
# Thèse

**présentée pour obtenir le grade de docteur**

**Ecole nationale supérieure des télécommunications**

**Informatique et Réseaux**


# Idris A. Rai


# Support de la Qualité de Service dans les Routeurs Accès de l'Internet

**Soutenue le 15 Septembre 2004 devant le jury composé de**

| | |
|---|---|
| **Walid Dabbous** | **Président** |
| **Mary W. Vernon** | **Rapporteurs** |
| **Philippe Nain** | |
| **Tijani Chahed** | **Examinateur** |
| **Ernst W. Biersack** | **Directeur de thèse** |
| **Guillaume Urvoy-Keller** | **Co-encadrant** |


**Ecole nationale supérieure des télécommunications**


# PhD thesis

## Presented to Obtain the Degree of Doctor of Philosophy

## Doctoral School of Computer Science and Telecommunications

## Computer Science and Networks


# Idris A. Rai


# QoS Support in Edge Routers

**Defense date: 15th September 2004**

**Jury members:**

| | |
|---|---|
| **Walid Dabbous** | **Chairman** |
| **Mary W. Vernon** | **Reporters** |
| **Philippe Nain** | |
| **Tijani Chahed** | **Examiners** |
| **Ernst W. Biersack** | **Thesis advisor** |
| **Guillaume Urvoy-Keller** | **Thesis co-advisor** |


**Ecole nationale supérieure des télécommunications**

*To Nasra and Manal*

*To my grand parents*

# Acknowledgement

First and foremost I would like to thank my thesis advisor Prof. Ernst W. Biersack for his never ended support that made this thesis possible. In addition to making technical research, I ought to mention that I learned a host of other things such as technical writing and presentation, through his scrupulous but veracious guidance of this thesis.

I am indepted to my thesis co-advisor, Dr. Guillaume Urvoy-Keller, for his continuing help and being always present for discussions. His help started from the first day of this thesis and never stopped until the end of the thesis. I have to acknowledge his important inputs in all chapters of this thesis and also for his invaluable help for French translation that was required for this thesis.

I gratefully acknowledge the precious opportunity I got to collaborate with Prof. Mary K. Vernon. I must say that my technical discussions with her tremendously improved my ability to analyze theoretical models and to relate them to their corresponding practical systems. Her support resulted in the valuable input in Chapter 5 of this thesis.

I also highly appreciate the opportunity I had to join Institut Eurecom and the financial support Eurecom provided for my studies after a year at RightVision.

I owe special thanks to my wife Nasra for her patience, love, and encouragement during these years of my studies. Finally, I would like to thank my family for their support during my student life, and for their patience and encouragement while have been separated for many years.

# Abstract

The goal of this thesis is to investigate scheduling mechanisms that allow to offer service differentiation in the Internet. For this purpose, we study *least attained service* (LAS) scheduling policy in packet switched networks to support *quality of service* (QoS). LAS has been known for decades as job scheduling policy but has never been considered for packet switched networks. When analyzed under M/M/1 queue LAS penalizes the largest jobs a lot. However, recent Internet traffic measurements have revealed that the Internet traffic exhibits a *high variability property*; many flows are short Web transfers and about 1% of the largest flows carry more than 50% of all bytes.

Motivated by the high variability property of Internet traffic, we first analyze the performance of LAS under the M/G/1 queue, where G is a job size distribution with varying degree of variability. The analysis for LAS shows that the conditional mean response time highly depends on the variability of a job size distribution and that the percentage of large jobs that see penalty under LAS is negligible when a job size distribution shows the high variability property. While *shortest remaining processing time* (SRPT) is known to be the optimal policy in reducing the mean response time, we show that LAS offers a conditional mean response time close to SRPT. When compared to *first-come first-serve* (FCFS), