'(\theta) = d\overline{T}(\theta)/d\theta = 0$, we approximate the derivative $\overline{T}'(\theta)$ by some function $\widetilde{\overline{T}}'(\theta)$ and obtain a solution for $\widetilde{\overline{T}}'(\tilde{\theta}_{opt}) = 0$.

Since in the Internet connections belong to two distinct classes with very different sizes of transfer (see Section 3.3.1), then to find the approximation of $\overline{T}'(\theta)$ we consider a particular case when $\mu_2 << \mu_1$. Let us introduce a small parameter ϵ such that

$$\mu_2 = \epsilon\mu_1, \quad p_1 = 1 - \frac{\epsilon\,(m\mu_1 - 1)}{1-\epsilon}, \quad p_2 = \frac{\epsilon\,(m\mu_1 - 1)}{1-\epsilon}.$$

We note that when $\epsilon \to 0$ the second moment of the job size distribution goes to infinity.

Lemma 3.1 *The following inequality holds:* $\mu_1\rho > \lambda$.

Proof. Since $p_1 > 0$ and $p_2 > 0$, then $m\mu_1 > 1$ and $m > \frac{1}{\mu_1}$. Taking into account that $\lambda m = \rho$, we get $\frac{\rho}{\lambda} > \frac{1}{\mu_1}$. Consequently, we have that $\mu_1\rho > \lambda$. ∎

Proposition 3.1 *The derivative of* $\overline{T}(\theta)$ *can be approximated by the following function:*

$$\widetilde{\overline{T}}'(\theta) = -e^{-\mu_1\theta}\mu_1 c_1 + e^{-\mu_2\theta}\mu_2 c_2,$$

where

$$c_1 = \frac{\rho(m\mu_1 - 1)}{\mu_1(m\mu_1 - \rho)(1-\rho)}, \quad c_2 = \frac{\rho m(m\mu_1 - 1)}{(m\mu_1 - \rho)^2}. \tag{3.13}$$

Namely,

$$\overline{T}'(\theta) - \widetilde{\overline{T}}'(\theta) = O(\mu_2/\mu_1).$$

Proof. Using the analytic expression for both $\overline{T}'(\theta)$ and $\widetilde{\overline{T}}'(\theta)$, we get the Taylor series for $\overline{T}'(\theta) - \widetilde{\overline{T}}'(\theta)$ with respect to ϵ, which shows that indeed

$$\overline{T}'(\theta) - \widetilde{\overline{T}}'(\theta) = O(\epsilon).$$

∎

Thus we have found an approximation of the derivative of $\overline{T}(\theta)$. Now we can find an approximation of the optimal threshold by solving the equation $\widetilde{\overline{T}}'(\theta) = 0$.

Theorem 3.2 *Let θ_{opt} denote the optimal value of the threshold. Namely, $\theta_{opt} = \arg\min \overline{T}(\theta)$. The value $\tilde{\theta}_{opt}$ given by*

$$\tilde{\theta}_{opt} = \frac{1}{\mu_1 - \mu_2} \ln\left(\frac{(\mu_1 - \lambda)}{\mu_2(1-\rho)}\right) \tag{3.14}$$

approximates θ_{opt} so that $\overline{T}'(\tilde{\theta}_{opt}) = o(\mu_2/\mu_1)$.

Proof. Solving the equation

$$\widetilde{\overline{T}}'(\theta) = 0,$$

we get an analytic expression for the approximation of the optimal threshold:

$$\widetilde{\theta}_{opt} = -\frac{1}{\mu_1(1-\epsilon)} \ln\left(\epsilon \frac{\mu_1(1-\rho)}{(\mu_1 - \lambda)}\right) = \frac{1}{\mu_1 - \mu_2} \ln\left(\frac{(\mu_1 - \lambda)}{\mu_2(1-\rho)}\right).$$

Let us show that the above threshold approximation is greater than zero. We have to show that $\frac{(\mu_1-\lambda)}{\mu_2(1-\rho)} > 1$. Since $\mu_1 > \mu_2$ and $\mu_1\rho > \lambda$ (see Lemma 3.1), we have

$$\begin{aligned} & \mu_1 > \mu_2 \\ \Longrightarrow \quad & \mu_1(1-\rho) > \mu_2(1-\rho) \\ \Longrightarrow \quad & \lambda < \mu_1\rho < \mu_1 - \mu_2(1-\rho) \\ \Longrightarrow \quad & (\mu_1 - \lambda) > \mu_2(1-\rho). \end{aligned}$$

Expanding $\overline{T}'(\widetilde{\theta}_{opt})$ as a power series with respect to ϵ gives:

$$\overline{T}'(\widetilde{\theta}_{opt}) = \epsilon^2(const_0 + const_1 \ln\epsilon + const_2 \ln^2\epsilon),$$

where $const_i$, $i = 1, 2$ are some constant values[1] with respect to ϵ. Thus,

$$\overline{T}'(\widetilde{\theta}_{opt}) = o(\epsilon) = o(\mu_2/\mu_1),$$

which completes the proof. ∎

From formula (3.14) we can see that $\widetilde{\theta}_{opt}$ is of the same order as $1/\mu_1 \ln(1/\epsilon)$. As a consequence $\widetilde{\theta}_{opt}$ goes to infinity when $\epsilon \to 0$. Also formula (3.14) indicates that the value of the threshold should be chosen between $1/\mu_1$ and $1/\mu_2$.

In the next proposition we characterize the limiting behavior of $\overline{T}(\theta_{opt})$ and $\overline{T}(\widetilde{\theta}_{opt})$ as $\epsilon \to 0$. In particular, we show that $\overline{T}(\widetilde{\theta}_{opt})$ tends to the exact minimum of $\overline{T}(\theta)$ when $\epsilon \to 0$.

[1] The expressions for the constants $const_i$ are cumbersome and can be found using Maple command "series".

Proposition 3.2

$$\lim_{\epsilon\to 0}\overline{T}(\theta_{opt}) = \lim_{\epsilon\to 0}\overline{T}(\widetilde{\theta}_{opt}) = \frac{m}{1-\rho} - c_1,$$

where c_1 is given by (3.13).

Proof. We find the following limit as $\epsilon \to 0$:

$$\lim_{\epsilon\to 0}\overline{T}(\theta) = \frac{m}{1-\rho} - c_1 + c_1 e^{-\mu_1\theta},$$

where c_1 is given by (3.13). Since $\lim_{\epsilon\to 0}\overline{T}(\theta)$ is a decreasing function, the optimal threshold for it is $\theta_{opt} = \infty$. Thus,

$$\lim_{\epsilon\to 0}\overline{T}(\theta_{opt}) = \lim_{\theta\to\infty}\lim_{\epsilon\to 0}\overline{T}(\theta) = \frac{m}{1-\rho} - c_1.$$

On the other hand, we obtain

$$\lim_{\epsilon\to 0}\overline{T}(\widetilde{\theta}_{opt}) = \frac{m}{1-\rho} - c_1,$$

which proves the proposition. ∎

Let us denote by $g(\rho) = \frac{\overline{T}^{PS} - \overline{T}(\widetilde{\theta}_{opt})}{\overline{T}^{PS}}$ the relative gain of the TLPS system with the optimal threshold approximation (3.14) with respect to the PS system. In the next proposition we study the limiting behavior of $g(\rho)$ when $\epsilon \to 0$ and when the load of the system $\rho \to 1$.

Proposition 3.3 *The gain of the TLPS system with the value of the threshold as in (3.14) according to the standard PS system has the following properties:*

$$\lim_{\epsilon\to 0} g(\rho) = \frac{\overline{T}^{PS} - \lim_{\epsilon\to 0}\overline{T}(\widetilde{\theta}_{opt})}{\overline{T}^{PS}} = \frac{\rho(m\mu_1 - 1)}{m\mu_1(m\mu_1 - \rho)},$$

$$\lim_{\rho\to 1}\lim_{\epsilon\to 0} g(\rho) = \frac{1}{m\mu_1}.$$

The limit $\lim_{\epsilon\to 0} g(\rho)$ is an increasing function of ρ.

Proof. Follows from the previous derivations. ∎

One can see that the limit $\lim_{\epsilon\to 0} g(\rho)$ can be made arbitrarily close to one by choosing μ_1 sufficiently close to $1/m$ and the load sufficiently close to one. This in turn implies that the ratio $\lim_{\epsilon\to 0}\overline{T}(\widetilde{\theta}_{opt})/\overline{T}^{PS}$ can be made as close to zero as one wants.

This is a striking result as it shows that the performance of the TLPS system can be arbitrarily better than the performance of the PS system for some selection of the parameters. However, in the next session with the numerical results we show that this set of parameters is very small and for realistic parameters the gain is in the order of 50%.

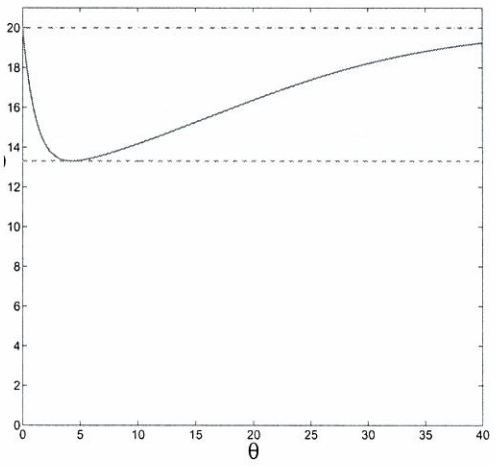

Figure 3.1: $\overline{T}(\theta)$ - solid line, $\overline{T}^{PS}$ - dash dot line, $\overline{T}(\widetilde{\theta}_{opt})$ - dash line.

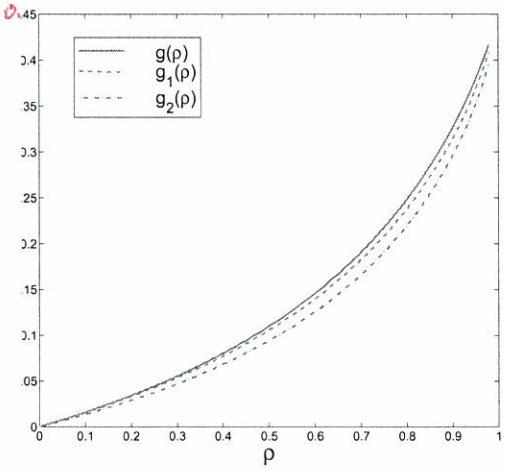

Figure 3.2: $g(\rho)$ - solid line, $g_1(\rho)$ - dash line, $g_2(\rho)$ - dash dot line.

3.4.4 Numerical results

For plots in Figures 3.1-3.2 we use the following parameters: $\rho = 0.909$, $m = 1.818$, $\mu_1 = 1$, $\mu_2 = 0.1$, so $\lambda = 0.5$ and $\epsilon = \mu_2/\mu_1 = 0.1$. Then, $p_1 = 0.909$ and $p_2 = 0.0909$.

In Figure 3.1 we plot $\overline{T}(\theta)$, $\overline{T}^{PS}$ and $\overline{T}(\widetilde{\theta}_{opt})$. We note that the expected sojourn time in the standard PS system $\overline{T}^{PS}$ is equal to $\overline{T}(0)$. We observe that $\overline{T}(\widetilde{\theta}_{opt})$ corresponds well to the optimum even though $\epsilon = 1/10$ is not too small.

Let us now study the gain that we obtain using TLPS, by setting $\theta = \widetilde{\theta}_{opt}$, in comparison with the standard PS. To this end, we plot the ratio $g(\rho) = \frac{\overline{T}^{PS} - \overline{T}(\widetilde{\theta}_{opt})}{\overline{T}^{PS}}$ in Figure 3.2. The gain in the system performance with TLPS in comparison with PS strongly depends on ρ, the load of the system. One can see that the gain of the TLPS system with respect to the standard PS system goes up to 45% when the load of the system increases.

To study the sensitivity of the TLPS system with respect to θ, we plot in Figure 3.2 the ratios $g_1(\rho) = \frac{\overline{T}^{PS} - \overline{T}(\frac{3}{2}\widetilde{\theta}_{opt})}{\overline{T}^{PS}}$ and $g_2(\rho) = \frac{\overline{T}^{PS} - \overline{T}(\frac{1}{2}\widetilde{\theta}_{opt})}{\overline{T}^{PS}}$. Thus, even with the 50% error of the $\widetilde{\theta}_{opt}$ value, the system performance is close to optimal.

One can see that it is beneficial to use TLPS instead of PS in the case of heavy and moderately heavy loads. We also observe that the TLPS system is not too sensitive to the choice of the threshold near its optimal value, when the job size distribution is hyper-exponential with two phases. Nevertheless, it is better to choose larger rather than smaller values of the threshold.

3.4.5 Simulation results

Using NS-2 simulator we implemented the algorithm based on the TLPS scheduling scheme and provide experimental results for the case of two phase hyper-exponential job size distribution. The algorithm is implemented in the router queue. In the router we keep the trace of the attained service of all flows in the system. The trace is kept during some time after which the router does not receive more packets from this flow. Every time to dequeue the packet from the bottleneck router, the router checks, if the first packet in the queue belongs to the flow which already received θ amount of service. If the flow did not receive θ amount of service, then the packet is served, else, the next packet is considered. If in the queue there are no packets which belong to the flows which did not yet received θ amount of service, the first packet in the queue is served.

In [AABN04] authors provide RuN2C, the scheduling algorithm based on the TLPS scheme. RuN2C takes the decision of the packet service according to the packet sequence number. In the current work we do not use the packets sequence numbers to take the scheduling decision, but we keep the track of the attained service for every flow in the system.

The simulation topology is the following. The files are generated by the FTP sources which are connected to the TCP senders. All TCP senders send files to the TCP destination nodes using the same bottleneck link. Every FTP source belongs to one of two sending classes in the system. Each class i, $i = 1, 2$ sends files with Poisson process with rate λ_i and has the exponential file size distribution with mean m_i. We consider that all connection have the same propagation delays. The bottleneck capacity is μ. We apply the TLPS scheduling algorithm to schedule the packets in the queue of the bottleneck link.

The proposed scheme is equivalent to the case of hyper-exponential job size distribution with two phases, where $p_1 = \lambda_1/\lambda$, $p_2 = \lambda_2/\lambda$, $1/\mu_1 = m_1/\mu$, $1/\mu_2 = m_2/\mu$, $\rho = \lambda(p_1/\mu_1 + p_2/\mu_2)$. Then we can use the approximated value of the optimal threshold given by (3.14)

$$\tilde{\theta}_{opt} = \frac{1}{\mu_1 - \mu_2} \ln\left(\frac{(\mu_1 - \lambda)}{\mu_2(1-\rho)}\right).$$

After we found the approximated value of the optimal threshold for the analytical model, we have to multiply $\tilde{\theta}_{opt}$ by the bottleneck capacity to get the real threshold value we use in the simulations, $\tilde{\theta}_{opt}^{simul} = \tilde{\theta}_{opt}\mu$.

For the simulations we select the following system parameters. The bottleneck capacity μ=100 Mbit/s. All the connections have a Maximum Segment Size (MSS) of 540 B. The propagation delay of every link equals to 2 ms. The duration of every simulation is 2000 s. Other system parameters and the approximated value of the optimal threshold are given in Table 3.1. In the current simulation model the short flows take $\rho_1 = 0.25$ and the long flows $\rho_2 = 0.61$ of the total bottleneck capacity. The total load in the system is $\rho = 0.86$.

m_1	m_2	λ_1	λ_2	ρ_1	ρ_2	ρ	$\tilde{\theta}_{opt}^{simul}$
1157 MSS	11574 MSS	5.0	1.22	0.25	0.61	0.86	7638 MSS

Table 3.1: Simulation parameters

We compare the mean waiting time in the system under TLPS, LAS and DropTail policies. For the TLPS policy we provide the simulation results for different values of θ, which is varied from 1157 MSS to 80000 MSS. The results are presented in Figure 3.3. As one can see the found approximated value of the optimal threshold $\tilde{\theta}_{opt}^{simul}$ =7.6x10^3 MSS minimizes the mean waiting time in the TLPS system between other threshold values. The mean waiting time in the TLPS system is very close to the optimal value of the mean waiting time achieved with the LAS policy. The maximal achieved relative gain with the TLPS policy when $\theta = \tilde{\theta}_{opt}^{simul}$ in comparison with the DropTail policy equals to 35.7%, while the relative gain with the optimal LAS policy in comparison with the DropTail policy is 36.7%.

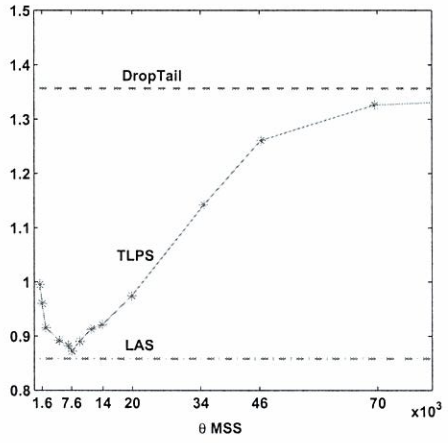

Figure 3.3: Mean waiting time in the system (s): TLPS - solid line with stars, DropTail - dash line, LAS - dash dot line.

3.5 Hyper-exponential job size distribution with more than two phases

3.5.1 Notation and motivation

In the second part of the present work we analyze the TLPS discipline with the hyper-exponential job size distribution with more than two phases. Using hyper-exponential distribution with more than two phases we obtain a more realistic representation of the file size

distribution in the Internet. In particular it was shown in [BM06, KSH03, FW98] that the hyper-exponential distribution with a significant number of phases models well the file size distribution in the Internet. Thus, we will use

$$F(x) = 1 - \sum_{i=1}^{N} p_i\, e^{-\mu_i x}, \quad \sum_{i=1}^{N} p_i = 1, \quad \mu_i > 0,\ p_i \geq 0, \quad i = 1, \ldots, N, \quad 1 < N \leq \infty.$$

It appears that in the case of many phases finding an explicit expression for the optimal threshold value is quite a challenging problem. In order to deal with a general hyper-exponential distribution we proceed with the derivation of a tight upper bound on the expected sojourn time function. The upper bound has a simple expression in terms of the system parameters and can lend itself to efficient numerical optimization.

In the following we write simply $\sum_i$ instead of $\sum_{i=1}^{N}$.

The mean job size m, the second moment d, the parameters $\overline{F_\theta^i}$, $\overline{X_\theta^1}$, $\overline{X_\theta^2}$ and ρ_θ are defined as in Section 3.3.1 and Section 3.3.2 by formulas (3.2), (3.3), (3.4), (3.5) for any $1 \leq N \leq \infty$. The formulas presented in Section 3.3.2 can still be used to calculate $b(\theta)$, $\overline{B}(x)$, $\overline{W}(\theta)$, $\gamma(\theta)$, $\delta_\rho(\theta)$, $\overline{T}^{TLPS}(x)$, $\overline{T}(\theta)$. We also need the following operator notations:

$$\Phi_1(\beta(x)) = \gamma(\theta) \int_0^\infty \beta(y)\overline{F}(x+y+\theta)dy + \gamma(\theta) \int_0^x \beta(y)\overline{F}(x-y+\theta)dy,$$
$$\Phi_2(\beta(x)) = \int_0^\infty \beta(y)\overline{F}(y+\theta)dy,$$

for any function $\beta(x)$. In particular, for some given constant c, we have

$$\Phi_1(c) = c\,\gamma(\theta)(m - \overline{X_\theta^1}) = c\,q, \tag{3.15}$$
$$\Phi_2(c) = c\,(m - \overline{X_\theta^1}), \tag{3.16}$$

where

$$q = \gamma(\theta)(m - \overline{X_\theta^1}) = \frac{\lambda(m - \overline{X_\theta^1})}{1 - \rho_\theta} = \frac{\rho - \rho_\theta}{1 - \rho_\theta} < 1. \tag{3.17}$$

The integral equation (3.6) can now be rewritten in the form

$$\alpha'(x) = \Phi_1(\alpha'(y)) + \frac{b(\theta)}{\overline{F}(\theta)}\overline{F}(x+\theta) + 1 \tag{3.18}$$

and equation (3.8) for $\overline{T}^{BPS}(\theta)$ takes form

$$\overline{T}^{BPS}(\theta) = \Phi_2(\alpha'(x)). \tag{3.19}$$

3.5.2 Linear system based solution

Using the Laplace transform based method described in [Ban03] we prove the following proposition.

Proposition 3.4 *The following formula holds:*

$$\overline{T}^{BPS}(\theta) = \sum_i \overline{F_\theta^i} L_i, \tag{3.20}$$

with

$$L_i = L_i^* + \frac{1}{\delta_\rho(\theta)\mu_i}, \quad i = 1, \ldots, N,$$

where the L_i^*, $i = 1, \ldots, N$ are the solution of the linear system

$$L_p^* \left(1 - \gamma(\theta) \sum_i \frac{\overline{F_\theta^i}}{\mu_p + \mu_i}\right) = \gamma(\theta) \sum_i \frac{\overline{F_\theta^i} L_i^*}{\mu_p + \mu_i} + \frac{b(\theta)}{\overline{F}(\theta)} \sum_i \frac{\overline{F_\theta^i}}{\mu_p + \mu_i}, \quad p = 1, \ldots, N. \tag{3.21}$$

Proof. To find $\overline{T}^{BPS}(\theta)$ we need to solve the integral equation (3.6). Let us recall that $\gamma(\theta) = \lambda/(1 - \rho_\theta)$, then we can rewrite (3.6) in the following way

$$\alpha'(x) = \gamma(\theta) \int_0^\infty \alpha'(y)\overline{F}(x + y + \theta)dy + \gamma(\theta) \int_0^x \alpha'(y)\overline{F}(x - y + \theta)dy + b(\theta)\overline{B}(x) + 1,$$

$$\alpha'(x) = \gamma(\theta) \sum_i \overline{F_\theta^i} e^{-\mu_i x} \int_0^\infty \alpha'(y)e^{-\mu_i y}dy + \gamma(\theta) \int_0^x \alpha'(y)\overline{F}(x - y + \theta)dy + b(\theta)\overline{B}(x) + 1.$$

We note that in the latter equation $\int_0^\infty \alpha'(y)e^{-\mu_i y}dy$, $i = 1, \ldots, N$ are the Laplace transforms of $\alpha'(y)$ evaluated at μ_i, $i = 1, \ldots, N$. Denote by $L_{\alpha'}(s) = \int_0^\infty \alpha'(x)e^{-sx}dx$ the Laplace transform of $\alpha'(x)$ and let $L_i = L_{\alpha'}(\mu_i)$, $i = 1, \ldots, N$. Then, we have

$$\alpha'(x) = \gamma(\theta) \sum_i \overline{F_\theta^i} L_i e^{-\mu_i x} + \gamma(\theta) \int_0^x \alpha'(y)\overline{F}(x - y + \theta)dy + b(\theta)\overline{B}(x) + 1.$$

Now taking the Laplace transform of the above equation and using the convolution property, we get

$$L_{\alpha'}(s) = \gamma(\theta) \sum_i \frac{\overline{F_\theta^i} L_i}{s + \mu_i} + \gamma(\theta) \sum_i \frac{\overline{F_\theta^i} L_{\alpha'}(s)}{s + \mu_i} + \frac{b(\theta)}{\overline{F}(\theta)} \sum_i \frac{\overline{F_\theta^i}}{s + \mu_i} + \frac{1}{s}$$

$$\Longrightarrow \quad L_{\alpha'}(s) \left(1 - \gamma(\theta) \sum_i \frac{\overline{F_\theta^i}}{s + \mu_i}\right) = \gamma(\theta) \sum_i \frac{\overline{F_\theta^i} L_i}{s + \mu_i} + \frac{b(\theta)}{\overline{F}(\theta)} \sum_i \frac{\overline{F_\theta^i}}{s + \mu_i} + \frac{1}{s}.$$

Then, we substitute into the above equation $s = \mu_i$, $i = 1, \ldots, N$ and get L_i, $i = 1, \ldots, N$ as a solution of the linear system

$$L_p \left(1 - \gamma(\theta) \sum_i \frac{\overline{F_\theta^i}}{\mu_p + \mu_i}\right) = \gamma(\theta) \sum_i \frac{\overline{F_\theta^i} L_i}{\mu_p + \mu_i} + \frac{b(\theta)}{\overline{F}(\theta)} \sum_i \frac{\overline{F_\theta^i}}{\mu_p + \mu_i} + \frac{1}{\mu_p}, \quad p = 1, \ldots, N.$$

If now we set $L_p = L_p^* + \frac{1}{\delta_\rho(\theta)\mu_p}$, $p = 1, \ldots, N$, then L_p^* are the solutions of the linear system (3.21). Next we need to calculate $\overline{T}^{BPS}(\theta)$.

$$\overline{T}^{BPS}(\theta) = \int_0^\infty \alpha'(x)\overline{F}(x+\theta)dx = \int_0^\infty \alpha'(x) \sum_i \overline{F_\theta^i} e^{-\mu_i x} dx = \sum_i \overline{F_\theta^i} L_i.$$

Finally, we have (3.20). ∎

Unfortunately, the system (3.21) does not seem to have a tractable finite form analytic solution. Therefore, in the ensuing subsections we propose an alternative solution based on an operator series and construct a tight upper bound.

3.5.3 Operator series form for the expected sojourn time

Since the operator Φ_1 is a contraction [AABN04, AAB05], we can iterate equation (3.18) starting from some initial point α'_0. The initial point can be simply a constant. As shown in [AABN04, AAB05] the iterations converge to the unique solution of (3.18). Specifically, we make iterations in the following way:

$$\alpha'_{n+1}(x) = \Phi_1(\alpha'_n(x)) + \frac{b(\theta)}{\overline{F}(\theta)}\overline{F}(x+\theta) + 1, \quad n = 0, 1, 2, \ldots. \tag{3.22}$$

At every iteration step we construct the following approximation of $\overline{T}^{BPS}(\theta)$ according to (3.19):

$$\overline{T}^{BPS}_{n+1}(\theta) = \Phi_2(\alpha'_{n+1}(x)). \tag{3.23}$$

Using (3.22) and (3.23), we construct the operator series expression for the expected sojourn time in the TLPS system.

Theorem 3.3 *The expected sojourn time $\overline{T}(\theta)$ in the TLPS system with the hyper-exponential job size distribution is given by*

$$\overline{T}(\theta) = \frac{\overline{X_\theta^1} + \overline{W}(\theta)\overline{F}(\theta)}{1-\rho_\theta} + \frac{m - \overline{X_\theta^1}}{1-\rho} + \frac{b(\theta)}{\overline{F}(\theta)(1-\rho_\theta)} \left(\sum_{i=0}^{\infty} \Phi_2\left(\Phi_1^i(\overline{F}(x+\theta))\right) \right). \tag{3.24}$$

Proof. From (3.22) we have

$$\alpha'_n = q^n \alpha'_0 + \sum_{i=1}^{n-1} q^i + \frac{b(\theta)}{\overline{F}(\theta)} \sum_{i=1}^{n-1} \Phi_1^i(\overline{F}(x+\theta)) + \frac{b(\theta)}{\overline{F}(\theta)}\overline{F}(x+\theta) + 1,$$

and then from (3.23) and (3.15) it follows that

$$\overline{T}^{BPS}_n(\theta) = (m - \overline{X_\theta^1}) \left(q^n \alpha'_0 + \sum_{i=0}^{n-1} q^i \right) + \frac{b(\theta)}{\overline{F}(\theta)} \left(\Phi_2 \left(\sum_{i=0}^{n-1} \Phi_1^i(\overline{F}(x+\theta)) \right) \right).$$

Using the facts (see (3.17)):

$$1.\ q < \rho < 1 \Longrightarrow q^n \to 0 \text{ as } n \to \infty,$$
$$2.\ \sum_{i=0}^{\infty} q^i = \frac{1}{1-q} = \frac{1-\rho_\theta}{1-\rho},$$

we conclude that

$$\overline{T}^{BPS}(\theta) = \lim_{n\to\infty} \overline{T}_n^{BPS}(\theta) = (m - \overline{X_\theta^1}) \left(\frac{1-\rho_\theta}{1-\rho}\right) + \frac{b(\theta)}{\overline{F}(\theta)} \left(\sum_{i=0}^{\infty} \Phi_2\left(\Phi_1^i(\overline{F}(x+\theta))\right)\right).$$

Finally, using (3.7) we obtain (3.24). ∎

The resulting formula (3.24) is still difficult to analyze. Therefore, in the next subsection using (3.24) we find an approximation, which is also an upper bound, of the expected sojourn time function in a more explicit form.

3.5.4 Upper bound for the expected sojourn time

Let us start with auxiliary results.

Lemma 3.2 *For any function $\beta(x) \geq 0$ with $\beta_j = \int_0^\infty \beta(x) e^{-\mu_i x} dx$,*

$$\text{if} \quad \frac{d(\beta_j \mu_j)}{d\mu_j} \geq 0, \quad j = 1, \ldots, N \quad \text{it follows that} \quad \Phi_2\left(\Phi_1(\beta(x))\right) \leq q\Phi_2\left(\beta(x)\right).$$

Proof. See Appendix. ∎

Lemma 3.3 *For the TLPS system with the hyper-exponential job size distribution the following statement holds:*

$$\Phi_2\left(\Phi_1(\alpha'(x))\right) \leq q\Phi_2\left(\alpha'(x)\right). \tag{3.25}$$

Proof. We define $\alpha'_j = \int_0^\infty \alpha'(x) e^{-\mu_j x} dx$, $j = 1, \ldots, N$. As was shown in [Osi08a], $\alpha'(x)$ has the following structure:

$$\alpha'(x) = a_0 + \sum_k a_k e^{-b_k x}, \quad a_0 \geq 0,\ a_k \geq 0,\ b_k > 0, \quad k = 1, \ldots, N.$$

Then, we have that $\alpha'(x) \geq 0$ and

$$\alpha'_j = \frac{a_0}{\mu_j} + \sum_k \frac{a_k}{b_k + \mu_j}, \quad j = 1, \ldots, N,$$
$$\Longrightarrow \quad \frac{d(\alpha'_j \mu_j)}{d\mu_j} = \sum_k \frac{a_k}{b_k + \mu_j} - \sum_k \frac{a_k \mu_j}{(b_k + \mu_j)^2} = \sum_k \frac{a_k b_k}{(b_k + \mu_j)^2} \geq 0, \quad j = 1, \ldots, N,$$

as $a_k \geq 0$, $b_k > 0$, $k = 1, \ldots, N$. So, then, according to Lemma 3.2 we have (3.25). ∎

Let us state the following Theorem:

Theorem 3.4 *An upper bound for the expected sojourn time function $\overline{T}(\theta)$ in the TLPS system with the hyper-exponential job size distribution function with many phases is given by $\overline{\Upsilon}(\theta)$:*

$$\overline{T}(\theta) \le \overline{\Upsilon}(\theta) = \frac{\overline{X_\theta^1} + \overline{W}(\theta)\overline{F}(\theta)}{1-\rho_\theta} + \frac{m - \overline{X_\theta^1}}{1-\rho} + \frac{b(\theta)}{\overline{F}(\theta)(1-\rho)} \sum_{i,j} \frac{\overline{F_\theta^i}\,\overline{F_\theta^j}}{\mu_i + \mu_j}. \tag{3.26}$$

Proof. According to the recursion (3.22), we consider $\widetilde{\alpha}'(x)$ as a candidate for the approximation of $\alpha'(x)$. Namely, $\widetilde{\alpha}'(x)$ satisfies the following equation:

$$\widetilde{\alpha}'(x) = \widetilde{\alpha}'(x)\Phi_1(1) + \frac{b(\theta)}{\overline{F}(\theta)}\overline{F}(x+\theta) + 1.$$

Then, using (3.15), we can find the analytic expression for $\widetilde{\alpha}'(x)$:

$$\begin{aligned} & \widetilde{\alpha}'(x) = q\widetilde{\alpha}'(x) + \frac{b(\theta)}{\overline{F}(\theta)}\overline{F}(x+\theta) + 1, \\ \Longrightarrow \quad & \widetilde{\alpha}'(x) = \frac{1}{1-q}\left(\frac{b(\theta)}{\overline{F}(\theta)}\overline{F}(x+\theta) + 1\right). \end{aligned}$$

We take $\overline{\Upsilon}^{BPS}(\theta) = \Phi_2(\widetilde{\alpha}'(x))$ as an approximation for $\overline{T}^{BPS}(\theta) = \Phi_2(\alpha'(x))$. Then

$$\overline{\Upsilon}^{BPS}(\theta) = \Phi_2(\widetilde{\alpha}'(x)) = \frac{(m - \overline{X_\theta^1})}{1-q} + \frac{b(\theta)}{\overline{F}(\theta)}\Phi_2(\overline{F}(x+\theta)) = \frac{(m - \overline{X_\theta^1})}{1-q} + \frac{b(\theta)}{\overline{F}(\theta)} \sum_{i,j} \frac{\overline{F_\theta^i}\,\overline{F_\theta^j}}{\mu_i + \mu_j}.$$

Let us prove that

$$\overline{T}^{BPS}(\theta) \le \overline{\Upsilon}^{BPS}(\theta),$$

or equivalently

$$\overline{T}^{BPS}(\theta) - \overline{\Upsilon}^{BPS}(\theta) = \Phi_2(\alpha'(x)) - \Phi_2(\widetilde{\alpha}'(x)) \le 0.$$

Let us look at

$$\begin{aligned} & \Phi_2(\alpha'(x)) - \Phi_2(\widetilde{\alpha}'(x)) = \\ & = \Phi_2(\Phi_1(\alpha'(x))) + \Phi_2\left(\frac{b(\theta)}{\overline{F}(\theta)}\overline{F}(x+\theta) + 1\right) - \left(q\Phi_2(\widetilde{\alpha}'(x)) + \Phi_2\left(\frac{b(\theta)}{\overline{F}(\theta)}\overline{F}(x+\theta) + 1\right)\right) \\ & = \Phi_2(\Phi_1(\alpha'(x))) - q\Phi_2(\alpha'(x)) + q\left(\Phi_2(\alpha'(x)) - \Phi_2(\widetilde{\alpha}'(x))\right) \\ & \Longrightarrow \\ & \Phi_2(\alpha'(x)) - \Phi_2(\widetilde{\alpha}'(x)) = \frac{1}{1-q}\left(\Phi_2(\Phi_1(\alpha'(x))) - q\Phi_2(\alpha'(x))\right). \end{aligned}$$

Now from Lemma 3.3 and formula (3.7) we conclude that (3.26) is true. ∎

In this subsection we found the analytic expression of the upper bound of the expected sojourn time in the case when the job size distribution is a hyper-exponential function with many phases. In the experimental results of the following subsection we show that the obtained upper bound is also a close approximation. The analytic expression of the upper bound which we obtained is more clear and easier to analyze than the expression (3.26) for the expected sojourn time. It can be used in efficient numerical optimization of the TLPS performance.

3.5.5 Numerical results

We calculate $\overline{T}(\theta)$ and $\overline{\Upsilon}(\theta)$ for different numbers of phases N of the job size distribution function. We take $N = 10, 100, 500, 1000$. To calculate $\overline{T}(\theta)$ we find the numerical solution of the system of linear equations (3.21) using the Gauss method. Then using the result of Proposition 3.4 we find $\overline{T}(\theta)$. For $\overline{\Upsilon}(\theta)$ we use equation (3.26).

As was mentioned in Subsection 3.3.1, by using the hyper-exponential distribution with many phases, one can approximate a heavy-tailed distribution. In our numerical experiments we fix ρ, m, and select p_i and μ_i in such a way that by increasing the number of phases we let the second moment d (see (3.2)) increase as well. Here we take

$$\rho = 0.909,\ m = 1.818,\ p_i = \frac{\nu}{i^{2.5}},\ \mu_i = \frac{\eta}{i^{1.2}},\quad i = 1, \dots, N.$$

In particular, we have

$$\sum_i p_i = 1, \quad \Longrightarrow \nu = \frac{1}{\sum_i i^{-2.5}},$$
$$\sum_i \frac{p_i}{\mu_i} = m, \quad \Longrightarrow \eta = \frac{\nu}{m} \sum_i i^{-1.3}.$$

In Figure 3.4 one can see the plots of the expected sojourn time and its upper bound as functions of θ when N equals to 10, 100 , 500 and 1000. In Figure 3.5 we plot the relative error of the upper bound

$$\Delta(\theta) = \frac{\overline{\Upsilon}(\theta) - \overline{T}(\theta)}{\overline{T}(\theta)},$$

when N equals to 10, 100 , 500 and 1000. As one can see, the upper bound (3.26) is very tight.

We find the maximum gain of the expected sojourn time of the TLPS system with respect to the standard PS system. As previously we denote the gain by $g(\theta) = \frac{\overline{T}^{PS} - \overline{T}(\theta)}{\overline{T}^{PS}}$, where $\overline{T}^{PS}$ is the expected sojourn time in the standard PS system. The data and results are summarized in Table 3.2.

We can make the following conclusions when increasing number of phases:

1. the maximum gain $\max_\theta g(\theta)$ in expected sojourn time in comparison with PS increases;

N	η	d	θ_{opt}	$\max_\theta g(\theta)$	$\max_\theta \Delta(\theta)$
10	0.95	7.20	5	32.98%	0.0640
100	1.26	32.28	12	45.75%	0.0807
500	1.40	113.31	21	49.26%	0.0766
1000	1.44	200.04	26	50.12%	0.0743

Table 3.2: Increasing the number of phases

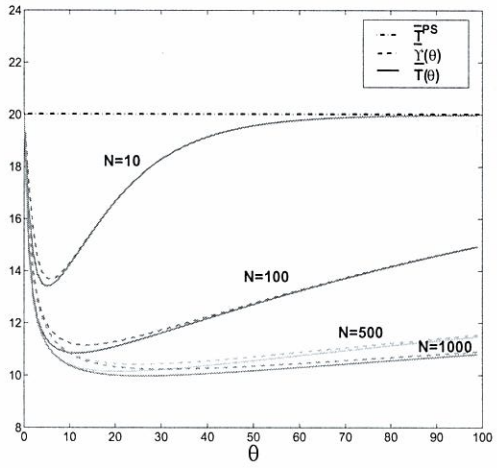

Figure 3.4: The expected sojourn time $\overline{T}(\theta)$ and its upper bound $\overline{\Upsilon}(\theta)$ for $N = 10$, 100, 500, 1000.

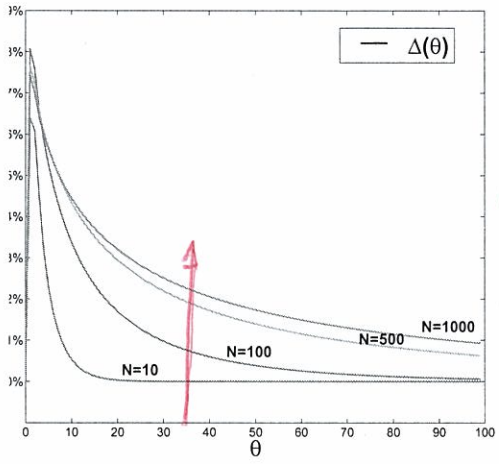

Figure 3.5: The relative error $\Delta(\theta)$ for $N = 10$, 100, 500, 1000.

2. the relative error $\Delta(\theta)$ of the upper bound according to the expected sojourn time decreases after the number of phases becomes sufficiently large;

3. the sensitivity of the system performance with respect to the selection of the sub-optimal threshold value decreases.

Thus the TLPS system produces better and more robust performance as the variance of the job size distribution increases.

3.6 Conclusion

We analyze the TLPS scheduling mechanism with the hyper-exponential job size distribution function.

In Section 3.4 we analyze the system when the job size distribution function has two phases and find the analytic expressions of the expected conditional sojourn time and the expected sojourn time of the TLPS system.

Connections in the Internet belong to two distinct classes: short HTTP and P2P signaling connections and long downloads such as PDF, MP3, and so on. Thus, according to this observation, we consider a special selection of the parameters of the job size distribution function with two phases and find the approximation of the optimal threshold, when the variance of the job size distribution goes to infinity. We show that the approximated value of the threshold tends to the optimal threshold, when the second moment of the distribution function goes to infinity.

We found that the ratio between the expected sojourn time of the TLPS system and the expected sojourn time of the standard PS system can be arbitrary small for very high loads. For realistic loads this ratio can reach 1/2. Also we show the system is not too sensitive to the selection of the optimal value of the threshold.

With NS-2 simulator we implement TLPS scheduling scheme in the router of the bottleneck link. We show that the analytically found approximation of the optimal threshold value minimizes the mean waiting time in the TLPS system between other threshold values. With the simulation results we show that TLPS with the found approximated value of the optimal threshold can give up to 35% gain in comparison with the DropTail policy and almost the same gain as the optimal LAS policy.

In Section 3.5 we study the TLPS model when the job size distribution is a hyper-exponential function with many phases. We provide an expression of the expected conditional sojourn time as a solution of a system of linear equations. Also we apply the iteration method to find the expression of the expected conditional sojourn time in the form of operator series and using the obtained expression we provide an upper bound for the expected sojourn time function. With the experimental results we show that the upper bound is very tight and can be used as an approximation of the expected sojourn time function. We show numerically that the relative error between the upper bound and the expected sojourn time function decreases when the variation of the job size distribution function increases. The obtained upper bound can be used to identify an approximation of the optimal threshold value for the TLPS system when the job size distribution is heavy-tailed.

We study the properties of the expected sojourn time function, when the parameters of the job size distribution function are selected in such a way that it approximates a heavy-tailed distribution as the number of phases of the job size distribution increases. As the number of phases increases the gain of the TLPS system compared with the standard PS system increases and the sensitivity of the system with respect to the selection of the optimal threshold decreases.

3.7 Appendix: Proof of Lemma 3.2

Let us consider any function $\beta(x) \geq 0$ and define $\beta_j = \int_0^\infty \beta(x) e^{-\mu_j x} dx, \quad j = 1, \ldots, N$. Let us show for $\beta(x) \geq 0$ that if

$$\frac{d(\beta_j \mu_j)}{d\mu_j} \geq 0, \quad j = 1, \ldots, N, \qquad \text{then it follows that} \qquad \Phi_2\left(\Phi_1(\beta(x))\right) \leq q\Phi_2\left(\beta(x)\right).$$

As

$$\int_0^\infty \int_0^x \beta(y)\overline{F}(x-y+\theta)\overline{F}(x+\theta)dydx = \int_0^\infty \int_0^\infty \beta(y)\overline{F}(x_1+\theta)\overline{F}(x_1+y+\theta)dx_1 dy$$

and

$$\begin{aligned}\Phi_2(\Phi_1(\beta(x))) = \gamma(\theta) &\int_0^\infty \int_0^\infty \beta(y)\overline{F}(x+y+\theta)\overline{F}(x+\theta)dydx \\ &+ \gamma(\theta)\int_0^\infty \int_0^x \beta(y)\overline{F}(x-y+\theta)\overline{F}(x+\theta)dydx,\end{aligned}$$

then

$$\begin{aligned}\Phi_2(\Phi_1(\beta(x))) &= 2\gamma(\theta)\int_0^\infty \int_0^\infty \beta(x)\overline{F}(x+\theta)\overline{F}(x+y+\theta)dydx = \\ &= 2\gamma(\theta)\int_0^\infty \beta(x)\sum_{i,j}\frac{\overline{F_\theta^i}\overline{F_\theta^j}}{\mu_i+\mu_j}e^{-\mu_j x}dx = 2\gamma(\theta)\sum_{i,j}\frac{\overline{F_\theta^i}\overline{F_\theta^j}}{\mu_i+\mu_j}\beta_j.\end{aligned}$$

Also for $\Phi_2\left(\beta(x)\right)$, taking into account that $q = \gamma(\theta)\sum_i \frac{\overline{F_\theta^i}}{\mu_i}$, we obtain

$$q\Phi_2\left(\beta(x)\right) \;=\; \gamma(\theta)\sum_i \frac{\overline{F_\theta^i}}{\mu_i}\sum_j \overline{F_\theta^j}\int_0^\infty \beta(x)e^{-\mu_j x}dx = \gamma(\theta)\sum_{i,j}\frac{\overline{F_\theta^i}\overline{F_\theta^j}}{\mu_i}\beta_j.$$

Thus, a sufficient condition for the inequality $\Phi_2\left(\Phi_1(\beta(x))\right) \leq q\Phi_2\left(\beta(x)\right)$ to be satisfied is that for every pair i, j:

$$\frac{2}{\mu_i+\mu_j}\beta_j + \frac{2}{\mu_j+\mu_i}\beta_i \leq \frac{1}{\mu_i}\beta_j + \frac{1}{\mu_j}\beta_i \quad \Longleftrightarrow \quad -(\beta_j\mu_j - \beta_i\mu_i)(\mu_j - \mu_i) \leq 0.$$

The inequality is indeed satisfied when $\beta_j\mu_j$ is an increasing function of μ_j. We conclude that $\Phi_2\left(\Phi_1(\beta(x))\right) \leq q\Phi_2\left(\beta(x)\right)$, which proves Lemma 3.2.

Chapter 4

Comparison of the Discriminatory Processor Sharing Policies

4.1 Summary

The DPS policy was introduced by Kleinrock. Under the DPS policy jobs are organized in classes, which share a single server. The capacity that each class obtains depends on the number of jobs currently presented in all classes and is controlled by the vector of weights. Varying DPS weights it is possible to give priority to different classes at the expense of others, control their instantaneous service rates and optimize different system characteristics as mean sojourn time and so on. So, the proper weight selection is an important task, which is not easy to solve because of the model's complexity.

We study the comparison of two DPS policies with different weight vectors. We show the monotonicity of the expected sojourn time of the system depending on the weight vector under certain condition on the system. The restrictions on the system are such that the result is true for systems for which the values of the job size distribution means are very different from each other. The restriction can be overcome by setting the same weights for the classes, which have similar means. The condition on means is a sufficient, but not a necessary condition. It becomes less strict when the system is less loaded.

The results of the current work can be found in [Osi08b].

4.2 Introduction

The Discriminatory Processor Sharing (DPS) policy was introduced by Kleinrock [Kle67]. Under the DPS policy jobs are organized in classes, which share a single server. The capacity that each class obtains depends on the number of jobs currently presented in all classes. All jobs present in the system are served simultaneously at rates controlled by the vector of weights $g_k > 0$, $k = 1, \ldots, M\}$, where M is the number of classes. If there are N_j jobs in class j, then each job of this class is served with the rate $g_j / \sum_{k=1}^{M} g_k N_k$. When all weights are equal, DPS system is equivalent to the standard PS policy.

The DPS policy model has recently received a lot of attention due to its wide range of application. For example, DPS could be applied to model flow level sharing of TCP flows with different flow characteristics such as different RTTs and packet loss probabilities. DPS also provides a natural approach to model the weighted round-robin discipline, which is used in operating systems for task scheduling. In the Internet one can imagine the situation that servers provide different service according to the payment rates. For more applications of DPS in communication networks see [AJK04], [BT01], [CvdBB+05], [GM02b], [HT05].

Varying DPS weights it is possible to give priority to different classes at the expense of others, control their instantaneous service rates and optimize different system characteristics as mean sojourn time and so on. So, the proper weight selection is an important task, which is not easy to solve because of the model's complexity.

The previously obtained results on DPS model are the following. Kleinrock in [Kle67] was first studying DPS. Then the paper of Fayolle et al. [FMI80] provided results for the DPS model. For the exponentially distributed required service times the authors obtained the expression of the expected sojourn time as a solution of a system of linear equations. The authors show that independently of the weights the slowdown for the expected conditional response time under the DPS policy tends to the constant slowdown of the PS policy as the service requirements increases to infinity.

Rege and Sengupta in [RS94] proved a decomposition theorem for the conditional sojourn time. For exponential service time distributions in [RS96] they obtained higher moments of the queue length distribution as the solutions of linear equations system and also provided a theorem for the heavy-traffic regime. Van Kessel et al. in [KNQB05], [KNQB04] study the performance of DPS in an asymptotic regime using time scaling. For general distributions of the required service times the approximation analysis was carried out by Guo and Matta in [GM02b]. Altman et al. [AJK04] study the behavior of the DPS policy in overload. Most of the results obtained for the DPS queue were collected together in the survey paper of Altman et al. [AAA06].

Avrachenkov et al. in [AABNQ05] proved that the mean unconditional response time of

each class is finite under the usual stability condition. They determine the asymptote of the conditional sojourn time for each class assuming finite service time distribution with finite variance.

The problem of weights selection in the DPS policy when the job size distributions are exponential was studied by Avrachenkov et al. in [AABNQ05] and by Kim and Kim in [KK06]. In [KK06] it was shown that the DPS policy reduces the expected sojourn time in comparison with PS policy when the weights increase in the opposite order with the means of job classes. Also in [KK06] the authors formulate a conjecture about the monotonicity of the expected sojourn time of the DPS policy. The idea of conjecture is that comparing two DPS policies, one which has a weight vector closer to the optimal policy, provided by $c\mu$-rule, see [Rig94], has smaller expected sojourn time. Using the method described in [KK06] in the present chapter we prove this conjecture with some restrictions on the system parameters. The restrictions on the system are such that the result is true for systems for which the values of the job size distribution means are very different from each other. The restriction can be overcome by setting the same weights for the classes, which have similar means. The condition on means is a sufficient, but not a necessary condition. It becomes less strict when the system is less loaded.

The chapter is organized as follows. In Section 4.3 we give general definitions of the DPS policy and formulate the problem of expected sojourn time minimization. In Section 4.4 we formulate the main Theorem and prove it. In Section 4.5 we give the numerical results. Some technical proofs can be found in the Appendix.

4.3 Previous results and problem formulation

We consider the DPS model. All jobs are organized in M classes and share a single server. Jobs of class $k = 1, \dots, M$ arrive with a Poisson process with rate λ_k and have required service-time distribution $F_k(x) = 1 - e^{-\mu_k x}$ with mean $1/\mu_k$. The load of the system is $\rho = \sum_{k=1}^{M} \rho_k$ and $\rho_k = \lambda_k/\mu_k$, $k = 1, \dots, M$. We consider that the system is stable, $\rho < 1$. Let us denote $\lambda = \sum_{k=1}^{M} \lambda_k$.

The state of the system is controlled by a vector of weights $g = (g_1, \dots, g_M)$, which denotes the priority for the job classes. If in the class k there are currently N_k jobs, then each job of class k is served with the rate equal to $g_j / \sum_{k=1}^{M} g_k N_k$, which depends on the current system state, or on the number of jobs in each class.

Let $\overline{T}^{DPS}$ be the expected sojourn time of the DPS system. We have

$$\overline{T}^{DPS} = \sum_{k=1}^{M} \frac{\lambda_k}{\lambda} \overline{T}_k,$$

where $\overline{T}_k$ are expected sojourn times for class k. The expressions for the expected sojourn times

$\overline{T}_k$, $k = 1, \ldots, M$ can be found as a solution of the system of linear equations, see [FMI80],

$$\overline{T}_k \left(1 - \sum_{j=1}^{M} \frac{\lambda_j g_j}{\mu_j g_j + \mu_k g_k} \right) - \sum_{j=1}^{M} \frac{\lambda_j g_j \overline{T}_j}{\mu_j g_j + \mu_k g_k} = \frac{1}{\mu_k}, \tag{4.1}$$

$k = 1, \ldots, M$. Let us notice that for the standard Processor Sharing system $\overline{T}^{PS} = \frac{\rho/\lambda}{1-\rho}$.

One of the problems when studying DPS is to minimize the expected sojourn time $\overline{T}^{DPS}$ with some weight selection. Namely, find g^* such as

$$\overline{T}^{DPS}(g^*) = \min_g \overline{T}^{DPS}(g).$$

This is a general problem and to simplify it the following subcase is considered. To find a set G such that

$$\overline{T}^{DPS}(g^*) \leq \overline{T}^{PS}, \quad \forall g^* \in G. \tag{4.2}$$

For the case when job size distributions are exponential the solution of (4.2) is given by Kim and Kim in [KK06] and is as follows. If the means of the classes are such as $\mu_1 \geq \mu_2 \geq \ldots \geq \mu_M$, then G consists of all such vectors which satisfy

$$G = \{g |\ g_1 \geq g_2 \geq \ldots \geq g_M\}.$$

Using the approach of [KK06] we solve more general problem about the monotonicity of the expected sojourn time in the DPS system, which we formulate in the following section as Theorem 4.1.

4.4 Expected sojourn time monotonicity

Let us formulate and prove the following Theorem.

Theorem 4.1 *Let the job size distribution for every class be exponential with means* $1/\mu_i$, $i = 1, \ldots, M$ *and we enumerate them in the following way*

$$\mu_1 \geq \mu_2 \geq \ldots \geq \mu_M.$$

Let us consider two different weight policies for the DPS system, which we denote as α *and* β. *Let* $\alpha, \beta \in G$, *or*

$$\alpha_1 \geq \alpha_2 \geq \ldots \geq \alpha_M,$$
$$\beta_1 \geq \beta_2 \geq \ldots \geq \beta_M.$$

The expected sojourn time of the DPS policies with weight vectors α and β satisfies

$$\overline{T}^{DPS}(\alpha) \leq \overline{T}^{DPS}(\beta), \tag{4.3}$$

if the weights α and β are such that:

$$\frac{\alpha_{i+1}}{\alpha_i} \leq \frac{\beta_{i+1}}{\beta_i}, \quad i = 1, \ldots, M-1, \tag{4.4}$$

and the following restriction is satisfied:

$$\frac{\mu_{j+1}}{\mu_j} \leq 1 - \rho, \tag{4.5}$$

for every $j = 1, \ldots, M$.

Remark 4.1 *If for some classes j and $j+1$ condition (4.5) is not satisfied, then by choosing the weights of these classes to be equal, we can still use Theorem 4.1. Namely, for classes j and $j+1$ such as $\frac{\mu_{j+1}}{\mu_j} > 1 - \rho$, if we set $\alpha_{j+1} = \alpha_j$ and $\beta_{j+1} = \beta_j$, then still the statement (4.3) of Theorem 4.1 holds.*

Remark 4.2 *Theorem 4.1 shows that the expected sojourn time $\overline{T}^{DPS}(g)$ is monotonous according to the selection of weight vector g. The closer is the weight vector to the optimal policy, provided by $c\mu$-rule, the smaller is the expected sojourn time. This is shown by the condition (4.4), which shows that vector α is closer to the optimal $c\mu$-rule policy than vector β.*

Theorem 4.1 is proved with restriction (4.5). This restriction is a sufficient and not a necessary condition on system parameters. It shows that the means of the job classes have to be quite different from each other. This restriction can be overcome, giving the same weights to the job classes, which mean values are similar. Condition (4.5) becomes less strict as the system becomes less loaded.

To prove Theorem 4.1 let us first give some notations and prove additional Lemmas. Let us rewrite linear system (4.1) in the matrix form. Let $\overline{T}^{(g)} = [\overline{T}_1^{(g)}, \ldots, \overline{T}_M^{(g)}]^T$ be the vector of $\overline{T}_k^{(g)}$, $k = 1, \ldots, M$. Here by $[\,]^T$ we mean transpose sign, so $[\,]^T$ is a vector. By $[\,]^{(g)}$ we note that this element depends on the weight vector selection $g \in G$. Let us consider that later in the chapter vectors $g, \alpha, \beta \in G$, if the opposite is not noticed.

Let us give the following notations.

$$\sigma_{ij}^{(g)} = \frac{g_j}{\mu_i g_i + \mu_j g_j}.$$

Using the notation of $\sigma_{ij}^{(g)}$ let us define matrices $A^{(g)}$ and $D^{(g)}$ in the following way.

$$A_{ij}^{(g)} = \sigma_{ij}^{(g)}, \quad i, j = 1, \ldots, M, \tag{4.6}$$

$$D_{ij}^{(g)} = \begin{cases} \sum_{k=1}^{M} \sigma_{ik}^{(g)}, & i, j = 1, \ldots, M, \quad i = j, \\ 0, & i, j = 1, \ldots, M, \quad i \neq j. \end{cases} \tag{4.7}$$

Then linear system (4.1) becomes

$$(E - D^{(g)} - A^{(g)})\overline{T}^{(g)} = \left[\frac{1}{\mu_1} \dots \frac{1}{\mu_M}\right]^T. \tag{4.8}$$

Let us denote

$$B^{(g)} = A^{(g)} + B^{(g)}, \quad \forall g.$$

We need to find the expected sojourn time of the DPS system $\overline{T}^{DPS}(g)$. According to the definition of $\overline{T}^{DPS}(g)$ and equation (4.8) we have

$$\begin{aligned} \overline{T}^{DPS}(g) &= \lambda^{-1}[\lambda_1, \dots, \lambda_M]\overline{T}^{(g)} \\ &= \lambda^{-1}[\lambda_1, \dots, \lambda_M](E - B^{(g)})^{-1}\left[\frac{1}{\mu_1}, \dots, \frac{1}{\mu_M}\right]^T. \end{aligned}$$

Let us consider the case when $\lambda_i = 1$ for $i = 1, \dots, M$. This results can be extended for the case when λ_i are different. We prove it following the approach of [KK06] in Proposition 4.1 at the end of the current Section. Then the previous equation becomes

$$\overline{T}^{DPS}(g) = \lambda^{-1}\underline{1}'(E - B^{(g)})^{-1}[\rho_1, \dots, \rho_M]^T. \tag{4.9}$$

Let us show some properties of $\sigma_{ij}^{(g)}$. From the definition of $\sigma_{ij}^{(g)}$ it follows

$$\begin{aligned} &\sigma_{ij}^{(g)} g_i = \sigma_{ji}^{(g)} g_j, \\ &\frac{\sigma_{ij}^{(g)}}{\mu_i} + \frac{\sigma_{ji}^{(g)}}{\mu_j} = \frac{1}{\mu_i \mu_j}, \end{aligned} \tag{4.10}$$

$i, j = 1, \dots, M$. Also we prove Lemma 4.1.

Lemma 4.1 *If α and β satisfy (4.4), then*

$$\sigma_{ij}^{(\alpha)} \le \sigma_{ij}^{(\beta)}, \quad i \le j, \tag{4.11}$$

$$\sigma_{ij}^{(\alpha)} \ge \sigma_{ij}^{(\beta)}, \quad i \ge j. \tag{4.12}$$

Proof. If α and β satisfy (4.4), then for $i = 1, \dots, M-1$ $\frac{\alpha_j}{\alpha_i} \le \frac{\beta_j}{\beta_i}$, $i \le j$. From here $\alpha_j \mu_i \beta_i \le \beta_j \mu_i \alpha_i$, $i \le j$. Adding to both parts $\alpha_j \mu_j \beta_j$ and dividing both parts by $(\mu_i \beta_i + \mu_j \beta_j)$ we get (4.11). We prove (4.12) in a similar way. ∎

Lemma 4.2 *If α, β satisfy (4.4), then*

$$\overline{T}^{DPS}(\alpha) \le \overline{T}^{DPS}(\beta),$$

when the elements of vector $y = \underline{1}'(E - B^{(\alpha)})^{-1}$ are such that $y_1 \ge y_2 \ge \dots \ge y_M$.

Proof. Using expression (4.9) for $g = \alpha, \beta$ we get the following.

$$\begin{aligned}\overline{T}^{DPS}(\alpha) - \overline{T}^{DPS}(\beta) &= \lambda^{-1}\underline{1}'((E - B^{(\alpha)})^{-1} - (E - B^{(\beta)})^{-1})\,[\rho_1, \ldots, \rho_M]^T \\ &= \lambda^{-1}\underline{1}'((E - B^{(\alpha)})^{-1}(B^{(\alpha)} - B^{(\beta)})(E - B^{(\beta)})^{-1})\,[\rho_1, \ldots, \rho_M]^T.\end{aligned}$$

Let us denote M as a diagonal matrix $M = diag(\mu_1, \ldots, \mu_M)$ and

$$y = \underline{1}'(E - B^{(\alpha)})^{-1}M. \tag{4.13}$$

Then

$$\begin{aligned}\overline{T}^{DPS}(\alpha) - \overline{T}^{DPS}(\beta) &= \underline{1}'(E - B^{(\alpha)})^{-1}MM^{-1}(B^{(\alpha)} - B^{(\beta)})\overline{T}^{(\beta)} \\ &= yM^{-1}(B^{(\alpha)} - B^{(\beta)})\overline{T}^{(\beta)} \\ &= \sum_{i,j}\left(\frac{y_j}{\mu_j}\sigma_{ji}^{(\alpha)} + \frac{y_i}{\mu_i}\sigma_{ij}^{(\alpha)} - \left(\frac{y_j}{\mu_j}\sigma_{ji}^{(\beta)} + \frac{y_i}{\mu_i}\sigma_{ij}^{(\beta)}\right)\right)\overline{T}_j^{(\beta)} \\ &= \sum_{i,j}\left(y_j\left(\frac{\sigma_{ji}^{(\alpha)}}{\mu_j} - \frac{\sigma_{ji}^{(\beta)}}{\mu_j}\right) + \frac{y_i}{\mu_i}(\sigma_{ij}^{(\alpha)} - \sigma_{ij}^{(\beta)})\right)\overline{T}_j^{(\beta)}.\end{aligned}$$

As (4.10), $\frac{\sigma_{ji}^{(g)}}{\mu_j} = \frac{1}{\mu_i\mu_j} - \frac{\sigma_{ij}^{(g)}}{\mu_i}$, $g = \alpha, \beta$, then

$$\begin{aligned}\overline{T}^{DPS}(\alpha) - \overline{T}^{DPS}(\beta) &= \sum_{i,j}\left(-\frac{y_j}{\mu_i}\left(\sigma_{ij}^{(\alpha)} - \sigma_{ij}^{(\beta)}\right) + \frac{y_i}{\mu_i}(\sigma_{ij}^{(\alpha)} - \sigma_{ij}^{(\beta)})\right)\overline{T}_j^{(\beta)} \\ &= \sum_{i,j}\left(\left(\sigma_{ij}^{(\alpha)} - \sigma_{ij}^{(\beta)}\right)(y_i - y_j)\frac{1}{\mu_i}\right)\overline{T}_j^{(\beta)}.\end{aligned}$$

Using Lemma 4.1 we get that expression $\left(\sigma_{ij}^{(\alpha)} - \sigma_{ij}^{(\beta)}\right)(y_i - y_j) \leq 0$, $i, j = 1, \ldots, M$ when $y_1 \geq y_2 \geq \ldots \geq y_M$. This proves the statement of Lemma 4.2. ∎

Lemma 4.3 *Vector y given by (4.13) satisfies*

$$y_1 \geq y_2 \geq \ldots \geq y_M,$$

if the following is true:

$$\frac{\mu_{i+1}}{\mu_i} \leq 1 - \rho,$$

for every $i = 1, \ldots, M$.

Proof. The proof could be found in the appendix. ∎

Remark 4.3 *For the job classes such as* $\frac{\mu_{i+1}}{\mu_i} > 1-\rho$ *we prove that it is sufficient to give this classes the same weights,* $\alpha_{i+1} = \alpha_i$ *to to keep* $y_1 \geq y_2 \geq \ldots \geq y_M$,. *The proof could be found in the Appendix.*

Combining the results of Lemmas 4.1, 4.2 and 4.3 we prove the statement of Theorem 4.1. Remark 4.3 gives Remark 4.1 after Theorem 4.1. Now in Proposition 4.1 we prove the extension of Theorem 4.1 on the case when $\lambda_i \neq 1$.

Proposition 4.1 *The result of Theorem 4.1 is extended to the case when* $\lambda_i \neq 1$.

Proof. Let us first consider the case when all $\lambda_i = q$, $i = 1, \ldots, M$. It can be shown that for this case the proof of Theorem 4.1 is equivalent to the proof of the same Theorem but for the new system with $\lambda_i^* = 1$, $\mu_i^* = q\mu_i$, $i = 1, \ldots, M$. For this new system the results of Theorem 4.1 is evidently true and restriction (4.5) is not changed. Then, Theorem 4.1 is true for the initial system as well.

If λ_i are rational, then they could be written in $\lambda_i = \frac{p_i}{q}$, where p_i and q are positive integers. Then each class can be presented as p_i classes with equal means $1/\mu_i$ and intensity $1/q$. So, the DPS system can be considered as the DPS system with $p_1 + \ldots + p_K$ classes with the same arrival rates $1/q$. The result of Theorem 4.1 is extended on this case.

If λ_i, $i = 1, \ldots, M$ are positive and real we apply the previous case of rational λ_i and use continuity. ∎

In the following section we give numerical results on Theorem 4.1. We consider two cases, when condition (4.5) is satisfied and when it is not satisfied. We show that condition (4.5) is a sufficient and not a necessary condition on the system parameters.

4.5 Numerical results

Let us consider a DPS system with 3 classes. Let us consider the set of normalized weights vectors $g(x) = (g_1(x), g_2(x), g_3(x))$, $\sum_{i=1}^3 g_i(x) = 1$, $g_i(x) = x^{-i}/(\sum_{i=1}^3 x^{-i})$, $x > 1$. Every point $x > 1$ denotes a weight vector. Vectors $g(x), g(y)$ satisfy property (4.4) when $1 < y \leq x$, namely $g_{i+1}(x)/g_i(x) \leq g_{i+1}(y)/g_i(y)$, $i = 1, 2$, $1 < y \leq x$. On Figures 4.1, 4.2 we plot $\overline{T}^{DPS}(g(x))$ with weights vectors $g(x)$ as a function of x, the expected sojourn times $\overline{T}^{PS}$ for the PS policy and $\overline{T}^{opt}$ for the optimal $c\mu$-rule policy.

On Figure 4.1 we plot the expected sojourn time for the case when condition (4.5) is satisfied for three classes. The parameters are: $\lambda_i = 1$, $i = 1, 2, 3$, $\mu_1 = 160$, $\mu_2 = 14$, $\mu_3 = 1.2$, then $\rho = 0.911$. On Figure 4.2 we plot the expected sojourn time for the case when condition (4.5) is *not* satisfied for three classes. The parameters are: $\lambda_i = 1$, $i = 1, 2, 3$, $\mu_1 = 3.5$, $\mu_2 = 3.2$,

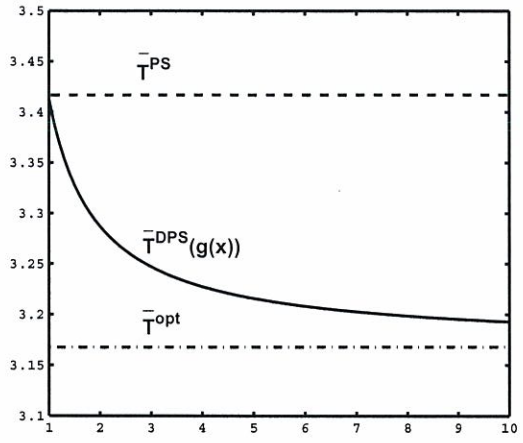

Figure 4.1: $\overline{T}^{DPS}(g(x))$, $\overline{T}^{PS}$, $\overline{T}^{opt}$ functions, condition satisfied.

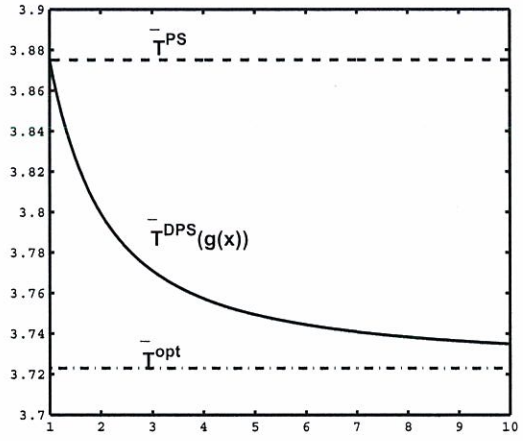

Figure 4.2: $\overline{T}^{DPS}(g(x))$, $\overline{T}^{PS}$, $\overline{T}^{opt}$ functions, condition not satisfied

$\mu_3 = 3.1$, then $\rho = 0.92$. One can see that $\overline{T}^{DPS}(g(x)) \leq \overline{T}^{DPS}(g(y))$, $1 < y \leq x$ even when the restriction (4.5) is not satisfied.

4.6 Conclusion

We study the DPS policy with exponential job size distribution. One of the main problems studying DPS is the expected sojourn time minimization according to the weights selection. In the present work we compare two DPS policies with different weights. We show that the expected sojourn time is smaller for the policy with the weigh vector closer to the optimal policy vector, provided by $c\mu$-rule. So, we prove the monotonicity of the expected sojourn time for the DPS policy according to the weight vector selection.

The result is proved with some restrictions on system parameters. The found restrictions on the system parameters are such that the result is true for systems such as the mean values of the job class size distributions are very different from each other. We found, that to prove the main result it is sufficient to give the same weights to the classes with similar means. The found restriction is a sufficient and not a necessary condition on a system parameters. When the load of the system decreases, the condition becomes less strict.

4.7 Appendix

In the following proof we do not use the dependency of the parameters on g to simplify the notations. We consider that vector $g \in G$, or $g_1 \geq g_2 \ldots \geq g_M$. To simplify the notations let us

use $\sum_k$ instead of $\sum_{k=1}^{M}$.

Lemma 4.3. Vector $y = \underline{1}'(E-B)^{-1}M$ satisfies $y_1 \geq y_2 \geq \ldots \geq y_M$, if $\frac{\mu_{i+1}}{\mu_i} \leq 1-\rho$, for every $i = 1, \ldots, M-1$.

Proof. Using the results of the following Lemmas 4.4, 4.5, 4.6, 4.7, 4.8 we prove the statement of Lemma 4.3 and give the proof for Remark 4.3. ∎

Let us give the following notations

$$\tilde{\mu} = \mu^T (E-D)^{-1}, \tag{4.14}$$

$$\tilde{A} = M^{-1}AM(E-D)^{-1}. \tag{4.15}$$

We define $f(x) = \sum_k \frac{x}{\mu_k(x+\mu_k g_k)}$ and notice that $1 - \sum_k \sigma_{jk} = 1 - \rho + f(\mu_j g_j)$. Then

$$\begin{aligned}
(E-D)^{-1}_{jj} &= \frac{1}{1-\sum_k \sigma_{jk}} = \frac{1}{1-\rho+f(\mu_j g_j)} > 0, \\
(E-D)^{-1}_{ij} &= 0, \quad i \neq j, \\
\tilde{A}_{ij} &= \left(\frac{\mu_j g_j}{\mu_i(\mu_i g_i + \mu_j g_j)}\right)(E-D)^{-1}_{jj} \\
&= \frac{\mu_j g_j}{\mu_i(\mu_i g_i + \mu_j g_j)(1-\rho+f(\mu_j g_j))} > 0,
\end{aligned}$$

$i, j = 1, \ldots, M$. Let us prove additional Lemma.

Lemma 4.4 *Matrix $\tilde{A} = M^{-1}AM(E-D)^{-1}$ is a positive contraction.*

Proof. Matrix $\tilde{A}$ is a positive operator as its elements $\tilde{A}_{ij}$ are positive. Let $\Omega = \{X | x_i \geq 0,\ i = 1, \ldots, M\}$. If $X \in \Omega$, then $\tilde{A}X \in \Omega$. Then to prove that matrix $\tilde{A}$ is a contraction it is enough to show that

$$\exists q, \quad 0 < q < 1, \quad ||\tilde{A}X|| \leq q||X||, \quad \forall\, X \in \Omega. \tag{4.16}$$

As $X \in \Omega$, then we can take $||X|| = \underline{1}'X = \sum_i x_i$. Then

$$\begin{aligned}
\underline{1}'\tilde{A}X &= \sum_j x_j \sum_i \tilde{A}_{ij} = \sum_j x_j \frac{\sum_i \frac{\mu_j g_j}{\mu_i(\mu_j g_j + \mu_i g_i)}}{(1-\rho+f(\mu_j g_j))} \\
&= \sum_j x_j \frac{f(\mu_j g_j)}{1-\rho+f(\mu_j g_j)} \\
&= \sum_j x_j \left(1 - \frac{1-\rho}{1-\rho+f(\mu_j g_j)}\right) \\
&= \sum_j x_j \left(1 - (1-\rho)\frac{\sum_j \frac{x_j}{1-\rho+f(\mu_j g_j)}}{\sum_j x_j}\right).
\end{aligned}$$

Let us denote $\Delta_0 = \frac{\sum_j \frac{x_j}{1-\rho+f(\mu_j g_j)}}{\sum_j x_j}$. Then $\underline{1}'\tilde{A}X = \sum_j x_j\,(1-(1-\rho)\Delta_0)$.

We need to find the value of q, which satisfies the following

$$\begin{aligned}
&\underline{1}'\tilde{A}X \le q\underline{1}'X,\\
&\sum_j x_j\,(1-(1-\rho)\Delta_0) \le q\sum_j x_j,\\
&1-(1-\rho)\Delta_0 \le q.
\end{aligned}$$

As $f(\mu_j g_j) > 0$, then $0 < 1-(1-\rho)\Delta_0 < 1$. We define $\delta = \frac{1}{1-\rho+\max_j f(\mu_j g_j)}$. Let us notice that $\max_j f(\mu_j g_j)$ always exists as the values of $\mu_j g_j$, $j = 1,\dots,M$ are finite. As $\delta < \Delta_0$, then if we select $q = 1-(1-\rho)\delta$, then the found q is $0 < q < 1$ and satisfies condition (4.16). This completes the proof. ∎

Lemma 4.5 *If*

$$y_1^{(0)} = [0,\dots,0], \tag{4.17}$$

$$y^{(n)} = \tilde{\mu} + y^{(n-1)}\tilde{A}, \quad n = 1,2,\dots, \tag{4.18}$$

then $y^{(n)} \to y$, *when* $n \to \infty$.

Proof. Let us recall that $y = \underline{1}(E-B)^{-1}M$ and $B = E-A-D$, then

$$\begin{aligned}
&yM^{-1}(E-D-A) = \underline{1},\\
&yM^{-1}(E-D) = yM^{-1}A + \underline{1},\\
&y = yM^{-1}A(E-D)^{-1}M + \underline{1}(E-D)^{-1}M.
\end{aligned}$$

As matrices D and M are diagonal, the $MD = DM$ and then

$$y = \mu^T(E-D)^{-1} + yM^{-1}AM(E-D)^{-1},$$

where $\mu = [\mu_1,\dots,\mu_M]$. According to notations (4.14) and (4.15) we have the following

$$y = \tilde{\mu} + y\tilde{A}.$$

Let us denote $y^{(n)} = [y_1^{(n)},\dots,y_1^{(n)}]$, $n = 0,1,2,\dots$ and let define $y_1^{(0)}$ and $y^{(n)}$ by (4.17) and (4.18). According to Lemma 4.4 operator $\tilde{A}$ is a positive reflexion and is a contraction. Also $\tilde{\mu}_i$ are positive. Then $y^{(n)} \to y$, when $n \to \infty$ and we prove the statement of Lemma 4.5. ∎

Lemma 4.6 *Let* $y^{(n)}$, $n = 0,1,2,\dots$ *are defined as in Lemma 4.5, then*

$$y_1^{(n)} \ge y_2^{(n)} \ge \dots \ge y_M^{(n)}, \quad n = 1,2,\dots \tag{4.19}$$

if $\frac{\mu_{i+1}}{\mu_i} \le 1-\rho$ *for every* $i = 1,\dots,M-1$.

Proof. We prove the statement (4.19) by induction. For $y^{(0)}$ the statement (4.19) is true. Let us assume that (4.19) is true for the $(n-1)$ step, $y_1^{(n-1)} \geq y_2^{(n-1)} \geq \ldots \geq y_M^{(n-1)}$. To prove the induction statement we have to show that $y_1^{(n)} \geq y_2^{(n)} \geq \ldots \geq y_M^{(n)}$, when $\frac{\mu_{i+1}}{\mu_i} \leq 1-\rho$ for every $i = 1, \ldots, M-1$. Let us consider $y_j^{(n)} - y_p^{(n)}$, $j \leq p$. As

$$y_j^{(n)} = \tilde{\mu}_j + \sum_{i=1}^{M} y_i^{(n-1)} \tilde{A}_{ij},$$

then

$$y_j^{(n)} - y_p^{(n)} = \tilde{\mu}_j - \tilde{\mu}_p + \sum_{i=1}^{M} y_i^{(n-1)} (\tilde{A}_{ij} - \tilde{A}_{ip}).$$

In Lemma 4.7 we show that $\tilde{\mu}_j \geq \tilde{\mu}_p$, $j \leq p$, when $\frac{\mu_{i+1}}{\mu_i} \leq 1-\rho$ for every $i = 1, \ldots, M-1$. Let us regroup the sum $\sum_{i=1}^{M} y_i^{(n-1)} (\tilde{A}_{ij} - \tilde{A}_{ip})$ in the following way

$$\sum_{i=1}^{M} y_i^{(n-1)} (\tilde{A}_{ij} - \tilde{A}_{ip}) = y_M^{(n-1)} \sum_{k=1}^{M} (\tilde{A}_{kj} - \tilde{A}_{kp}) + \sum_{i=1}^{M-1} (y_i^{(n-1)} - y_{i+1}^{(n-1)}) \sum_{k=1}^{r} (\tilde{A}_{kj} - \tilde{A}_{kp}).$$

As $y_i^{(n-1)} \geq y_{i+1}^{(n-1)}$, $i = 1, \ldots, M-1$, according to the induction step, then to show that $\sum_{i=1}^{M} y_i^{(n-1)} (\tilde{A}_{ij} - \tilde{A}_{ip}) \geq 0$, $j \leq p$ it is enough to show that $\sum_{i=1}^{r} (\tilde{A}_{ij} - \tilde{A}_{ip}) \geq 0$, $j \leq p$, $r = 1, \ldots, M$. We show this in Lemma 4.8. Using the previous discussion we prove the induction step and so prove Lemma 4.6. ∎

Now let us prove Lemmas 4.7 and 4.8.

Lemma 4.7

$$\tilde{\mu}_1 \geq \tilde{\mu}_2 \ldots \geq \tilde{\mu}_M,$$

if $\frac{\mu_{i+1}}{\mu_i} \leq 1-\rho$ *for every* $i = 1, \ldots, M-1$.

Proof. We consider $\tilde{\mu}_j - \tilde{\mu}_p$, $j < p$. Let us recall that $g_1 \geq g_2 \ldots \geq g_M$ and $\mu_1 \geq \mu_2 \ldots \geq \mu_M$.

Let us denote $f_2(x) = \sum_k \frac{g_k}{x+\mu_k g_k}$, then $\tilde{\mu}_i = \frac{\mu_i}{1-f_2(\mu_i g_i)}$ and

$$\tilde{\mu}_j - \tilde{\mu}_p = \frac{\mu_j - \mu_p - (\mu_j f_2(\mu_p g_p) - \mu_p f_2(\mu_j g_j))}{(1-f_2(\mu_j g_j))(1-f_2(\mu_p g_p))}.$$

Let us denote $\Delta_1 = \mu_j - \mu_p - (\mu_j f_2(\mu_p g_p) - \mu_p f_2(\mu_j g_j))$. As $0 < f_2(x) < \rho$, then

$$\Delta_1 > (\mu_j - \mu_p) \left(1 - \rho \left(\frac{1}{1-\frac{\mu_p}{\mu_j}}\right)\right) \geq 0,$$

when

$$\frac{\mu_p}{\mu_j} \leq 1 - \rho.$$

So, $\tilde{\mu}_j - \tilde{\mu}_p \geq 0$, $j < p$ if $\frac{\mu_p}{\mu_j} \leq 1 - \rho$.

Let us consider Δ_1 when $\mu_j > \mu_p$ and $g_j = g_p$. In this case

$$\Delta_1 = (\mu_j - \mu_p)\left(1 - \sum_{k=1}^{M} \frac{g_k(g_j(\mu_j + \mu_p) + \mu_k g_k)}{(\mu_p g_j + \mu_k g_k)(\mu_p g_j + \mu_k g_k)}\right).$$

We can show that

$$\frac{g_k(g_j(\mu_j + \mu_p) + \mu_k g_k)}{(\mu_j g_j + \mu_k g_k)(\mu_p g_j + \mu_k g_k)} < \frac{1}{\mu_k}, \quad k = 1, \ldots, M.$$

Then

$$\Delta_1 > (\mu_j - \mu_p)\left(1 - \sum_{k=1}^{M} \frac{1}{\mu_k}\right) = (\mu_j - \mu_p)(1 - \rho) > 0.$$

In the case when $\mu_j = \mu_p$ and $g_j = g_p$, then $\tilde{\mu}_j - \tilde{\mu}_p = 0$.

Then we have proved the following:

$$\begin{array}{lll} \text{If} \quad g_j = g_p \ \mu_j = \mu_p, & & \text{then} \quad \tilde{\mu}_j = \tilde{\mu}_p, \\ \text{If} \quad g_j = g_p \ \mu_j > \mu_p, & & \text{then} \quad \tilde{\mu}_j > \tilde{\mu}_p, \\ \text{If} \quad g_j \geq g_p \ \mu_j \geq \mu_p, & \frac{\mu_p}{\mu_j} \leq 1 - \rho, & \text{then} \quad \tilde{\mu}_j \geq \tilde{\mu}_p. \end{array}$$

Setting $p = j + 1$ and recalling that $\mu_1 \geq \ldots \geq \mu_M$, we get that $\tilde{\mu}_1 \geq \tilde{\mu}_2 \ldots \geq \tilde{\mu}_M$ is true when $\frac{\mu_{i+1}}{\mu_i} \leq 1 - \rho$ for every $i = 1, \ldots, M - 1$. That proves the statement of Lemma 4.7.

Returning back to the main Theorem 4.1, Lemma 4.7 gives condition (4.5) as a restriction on system parameters.

Let us notice that if for the job classes i and $i + 1$ $\frac{\mu_{i+1}}{\mu_i} < 1 - \rho$, then setting the weights for these classes equal, still $\tilde{\mu}_i \geq \tilde{\mu}_{i+1}$. This condition gives us as a result Remark 4.3 and Remark 4.1. ∎

Lemma 4.8

$$\sum_{i=1}^{r} \tilde{A}_{i1} \geq \sum_{i=1}^{r} \tilde{A}_{i2} \geq \ldots \geq \sum_{i=1}^{r} \tilde{A}_{iM}, \quad r = 1, \ldots, M.$$

Proof. Let us recall $\tilde{A} = M^{-1}AM(E - D)^{-1}$ and let us fix r in the following proof. Let us define

$$f_3(x) = \frac{\sum_{i=1}^{r} \frac{x}{\mu_i(x + \mu_i g_i)}}{1 - \rho + \sum_{k=1}^{M} \frac{x}{\mu_k(x + \mu_k g_k)}} = \frac{h_1(x)}{1 - \rho + h_1(x) + h_2(x)},$$

where $h_1(x) = \sum_{i=1}^{r} \frac{x}{\mu_i(x+\mu_i g_i)} > 0$, and $h_2(x) = \sum_{j=r+1}^{M} \frac{x}{\mu_j(x+\mu_j g_j)} > 0$. Then $\sum_{i=1}^{r} \tilde{A}_{ij} = f_3(\mu_j g_j)$. To prove the statement of the Theorem it is enough to show that the function $f_3(x)$ is increasing in x. For that it is enough to show that $\frac{df_3(x)}{dx} \geq 0$. Let us consider

$$\frac{df_3(x)}{dx} = \frac{h_1'(x)(1-\rho) + h_1'(x)h_2(x) - h_1(x)h_2'(x)}{(1-\rho+h_1(x)+h_2(x))^2}.$$

We can show that

$$h_1'(x)h_2(x) - h_1(x)h_2'(x) = \sum_{i=1}^{r}\sum_{k=r+1}^{M} \frac{x^2(\mu_i g_i - \mu_k g_k)}{(x+\mu_i g_i)^2(x+\mu_k g_k)^2 \mu_k \mu_i} \geq 0,$$

as $\mu_j g_j \geq \mu_p g_p$, $j \leq p$. Since $h_1'(x) > 0$ and $1-\rho > 0$, then $\frac{df_3(x)}{dx} \geq 0$, $f_3(x)$ is an increasing function of x and we prove the statement of Lemma 4.8. ∎

Chapter 5

Optimal policy for multi-class scheduling in a single server queue

5.1 Summary

In this chapter we apply the Gittins optimality result to characterize the optimal scheduling discipline in a multi-class $M/G/1$ queue. We apply the general result to several cases of practical interest where the service time distributions belong to the set of DHR distributions, like Pareto or hyper-exponential. When there is only one class it is known that in this case the LAS policy is optimal. We show that in the multi-class case the optimal policy is a priority discipline, where jobs of the various classes depending on their attained service are classified into several priority levels. Using a tagged-job approach we obtain, for every class, the mean conditional sojourn time. This allows us to compare numerically the mean sojourn time in the system between the Gittins optimal and popular policies like PS, FCFS and LAS.

Our results may be applicable for instance in an Internet router, where packets generated by different applications must be served or service is non-preemptive. Typically a router does not have access to the exact required service time (in packets) of the TCP connections, but it may have access to the attained service of each connection. Thus we implement the Gittins' optimal algorithm in NS-2 and we perform numerical experiments to evaluate the achievable performance gain.

5.2 Introduction

We are interested to schedule the jobs in the $M/G/1$ queue with the aim to minimize the mean sojourn time in the system as well as the mean number of jobs in the system. In our study we restrict ourselves to the non-anticipating scheduling policies. Let us recall that the policy in non-anticipating if it does not use information about the size of the arriving jobs. In [Git89] Gittins considered an $M/G/1$ queue and proved that the so-called Gittins index rule minimizes the mean delay. At every moment of time the Gittins rule calculates, depending on the service times of jobs, which job should be served. Gittins derived this result as a byproduct of his groundbreaking results on the multi-armed bandit problem. The literature on multi-armed bandit related papers that build on Gittins' result is huge (see for example [VWB85, Whi88, Web92, Tsi93, DGNM96, FW99, BNM00]). However, the optimality result of the Gittins index in the context of the $M/G/1$ queue has not been fully exploited, and it has not received the attention it deserves.

In the present work we generalize the Gittins index approach to the scheduling of the multi-class $M/G/1$ queue. We emphasize that Gittins' optimality in a multi-class queue holds under much more general conditions than the condition required for the optimality of the well-known $c\mu$-rule. We recall that the $c\mu$-rule is the discipline that gives strict priority in descending order of $c_k\mu_k$, where c_k and μ_k refer to a cost and the inverse of the mean service requirement, respectively, of class k. Indeed it is known (see for example [BVW85, SY92, NT94]) that the $c\mu$-rule minimizes the weighted mean number of customers in the queue in two main settings: (i) generally distributed service requirements among all non-preemptive disciplines and (ii) exponentially distributed service requirements among all preemptive non-anticipating disciplines. In the preemptive case the $c\mu$-rule is only optimal if the service times are exponentially distributed. On the other hand, by applying Gittins' framework to the multi-class queue one can characterize the optimal policy for arbitrary service time distributions. We believe that our results open an interesting avenue for further research. For instance well-known optimality results in a single-class queue like the optimality of the LAS discipline when the service times are of type decreasing hazard rate or the optimality of FCFS when the service time distribution is of type New-Better-than-Used-in-Expectation can all be derived as corollaries of Gittins' result. The optimality of the $c\mu$-rule can also easily be derived from the Gittins' result.

In order to get insights into the structure of the optimal policy in the multi-class case we consider several relevant cases where the service time distributions are Pareto or hyper-exponential. We have used these distributions due to the evidence that the file size distributions in the Internet are well presented by the heavy-tailed distributions such as Pareto distributions with the infinite second moment. Also it was shown that the job sizes in the Internet are well modelled with the distributions with the decreasing hazard rate. We refer to [NMM98,

CB97, Wil01] for more details on this area, see also Subsection 1.1.3. In particular, we study the optimal multi-class scheduling in the following cases of the service time distributions: two Pareto distributions, several Pareto distributions, one hyper-exponential and one exponential distributions. Using a tagged-job approach and the collective marks method we obtain, for every class, the mean conditional sojourn time. This allows us to compare numerically the mean sojourn time in the system between the Gittins optimal and popular policies like PS, FCFS and LAS. We find that in a particular example with two classes and Pareto-type service time distribution the Gittins' policy outperforms LAS by nearly 25% under moderate load. We demonstrate that in particular cases the PS has much worse performance than the Gittins policy.

From an application point of view, our findings could be applied in Internet routers. Imagine that incoming packets are classified based on the application or the source that generated them. Then it is reasonable to expect that the service time distributions of the various classes may differ from each other. A router in the Internet does not typically have access to the exact required service time (in packets) of the TCP connections, but it may have access to the attained service of each connection. Thus we can apply our theoretical findings in order to obtain the optimal (from the connection-level performance point of view) scheduler at the packet level. We implement the Gittins' scheduling in the NS-2 simulator and perform experiments to evaluate the achievable performance gain.

The structure of the chapter is as follows: In Section 2 we review the Gittins index policy for the single-class $M/G/1$ queue and then provide a general framework of the Gittins index policy for the multi-class $M/G/1$ queue. In Section 3, we study the Gittins index policy for the case of two Pareto distributed classes. In particular, we derive analytic expressions for the mean conditional sojourn times, study various properties of the optimal policy, provide numerical examples and NS-2 simulations. At the end of Section 3 we generalized the results to multiple Pareto classes. In Section 4 we study the case of two distributions: one distribution being exponential and the other distribution being hyper-exponential with two phases. For the case of exponential and hyper-exponential distributions, we also obtain analytical results and provide numerical examples. Section 5 concludes the chapter.

5.3 Gittins policy in multi-class $M/G/1$ queue

Let us first recall the basic results related to the Gittins index policy in the context of a single-class $M/G/1$ queue.

Let Π denote the set of non-anticipating scheduling policies. Popular disciplines such as PS, FCFS and LAS, also called FB, belong to Π. Important disciplines that do not belong to Π are SRPT and Shortest Processing Time (SPT).

We consider a single-class $M/G/1$ queue. Let X denote the service time with distribution

$P(X \leq x) = F(x)$. The density is denoted by $f(x)$, the complementary distribution by $\overline{F}(x) = 1 - F(x)$ and the hazard rate function by $h(x) = f(x)/\overline{F}(x)$. Let $\overline{T}^{\pi}(x)$, $\pi \in \Pi$ denote the mean conditional sojourn time for the job of size x in the system under the scheduling policy π, and $\overline{T}^{\pi}$, $\pi \in \Pi$ denote the mean sojourn time in the system under the scheduling policy π.

Let us give some definitions.

Definition 5.1 *For any $a, \Delta \geq 0$, let*

$$J(a, \Delta) = \frac{\int_0^{\Delta} f(a+t)dt}{\int_0^{\Delta} \overline{F}(a+t)dt} = \frac{\overline{F}(a) - \overline{F}(a+\Delta)}{\int_0^{\Delta} \overline{F}(a+t)dt}. \tag{5.1}$$

For a job that has attained service a and is assigned Δ units of service, equation (5.1) can be interpreted as the ratio between (i) the probability that the job will complete with a quota of Δ (interpreted as payoff) and (ii) the expected processor time that a job with attained service a and service quota Δ will require from the server (interpreted as investment). Note that for every $a > 0$

$$J(a, 0) = \frac{f(a)}{\overline{F}(a)} = h(a),$$
$$J(a, \infty) = \frac{\overline{F}(a)}{\int_0^{\infty} \overline{F}(a+t)\, dt} = 1/E[X - a | X > a].$$

Note further that $J(a, \Delta)$ is continuous with respect to Δ.

Definition 5.2 *The Gittins index function is defined by*

$$G(a) = \sup_{\Delta \geq 0} J(a, \Delta), \tag{5.2}$$

for any $a \geq 0$.

We call $G(a)$ the *Gittins index* after the author of book [Git89], which handles various static and dynamic scheduling problems. Independently, Sevcik defined a corresponding index when considering scheduling problems without arrivals in [Sev74]. In addition, this index has been dealt with by Yashkov, see [Yas92] and references therein, in particular the works by Klimov [Kli74, Kli78].

Definition 5.3 *For any $a \geq 0$, let*

$$\Delta^*(a) = \sup\{\Delta \geq 0 \mid J(a, \Delta) = G(a)\}. \tag{5.3}$$

By definition, $G(a) = J(a, \Delta^*(a))$ for all a.

Definition 5.4 *The Gittins index policy π_g is the scheduling discipline that at every instant of time gives service to the job in the system with highest $G(a)$, where a is the job's attained service.*

Theorem 5.1 *The Gittins index policy minimizes the mean sojourn time in the system between all non-anticipating scheduling policies. Otherwise, in the $M/G/1$ queue for any $\pi \in \Pi$,*

$$\overline{T}^{\pi_g} \leq \overline{T}^{\pi}.$$

Proof. See [Git89]. ∎

Note that by Little's law the Gittins index policy also minimizes the mean number of jobs in the system.

We generalize the result of Theorem 5.1 to the case of the multi-class single server queue. Let us consider a multi-class $M/G/1$ queue. Let X_i denote the service time with distribution $P(X_i \leq x) = F_i(x)$ for every class $i = 1, \ldots, M$. The density is denoted by $f_i(x)$ and the complementary distribution by $\overline{F}_i(x) = 1 - F_i(x)$. The jobs of every class-i arrive with the Poisson process with rate λ_i, the total arrival rate is $\lambda = \sum_{i=1}^{M} \lambda_i$. For every class $i = 1, \ldots, M$ we define $J_i(a, \Delta) = \frac{\int_0^{\Delta} f_i(a+t)dt}{\int_0^{\Delta} \overline{F}_i(a+t)dt}$ and then the Gittins index of a class-i job is defined as $G_i(a) = \sup_{\Delta \geq 0} J_i(a, \Delta)$.

The mean conditional sojourn time $\overline{T}_i^{\pi}(x)$ for the class-i job of size x, $i = 1, \ldots, M$, and the mean sojourn time $\overline{T}^{\pi}$ in the system under the scheduling policy $\pi \in \Pi$ are defined as in the previous section.

Proposition 5.1 *In a multi-class $M/G/1$ queue the policy that schedules the job with highest Gittins index $G_i(a)$, $i = 1, \ldots, M$ in the system, where a is the job's attained service, is the optimal policy that minimizes the mean sojourn time.*

Proof. The result follows directly from the application Gittins Index Definition 5.2 and Theorem 5.1 to a multi-class $M/G/1$ queue. ∎

Let $h_i(x) = f_i(x)/\overline{F}_i(x)$ denote the hazard rate function of class $i = 1, \ldots, M$. Let the service time distribution of class-i have a decreasing hazard rate. It is possible to show, see [AA07], that if $h_i(x)$ is non-increasing, the function $J_i(a, \Delta)$ is non-increasing in Δ. Thus

$$G_i(a) = J_i(a, 0) = h_i(a). \tag{5.4}$$

As a consequence we obtain the following proposition.

Proposition 5.2 *In a multi-class $M/G/1$ queue with non-increasing hazard rate functions $h_i(x)$ for every class $i = 1, \ldots, M$, the policy that schedules the job with highest $h_i(a)$, $i =$*

$1, \ldots, M$ in the system, where a is the job's attained service, is the optimal policy that minimizes the mean sojourn time.

Proof. Follows immediately from the Gittins policy Definition 5.4, Proposition 5.1 and equation (5.4). ∎

The policy presented in Proposition 5.2 is an optimal policy for the multi-class single-server queue. Let us notice that for the single class single server queue the Gittins policy becomes a LAS policy, as the hazard rate function is the same for all the jobs and so the job with the maximal value of the hazard rate function from attained service is the job with the least attained service. When we serve the jobs with the Gittins policy in the multi-class queue to find a job which has to be served next we need to calculate the hazard rate of the attained service of every job in the system. The job which has the maximal value of the hazard rate function from it's attained service is served the next.

Now let us consider several subcases of the described general approach. Depending on the behavior of the hazard rate functions of the job classes the policy is different. We consider the case with two job classes in the system and two subcases: (a) both job classes are distributed with Pareto and the hazard rate functions do not cross and (b) job size distributions are hyper-exponential with one and two phases and they cross at one point. Then we extend the case of two Pareto job classes to the case of N Pareto job classes. We provide the analytical expressions for the mean conditional sojourn times in the system and numerical results. We implemented the algorithm for the case of two Pareto classes with the NS-2 simulator on the packet level.

5.4 Two Pareto classes

Let us first present the case when job sizes are distributed according to Pareto distribution.

5.4.1 Model description

We consider the case when the job size distribution functions are Pareto. We consider the two-class single server $M/G/1$ queue. The jobs of each class arrive to the server with Poisson process with rates λ_1 and λ_2. The job sizes are distributed according to the Pareto distributions, namely

$$F_i(x) = 1 - \frac{b_i}{(x+b_i)^{c_i}}, \quad i = 1, 2. \tag{5.5}$$

Here $b_i = m_i(c_i - 1)$, where m_i is the mean of class-i, $i = 1, 2$. Then $f_i(x) = b_i^{c_i} c_i/(x+b_i)^{c_i+1}$, $i = 1, 2$ and the hazard rate functions are

$$h_i(x) = \frac{c_i}{(x+b_i)}, \quad i = 1, 2.$$

$$\frac{c_1}{x+b_1} = \frac{c_2}{x+b_2} \rightarrow (c_1 - c_2)x = c_2 b_1 - c_1 b_2$$

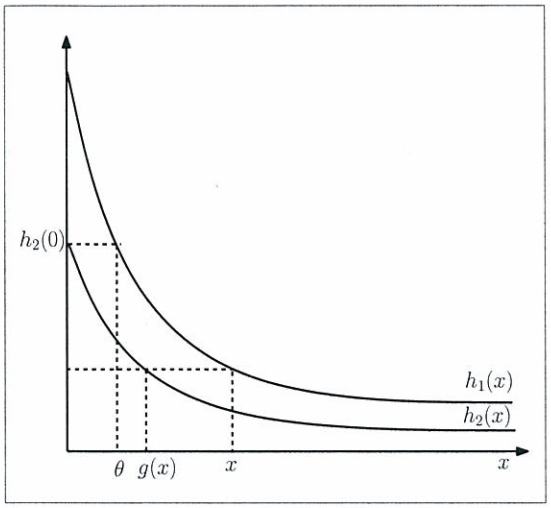

Figure 5.1: Two Pareto classes, hazard rates

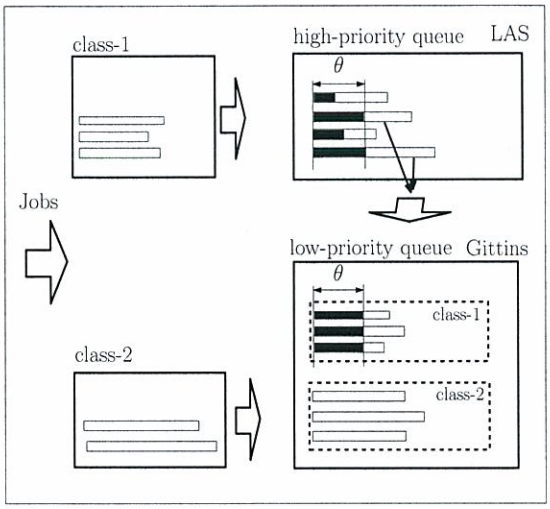

Figure 5.2: Two Pareto classes, policy scheme

This function cross at the point

$$a^{**} = \frac{c_2 b_1 - c_1 b_2}{c_1 - c_2}.$$

Without loss of generality suppose that $c_1 > c_2$. Then the behavior of the hazard rate functions depends on the values of b_1 and b_2. Let us first consider the case when the hazard rate function do not cross, so $a^{**} < 0$. This happens when $b_1/b_2 < c_1/c_2$. Then the hazard-rate functions are decreasing and never cross and $h_1(x) > h_2(x)$, for all $x \geq 0$.

Let us denote θ and function $g(x)$ in the following way that

$$h_1(x) = h_2(g(x)), \quad h_1(\theta) = h_2(0).$$

We can see that $g(\theta) = 0$. For given expressions of $h_i(x)$, $i = 1, 2$ we get

$$g(x) = \frac{c_2}{c_1}(x + b_1) - b_2, \quad \theta = \frac{c_1 b_2 - c_2 b_1}{c_2}.$$

According to the definition of function $g(x)$, the class-1 job of size x and the class-2 job of size $g(x)$ have the same value of the hazard rate when they are fully served, see Figure 5.1. Then the optimal policy structure is the following. The optimal policy scheme is given on Figure 5.2.

5.4.2 Optimal policy

The jobs in the system are served in two queues, low and high priority queues. The class-1 jobs which have attained service $a < \theta$ are served in the high priority queue with LAS policy. When the class-1 job achieves θ amount of service it is moved to the second low priority queue.

The class-2 jobs are moved immediately to the low priority queue. The low priority queue is served only when the high priority queue is empty. In the low priority queue the jobs are served in the following way: the service is given to the job with the highest $h_i(a)$, where a is the job's attained service. So, for every class-1 job with a attained service the function $h_1(a)$ is calculated, for every class-2 job with a attained service the function $h_2(a)$ is calculated. After all values of $h_i(a)$ are compared and the job which has the highest $h_i(a)$ is served.

Now let us calculate the expressions of the mean conditional sojourn time for the class-1 and class-2 jobs.

5.4.3 Mean conditional sojourn times

Let us denote by indices $[]^{(1)}$ and $[]^{(2)}$ the values for class-1 and class-2 accordingly.

Let us define as $\overline{X_y^n}^{(i)}$ the n-th moment and $\rho_y^{(i)}$ be the utilization factor for the distribution $F_i(x)$ truncated at y for $i = 1, 2$. The distribution truncated at y equals to $F(x)$ for $x \leq y$ and equals to 1 when $x > y$. Let us denote $W_{x,y}$ the mean workload in the system which consists only of class-1 jobs of size less than x and of class-2 jobs of size less than y. According to the Pollaczek-Khinchin formula

$$W_{x,y} = \frac{\lambda_1 \overline{X_x^2}^{(1)} + \lambda_2 \overline{X_y^2}^{(2)}}{2(1 - \rho_x^{(1)} - \rho_y^{(2)})}.$$

Now let us formulate the following Theorem which we prove in the Appendix.

Theorem 5.2 *In the two-class $M/G/1$ queue where the job size distributions are Pareto, given by (5.5), and which is scheduled with the Gittins policy described in Subsection 5.4.2, the mean conditional sojourn times for class-1 and class-2 jobs are*

$$T_1(x) = \frac{x + W_{x,0}}{1 - \rho_x^{(1)}}, \quad x < \theta, \tag{5.6}$$

$$T_1(x) = \frac{x + W_{x,g(x)}}{1 - \rho_x^{(1)} - \rho_{g(x)}^{(2)}}, \quad x \geq \theta, \tag{5.7}$$

$$T_2(g(x)) = \frac{g(x) + W_{x,g(x)}}{1 - \rho_x^{(1)} - \rho_{g(x)}^{(2)}}, \quad x \geq \theta. \tag{5.8}$$

Proof. The proof is very technical and is given in the Appendix. Let us give a very general idea of the proof. To obtain expressions (5.7), (5.8) we use the fact that the second low priority queue is the queue with batch arrivals. To obtain expressions of the mean batch size with and without the tagged job we apply the Generating function analysis using the method of the collective marks. ∎

The obtained expressions (5.6), (5.7) and (5.8) can be interpreted using the tagged-job and mean value approach.

Let us consider class-1 jobs. For the job of size $x \leq \theta$ the mean conditional sojourn time is known, [Kle76a, Sec. 4.6], $T_1(x) = \frac{x+W_{x,0}}{1-\rho_x^{(1)}}$, $x < \theta$, where $W_{x,0}$ is the mean workload and $\rho_x^{(1)}$ is the mean load in the system for class-1 job of size x. The mean workload $W_{x,0}$ and mean load $\rho_x^{(1)}$ consider only the jobs of the high priority queue of class-1.

For the jobs of size $x > \theta$ the expression (5.8) can be presented in the following way, $T_1(x) = x + W_{x,g(x)} + T_1(x)(\rho_x^{(1)} + \rho_{g(x)}^{(2)})$, where

- x is time which is actually spent to serve the job;
- $W_{x,g(x)}$ is the mean workload which the tagged job finds in the system and which has to be processed before it;
- $T_1(x)(\rho_x^{(1)} + \rho_{g(x)}^{(2)})$ is the mean time to serve the jobs which arrive to the system during the service time of the tagged job and which have to be served before it.

Let us provide more explanations. Let us find the expression for the mean workload in the system for the class-1 job of size x, which is the tagged job. According to the PASTA property of Poisson arrivals, all jobs arriving to the system see the system in the same steady state. So, class-1 and class-2 jobs see the same mean workload in the system when they arrive. As we need to take into account only the mean workload which is served before the tagged job, then for each job the mean workload $W_{x,g(x)}$ depends on the size of the tagged job, x. The jobs which have to be served before the tagged job of class-1 of size x are the class-1 jobs of size less than x and class-2 jobs of size less than $g(x)$. Then using Pollaczek-Khinchin formula (5.6) for the class-1 jobs of size less than x and class-2 jobs less than $g(x)$ we conclude that $W_{x,g(x)}$ gives the mean workload in the system for the class-1 job of size x, which has to be done before it. Let us notice that the mean workload in the system for the class-2 job of size $g(x)$ is the same, $W_{x,g(x)}$.

Now let us find the mean workload which arrive during the service time of the tagged job. The service time of the tagged job is $T_1(x)$. The mean load of jobs arrival to the system is: for the class-1 of size less than x is $\lambda_1 \overline{X_x^1}^{(1)} = \rho_x^{(1)}$ and for the class-2 with size less than $g(x)$ is $\lambda_2 \overline{X_{g(x)}^1}^{(2)} = \rho_{g(x)}^{(2)}$. Then $T_1(x)(\rho_x^{(1)} + \rho_{g(x)}^{(2)})$ is the mean workload which arrive during the service time of the tagged job of class-1 of size x.

Now we use the similar analysis to give interpretation to the expression of $T_2(g(x))$ for the class-2 job of size $g(x)$. We can rewrite expression (5.8) in the following way $T_2(g(x)) = g(x) + W_{x,g(x)} + T_2(g(x))(\rho_x^{(1)} + \rho_{g(x)}^{(2)})$.

In the case of the tagged job of class-2 of size $g(x)$ the jobs which have to be served before the tagged job are jobs of class-1 of size less than x and jobs of class-2 of size less than $g(x)$.

Then in the previous expression $g(x)$ is the time to serve the class-2 job of size $g(x)$; $W_{x,g(x)}$ is the mean workload in the system for the class-2 job of size $g(x)$ which has to be served before it; $T_2(g(x))(\rho_x^{(1)} + \rho_{g(x)}^{(2)})$ is the mean work which arrives during the service time $T_2(x)$ and which has to be served before class-2 job of size $g(x)$.

Let us describe several properties of the optimal policy.

5.4.4 Properties of the optimal policy

Property 5.1 *When the class-2 jobs arrive to the server they are not served immediately, but wait until the high priority queue is empty. The mean waiting time is the limit* $\lim_{g(x)\to 0} T_2(g(x))$. *As* $\lim_{x\to\theta} g(x) = 0$, *then*

$$\lim_{g(x)\to 0} T_2(g(x)) = \frac{W_{\theta,0}}{1-\rho_\theta^{(1)}} = \frac{\lambda_1 \overline{X_\theta^2}^{(1)}}{(1-\rho_\theta^{(1)})^2}.$$

Let us notice that

$$\lim_{g(x)\to 0} T_2(g(x)) \neq T_1(\theta) = \frac{\theta + W_{\theta,0}}{1-\rho_\theta^{(1)}}.$$

That means that the mean waiting time of the class-2 jobs is not equal to the time which the parts of the class-1 jobs of size more than θ wait in the system before start to be served.

Property 5.2 *Let us consider the condition of no new arrival. According to the optimal policy structure in the low priority queue the jobs are served according to the LAS policy with different rates, which depend on the number of jobs in each class and hazard rate functions. For the case when there are no new arrivals in the low priority queue we can calculate the rates with which the class-1 jobs and class-2 jobs are served in the system at every moment of time. We consider that all the class-1 jobs and all the class-2 jobs already received the same amount of service. Let n_1 and n_2 be the number of jobs in class-1 and class-2 and let x_1 and x_2 be the attained services of every job in these classes. Then at any moment*

$$h_1(x_1) = h_2(x_2).$$

If the total capacity of the server is Δ, then let Δ_1 and Δ_2 be the capacities which each job of class-1 and class-2 receives. Then

$$n_1\Delta_1 + n_2\Delta_2 = \Delta.$$

Also

$$h_1(x_1 + \Delta_1) = h_2(x_2 + \Delta_2).$$

As Δ is very small (and so as well Δ_1 and Δ_2) according to the LAS policy, then we can approximate

$$h_i(x + \Delta_i) = h_i(x) + \Delta_i h_i'(x), \quad i = 1, 2.$$

Then from the previous equations we have

$$\Delta_1 h_1'(x_1) = \Delta_2 h_2'(x_2).$$

Then

$$\frac{\Delta_1}{\Delta} = \frac{h_2'(x_1)}{n_1 h_2'(x_2) + n_2 h_1'(x_1)},$$
$$\frac{\Delta_2}{\Delta} = \frac{h_1'(x_1)}{n_1 h_2'(x_2) + n_2 h_1'(x_1)}.$$

This result is true for any two distributions for which the hazard rates are decreasing and never cross. For the case of two Pareto distributions given by (5.5) we have the following:

$$\frac{\Delta_1}{\Delta} = \frac{c_2}{n_1 c_2 + n_2 c_1}, \quad \frac{\Delta_2}{\Delta} = \frac{c_1}{n_1 c_2 + n_2 c_1}.$$

So, for the case of two Pareto distributions the service rates of class-1 and class-2 jobs do not depend on the current jobs' attained services.

Property 5.3 *As one can see from the optimal policy description, the class-1 and class-2 jobs quit the system together if they have the same values of the hazard rate functions of their sizes and if they find each other in the system. According to the definition of the $g(x)$ function we can conclude that the class-1 job of size x and class-2 job of size $g(x)$, if they find each other in the system, quit the system together. But these jobs do not have the same conditional mean sojourn time,*

$$T_1(x) \neq T_2(g(x)).$$

Let us prove this fact formally. Let $\tau_1(x|P)$ be the sojourn time of class-1 job of size x, given the sample-path P. By sample-path we understand the jobs (and their residual service times) present when the job arrives, and also all the future arrivals and service times. It holds that:

$$\tau_1(x|P) \neq \tau_2(g(x)|P). \tag{5.9}$$

This is true as if we consider P_1 the sample-path such as the system is empty and there is no arriving jobs during the serving time. Then $\tau_1(x|P_1) = x$ and $\tau_2(g(x)|P_1) = g(x)$, so they are not equal at least on one sample path. The probability that the sample-path P_1 takes place is non-zero. That proves (5.9).

By unconditioning on the sample-path that the job finds upon arrival, we can define the mean conditional sojourn time as:

$$T_1(x) = \int_{P\in\Omega} \tau_1(x|P) dF_1(P), \tag{5.10}$$

where $dF_1(P)$ is the probability that the sample path upon arrival is $P \in \Omega$ equals to all possible sample-paths. Similarly

$$T_2(g(x)) = \int_{P\in\Omega} \tau_2(g(x)|P) dF_2(P). \tag{5.11}$$

Now, by the PASTA property of Poisson arrivals, job arrivals see the steady state of the system, which means that class-1 and class-2 jobs see the same system upon arrival. This implies that $dF_1(P) = dF_2(P)$, for all P. Now, in view of (5.9), this implies that $T_1(x) \neq T_2(g(x))$, which proves the property.

5.4.5 Two Pareto classes with intersecting hazard rate functions

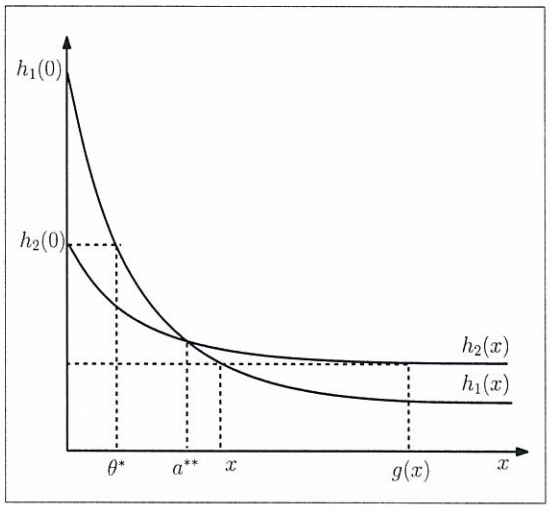

Figure 5.3: Two Pareto extension classes, hazard rates

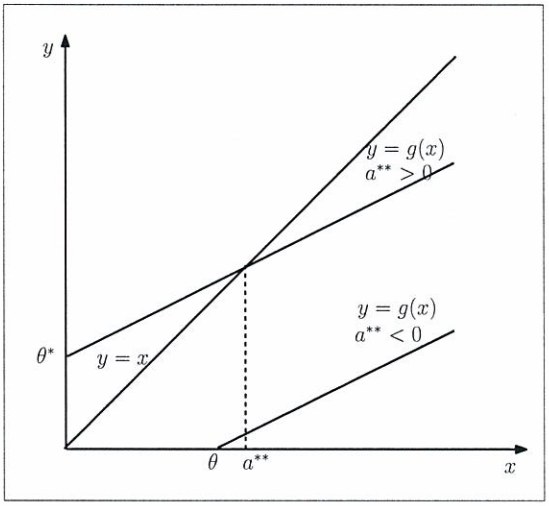

Figure 5.4: Two Pareto extension classes, $g(x)$ function behavior

Now let us consider the case when the hazard rate function cross, then $a^{**} = (c_2b_1 - c_1b_2)/(c_1 - c_2) \geq 0$, see Figure 5.3. As we considered $c_1 > c_2$, then $h_1(0) < h_2(0)$ and then class-2 jobs are served in the high priority queue until they receive $\theta^* = (c_1b_2 - c_2b_1)/c_2$ amount of service. Here θ^* is such that $h_2(\theta^*) = h_1(0)$ and $g(\theta^*) = 0$. In this case the $g(x)$ function crosses the $y = x$ function at point a^{**}, see Figure 5.4, and so in the low priority queue the class-2 jobs are served with higher priority with comparison to the class-1 jobs until they receive a^{**} amount of service. After class-1 and class-2 jobs received a^{**} amount of service the priority changes and

class-1 jobs receive more capacity of the server in the system. According to this analysis we can rewrite the expressions of mean conditional sojourn times of Section 5.4 Theorem 5.2 in the following way

Corollary 5.1 *In the two-class $M/G/1$ queue where the job size distributions are Pareto, given by (5.5) such that the hazard rate functions cross, and which is scheduled with the Gittins optimal policy, the mean conditional sojourn times for class-1 and class-2 jobs are*

$$T_1(x) = \frac{x + W_{x,g(x)}}{1 - \rho_x^{(1)} - \rho_{g(x)}^{(2)}}, \quad x \geq 0,$$
$$T_2(x) = \frac{x + W_{x,0}}{1 - \rho_x^{(2)}}, \quad x < \theta^*,$$
$$T_2(g(x)) = \frac{g(x) + W_{x,g(x)}}{1 - \rho_x^{(1)} - \rho_{g(x)}^{(2)}}, \quad x \geq \theta^*.$$

Proof. The proof follows from the previous discussion. ∎

5.4.6 Numerical results

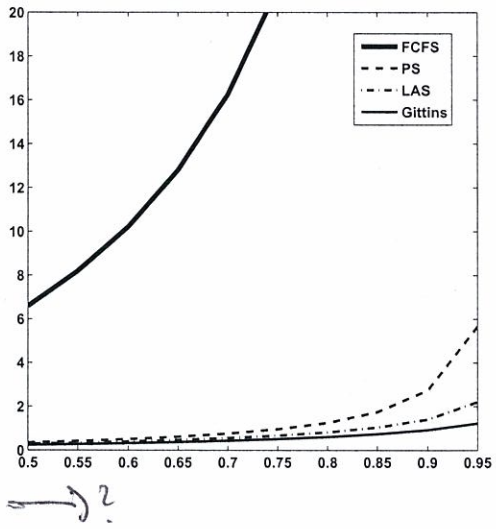

Figure 5.5: Two Pareto classes, mean sojourn times comparison, V_1

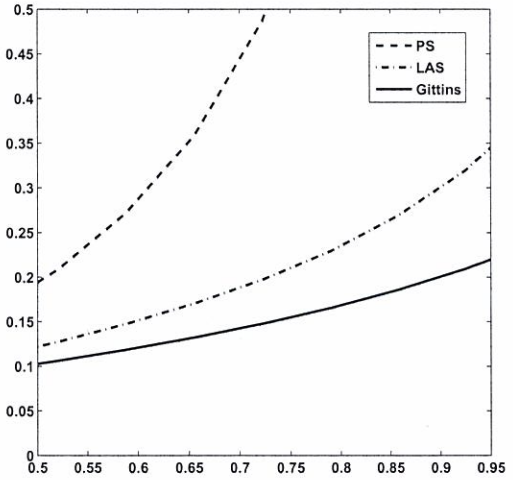

Figure 5.6: Two Pareto classes, mean sojourn times comparison, V_2

We consider two classes with parameters presented in Table 5.1. We provide the results for two different parameters sets, which we call V_1 and V_2

Table 5.1: Two Pareto classes, parameters

V	c_1	c_2	m_1	m_2	ρ_1	ρ_2	ρ
V_1	25.0	2.12	0.04	0.89	0.1	0.4..0.85	0.5..0.95
V_2	10.0	1.25	0.05	1.35	0.25	0.06..0.74	0.31..0.99

It is known that in the Internet the most of the traffic is presented by the large files (80%), while most of the files are very small (90%). This phenomenon is referred to as "mice-elephant" effect. Also it is known that the file sizes are well presented by the heavy-tailed distributions like Pareto. Here the class-1 jobs represent "mice" class and class-2 jobs "elephants". We consider that the load of the small files is fixed and find the mean sojourn time in the system according to the different values of the "elephant" class arrival rate.

We compare the mean sojourn time for the Gittins policy, PS, FCFS and LAS policies. These policies can be applied either in the Internet routers or in the Web service. The expected sojourn times are

$$\overline{T}^{PS} = \frac{\rho/\lambda}{1-\rho},$$
$$\overline{T}^{FCFS} = \rho/\lambda + W_{\infty,\infty},$$

here $W_{\infty,\infty}$ means the total mean unfinished work in the system.

$$\overline{T}^{LAS} = \frac{1}{\lambda_1+\lambda_2}\int_0^\infty \overline{T}^{LAS}(x)(\lambda_1 f_1(x) + \lambda_2 f_2(x))dx,$$

where

$$\overline{T}^{LAS}(x) = \frac{x + W_{x,x}}{1-\rho_x^{(1)}-\rho_x^{(2)}}.$$

The results are presented on Figures 5.5,5.6. For the results of V_2 we do not plot the mean sojourn time for the FCFS policy as class-2 has an infinite second moment. As one can see Gittins policy minimizes the mean sojourn time. In particular, it outperforms the LAS policy by almost $25-30\%$ when the system is loaded by around 90%. We note that surprisingly the PS policy produces much worse results than the LAS and Gittins policies.

5.4.7 Simulation results

We implemented Gittins policy algorithm for the case of two Pareto distributed classes with NS-2 simulator. The algorithm is implemented in the router queue. In the router we keep the trace of the attained service for the flows in the system. We keep the trace during some time interval after which there are no more packets from the flow in the queue.

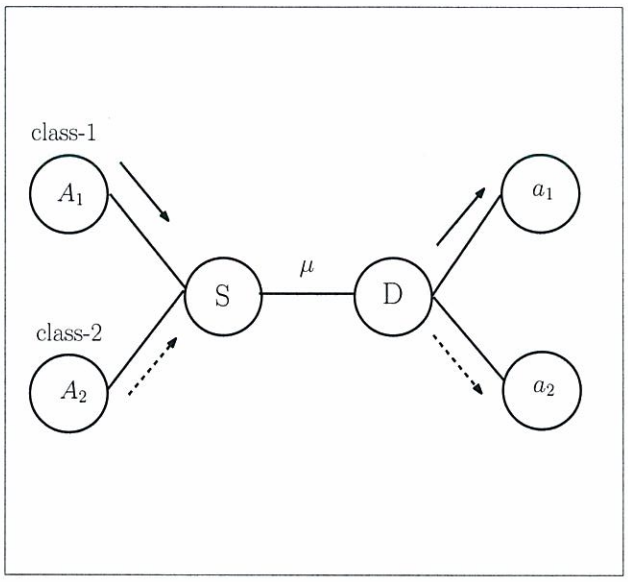

Figure 5.7: NS-2 simulation scheme.

It is possible to select the packet with the minimal sequence number of the flow which has to be served instead of selecting the first packet in the queue. In the current simulation this parameter does not play a big role according to the selected model scheme and parameters. (There are no drops in the system, so there are no retransmitted packets. Then all the packets arrive in the same order as they were sent.)

The algorithm which is used for the simulations is as follows:

Algorithm

on packet dequeue
select the flow f with the max $h_i(a_f)$, where
a_f is the flow's attained service
select the first packet p_f of the flow f in the queue
dequeue selected packet p_f
set $a_f = a_f + 1$

To compare Gittins policy with the LAS policy we also implemented LAS algorithm in the router queue. According to the LAS discipline the packet to dequeue is the packet from the flow with the least attained service.

The simulation topology scheme is given in Figure 5.7. The jobs arrive to the bottleneck router in two classes, which represent mice and elephants in the network. The jobs are generated by the FTP sources which are connected to TCP senders. The file size distributions are Pareto,

$F_i = 1 - b_i^{c_i}/(x+b_i)^{c_i}$, $i = 1, 2$. The jobs arrive according to the Poisson arrivals with rates λ_1 and λ_2. For the simulations we selected the scenario described in Subsection 5.4.5.

We consider that all connections have the same propagation delays. The bottleneck link capacity is μ. The simulation run time is 2000 seconds. We provide two different versions of parameters selection, which we call Vs_1 and Vs_2. In Vs_1 first class takes 25% of the total bottleneck capacity and in Vs_2 it takes 50%.

The parameters we used are given in Table 5.2.

Table 5.2: Two Pareto classes, simulation parameters

Ver.	c_1	c_2	m_1	m_2	ρ_1	ρ_2	ρ
Vs_1	10.0	1.25	0.5	6.8	0.25	0.50	0.75
Vs_2	10.0	2.25	0.5	4.5	0.50	0.37	0.87

The results are given in Table 5.3. We provide results for the NS-2 simulations and the values of the numerical mean sojourn times with the same parameters. We calculate the related gain of the Gittins policy in comparison with DropTail and LAS policies, $g_1 = \frac{\overline{T}^{DT} - \overline{T}^{Gitt}}{\overline{T}^{DT}}$ and $g_2 = \frac{\overline{T}^{LAS} - \overline{T}^{Gitt}}{\overline{T}^{LAS}}$

Table 5.3: Mean sojourn times

Ver.	$\overline{T}^{DT}$	$\overline{T}^{LAS}$	$\overline{T}^{Gitt}$	g_1	g_2
Vs_1 NS-2	18.72	2.10	2.08	88.89%	0.95%
Vs_1 theory	PS: 4.71	1.58	1.01	78.56%	36.08%
Vs_2 NS-2	6.23	2.03	1.83	70.63%	9.85%
Vs_2 theory	PS: 6.46	3.25	2.19	66.10%	32.62%

We found that with the NS-2 simulations the gain of the Gittins policy in comparison with LAS policy is not so significant when the small jobs do not take a big part of the system load. As one can see in Vs_2 when the class-1 load is 50% the related gain of the Gittins policy in comparison with LAS policy is 10%. In both versions the relative gain for the corresponding analytical system is much higher and reaches up to 36%. We explain this results with the phenomena related to the TCP working scheme.

5.4.8 Multiple Pareto classes

We consider a multi-class single server $M/G/1$ queue. The jobs arrive to the system in N classes. Jobs of i-th class, $i = 1, \ldots, N$ arrive according to the Poisson arrival processes with rates λ_i. Jobs size distributions are Pareto, namely

$$F_i(x) = 1 - \frac{1}{(x+1)^{c_i}}, \quad i = 1, \ldots, N.$$

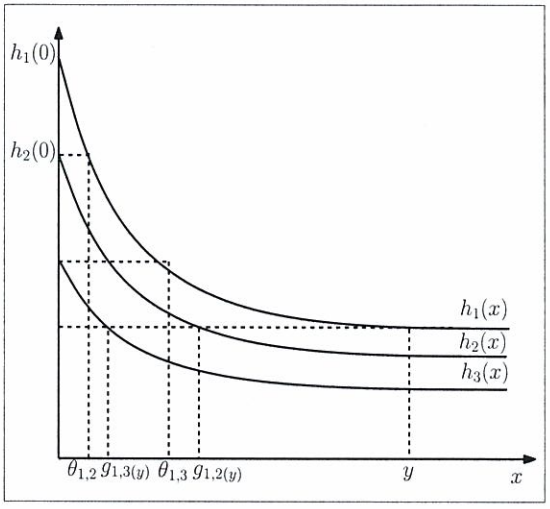

Figure 5.8: N Pareto classes, hazard rates

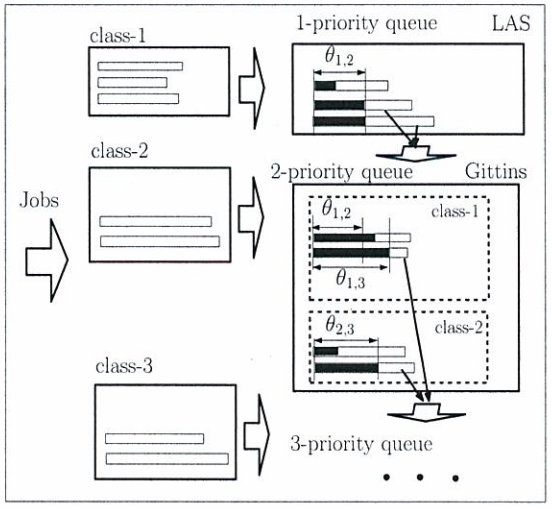

Figure 5.9: N Pareto classes, policy scheme

Then, the hazard rates

$$h_i(x) = \frac{c_i}{(x+1)}, \quad i = 1, \ldots, N,$$

never cross. Without loss of generality, let us consider that $c_1 > c_2 > \ldots > c_N$. Let us define the values of $\theta_{i,j}$ and $g_{i,j}(x)$, $i, j = 1, \ldots, N$ in the following way

$$h_i(\theta_{i,j}) = h_j(0),$$
$$h_i(x) = h_j(g_{i,j}(x)).$$

Then we get

$$g_{i,j}(x) = \frac{c_j}{c_i}(x+1), \quad \theta_{i,j} = \frac{c_i}{c_j} - 1.$$

Let us notice that $\theta_{k,i} < \theta_{k,i+1}$ and $\theta_{i,k} > \theta_{i+1,k}$, $k = 1, \ldots, N$, $i = 1, \ldots, N-1$, $i \neq k$, $i \neq k+1$, see Figure 5.8.

The optimal policy is the following. The scheme of the optimal policy is given on Figure 5.9.

Optimal policy.

There are N queues in the system. The class-1 jobs arrive to the system and go to the first-priority queue-1. There they are served until they get $\theta_{1,2}$ of service. Then they are moved to the queue-2, which is served only when the queue-1 is empty. In the queue-2 the job of class-1 are served with the jobs of class-2 with the Gittins policy. When the jobs of class-1 attain service $\theta_{1,3}$ they are moved to the queue-3. When the jobs of class-2 attain service $\theta_{2,3}$ they are also moved to the queue-3. And so on.

To find the expressions for the mean conditional sojourn times in the system we use the analysis which we used in interpretation of the mean conditional sojourn times expressions in the case of two class system, see Section. For job of every class its mean conditional sojourn time consists if the time which is spent to serve the job when the system is empty, the mean workload in the system which has to be done before the tagged job and the mean workload which arrives during the service time of the tagged job and has to be served before it.

Let the tagged job be from class-1 of size x. The jobs which have the same priority in the system and which have to be served before the tagged job are: class-1 jobs of size less than x, class-i jobs of size less than $g_{1,i}(x)$.

We denote $\overline{X_x^n}^{(i)}$ the n-th moment and $\rho_x^{(i)}$ the utilization factor for the distribution $F_i(x)$ of the class-i, $i = 1, \ldots, N$ truncated at x. The mean workload in the system which has to be done before the tagged job is then found with Pollaczek-Khinchin formula and equals to

$$W_{x,g_{1,2}(x),\ldots,g_{1,N}(x)} = \frac{\sum_{i=1}^{N} \lambda_i \overline{X^2_{g_{1,i}(x)}}}{2(1 - \sum_{i=1}^{N} \rho_{g_{1,i}(x)})}.$$

Then we formulate the theorem.

Theorem 5.3 *For the class-1 jobs of size x such as $\theta_{1,p} < x < \theta_{1,p+1}$, $p = 1, \ldots, N$ and corresponding class-k jobs with sizes $g_{1,k}(x)$, $k = 2, \ldots, N$ the mean conditional sojourn times are given by*

$$T_1(x) = \frac{x + W(x, g_{1,2}(x), \ldots, g_{1,p}(x))}{1 - \rho_1(x) - \rho_2(g_{1,2}(x)) - \ldots - \rho_p(g_{1,p}(x))},$$
$$T_k(g_{1,k}(x)) = \frac{g_{1,k}(x) + W(x, g_{1,2}(x), \ldots, g_{1,p}(x))}{1 - \rho_1(x) - \rho_2(g_{1,2}(x)) - \ldots - \rho_p(g_{1,p}(x))},$$

where $\theta_{1,p} < x < \theta_{1,p+1}$ and we consider that $\theta_{i,N+1} = \infty$, $i = 1, \ldots, N$.

Proof. It is similar to the proof of Theorem 5.2. ∎

5.5 Hyper-exponential and exponential classes

We consider two class $M/G/1$ queue. The jobs of each class arrive with the Poisson arrival process with rates λ_1 and λ_2. The job size distribution of class-1 is exponential with mean $1/\mu_1$, and hyper-exponential with two phases for class-2 with the mean $(\mu_3 p + (1-p)\mu_2)/(\mu_2\mu_3)$. Namely,

$$F_1(x) = 1 - e^{-\mu_1 x}, \; F_2(x) = 1 - pe^{-\mu_2 x} - (1-p)e^{-\mu_3 x}. \tag{5.12}$$

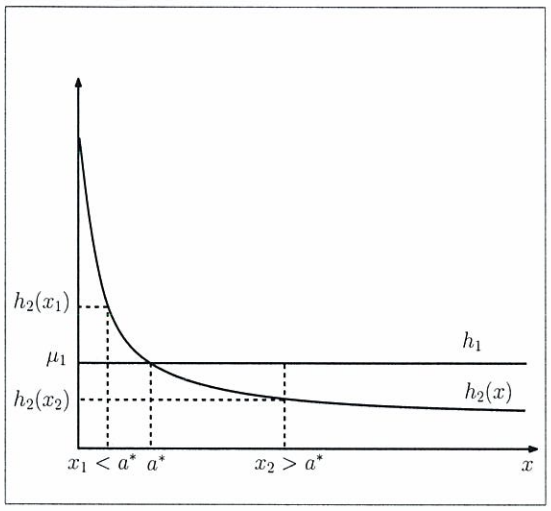

Figure 5.10: Exponential and HE classes, hazard rates.

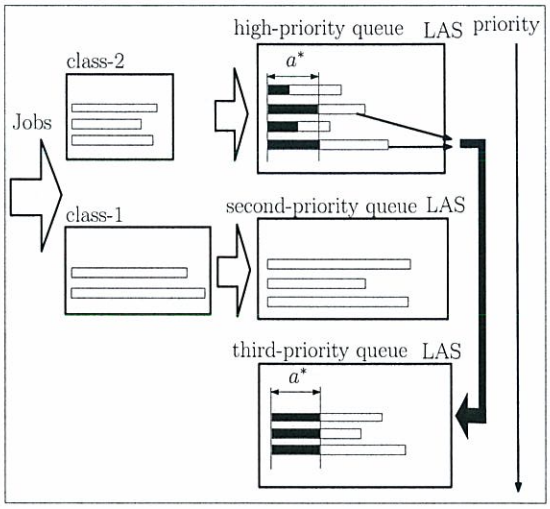

Figure 5.11: Exponential and HE classes, policy description.

Note that the hazard rates are

$$h_1(x) = \mu_1, \; h_2(x) = \frac{p\mu_2 e^{-\mu_2 x} + (1-p)\mu_3 e^{-\mu_3 x}}{pe^{-\mu_2 x} + (1-p)e^{-\mu_3 x}}, \; x \geq 0.$$

The hazard rate function of class-1 is a constant and equals to $h_1 = \mu_1$. The hazard rate function $h_2(x)$ of the class-2 is decreasing in x. As both hazard rate functions are non-increasing the optimal policy which minimizes the mean sojourn time is Gittins policy based on the value of the hazard function, which gives service to the jobs with the maximal hazard rate of the attained service.

For the selected job size distributions the hazard rate functions behave in different ways depending on parameters μ_1, μ_2, μ_3 and p. The possible behaviors of the hazard rate functions determine the optimal policy in the system. If the hazard rate functions never cross, the hazard rate of class-1 is higher than the hazard rate of class-2, then the class-1 jobs are served with priority to class-2 jobs. This happens when $h_1 > h_2(x)$, $x \in (0, \infty)$. As $h_2(x)$ is decreasing, then this happens when $\mu_1 > h_2(0)$. Let us consider that $\mu_2 > \mu_3$, then as $h_2(0) = p\mu_2 + (1-p)\mu_3$ and $\mu_1 > h_2(0)$ if $\mu_1 > \mu_2 > \mu_3$. For this case it is known that the optimal policy is a strict priority policy, which serves the class-1 jobs with the strict priority with respect to the class-2 jobs. From our discussion it follows that this policy is optimal even if $\mu_2 > \mu_1 > \mu_3$, but still $\mu_1 > p\mu_2 + (1-p)\mu_3$.

Let us consider the case when $\mu_2 > \mu_1 > \mu_3$ and $\mu_1 < p\mu_2 + (1-p)\mu_3$. Then it exists the unique point of intersection of $h_2(x)$ and h_1. Let us denote a^* the point of this intersection.

The value of a^* is the solution of $\frac{p\mu_2 e^{-\mu_2 x}+(1-p)\mu_3 e^{-\mu_3 x}}{pe^{-\mu_2 x}+(1-p)e^{-\mu_3 x}} = \mu_1$. Solving this equation, we get that

$$a^* = \frac{1}{\mu_2 - \mu_3} \ln\left(\frac{p}{1-p}\frac{\mu_2 - \mu_1}{\mu_1 - \mu_3}\right).$$

The hazard rate function scheme is given on Figure 5.10. Then, the optimal policy is the following.

5.5.1 Optimal policy.

There are three queues in the system, which are served with the strict priority between them. The second priority queue is served only when the first priority queue is empty and the third priority queue is served only when the first and second priority queues are empty. The class-2 jobs arrive to the system are served in the first priority queue with the LAS policy until they get a^* amount of service. After they get a^* amount of service they are moved to the third priority queue, where they are served according to the LAS policy. The class-1 jobs arrive to the system and go to the second priority queue, where they are served with LAS policy. Since $h_1(x) = \mu_1$, class-1 jobs can be served with any non-anticipating scheduling policy. The scheme of the optimal policy is given on Figure 5.11.

According to this optimal policy we find the expressions of the expected sojourn times for the class-1 and class-2 jobs.

5.5.2 Expected sojourn times

Let us recall that the mean workload in the system for the class-1 jobs of size less than x and class-2 jobs of size less than y is $W_{x,y}$ and is given by (5.6). We prove the following Theorem.

Theorem 5.4 *The mean conditional sojourn times in the $M/G/1$ queue with job size distribution given by (5.12) under Gittins optimal policy described in subsection5.5.1 are given by*

$$T_1(x) = \frac{x + W_{x,a^*}}{1 - \rho_x^{(1)} - \rho_{a^*}^{(2)}}, \quad x \in [0, \infty], \tag{5.13}$$

$$T_2(x) = \frac{x + W_{0,x}}{1 - \rho_x^{(2)}}, \quad x \in [0, a^*], \tag{5.14}$$

$$T_2(x) = \frac{x + W_{\infty,x}}{1 - \rho_\infty^{(1)} - \rho_x^{(2)}}, \quad y \in (a^*, \infty). \tag{5.15}$$

Proof. To find expressions of the mean conditional sojourn times we use the mean-value analysis and tagged job approach. The mean conditional sojourn time for the class-1 job of size x consists of the following elements.

- x, time need to serve the job itself.

- W, mean workload in the system which has to be served before the tagged job.
- $\overline{T}^*$, mean time to serve the jobs which arrive to the system during the service time of the current job and which have to be served before the tagged job.

When the tagged job is a class-1 job of size x the jobs which have to be served before it are all class-1 jobs of size x and all class-2 jobs of size less than a^*. Then the mean workload which the tagged job finds in the system and which has to be done before it is $W = W_{x,a^*}$. To find the mean work which arrive to the system during the service time of the tagged job, which is $T_1(x)$ we take into account only the jobs which have to be served before it. So, $\overline{T}^* = T_1(x)(\rho_x^{(1)} + \rho_{a^*}^{(2)})$.

For the tagged job of class-2 of size $x \leq a*$ the jobs which have to be served before it are the class-2 jobs of size less than x. Then the mean workload which the tagged job finds in the system and which has to be done before it is $W_{0,x}$ and the mean time to serve the jobs which arrive to the system during $\overline{T}_2(x)$ is $T_2(x)\rho_x^{(2)}$.

For the class-2 job of size $x > a^*$ the jobs which have to be served before it are all class-1 jobs and class-2 jobs of size less than x. Then the mean workload which has to be done before the tagged job is $W_{\infty,x}$ and the mean time spend to serve the jobs which arrive during the service time of the current job is $T_2(x)(\rho_\infty^{(1)} + \rho_x^{(2)})$.

Summarizing the results of the previous discussion we get

$$
\begin{aligned}
T_1(x) &= x + W_{x,a^*} + T_1(x)(\rho_x^{(1)} + \rho_{a^*}^{(2)}) \quad x \in [0, \infty], \\
T_2(x) &= y + W_{0,x} + T_2(x)\rho_x^{(2)}, \quad x \in [0, a^*], \\
T_2(x) &= y + W_{\infty,x} + T_2(x)(\rho_\infty^{(1)} + \rho_x^{(2)}) \quad x \in (a^*, \infty).
\end{aligned}
$$

from here we get the proof of the Theorem. ∎

5.5.3 Numerical results

Let us calculate numerically for some examples the mean sojourn time in the system when the Gittins policy is used. We consider two classes with the parameters given in Table 5.4. Also here $p = 0.1$ and the threshold value is $a^* = 7.16$. We compare the obtained results with the mean sojourn times when the system is scheduled with FCFS, PS and LAS policies, the results are given on Figure 5.12.

Table 5.4: Exponential and HE classes, simulation parameters

μ_1	μ_2	μ_3	m_1	m_2	ρ_1	ρ_2	ρ
0.6	1.0	0.5	1.6	1.1	0.1	0.4..0.85	0.5..0.95

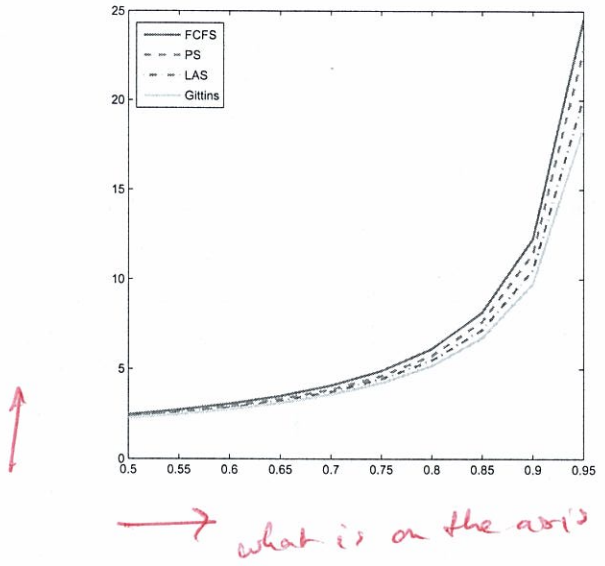

Figure 5.12: Exponential and HE classes, mean sojourn time

5.5.4 Pareto and exponential classes

We can apply the same analysis for the case when class-1 job size distribution is exponential and class-2 job size distribution is Pareto. Let us consider the case when the hazard rate functions of class-1 and class-2 cross at one point.

Let $F_1(x) = 1 - e^{-\mu_1 x}$ and $F_2(x) = 1 - b_2/(x+b_2)^{c_2}$. Then $h_1 = \mu_1$ and $h_2(x) = c_2/(x+b_2)$. The crossing point is $a^* = c_2/\mu_1 - b_2$. When $a^* \leq 0$ the hazard rate functions do not cross and then the optimal policy is to give strict priority to the class-1 jobs. If $a^* > 0$ then the hazard rate functions cross at one point and the optimal policy is the same as in the previous section. Then the expressions of the mean conditional sojourn timed of class-1 and class-2 are also (5.13), (5.14) and (5.15).

5.6 Conclusions

In [Git89] Gittins considered an $M/G/1$ queue and proved that the so-called Gittins index rule minimizes the mean delay. The Gittins rule determines, depending on the service times of jobs, which job should be served next. Gittins derived this result as a by-product of his groundbreaking results on the multi-armed bandit problem. Gittins' results on the multi-armed bandit problem have had a profound impact and it is extremely highly cited. However, and in despite of the big body of literature on scheduling disciplines in single server queues, Gittins work in the $M/G/1$ context has not received much attention.

In [AA07] authors showed that Gittins' policy could be used to characterize the optimal scheduling policy when the hazard rate of the service time distribution is not monotone. In the current work we have used Gittins' policy to characterize the optimal scheduling discipline in a multi-class queue. Our results show that, even though all service times have a decreasing hazard rate, the optimal policy can significantly differ from LAS, which is known to be optimal in the single-server case. We demonstrate that in particular cases surprisingly the PS has much worse performance than Gittins policy.

Using NS-2 simulator we implemented the Gittins' optimal policy in the router queue and provided simulations for several particular schemes. With the simulation results we found that Gittins policy can achieve 10% gain in comparison with LAS policy and provides much better performance than DropTail policy.

In future research we may consider other types of service time distributions. The applicability of our results in real systems like the Internet should also be more carefully evaluated. We also would like to investigate the conditions under which Gittins' policy gives significantly better performance than LAS policy.

5.7 Appendix: Proof of Theorem 2

We prove that the mean conditional sojourn times in the system described in Section 5.4 scheduled with the optimal Gittins policy given in Subsection 5.4.2 are given with (5.6), (5.7) and (5.8).

The class-1 jobs of size $x < \theta$ are served in the high priority queue with LAS policy, so the expression for the mean conditional sojourn time for this case is known, see [Kle76a, sec. 4.6], as is given by (5.6).

Let us consider class-1 jobs with sizes $x > \theta$ and class-2 jobs, which are served in the low priority queue. There are two queues in the system, each of them is served with LAS policy and there is a strict priority between the queues. The low priority queue is served only when the high priority queue is empty. Then the low priority queue is a queue with batch arrivals. To find the expressions of the mean conditional sojourn times in the system we use the analysis similar to the one of Kleinrock for Multi Level Processor Sharing queue in [Kle76a, sec. 4.7]. We formulate the following Lemma.

Lemma 5.1 *The mean conditional sojourn times for class-1 job of size x and for class-2 job of size $g(x)$ equal to*

$$T_1(x) = \frac{\theta + W_{\theta,0}}{1-\rho_\theta^{(1)}} + \frac{\alpha_1(x-\theta, g(x))}{1-\rho_\theta^{(1)}}, \tag{5.16}$$

$$T_2(g(x)) = \frac{W_{\theta,0}}{1-\rho_\theta^{(1)}} + \frac{\alpha_2(x-\theta, g(x))}{1-\rho_\theta^{(1)}}, \tag{5.17}$$

where $\alpha_1(x-\theta,g(x))/(1-\rho_\theta^{(1)})$ *and* $\alpha_2(x-\theta,g(x))/(1-\rho_\theta^{(1)})$ *are the times spent in the low priority queue by the class-*1 *and class-*2 *jobs respectively and equal to*

$$\alpha_1(x-\theta,g(x)) = \frac{x-\theta+A_1(x)+W_b}{1-\rho_b},$$
$$\alpha_2(x-\theta,g(x)) = \frac{g(x)+A_2(g(x))+W_b}{1-\rho_b},$$

where W_b *is the mean workload in the low priority queue which the tagged batch sees when arrive to the low priority queue,* ρ_b *is the mean load in the low priority queue and* $A_i(x)$, $i=1,2$ *are the mean works which arrive to the low priority queue with the tagged job in the batch.*

Proof. Let us consider that the tagged job is from class-1 and has a size $x>\theta$. The time it spends in the system consists of the mean time it spends in the high priority queue. This time is $\frac{\theta+W_{\theta,0}}{1-\rho_\theta}$ as it has to be served only with the class-1 jobs until it gets θ amount of service. After the tagged job is moved to the low priority queue after waiting while the high priority queue becomes empty. This is the time $\alpha_1(x)/(1-\rho_\theta^{(1)})$. The time $\alpha_1(x-\theta)$ is the time spent by the tagged job in the low priority queue. This time consists of the time spent to serve the job itself, $x-\theta$, of the mean workload in the low priority queue which the tagged job finds, W_b, of the mean work which arrives in the batch with the tagged job, $A_1(x)$ and of the mean work which arrive during the service time of the tagged job, $\alpha_1(x-\theta)\rho_b$.

We use the same analysis for the mean conditional sojourn time of the class-2 job of size $g(x)$. ∎

Now let us find the expressions for the W_b, ρ_b, $A_1(x)$ and $A_2(x)$. Let us define the truncated distribution $F_{1,\theta,x}(y)=F_1(y),\theta<y<x$ and $F_{1,\theta,x}(y)=0,y<\theta,y>x$. Let $\overline{X_{\theta,x}^n}^{(i)}$ be the n-th moment and $\rho_{\theta,x}^{(i)}$, $i=1,2$ be the utilization factor for this truncated distribution. We use this notation because the jobs of class-1 which find themselves in a batch are already served until θ.

Let N_i be the random variable which denotes the number of jobs in a batch of class-i, $i=1,2$. We define $X_{\theta,x}^{(1)}$ as the random variable which denotes the size of class-1 job in a batch. Let $X_{g(x)}^{(2)}$ be the random variable which corresponds to the size of the class-2 job in a batch. Then

$$Y_b = \sum_{i=1}^{N_1} X_{i,\theta,x}^{(1)} + \sum_{i=1}^{N_2} X_{i,g(x)}^{(2)},$$

is the random variable which denotes the size of the batch. Let us denote as λ_b the batch arrival rate. We know that $\lambda_b=\lambda_1+\lambda_2$. According to the previous notations we can write

$$\rho_b = \lambda_b E[Y_b],$$

here $E[Y_b]$ is the mean work that a batch brings and by Pollaczek-Khinchin

$$W_b = \frac{\lambda_b E[Y_b^2]}{2(1-\rho_b)}.$$

Let us note that W_b does not depend from which class the tagged job comes. As we know the first and the second moments of $X^{(1)}_{\theta,x}$, $X^{(2)}_{g(x)}$, to find ρ_b and W_b we need to know the first and the second moments of N_i, $i = 1, 2$. To find this values we we use the method of the Generating functions, which is described in the following section.

5.7.1 Generating function calculation

Lemma 5.2 *The Generating function equals to*

$$\begin{aligned} G(z_1, z_2) &= \frac{\lambda_1}{\lambda_b}\Big(\int_0^\theta e^{-\lambda_1 x(1-G_1(z_1,z_2))-\lambda_2 x(1-z_2)} dF_1(x) + \\ &+ z_1 e^{-\lambda_1\theta(1-G_1(z_1,z_2))-\lambda_2\theta(1-z_2)}\overline{F}_1(\theta)\Big) + \frac{\lambda_2}{\lambda_b} z_2. \end{aligned} \tag{5.18}$$

Proof. We propose a two dimension generating function $G(z_1, z_2)$, which we obtain using collective marks method. The method of the collective marks is described in [Kle76b, Ch. 7].

Let us mark the jobs in a batch in the following way. We mark the job of class-1 with a probability $1 - z_1$, then z_1 is a probability that the job of class-1 is not marked. The same is defined for the jobs of class-2 as z_2. Let p_{n_1,n_2} be the probability that n_1 class-1 and n_2 class-2 jobs arrive in the batch. Then

$$G(z_1, z_2) = \sum_{n1}\sum_{n2} z_1^{n_1} z_2^{n_2} p_{n_1,n_2}$$

is a generation function and it gives a probability that there are no marked jobs in the batch.

Let us define as a "starter" or S a tagged job. Let us distinguish the cases when the starter S belongs to class-1 or class-2 and denote by $G_1(z_1, z_2)$ and $G_2(z_1, z_2)$ the probabilities that there are no marked jobs in the batch if the starter is from the class-1 and class-2. When the $S \in$ class-1, we consider two cases depending on the size of the starter ($S \leq, > \theta$). Then

$$G(z_1, z_2) = \frac{\lambda_1}{\lambda_b}([G_1(z_1, z_2), S \leq \theta] + [G_1(z_1, z_2), S > \theta]) + \frac{\lambda_2}{\lambda_b} G_2(z_1, z_2).$$

Let us calculate $G_1(z_1, z_2)$. When the class-1 job arrives to the system it creates the busy period. Still this job does not receive θ amount of service the low priority queue is not served. So, the jobs which arrive to the low priority queue and the jobs which are already in the low priority queue are waiting and so they create a batch. The probability that there are no marked job in this batch is $G_1(z_1, z_2)$.

Let the class-1 job of size x arrive to the system. Let $x \leq \theta$. The probability that k_1 class-1 jobs arrive in the period $(0, x)$ is $P_1(x) = e^{-\lambda_1 x}(\lambda_1 x)^{k_1}/k_1!$. The probability that all the batches generated by this arrived k_1 jobs of class-1 is $G_1(z_1, z_2)^{k_1}$, because each of them generates the batch which does not have marked jobs with probability $G_1(z_1, z_2)$. During time

$(0,x)$ the probability that k_2 class-2 jobs arrive to the system is $P_2(x) = e^{-\lambda_2 x}(\lambda_2 x)^{k_2}/k_2!$. The probability that this jobs are not marked is not included in $G_1(z_1,z_2)$ and equals to $z_2^{k_2}$. Then we summarize on k_1 and k_2, integrate on x in $(0,\theta)$ with $dF_1(x)$, as only the class-1 jobs generate busy periods. We get that the probability that there are no marked jobs in the batch is

$$\begin{aligned}[G_1(z_1,z_2), S \le \theta] &= \int_0^\theta \left(\sum_{k_1=0}^{\infty} P_1(x) G_1(z_1,z_2)^{k_1} P_2(x) z_2^{k_1} \right) dF_1(x) = \\ &= \int_0^\theta e^{-\lambda_1 x(1-G_1(z_1,Z_2))-\lambda_2 x(1-z_2)} dF_1(x).\end{aligned}$$

Let class-1 job of size $x > \theta$ arrive to the system. The class-1 job is first served in the high priority queue until it gets θ of service. Then it is moved to the low priority queue. The probability that k_1 class-1 jobs arrive in the period $(0,\theta)$ is $P_1(\theta) = e^{-\lambda_1\theta}(\lambda_1\theta)^{k_1}/k_1!$. The probability that there are no marked jobs in all the batches generated by this arrived k_1 class-1 jobs is $G_1(z)^{k_1}$. The probability that k_2 class-2 jobs arrive to the system in the period $(0,\theta)$ is $P_2(\theta) = e^{-\lambda_2\theta}(\lambda_2\theta)^{k_2}/k_2!$. The probability that all this jobs are not marked is z^{k_2}.

We have to take into account the "starter" itself, as it has the size more than θ and it comes in the batch. The probability that the starter is not marked is z_1. Then we summarize on k_1 and k_2, integrate on x on (θ,∞) with $dF_1(x)$, as only the class-2 jobs generate busy periods. We get

$$\begin{aligned}[G_1(z_1,z_2), S > \theta] &= \int_\theta^\infty \left(\sum_{k_1=0}^{\infty} P_1(\theta) G_1(z_1,z_2)^{k_1} z_1 P_2(\theta) z_2^{k_1} \right) dF_1(x) = \\ &= z_1 e^{-\lambda_1\theta(1-G_1(z_1,z_2))-\lambda_2\theta(1-z_2)} \overline{F}_1(\theta).\end{aligned}$$

Let us find $G_2(z_1,z_2)$. When the job of the second class arrives to the system it generates the batch of size one, then the probability that the jobs of this batch are not marked is z_2. Then $G_2(z_1,z_2) = z_2$.

$$[G_2(z_1,z_2)] = \int_0^\infty z_2 dF_2(x) = z_2.$$

Finally

$$G(z_1,z_2) = \frac{\lambda_1}{\lambda_b} G_1(z_1,z_2) + \frac{\lambda_2}{\lambda_b} G_2(z_1,z_2),$$

and we get (5.18). Let us notice that $G(1,1) = 1$. ∎

Now we can calculate $E[N_1]$, $E[N_2]$ and so ρ_b and W_b. After some mathematical calculations we get the following result.

Lemma 5.3

$$\rho_b = 1 - \frac{1 - \rho_x^{(1)} - \rho_{g(x)}^{(2)}}{1 - \rho_\theta^{(1)}},$$

$$W_b = W_{x,g(x)} - W_\theta(1 + \rho_b) - \theta\frac{\rho_x^{(1)} - \rho_\theta^{(1)}}{1 - \rho_\theta^{(1)}}.$$

Proof. We use the following equations. For $i = 1, 2$

$$E[N_i] = \frac{\partial G(z_1, z_2)}{\partial z_i}|_{1,1}$$

$$E[N_i(N_i - 1)] = E[N_i^2] - E[N_i] = \frac{\partial^2 G(z_1, z_2)}{\partial z_i^2}|_{1,1},$$

$$E[N_1 N_2] = \frac{\partial^2 G(z_1, z_2)}{\partial z_1 \partial z_2}|_{1,1}.$$

Using $b_i = \frac{E[N_i^2]}{E[N_i]} - 1$ after some mathematical calculations we obtain the result of the current Lemma. ∎

Now let us find expressions for $A_1(x)$ and $A_2(x)$.

Lemma 5.4 *The mean workload which comes with the tagged job of class-1 of size x in the batch and has to be done before it equals to*

$$A_1(x) = 2(W_\theta + \theta)\rho_b - \theta\frac{\rho_{g(x)}^{(2)}}{1 - \rho_\theta^{(1)}}.$$

Proof. The term $A_1(x)$ is the work that arrives with the tagged job of class-1 of size x and that gets served before its departure. Since the tagged job arrives from class-1 only when the batch is started by a class-1 job, the calculations now will depend on $G_1(z_1, z_2)$. We denote $b_{1|1}$ and $b_{2|1}$ the mean number of jobs of class-1 and class-2 which arrive in the batch with the tagged job of class-1 when the batch is initiated by a class-1 job. Then

$$A_1(x) \quad = \quad b_{1|1}E[X_{\theta,x}^{(1)}] + b_{2|1}E[X_{g(x)}^{(2)}] - E[X_{\theta,x}^{(1)}].$$

Here

$$b_{1|1} = \sum_{n_1} n_1 \frac{n_1 P(n_1)}{E[N_{1|1}]} = \frac{E[N_{1|1}^2]}{E[N_{1|1}]},$$

where $N_{1|1}$ is the random variable which corresponds to the number of jobs of class-1 in the batch when the batch is initiated by the class-1 job. So the number of class-1 jobs that arrive

in addition to the tagged job is $\left(\frac{E[N_{1|1}^2]}{E[N_{1|1}]} - 1\right)$. Note that since we condition on the fact that the starter is a class-1 job, $N_{1|1}$ is now calculated from $G_1(z_1, z_2)$ so:

$$E[N_{1|1}] = \frac{\partial G_1(z_1, z_2)}{\partial z_1}|_{1,1},$$
$$E[N_{1|1}(N_{1|1} - 1)] = \frac{\partial^2 G_1(z_1, z_2)}{\partial z_1 \partial z_1}|_{1,1}.$$

Then we can find $(b_{1|1} - 1)$. Now we need to calculate $b_{2|1}$, that is, the mean number of class-2 jobs that the tagged job of class-1 job see. We have that from the Generating function $G_1(z_1, z_2)$ by conditioning on the number of class-1 jobs:

$$G_1(z_1, z_2) = \sum_{n1}\sum_{n2} z_1^{n_1} z_2^{n_2} p_{n_1,n_2} = \sum_{n1}\sum_{n2} z_1^{n_1} z_2^{n_2} p_{n_2|n_1} p_{n_1},$$
$$\frac{\partial^2 G_1(z_1, z_2)}{\partial z_1 \partial z_2}|_{1,1} = E[N_1] \sum_{n1}\sum_{n2} n_2 p_{n_2|n_1} \frac{n_1 p_{n_1}}{E[N_1]} = E[N_1] b_{2|1}.$$

Then we can calculate $b_{2|1}$

$$b_{2|1} = \frac{1}{E[N_{1|1}]} \frac{\partial^2 G_1(z_1, z_2)}{\partial z_1 \partial z_2}|_{(1,1)}.$$

Finally we find the expression for $A_1(x)$. ∎

Lemma 5.5 *The mean workload which comes with the tagged job of class-2 of size $g(x)$ in the batch and has to be done before it equals to*

$$A_2(g(x)) = 2(W_\theta + \theta)\rho_b - \theta \frac{\rho_{g(x)}^{(2)}}{1 - \rho_\theta^{(1)}} - \theta\rho_b.$$

Proof. The term $A_2(g(x))$ is the work that arrives with the tagged job of size $g(x)$ of class-2 and that gets served before its departure. When the tagged job arrives from class-2 the batch can be started by a class-1 or by a class-2 job, so the calculations depend on $G(z_1, z_2)$. We denote $b_{1|2}$ and $b_{2|2}$ the mean number of jobs of class-1 and class-2 which arrive in the batch with the tagged job of class-2. Then

$$\begin{aligned} A_2(g(x)) &= b_{1|2} E[X_{\theta,x}^{(1)}] + b_{2|2} E[X_{g(x)}^{(2)}] - E[X_{g(x)}^{(2)}] = \\ &= b_{1|2} E[X_{\theta,x}^{(1)}] + (b_{2|2} - 1) E[X_{g(x)}^{(2)}]. \end{aligned}$$

As the tagged job is from class-2, then $b_{2|2} = b_2$. We need to find the value of $b_{1|2}$. We use the fact that the jobs of class-1 and class-2 arrive independently from each other.

$$G(z_1, z_2) = \sum_{n1}\sum_{n2} z_1^{n_1} z_2^{n_2} p_{n_1,n_2} = \sum_{n1}\sum_{n2} z_1^{n_1} z_2^{n_2} p_{n_1|n_2} p_{n_2}$$

$$\frac{\partial^2 G(z_1, z_2)}{\partial z_1 \partial z_2}|_{1,1} = E[N_2] \sum_{n1} \sum_{n2} n_1 p_{n_1|n_2} \frac{n_2 p_{n_2}}{E[N_2]} = E[N_2] b_{1|2}.$$

Then

$$b_{1|2} = \frac{1}{E[N_2]} \frac{\partial^2 G(z_1, z_2)}{\partial z_1 \partial z_2}|_{1,1}.$$

From here we get the expression for $A_2(g(x))$. ∎

Now we can prove the result of Theorem 5.2.

Lemma 5.6 *Expressions (5.16), (5.17) and (5.7), (5.8) are equal.*

Proof. After simplification of the expressions (5.16), (5.17) we get equations (5.7), (5.8). ∎

Chapter 6

Improving TCP Fairness with the MarkMax Policy

6.1 Summary

We introduce MarkMax a new flow-aware AQM algorithm for Additive Increase Multiplicative Decreases protocols (like TCP). MarkMax sends a congestion signal to a selected connection whenever the total backlog reaches a given threshold. The selection mechanism is based on the state of large flows. Using a fluid model we derive some bounds that can be used to analyze the behavior of MarkMax and we compute the per-flow backlog. We conclude the chapter with simulation results, using NS-2, comparing MarkMax with Drop Tail and showing how MarkMax improves both the fairness and link utilization when connections have significantly different RTTs.

The results of this work are published in [OBA08].

6.2 Introduction

It has been known for a long time that if TCP connections with different RTTs share a bottleneck link, TCP connections with smaller RTTs take a larger share of the bandwidth [Flo91, Man90]. In [LM97] the authors have observed that under synchronization assumptions a TCP connection obtains a share of the link capacity proportional to RTT^{α} with $1 < \alpha < 2$. In [Bro00] the author has used a fluid approximation to derive a more rigorous model for the case when connections have different RTTs. Then, in [ABL+00] it was observed that in the case of not complete synchronization and, especially when RED [FJ93] is used, the distribution of the link capacity is more fair. In particular, the experiments of [ABL+00] have suggested that a TCP connection obtains a share of the link capacity proportional to $RTT^{0.85}$. This was later justified by an analytical model for the case of two competing TCP connections [AJN02]. In [AART06] the authors have used a fluid model to analyze what happens if only one connection reduce its sending rate when multiple connections share the same bottleneck link but they have ignored backlog dynamics: whenever the total arrival rate at the bottleneck link is equal to its capacity one of the connection reduces its sending rate, so that the backlog is always zero. In [SS07] the authors have proposed an MLC(l) AQM algorithm to approach maxmin fairness. In particular, for $l = 1$ the MLC(l) algorithm performs similar to RED and for $l = 2$ the MLC(l) algorithm performs similar to CHOKe [PPP00]. The authors of [SS07] argue that by choosing a significantly large parameter l one can be arbitrary close to the maxmin fairness. The present work indicates that this does not appear to be the case.

Building upon [AFG06, AART06] and [Sta07] we propose a new flow-aware active queue management packet dropping scheme (MarkMax). The main idea behind MarkMax is to identify which connection should reduce its sending rate instead of which packets should be dropped. To improve fairness we propose to cut flows with the largest sending rate during the congestion moments. Several AQM schemes previously proposed do not discriminate between flows. Typically they drop every incoming packet with a certain probability that is a function of the state of the queue.

When AQM was first introduced in the 1990s it was unfeasible to classify incoming packets in real time for high speed links but with technological advances this is now possible. Furthermore, to reduce the numbers of flows that need to be tracked, it is possible to concentrate on the larger flows using the heavy-hitter counters of [Sta07] to identify large flows. Then, according to [AABN04] we suggest to treat short flows with priority and mark large flows which have the largest backlog during the congestion moments. We also suggest to use ECN [RFB01] to minimize the number of dropped packets.

The chapter is organized as follows: In the next Section 6.3 we specify the algorithm. Then, in Section 6.4 we perform its theoretical analysis. We conclude the chapter with a section on

NS-2 simulations illustrating the performance of MarkMax.

6.3 The MarkMax algorithm

The algorithm has three parameters: the thresholds θ, θ_l, θ_h, selected in such a way that $\theta_l < \theta < \theta_h$. The threshold θ acts as a "trigger," whenever the queue size is above this value one connection is cut. We propose two different ways of selecting which connection to cut, as described later on. The other two thresholds are needed because we are dealing with a packet based system with non-zero propagation and queueing delays.

Let q be the queue size and *flag* be a Boolean variable initialized to TRUE. The following algorithm is executed every time a new packet arrives:

enqueue packet
if $q \leq \theta_l$ or $q \geq \theta_h$
  then *flag* $\leftarrow$ TRUE
if $q \geq \theta$ and *flag*=TRUE
  then a. select connection with MarkMax-B (full backlog based MarkMax) or MarkMax-T (backlog tail based MarkMax)
  b. set the ECN flag in the first packet of the selected connection from the head of the queue
  c. *flag* $\leftarrow$ FALSE

The θ_h and θ_l thresholds are used do determine whether a congestion signal should be sent or not, if $q \geq \theta$. After a congestion signal is sent the algorithm will not send another one as long as the queue remains in the interval $[\theta_l, \theta_h]$.

The θ_h threshold acts as a safety mechanism covering the cases when a single cut in the sending rate of the selected connection might not be enough to reduce the total arrival rate to a value smaller than the capacity of the outgoing link. Whenever the queue size is above θ_h we keep sending congestion signals to the selected connection. This does happen especially during the slow start phase.

Given that the system has non-zero propagation and queueing delays whenever we set the ECN bit of a certain connection we need to wait for the sender to receive the corresponding acknowledgment before it reduces its sending rate. Before such reduction is noticeable at the bottleneck link we still need to wait for the propagation and queueing delay between the sender and the bottleneck link. During this time the sending rate and the queue will keep growing so that, at the bottleneck link, it is not immediately possible to conclude whether a single cut is enough or not. Clearly if we set θ_h too high the system will respond slowly, whenever one

cut is not enough, and the queue will be larger. On the other hand if we set θ_h too close to θ unnecessary multiple cuts can take place.

The lower threshold θ_l is needed because the queue size can oscillate around θ, due to the arrival and departure of single packets and to the bursty nature of the arrival flows. If the queue size is close to θ the threshold can be crossed multiple times, so if we use only one threshold θ this could generate multiple congestion signals, potentially causing the sender to reduce its sending rate multiple times[1]. Furthermore it could happen that different connections are selected, causing, again, multiple and unnecessary cuts. Because of these oscillations using θ_l is the only way to determine whether the selected connection has reacted to the congestion signal.

Even if a single cut is enough to reduce the total sending rate to a value smaller than the capacity of the outgoing link the additive increase aspect of TCP will increase the sending rate again so that the backlog will, eventually, start to increase again. Clearly if we set θ_l too low the backlog might never reach it forcing the algorithm to use only the θ_h threshold and to send multiple spurious congestion signals.

In the next section we use a fluid approximation to further discuss the selection of θ and θ_h. Based on the simulations we run it looks reasonable to suggest that θ_h and θ_l can be set as follows: $\theta_h = 1.15 \cdot \theta$ and $\theta_l = 0.85 \cdot \theta$.

After enqueueing the arriving packet the algorithm sets the *flag* variable to TRUE if the queue size has grown too large or has sufficiently decreased. In both cases the queue size is sufficiently far from θ so that we should send a new congestion signal if $q \geq \theta$. This is done by the last **if** statement: at first a connection is selected, then the ECN flag is set in the first packet from the head of the queue of the selected connection. Finally the *flag* is set to FALSE to indicate the fact that one congestion signal has already been sent.

We propose two different criteria for selecting the connection to be cut: MarkMax-B selects the connection with the biggest (per connection) backlog and MarkMax-T selects the connection with the biggest backlog in the final part of the queue (the tail). As such the MarkMax-T variant has one extra parameter, expressed as a percentage, indicating the portion of the queue that will be considered.

The per connection backlog is related to the sending rate of each connection. Clearly a larger sending rate will result in larger backlog. More precisely the connection with the biggest backlog is the connection with the largest average sending rate since the beginning of the current busy period. Larger values of θ and corresponding larger queues lead to a larger averaging window, basically increasing the "memory" of the system. The idea behind MarkMax-T is to reduce the averaging window in order to identify the connection with the biggest instantaneous rate.

[1]The ECN specification does mention that senders should reduce the sending rate only once per round trip time, but this is not enough to guarantee that multiple cuts will not take place if we mark multiple packets.

6.4 Fluid model

Consider N TCP connections sharing a single bottleneck link with capacity μ. Let RTT_i be the round trip time of the i-th connection ($i = 1, \ldots, N$) and $\lambda_i(t)$ its sending rate at time t. We approximate the behavior of the system using a fluid model. Data is represented by a fluid that flows into the buffer with rate $\lambda(t) = \sum_i \lambda_i(t)$, and it leaves the buffer with rate μ if there is a non-zero backlog. Fluid models have been successfully used to model TCP connections. In [AAB00] it is shown that such a model adequately describes the behavior of a TCP connection, provided the average sending rate is large enough.

As in [AAP05] we assume that, between congestion signals, senders increase their sending rate linearly. If at time t_0 the sending rate of the i-th sender is $\lambda_{0,i}$ then at time $t > t_0$ its sending rate is $\lambda_i(t) = \lambda_{0,i} + \alpha_i t$, where $\alpha_i = 1/(RTT_i)^2$. For the sake of simplicity we assume that RTT_i is a constant, as if it often done (see, for example, [ABN+95, SZC90, AAP05]).

It is not too hard to see that, if at time t_0 the sending rates are $\lambda_{0,i}$ and the total backlog is x_0, the backlog $x(t)$ is given by:

$$x(t) = x_0 + (\lambda_0 - \mu)t + \frac{\alpha}{2}t^2. \tag{6.1}$$

Where $\lambda_0 = \sum_i \lambda_{0,i}$ and $\alpha = \sum_i \alpha_i$, provided x_0 and λ_0 are such that $x_0 \geq \frac{(\lambda_0-\mu)^2}{2\alpha}$. If $x_0 < \frac{(\lambda_0-\mu)^2}{2\alpha}$ and $\lambda_0 < \mu$ then, after a decreasing phase, the buffer will be empty for a certain time and will finally start increasing again. In this case

$$x(t) = \begin{cases} x_0 + (\lambda_0 - \mu)t + \frac{\alpha t^2}{2}, & \text{if } t \leq t_1 \\ 0, & \text{if } t_1 \leq t \leq \frac{\mu-\lambda_0}{\alpha} \\ \frac{\alpha}{2}\left(t - \frac{\mu-\lambda_0}{\alpha}\right)^2, & \text{if } t > \frac{\mu-\lambda_0}{\alpha} \end{cases} \tag{6.2}$$

where $t_1 = \frac{\mu-\lambda_0-\sqrt{(\mu-\lambda_0)^2-2\alpha x_0}}{\alpha}$.

Solving $\lambda(t) = \lambda_0 + \alpha t$ for t and substituting in (6.1) we have that

$$x(\lambda) = \frac{\lambda^2}{2\alpha} - \frac{\lambda\mu}{\alpha} + x_0 + \frac{\mu\lambda_0}{\alpha} - \frac{\lambda_0^2}{2\alpha} \tag{6.3}$$

provided $x_0 \geq \frac{(\lambda_0-\mu)^2}{2\alpha}$. A similar expression can be obtain substituting the value of t in (6.2).

Figure 6.1 shows some of the possible trajectories of the system. Note that all these parabolas have the same shape in the sense that as x_0 and λ_0 vary the only thing that changes is the height of the vertex on the $\lambda = \mu$ line.

One possible way of adapting the MarkMax algorithm to the fluid case is as follows: every time the total backlog $x(t)$ reaches θ we can "send a congestion signal" to the corresponding connection by multiplying its sending rate by β ($0 < \beta < 1$). Throughout the chapter we will

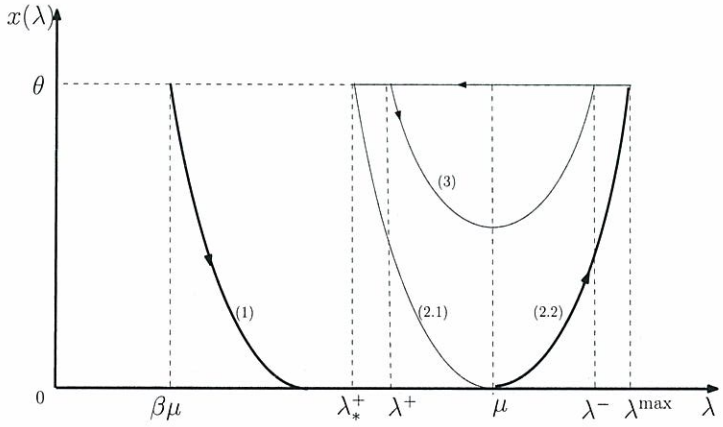

Figure 6.1: Some of the possible trajectories in the state space

use $\beta = 1/2$ (to model TCP New Reno) unless otherwise stated. The two selection methods previously discussed can easily be adapted as well: for MarkMax-B we select the connection with the biggest backlog, while for MarkMax-T we pick the connection with the biggest instantaneous sending rate. Recall that the idea behind MarkMax-T was exactly this and, with the fluid model, we know $\lambda_i(t)$ exactly so there is no need to approximate it.

To simplify the analysis, unless otherwise specified, we assume that the source reacts immediately to the congestion signals. Combining this with the fact that we know the sending rate after a cut and the are no short term oscillations in the queue size, it suffices to use only one threshold (θ). As a consequence whenever the backlog reaches θ the chosen connection, say j, immediately changes its rate to $\beta\lambda_j$. If $\sum_{i\neq j} \lambda_i + \beta\lambda_j > \mu$ (that is the arrival rate is still greater than μ) the procedure is repeated by selecting a new connection to cut (it can be the same one or not, depending on the specific case) until the total sending rate is less than μ. For the MarkMax-T version this procedure is guaranteed to terminate: eventually all connections will be cut. While for MarkMax-B this is not the case: if multiple cuts are needed the algorithm will always pick the same connection. As there is no feedback delay the backlog does not change. If the sum of the rates of the other connections is greater than μ even an infinite number of cuts will not suffice and the algorithm will not terminate. Given that this happens only in the fluid model and only for very large (and unrealistic) values of θ we decided not to address the problem.

It is worth noting that using this fluid model it is also possible to exactly compute the per connection backlog, at any given time t, using an approach based on network calculus [Cru91]. Let $R_{i,\text{in}}(t)$ be the total amount of traffic sent by the i-th connection until time t (this is generally called a "process" in network calculus), that is $R_{i,\text{in}}(t) = \int_0^t \lambda_i(u)du$. Similarly let $R_{i,\text{out}}(t)$ be the total amount of traffic of connection i that has left the buffer until time t. Clearly the

backlog at time t is $x_i(t) = R_{i,\text{in}}(t) - R_{i,\text{out}}(t)$ so that we need to compute $R_{i,\text{in}}(t)$ and $R_{i,\text{out}}(t)$ to find $x_i(t)$. Let $t_1, \ldots, t_n$ be the times at which a congestion signal was sent (to any of the connections). Between two congestion signals, say t_j and t_{j+1}, we know that if $\lambda_i(t) = \lambda_{i,j} + \alpha_i t$ then $R_{i,\text{in}}(t) = S_{i,j} + \lambda_{i,j}(t - t_j) + \frac{\alpha_i}{2}(t - t_j)^2$ where $\lambda_{i,j} \triangleq \lambda_i(t_j)$ and $S_{i,j} \triangleq R_{i,\text{in}}(t_j)$. This way we can also compute $R_{i,\text{in}}(\tau)$ for any $\tau \leq t$.

To compute $R_{i,\text{out}}(t)$ we can take advantage of the fact that we are dealing with a fluid FCFS queue with continuous inputs (the arrival rate is bounded) so that the delay for all the bits exiting at time t is the same and it is equal $d(t) = \inf\{u \geq 0 | R_{\text{in}}(t-u) \leq R_{\text{out}}(t)\}$ where $R_{\text{in}}(t) = \sum_i R_{i,\text{in}}(t)$ and $R_{\text{out}}(t) = \sum_i R_{i,\text{out}}(t)$. This implies that $R_{i,\text{out}}(t) = R_{i,\text{in}}(t - d(t))$. As we know $R_{i,\text{in}}(\tau)$ for any $\tau \leq t$ we only need to find $d(t)$ to compute $R_{i,\text{out}}(t)$.

Let $v \triangleq t - d(t)$, that is the bits that are exiting at time t joined the queue at time v. We can find v exploiting the fact that $R_{i,\text{in}}(v) = R_{\text{out}}(t)$ and that $R_{\text{out}}(t) = \mu(t-u)$, where u is the beginning of the system busy period containing t and can be found because $R_{i,\text{in}}(\tau)$ is known for all $\tau \leq t$. We also have that, if $t_k \leq \tau \leq t_{k+1}$:

$$R_{\text{in}}(\tau) = S_k + \lambda_0(\tau - t_k) + \frac{\alpha}{2}(\tau - t_k)^2, \tag{6.4}$$

where $k \triangleq \max\{j | S_j < S\}$ and $S_j \triangleq \sum_i S_{i,j}$. That is the traffic exiting at time t entered in the buffer between t_k and t_{k+1}. As $t_k \leq v \leq t_{K+1}$ we can use (6.4) to solve $R_{\text{in}}(v) = \mu(t-u)$ for v and finally compute $d(t) = t - v$. Knowing $d(t)$ we can use $R_{i,\text{out}}(t) = R_{i,\text{in}}(t - d(t))$ to compute $x_i(t) = R_{i,\text{in}}(t) - R_{i,\text{out}}(t)$.

Using this method we wrote a simulator for the fluid model (in Python) that implements both variants of MarkMax. Using this simulator we have noticed that, provided the value of θ is not too big, MarkMax-B and MarkMax-T behave in a very similar way. In the remainder of this section we present some results that can be derived using the fluid model.

6.4.1 Guideline bounds

Let t_θ be such that $x(t_\theta) = \theta$. Let λ^- and λ^+ be, respectively, the total sending rate before and after the cut(s) at time t_θ. Let

$$g(\lambda) = \begin{cases} 0, & \text{if } \lambda \leq \mu \\ \frac{(\lambda-\mu)^2}{2\alpha}, & \text{if } \lambda \geq \mu \end{cases},$$

(marked as (2.2) in Figure 6.1) and let $A = \{(\lambda, x) | x > g(\lambda)\}$. It is easy to verify that if $(\lambda_0, x_0) \in A$, then any trajectory starting at (λ_0, x_0) stays in A. Furthermore, given that we send the congestion signal(s) whenever $x(t) = \theta$ and that there is no feedback delay, the maximum rate $\lambda^{\max}$ corresponds to intersection between $g(\lambda)$ and $x = \theta$ in Figure 6.1. It is easy to see that $\lambda^{\max} = \mu + \sqrt{2\alpha\theta}$.

Clearly all the trajectories described by (6.3) intersect the $x = \theta$ line twice, once to the left of the $\lambda = \mu$ line and once to the right. Only the intersection points to the right correspond to an increasing backlog phase so that λ^- is always between μ and $\lambda^{\max}$. We can also bound λ^+: as we keep sending congestion signals until the arrival rate is less than μ we have $\lambda^+ \leq \mu$. The fact that $\lambda^- \geq \mu$ implies that λ^+ cannot be smaller than $\beta\mu$ (this happens when $\lambda^+ = \mu$ and either there is only one connection or, in the case of multiple connections, the biggest one is significantly bigger than the others). Combining all this we have:

$$\mu \leq \lambda^- \leq \mu + \sqrt{2\alpha\theta}, \tag{6.5}$$

$$\beta\mu \leq \lambda^+ \leq \mu. \tag{6.6}$$

After the cut(s) the total sending rate will be reduced by a factor $\tilde{\beta} \triangleq \lambda^+/\lambda^-$, which is always smaller than β.

Lemma 6.1 *If we use MarkMax-T then:*

$$\frac{\beta}{1 + \sqrt{2\alpha\theta}/\mu} \leq \tilde{\beta} \leq \frac{N + \beta - 1}{N}. \tag{6.7}$$

Proof. Let λ_i^- be the sending rate of i-th connection at time t_θ so that $\lambda^- = \sum_i \lambda_i^-$. And let j be such that $\lambda_j = \max_i\{\lambda_i\}$, then:

$$\begin{aligned}\tilde{\beta} = \frac{\lambda^+}{\lambda^-} &\leq 1 - (1-\beta)\frac{\lambda_j^-}{\lambda^-} \\ &\leq 1 - (1-\beta)\frac{\lambda^-}{\lambda^- N} = \frac{N - 1 + \beta}{N}.\end{aligned}$$

Where the first equality is the definition of $\tilde{\beta}$, the first inequality follows from the fact that $\lambda^+ \leq \beta\lambda_j^- + \sum_{i \neq j} \lambda_j^- = \lambda^- - \lambda_j^-(1-\beta)$; this inequality is true because the right hand side corresponds to the case where there is only a single cut and in this case λ^+ is largest. The second the inequality follows from the fact that $\lambda_j = \max_i\{\lambda_i\} \geq \frac{\lambda^-}{N}$.

By (6.6) and (6.5) we have that $\lambda^+ \geq \beta\mu$ and $\lambda^- \leq \mu + \sqrt{2\alpha\theta}$, combining these inequalities with the definition of $\tilde{\beta}$ we have the lower bound. ∎

Using the upper bounds in (6.5) and (6.7) we have:

$$\lambda^+ = \tilde{\beta}\lambda^- \leq (\mu + \sqrt{2\theta\alpha})\frac{N + \beta - 1}{N}. \tag{6.8}$$

As the upper bound on $\tilde{\beta}$ corresponds to the case where only one connection is cut, if the right hand side of (6.8) is less than μ then a single cut of the connection with the biggest rate will be enough. The following lemma follows immediately by setting the right hand side of (6.8) less than or equal to μ and solving for θ.

Lemma 6.2 *If we use MarkMax-T and if*

$$\theta \leq \frac{1}{2\alpha} \frac{\mu^2(1-\beta)^2}{(N-1+\beta)^2}, \tag{6.9}$$

then $\lambda^+ = \beta\lambda_j + \sum_{i \neq j} \lambda_i \leq \mu$ *(that is after a single cut* $\lambda < \mu$*), where* $\lambda_j = \max_i\{\lambda_i\}$.

Using the lower bound in (6.8) we can find a lower bound on θ so that there will be no underflow. That is the backlog is always positive and the link is fully utilized.

Lemma 6.3 *If we use MarkMax-T and if*

$$\theta > \frac{\mu^2(1-\zeta)^2}{2\alpha}, \tag{6.10}$$

where $\zeta = \frac{\beta}{1+\sqrt{2\alpha\theta}/\mu}$*, then the backlog is positive.*

Proof. We have that:

$$\begin{aligned} \lambda^+ &= \tilde{\beta}\lambda^- \geq \zeta\mu \\ &> \frac{\mu - \sqrt{2\alpha\theta}}{\mu}\mu = \mu - \sqrt{2\alpha\theta}, \end{aligned}$$

where the first inequality follows from (6.7) and (6.5) and the second from (6.10). It is easy to see that if $\lambda^+ > \lambda_*^+ = \mu - \sqrt{2\alpha\theta}$ then the backlog is always positive (see Figure (6.1): we want the vertex of the parabola (6.3) to be on the $x = 0$ axis), which completes the proof. ∎

We conclude with a bound that can be used as a guideline to set θ_{h}.

Lemma 6.4 *At time* $t = t_\theta + RTT_j$

$$x(t) \leq \theta + \sqrt{2\alpha\theta}RTT_j + \frac{\alpha}{2}RTT_j^2 \tag{6.11}$$

where $RTT_j = \max_i RTT_i$.

Proof. Consider a cycle that start at time t_θ then at time $t = t_\theta + RTT_i$

$$\begin{aligned} x(t) &= \theta + (\lambda^- - \mu)(t - t_\theta) + \frac{\alpha}{2}(t - t_\theta)^2 \\ &= \theta + (\lambda^- - \mu)RTT_i + \frac{\alpha}{2}RTT_i^2 \\ &\leq \theta + \sqrt{2\alpha\theta}\max_i\{RTT_i\} + \frac{\alpha}{2}\max_i\{RTT_i^2\}, \end{aligned}$$

where the first equality follows from (6.1), and the inequality from the upper bound in (6.5). ∎

Using (6.11) it is possible to know by how much the queue could grow between the time the threshold θ is reached and the time the "slowest" of the connections (i.e. the one with the biggest RTT) reacts to a congestion signal.

6.5 Simulation results

We have modified the NS-2 simulator in order to simulate the behavior of the proposed algorithm. We have implemented both the MarkMax-B and the MarkMax-T, referred to as MM-B and MM-T, respectively, in this section. For MM-T we only consider the last 10% of the queue (recall that for this version we are considering only the final part of the queue when determining the connection with the biggest backlog). We have compared MarkMax with the standard DropTail (DT) policy, by setting the queue size for DT equal to θ. For the MM case the buffer size was large enough to be considered unlimited so that we could verify that MM can stabilize the queue size.

We consider three scenarios, the corresponding topologies are presented in Figures 6.2,6.3. Each node s_i has a TCP connection with node d_i. All the connections have a Maximum Segment Size (MSS) of 540 B. The bottleneck link is the link between the nodes S and D and has capacity μ and propagation delay a_{btlnk}. The links (s_i, S) and (D, d_i) have capacity μ_i and propagation delay a_i. For the first scenario (see Figure 6.2) there are only two sources and two destinations while for the second scenario there is an additional TCP connection sending traffic in the opposite direction on the bottleneck link in order to introduce some variability in the flow of the acknowledgments for connections 1 and 2. The links used by this additional connection are represented as dotted lines in Figure 6.2. In the third scenario we consider 10 connections (see Figure 6.3) with all the traffic going in one direction. In all cases only the link (S, D) uses MM while all the other links use DT.

Let $\bar{q}$ be the average queue size at the bottleneck link and $\bar{q}_i$ $(i = 1, ..., N)$ be the average queue sizes for the i-th connection. Using Little's formula we have that the average queueing delay at the bottleneck link is $\bar{T} = \bar{q}/\mu$. We can express the round trip time of the i-th connection as: $RTT_i = 4a_i + 2a_{\text{btlnk}} + \bar{T}$, assuming the service time of each packet is negligible. Let $\delta_i \triangleq RTT_i - \bar{T} = 4a_i + 2a_{\text{btlnk}}$. By increasing δ_i for some connections we model different propagation and queueing delays of multiples links that, for the sake of simplicity, are not explicitly considered.

Let t_{f} be the total simulation time. Given that all the sources start sending data at time 0 we have that the bottleneck link could transmit at most μt_{f} units of data. Let $D(t_{\text{f}})$ be the total amount of data actually transmitted during the simulation so that the utilization of the link is $\rho \triangleq D(t_{\text{f}})/(\mu t_{\text{f}})$. Let $D_i(t_{\text{f}})$ be the total amount of data received by the i-th connection so that $g_i = D_i(t_{\text{f}})/t_{\text{f}}$ is the corresponding goodput. To compare the fairness of different solutions we use Jain's fairness index which is defined as:

$$J = \frac{\left(\sum_{i=1}^{N} g_i\right)^2}{N \sum_{i=1}^{N} g_i^2}.$$

Note that $\frac{1}{N} \leq J \leq 1$ and that bigger values indicates greater fairness.

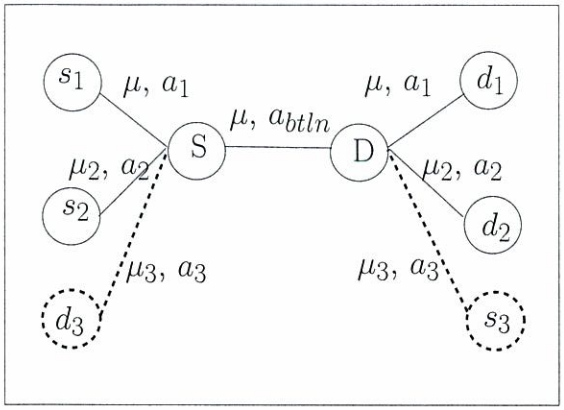

Figure 6.2: Scenarios 1 and 2

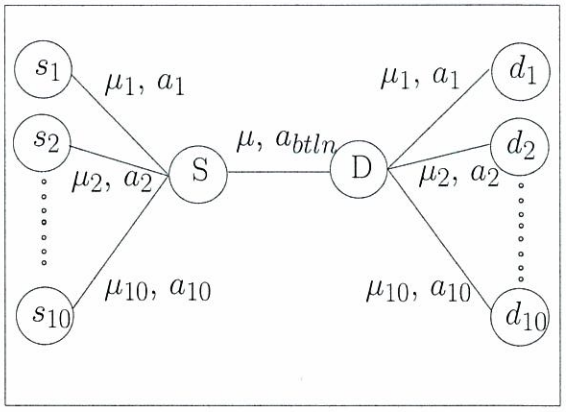

Figure 6.3: Scenario 3

6.5.1 Fluid model

Using the fluid model simulator we investigate the behavior of MarkMax-B for different values of θ. In this case μ =70 Mbit/s, RTT_1 =12 ms, $\alpha_i = \frac{540 \cdot 10^{-6}}{RTT_i^2}$MB/s, $i = 1, 2$. Table 6.1 shows the values of Jain's index and bottleneck link utilization for this case. As θ increases the utilization increases as well, due to the increase in the average backlog size. When θ is not sufficiently large the utilization is less than one due to periodic underflows. For each value of θ Jain's index decreases but it is not too far from 1.

	θ = 60 MSS		θ = 240 MSS		θ = 960 MSS	
$\frac{RTT_2}{RTT_1}$	J	ρ	J	ρ	J	ρ
3	0.9893	0.890	0.9906	0.9500	0.9815	0.9964
7	0.9874	0.892	0.9874	0.9401	0.9788	0.9990
10	0.9861	0.890	0.9869	0.9400	0.9760	0.9990
20	0.9846	0.889	0.9863	0.9440	0.9754	0.9990
50	0.9836	0.899	0.9821	0.9433	0.9664	0.9925

Table 6.1: Fluid Model: Jain's index, utilization.

6.5.2 Scenario 1

For Scenario 1 we set μ =70 Mbit/s, $\mu_1 = \mu_2$ =300 Mbit/s, δ_1 =12 ms, θ =240 MSS, θ_{l} =200 MSS, θ_{h} =280 MSS, θ_{DT} =240 MSS. Table 6.2 gives the values of Jain's index and link utilization for different values of δ_2/δ_1 and different queue management algorithms. Both MM variants outperform DT except in the first case when $\delta_2/\delta_1 = 3$. In this case Jain's index for DT is

bigger but the utilization is somewhat lower. At the same time the difference between Jain's index for DT and MM is significantly large for larger values of δ_2/δ_1. Table 6.3 shows that the average queue size for the MM algorithms is somewhat larger than for DT. This is due to the increased link utilization obtained by MM.

We have verified that in this case the hypothesis of Lemma 6.2 are satisfied and in the simulations it is indeed the case that one cut is always enough to reduce the total sending rate to a value less than μ.

As the difference between MM-B and MM-T is not significant we only use MM-B in the remaining scenarios.

	DT		MM-B		MM-T	
$\frac{\delta_2}{\delta_1}$	J	ρ	J	ρ	J	ρ
3	0.9893	0.9751	0.9853	0.9999	0.9633	0.9999
7	0.7540	0.9720	0.9625	0.9999	0.9515	0.9999
10	0.5361	0.9563	0.9494	0.9999	0.9501	0.9997
20	0.5484	0.9993	0.9561	0.9994	0.9258	0.9997

Table 6.2: Scenario 1: Jain's index, utilization.

	DT		MM-B		MM-T	
$\frac{\delta_2}{\delta_1}$	$\bar{q}$/ B	$\bar{T}$/ms	$\bar{q}$/ B	$\bar{T}$/ms	$\bar{q}$/ B	$\bar{T}$/ms
3	78373	8.9	87257	9.9	86753	9.9
7	74802	8.5	81723	9.3	81547	9.3
10	69219	7.9	80019	9.1	79502	9.1
20	68268	7.8	74297	8.4	74189	8.4

Table 6.3: Scenario 1: average queue size and delay.

6.5.3 Scenario 2

The only difference between the first and second scenario is that there is one additional TCP connection (s_3, d_3) sending data in the opposite direction on the bottleneck link. All the parameters are the same as in scenario 1 with the only difference being that the buffer size for the DT queue between D and S (that is the queue used by the data traffic of connection 3 and the acknowledgments of connections 1 and 2) is set to 240 MSS and the $\delta_3 = \delta_2$. Table 6.4 shows that as in the previous scenario MM-B outperforms DT. Not surprisingly the presence of

traffic competing with the acknowledgments on the (D, S) link does alter the performance of MM-B, for lower values of δ_2/δ_1 there is a slight increase in Jain's index but for higher values it decreases and the utilization is always lower than in the previous case. Most likely this is due to the fact that the presence of traffic disrupting the flow of the acknowledgments increases the round trip time.

	DT			MM-B		
$\frac{\delta_2}{\delta_1}$	J	ρ	$\bar{q}$/ B	J	ρ	$\bar{q}$/ B
7	0.8561	0.9338	34443	0.9637	0.9600	41966
10	0.7769	0.9497	32174	0.9632	0.9510	39486
20	0.6910	0.9146	28699	0.9228	0.9702	41350
50	0.5244	0.9262	29021	0.8572	0.9937	50408

Table 6.4: Scenario 2: Jain's index, utilization and average queue size.

6.5.4 Scenario 3

In the last scenario we have 10 connections sharing the (S, D) link and no connections using the reverse link, μ =70 Mbit/s, μ_i =300 Mbit/s, $i = 1, \ldots, 10$, δ_1 =12 ms, $\delta_{i+1} = \sqrt{2}\delta_i$, $i = 1, \ldots, 9$, θ =240 MSS, θ_l =200 MSS, θ_h =280 MSS, θ_DT =240 MSS. Table 6.5 shows that MM-B has a significantly higher Jain's index, and slightly higher utilization, at the expenses of a moderate increase in the average queue size.

	J	ρ	$\bar{q}$/ B	$\bar{T}$/ms
DT	0.5848	98,91	65207	7
MM-B	0.9313	99,99	98913	11

Table 6.5: Scenario 3: Jain's index, utilization and average queue size and delay

6.6 Conclusion and future work

We have introduced MarkMax: a simple flow-aware AQM algorithm. We have used a fluid model to set the parameters of the algorithm as well as to analyze its behavior. We have also shown how to compute the per-flow backlog using such a model. We have simulated the two proposed variants (MarkMax-B and MarkMax-T) using NS-2, showing how they improve the fairness and link utilization compared to the standard DropTail algorithm.

These results are definitely promising and warrant further analysis. Of all the issues that we plan on addressing we would like to mention performance and queue stability with large number of connections and comparison between MarkMax-B and MarkMax-T. So far we have conducted simulations with up to 10 connections but it is not immediately clear if the algorithm would perform equally well with more connections. It is conceivable that, at least in some cases, cutting a single connection could no be enough to bring the total sending rate to a value smaller than μ. We would also like to determine whether MarkMax-B always outperforms MarkMax-T as indicated by the simulations we run so far or if it the situation can be reversed by properly selecting the fraction of the queue that is considered while computing the per-connection backlog in MarkMax-T.

Chapter 7

Conclusions and perspectives

In the current thesis we propose several new contributions to improve the performance in computer networks. The obtained results concern the resource sharing problems in the Internet routers, Web servers and operating systems. We study several algorithms which decrease the mean waiting time in the system with efficient resource sharing, provide the possibility to introduce the Quality of Service, Network Pricing and flow differentiation to the networks. We show the effectiveness of the proposed algorithms and study the possibility of their implementation in the router queues. The studied problems open several directions for future work, some of which are the topics of our current research.

In Chapter 3 we study the TLPS scheduling scheme for the case of hyper-exponential job size distribution and find an approximation of the optimal threshold for the case of two phase job size distribution. We show that the mean waiting time in the system with the use of the found threshold approximation can be reduced up to 36% in comparison with the DropTail policy. Still the question of the threshold selection in the case when the job size distribution is hyper-exponential with many phases or has a different distribution stays open. We consider this to be an important topic future studies.

In Chapter 4 we prove the monotonicity of the mean conditional sojourn time in the DPS system under a restriction on the system parameters. As we did not find a counter-example and therefore the found restriction is probably not a necessary condition, we think that it is possible to prove the theorem for the general case without additional system constraints. Also the investigation of the system parameters to find the cases when the DPS system gives a significant gain in comparison with PS system is an interesting topic for future research.

In Chapter 5 we study the optimal Gittins policy in the multi-class single server queue. This topic opens a large area for future research, as we studied several particular cases of the

Gittins policy application. Taking into account the Internet traffic structure, we study the case when the jobs arrive to the system in two classes, which are Pareto distributed and represent "mice" and "elephants" in the Internet. For this case we describe the optimal system policy, find the analytical expression of the mean waiting time and implement the algorithm in the router queue. With the simulation results we show that with the found optimal policy the gain in the system can reach 10% in comparison with the LAS policy and 36% in comparison with the DropTail policy. Also we study several cases of particular interest when jobs arrive in classes with exponential distributions. As a future research we propose to consider the cases with more than two job classes in the system, also we may consider other types of service times distributions. It is important to investigate the system parameters to find when the Gittins policy gives a significant gain in comparison with the LAS policy. The applicability of our results in real systems like the Internet should also be more carefully evaluated.

In Chapter 6 we introduce a new flow-aware AQM scheme, MarkMax, which reduces the sending rate of the connection with the largest sending rate when the router buffer reaches some given threshold. With the fluid model we found the guidelines for the threshold selection. Using the NS-2 simulator we implement MarkMax in the router queue and show that it improves fairness in the system and provides better performance than the DropTail policy. As a future research topic we propose to study more complex system topologies and cases with the large number of connections share the bottleneck link. For this case we propose to cut several connections at once. The selection of the number of connections to cut and its dependency on the number of connections present in the network constitute a challenging study.

A possible research direction is a combination of MarkMax and a flow differentiation scheduling policy like TLPS or Gittins policies. Development of the new algorithm which gives priority to the short flows and at the same time improves fairness between the long flows can be an interesting and nontrivial task. We think that such an algorithm can improve both, fairness and mean waiting time in the system and provide better system performance.

List of Acronyms

ACK	Acknowledgment
AQM	Active Queue Management
DNS	Domain Name System
DPS	Discriminatory Processor Sharing
DT	DropTail
FB	Foreground Background
FCFS	First Come First Served
FTP	File Transfer Protocol
HTTP	Hypertext Transfer Protocol
HE	hyper-exponential
ICMP	Internet Control Message Protocol
IP	Internet Protocol
LAN	Local Area Network
LAS	Least Attained Service
MIME	Multipurpose Internet Mail Extension
MLPS	Multi Level Processor Sharing
MM	MarkMax
MSS	Maximum Segment Size
MTU	Maximum Transmission Unit
NS	Network Simulator
OSI	Open Systems Interconnection
PASTA	Poisson Arrivals See Time Averages
PS	Processor Sharing
RED	Random Early Dropping
RFC	Request for Comment
RTT	Round-Trip Time

SMTP	Simple Mail Transfer Protocol
SRPT	Shortest Remaining Processing Time
SPT	Shortest Processing Time
TCP	Transmission Control Protocol
Telnet	remote terminal protocol
TLPS	Two Level Processor Sharing
UDP	User Datagram Protocol
WAN	Wide Area Network
WWW	World Wide Web

BIBLIOGRAPHY

[AA06] S. Aalto and U. Ayesta. Mean delay analysis of Multilevel Processor Sharing disciplines. *Proceedings of IEEE INFOCOM 2006*, 2006. 13, 18, 30, 33

[AA07] S. Aalto and U. Ayesta. Mean delay optimization for the $M/G/1$ queue with Pareto type service times. In *Extended abstract in ACM SIGMETRICS 2007, San Diego, CA*, pages 383–384, 2007. 73, 91

[AAA06] E. Altman, K. Avrachenkov, and U. Ayesta. A survey on Discriminatory Processor Sharing. *Queueing Syst.*, 53(1-2):53–63, 2006. 14, 54

[AAB00] E. Altman, K. Avrachenkov, and C. Barakat. A stochastic model of TCP/IP with stationary random loses. In *ACM SIGCOMM 2000, Stockholm, Sweden*, volume 30, pages 231–242, 2000. 103

[AAB05] K. Avrachenkov, U. Ayesta, and P. Brown. Batch Arrival Processor-Sharing with Application to Multi-Level Processor-Sharing Scheduling. *Queueing Systems*, 50:459–480, 2005. 18, 44

[AABN04] K. Avrachenkov, U. Ayesta, P. Brown, and E. Nyberg. Differentiation between short and long TCP flows: Predictability of the response time. In *IEEE INFOCOM 2004*, volume 2, pages 762–773, 2004. 13, 18, 30, 40, 44, 100

[AABNQ05] K. Avrachenkov, U. Ayesta, P. Brown, and R. Núñez-Queija. Discriminatory Processor Sharing revisited. In *INFOCOM, 24th Annual Joint Conference of the IEEE Computer and Communications Societies*, pages 784–795. IEEE, 2005. 14, 54, 55

[AANB02] K. E. Avrachenkov, U. Ayesta, P. Nain, and C. Barakat. The effect of router buffer size on the TCP performance. In *In Proceedings of the LONIIS Workshop on Telecommunication Networks and Teletraffic Theory*, pages 116–121, 2002. 9

[AANO04] S. Aalto, U. Ayesta, and E. Nyberg-Oksanen. Two-level processor-sharing scheduling disciplines: mean delay analysis. *SIGMETRICS Perform. Eval. Rev.*, 32(1):97–105, 2004. 13, 30

[AANO05] S. Aalto, U. Ayesta, and E. Nyberg-Oksanen. $M/G/1/MLPS$ compared to $M/G/1/PS$. *Operation Reserch Letters*, 33(5):519–524, 2005. 13

[AAP05] K. Avrachenkov, U. Ayesta, and A. Piunovskiy. Optimal choice of the buffer size in the Internet routers. In *Decision and Control, 2005 and 2005 European Control Conference. CDC-ECC '05. 44th IEEE Conference on*, pages 1143–1148, December 2005. 103

[AART06] E. Altman, R. E. Azouzi, D. Ros, and B. Tuffin. Loss strategies for competing AIMD flows. *Comput. Networks*, 50(11):1799–1815, 2006. 100

[ABL+00] E. Altman, C. Barakat, E. Laborde, P. Brown, and D. Collange. Fairness analysis of TCP/IP. *Decision and Control, 2000. Proceedings of the 39th IEEE Conference*, 1:61–66, 2000. 100

[ABN+95] E. Altman, J. Bolot, P. Nain, D. Elouadghiri, M. Erramdani, P. Brown, and D. Collange. Performance modelling of TCP/IP in a Wide-Area network. In *34th IEEE Conference on Decision and Control*, December 1995. 103

[ABO07] K. Avrachenkov, P. Brown, and N. Osipova. Optimal choice of threshold in Two Level Processor Sharing. *Annals of Operations Research journal*, 2007. 15, 24, 29

[AFG06] K. Avrachenkov, L. Finlay, and V. Gaitsgory. Analysis of TCP-AQM interaction via periodic optimization and linear programming: the case of sigmoidal utility function. In *NEW2AN, Also LNCS v.4003*, pages 517–529, December 2006. 100

[AJK04] E. Altman, T. Jimenez, and D. Kofman. DPS queues with stationary ergodic service times and the performance of TCP in overload. In *in Proceedings of IEEE Infocom, Hong-Kong*, 2004. 14, 54

[AJN02] E. Altman, T. Jiménez, and R. Núñez-Queija. Analysis of two competing TCP/IP connections. *Perform. Eval.*, 49(1-4):43–55, 2002. 100

[AKM04] G. Appenzeller, I. Keslassy, and N. McKeown. Sizing router buffers. *SIGCOMM Comput. Commun. Rev.*, 34(4):281–292, 2004. 9

[All00] M. Allman. A Web server's view of the Transport layer. *ACM Computer Communication Review*, 30, 2000. 7

[APS99] M. Allman, V. Paxson, and W. Stevens. TCP Congestion Control. RFC 2581 (Proposed Standard), April 1999. Updated by RFC 3390. 9

[Ban03] N. Bansal. Analysis of the $M/G/1$ Processor-Sharing queue with bulk arrivals. *Operations Research Letters*, 31(5):401–405, 2003. 18, 36, 42

[BFOBR02] N. Benameur, S. B. Fredj, S. Oueslati-Boulahia, and J. W. Roberts. Quality of service and flow level admission control in the Internet. *Computer Networks*, 40(1):57–71, 2002. 7

[BGG03] M. Barthelemy, B. Gondran, and E. Guichard. Spatial structure of the Internet traffic. *Physica A Statistical Mechanics and its Applications*, 319:633–642, March 2003. 7

[BGG+08] N. Beheshti, Y. Ganjali, M. Ghobadi, N. McKeown, and G. Salmon. Experimental study of router buffer sizing. Systems and Networking Laboratory Technical Report TR08-UT-SN, University of Toronto, Department of Computer Science, May 2008. 9

[BM06] F. Baccelli and D. R. McDonald. A stochastic model for the rate of non-persistent TCP flows. *Proceedings of ValueTools 2006*, 2006. 19, 30, 32, 42

[BNM00] D. Bertsimas and J. Niño-Mora. Restless bandits, linear programming relaxations and a Primal-Dual index heuristic. *Operations Research*, 48:80–90, 2000. 70

[Bra89] R. Braden. Requirements for Internet hosts - communication layers. RFC 1122 (Standard), October 1989. Updated by RFCs 1349, 4379. 9

[Bre96] L. P. Breker. A survey of network pricing schemes. In *University of Saskatchewan*, 1996. 11

[Bro00] P. Brown. Resource sharing of TCP connections with different round trip times. In *INFOCOM 2000*, pages 1734–1741, 2000. 100

[Bro06] P. Brown. Comparing FB and PS scheduling policies. *SIGMETRICS Perform. Eval. Rev.*, 34(3):18–20, 2006. 13

[BT01] T. Bu and D. Towsley. Fixed point approximations for TCP behavior in an AQM network. In *SIGMETRICS '01: Proceedings of the 2001 ACM SIGMETRICS international conference on Measurement and modeling of computer systems*, pages 216–225, New York, NY, USA, 2001. ACM. 54

[BVW85] C. Buyukkoc, P. Varaya, and J. Walrand. The $c\mu$ rule revisited. *Adv. Appl. Prob.*, 17:237–238, 1985. 70

[CB97] M. E. Crovella and A. Bestavros. Self-similarity in World Wide Web traffic: evidence and possible causes. *IEEE/ACM Transactions on Networking*, 5:835–846, 1997. 7, 71

[CC08] D. Collange and J.-L. Costeux. Passive estimation of Quality of Experience. *Journal of Universal Computer Science*, 14(5):625–641, 2008. 7

[CJ07] N. Chen and S. Jordan. Throughput in Processor-Sharing queues. *IEEE Transactions on Automatic Control*, 52 (2):299–305, 2007. 12

[Cru91] R. L. Cruz. A calculus for network delay. I. Network elements in isolation. *Information Theory, IEEE Transactions on*, 37(1):114–131, 1991. 104

[CvdBB+05] S. K. Cheung, J. L. van den Berg, R. J. Boucherie, R. Litjens, and F. Roijers. An analytical packet/flow-level modelling approach for wireless LANs with quality-of-service support. In *in Proceedings of ITC-19*, 2005. 54

[DGNM96] M. Dacre, K. Glazebrook, and J. Niño-Mora. The achievable region approach to the optimal control of stochastic systems. *Journal of the Royal Statistical Society. Series B, Methodological*, 61(4):747–791, 1996. 70

[FBP+01] S. B. Fred, T. Bonald, A. Proutiere, G. Régnié, and J. W. Roberts. Statistical bandwidth sharing: a study of congestion at flow level. *SIGCOMM Comput. Commun. Rev.*, 31(4):111–122, 2001. 12

[FJ93] S. Floyd and V. Jacobson. Random early detection gateways for congestion avoidance. *IEEE/ACM Trans. Netw.*, 1(4):397–413, 1993. 11, 100

[Flo91] S. Floyd. Connections with multiple congested gateways in packet-switched networks part 1: one-way traffic. *SIGCOMM Comput. Commun. Rev.*, 21(5):30–47, 1991. 100

[Flo95] S. Floyd. TCP and Explicit Congestion Notification. *ACM Computer Communication Review*, 24(5):10–23, 1995. 11

[FM03a] H. Feng and V. Misra. Asymptotic bounds for $M^X/G/1$ Processor Sharing queues. *Technical report CUCS-00-04, Columbia University*, 2003. 18

[FM03b] H. Feng and V. Misra. Mixed scheduling disciplines for network flows. *ACM SIGMETRICS Performance Evaluation Review*, 31(2):36–39, 2003. 13, 30

[FMI80] G. Fayolle, I. Mitrani, and R. Iasnogorodski. Sharing a processor among many job classes. *Journal of the ACM*, 27:519–532, 1980. 14, 20, 54, 56

[FML+03] C. Fraleigh, S. Moon, B. Lyles, C. Cotton, M. Khan, D. Moll, R. Rockell, T. Seely, and C. Diot. Packet-level traffic measurements from the Sprint IP backbone. *IEEE Network*, 17:6–16, 2003. 7, 8

[FORR98] E. W. Fulp, M. Ott, D. Reininger, and D. S. Reeves. Paying for QoS: an optimal distributed algorithm for pricing network resources. *Quality of Service, 1998. (IWQoS 98) 1998 Sixth International Workshop on*, pages 75–84, May 1998. 11

[FR01] E. W. Fulp and D. S. Reeves. Optimal provisioning and pricing of differentiated services using QoS class promotion. In *In Proceedings of the INFORMATIK: Workshop on Advanced Internet Charging and QoS Technology*, 2001. 11

[FR04] E. W. Fulp and D. S. Reeves. Bandwidth provisioning and pricing for networks with multiple classes of service. *Comput. Netw.*, 46(1):41–52, 2004. 11

[FSKS02] W.-C. Feng, K. G. Shin, D. D. Kandlur, and D. Saha. The BLUE active queue management algorithms. *IEEE/ACM Trans. Netw.*, 10(4):513–528, 2002. 11

[FW98] A. Feldmann and W. Whitt. Fitting mixtures of exponentials to long-tail distributions to analyze network performance models. *Performance Evaluation*, 31:245–258, 1998. 19, 30, 32, 42

[FW99] E. Frostig and G. Weiss. Four proofs of Gittins' multiarmed bandit theorem. *Applied Probability Trust*, 1999. 70

[Git89] J. Gittins. Multi-armed Bandit Allocation Indices. *Wiley, Chichester*, 1989. 14, 15, 70, 72, 73, 90

[GM01] L. Guo and I. Matta. The war between mice and elephants. Technical report, Boston University, Boston, MA, USA, 2001. 13

[GM02a] L. Guo and I. Matta. Differentiated control of web traffic: A numerical analisys. *SPIE ITCOM, Boston*, 2002. 13, 30

[GM02b] L. Guo and I. Matta. Scheduling flows with unknown sizes: approximate analysis. In *in Proceedings of the 2002 ACM SIGMETRICS international conference on Measurement and modeling of computer systems*, pages 276–277, 2002. 13, 14, 54

[HBSBA03] M. Harchol-Balter, B. Schroeder, N. Bansal, and M. Agrawal. Size-based scheduling to improve web performance. *ACM Transactions on Computer Systems*, 21(2):207–233, 2003. 30

[HLN97] D. P. Heyman, T. V. Lakshman, and A. L. Neidhardt. A new method for analyzing feedback-based protocols with applications to engineering Web traffic over the Internet. In *SIGMETRICS '97: Proceedings of the 1997 ACM SIGMETRICS international conference on Measurement and modeling of computer systems*, pages 24–38, New York, NY, USA, 1997. ACM. 12

[HT05] Y. Hayel and B. Tuffin. Pricing for heterogeneous services at a Discriminatory Processor Sharing queue. In *Networking 2005*, volume 3462/2005, pages 816–827. Springer Berlin / Heidelberg, 2005. 54

[Jac88] V. Jacobson. Congestion avoidance and control. In *SIGCOMM '88: Symposium proceedings on Communications architectures and protocols*, volume 18, pages 314–329, New York, NY, USA, August 1988. ACM Press. 9

[JBB92] V. Jacobson, R. Braden, and D. Borman. TCP extensions for high performance. RFC 1323 (Proposed Standard), May 1992. 9

[KK06] B. Kim and J. Kim. Comparison of DPS and PS systems according to DPS weights. *Communications Letters, IEEE*, 10(7):558–560, July 2006. 55, 56, 58

[KK08] J. Kim and B. Kim. Concavity of the conditional mean sojourn time in the $M/G/1$ Processor Sharing queue with batch arrivals. *Queueing Systems*, 58(1):57–64, 2008. 17, 18, 22

[Kle67] L. Kleinrock. Time-shared Systems: a theoretical treatment. *J. ACM*, 14(2):242–261, 1967. 14, 54

[Kle76a] L. Kleinrock. *Queueing systems*, volume 2. John Wiley and Sons, 1976. 12, 13, 18, 20, 23, 24, 25, 30, 32, 34, 77, 91

[Kle76b] L. Kleinrock. *Queueing systems*, volume 1. John Wiley and Sons, 1976. 12, 93

[Kli74] G. Klimov. Time-sharing service systems. i. *Theory of Probability and Its Applications*, 19:532–551, 1974. 72

[Kli78] G. Klimov. Time-sharing service systems. ii. *Theory of Probability and Its Applications*, 23:314–321, 1978. 72

[KMR71] L. Kleinrock, R. R. Muntz, and E. Rodemich. The Processor-Sharing queueing model for time-shared systems with bulk arrivals. *Networks Journal*, 1:1–13, 1971. 18

[KNQB04] G. Van Kessel, R. Núñez-Queija, and S. Borst. Asymptotic regimes and approximations for Discriminatory Processor Sharing. *SIGMETRICS Perform. Eval. Rev.*, 32(2):44–46, 2004. 14, 54

[KNQB05] G. Van Kessel, R. Núñez-Queija, and S. Borst. Differentiated bandwidth sharing with disparate flow sizes. *INFOCOM 2005. 24th Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings IEEE*, 4:2425–2435, March 2005. 14, 54

[Kri00] J. Kristoff. TCP Congestion Control. Technical report, DePaul University, 2000. 9

[KSH03] R. E. A. Khayari, R. Sadre, and B.R. Haverkort. Fitting world-wide web request traces with the EM-algorithm. *Performance Evaluation*, 52(2-3):175–191, 2003. 30, 42

[Kur72] A. G. Kurosh. *Higher algebra.* MIR, 1972. 26

[LM97] T. V. Lakshman and U. Madhow. The performance of TCP/IP for networks with high bandwidth-delay products and random loss. *IEEE/ACM Trans. Netw.*, 5(3):336–350, 1997. 100

[Man90] A. Mankin. Random drop congestion control. In *SIGCOMM '90: Proceedings of the ACM symposium on Communications architectures & protocols*, pages 1–7, 1990. 100

[MR00] L. Massoulié and J. W. Roberts. Bandwidth sharing and admission control for elastic traffic. In *Telecommunication Systems*, volume 15, pages 185–201(17). Springer, 2000. 12

[NMM98] M. Nabe, M. Murata, and H. Miyahara. Analysis and modeling of World Wide Web traffic for capacity dimensioning of Internet access lines. *Perform. Eval.*, 34(4):249–271, 1998. 7, 12, 71

[NT94] P. Nain and D. Towsley. Optimal scheduling in a machine with stochastic varying processing rate. *IEEE/ACM Transactions on Automatic Control*, 39:1853–1855, 1994. 70

[NT02] W. Noureddine and F. Tobagi. Improving the performance of interactive TCP applications using service differentiation. In *Computer Networks Journal*, pages 2002–354. IEEE, 2002. 13

[NW08] M. Nuyens and A. Wierman. The Foreground-Background queue: A survey. *Perform. Eval.*, 65(3-4):286–307, 2008. 13

[OBA08] N. Osipova, A. Blanc, and K. Avrachenkov. Improving TCP fairness with the MarkMax policy. In *In Proceedings of the 15th International Conference on Telecommunications, ICT 2008*, 2008. 16, 99

[Osi07] N. Osipova. Batch Processor Sharing with Hyper-Exponential service time. Technical Report RR-6180, INRIA, 2007. 19

[Osi08a] N. Osipova. Batch Processor Sharing with Hyper-Exponential service time. *Operations Research Letters*, 36(3):372–376, 2008. 14, 17, 19, 45

[Osi08b] N. Osipova. Comparison of the Discriminatory Processor Sharing Policies. Technical Report RR-6475, INRIA, 2008. 15, 53

[Pos81] J. Postel. Transmission Control Protocol. RFC 793 (Standard), September 1981. Updated by RFC 3168. 9

[PPP00] R. Pan, B. Prabhakar, and K. Psounis. Choke: a stateless active queue management scheme for approximating fair bandwidth allocation. In *INFOCOM 2000. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, volume 2, pages 942–951, 2000. 11, 100

[RBUK05] I. A. Rai, E. W. Biersack, and G. Urvoy-Keller. Size-based scheduling to improve the performance of short TCP flows. *IEEE Network*, 19:12–17, 2005. 13

[RF99] K. Ramakrishnan and S. Floyd. A Proposal to add Explicit Congestion Notification (ECN) to IP. RFC 2481 (Experimental), January 1999. Obsoleted by RFC 3168. 11

[RFB01] K. Ramakrishnan, S. Floyd, and D. Black. The Addition of Explicit Congestion Notification (ECN) to IP. RFC 3168 (Proposed Standard), September 2001. 11, 100

[Rig94] R. Righter. Scheduling. *M. Shaked and J. Shanthikumar (eds), Stochastic Orders, New York: Academic Press*, pages 381–432, 1994. 55

[Rob01] J. Roberts. Traffic theory and the Internet. *IEEE Communication Magazine*, 39(1):94–99, 2001. 7

[RS89] R. Righter and J. Shanthikumar. Scheduling multiclass single server queueing systems to stochastically maximize the number of successful departures. *Probability in the Engineering and Informational Sciences*, 3:323–333, 1989. 13

[RS93] K. M. Rege and B. Sengupta. The $M/G/1$ Processor Sharing queue with bulk arrivals. In *In Proceedings of Modelling and Evaluation of ATM Networks*, pages 417–432, 1993. 18

[RS94] K. M. Rege and B. Sengupta. A decomposition theorem and related results for the Discriminatory Processor Sharing queue. *Queueing Systems*, 18(3-4):333–351, 1994. 14, 54

[RS96] K. M. Rege and B. Sengupta. Queue-length distribution for the Discriminatory Processor-Sharing queue. *Operations Research*, 44(4):653–657, 1996. 14, 54

[RUKB02] I. A. Rai, G. Urvoy-Keller, and E. W. Biersack. Size-based scheduling with differentiated services to improve response time of highly varying flows. In *in Proceedings of the 15th ITC Specialist Seminar, Internet Traffic Engineering and Traffic Management*, 2002. 13

[RUKB03] I. A. Rai, G. Urvoy-Keller, and E. W. Biersack. Analysis of LAS scheduling for job size distributions with high variance. In *SIGMETRICS '03: Proceedings of the 2003 ACM SIGMETRICS international conference on Measurement and modeling of computer systems*, pages 218–228, New York, NY, USA, 2003. ACM. 13

[RUKVB04] I. A. Rai, G. Urvoy-Keller, M. K. Vernon, and E. W. Biersack. Performance analysis of LAS-based scheduling disciplines in a packet switched network. *SIGMETRICS Perform. Eval. Rev.*, 32(1):106–117, 2004. 13

[SAM99] M. Murata S. Ata and H. Miyahara. Analysis and application of network traffic characteristics to design of high-speed routers. *Internet. Conference No2, Boston MA ,ETATS-UNIS (20/09/1999)*, 3842:14–24, 1999. 7

[SCEH96] S. Shenker, D. Clark, D. Estrin, and S. Herzog. Pricing in computer networks: reshaping the research agenda. *SIGCOMM Comput. Commun. Rev.*, 26(2):19–43, 1996. 11

[Sch68] L. E. Schrage. A proof of the optimality of the Shortest Remaining Processing Time discipline. *Operations Research*, 16(3):678–690, 1968. 12, 30

[Sev74] K. Sevcik. Scheduling for minimum total loss using service time distributions. *Journal of the ACM*, 21:66–75, 1974. 72

[SS07] R. Stanojevic and R. Shorten. Beyond CHOKe: Stateless fair queueing. In *NETCOOP 2007, LNCS v.4465*, pages 43–53, 2007. 11, 100

[Sta94] W. Stallings. *Data and Computer Communications: 4th edition.* Macmillian Publishing Company, 1994. 5, 6, 8

[Sta03] W. Stallings. *Computer Networking with Internet Protocols and Technology.* Pearson Education, Inc., Prentice Hall, 2003. 4, 6, 9

[Sta07] R. Stanojevic. *Router-based algorithms for improving Internet Quality of Service.* Phd thesis, Hamilton Institute, National University of Ireland Maynooth, 2007. 100

[Ste97] W. Stevens. TCP Slow Start, Congestion Avoidance, Fast Retransmit, and Fast Recovery Algorithms. RFC 2001 (Proposed Standard), January 1997. Obsoleted by RFC 2581. 9

[SY92] J. Shanthikumar and D. Yao. Multiclass queueing systems: Polymatroidal structure and optimal scheduling control. *Operations Research*, 40(2):293–299, 1992. 70

[SZC90] S. Schenker, L. Zhang, and D. D. Clark. Some observations on the dynamics of a congestion control algorithm. *SIGCOMM Comput. Commun. Rev.*, 20(5):30–39, 1990. 103

[Tan96] A. S. Tanenbaum. *Computer networks: 3rd edition.* Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1996. 4, 9

[TMW97] K. Thompson, G. J. Miller, and R. Wilder. Wide-area Internet traffic patterns and characteristics. *IEEE Network*, 11:10–23, 1997. 7

[Tsi93] J.N. Tsitsiklis. A short proof of the Gittins index theorem. In *IEEE CDC*, pages 389–390, 1993. 70

[VWB85] P. Varaiya, J. Walrand, and C. Buyukkoc. Extensions of the multiarmed bandit problem: the discounted case. *IEEE Transactions on Automatic Control*, 30:426–439, 1985. 70

[WBHB04] A. Wierman, N. Bansal, and M. Harchol-Balter. A note comparing response times in the $M/GI/1/FB$ and $M/GI/1/PS$ queues. *Operations Research Letters*, 32(1):73–76, 2004. 13

[Web92] R. Weber. On the Gittins index for multiarmed bandits. *Annals of Appllied Probability*, 2(4):1024–1033, 1992. 70

[WHB03] A. Wierman and M. Harchol-Balter. Classifying scheduling policies with respect to unfairness in an $M/GI/1$. *SIGMETRICS Perform. Eval. Rev.*, 31(1):238–249, 2003. 13

[Whi88] P. Whittle. Restless bandits: activity allocation in a changing world. *Journal of Applied Probability*, 25:287–298, 1988. 70

[Wil98] F. Wilder. *A guide to the TCP/IP protocol suite: second edition.* Artech House, INC, 1998. 9

[Wil01] C. Williamson. Internet traffic measurement. *IEEE Internet Computing*, 5:70–74, 2001. 7, 8, 71

[Wol89] R. W. Wolf. *Stochastic modeling and the theory of queues.* Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1989. 7

[WZ02] B. Wydrowski and M. Zukerman. GREEN: An Active Queue Management algorithm for a self managed Internet. In *ICC*, volume 4, pages 2368–2372, 2002. 11

[Yas87] S. F. Yashkov. Processor-Sharing queues: some progress in analysis. *Queueing Syst. Theory Appl.*, 2(1):1–17, 1987. 12

[Yas92] S. Yashkov. Mathematical problems in the theory of shared-processor systems. *Journal of Mathematical Sciences*, 58:101–147, 1992. 72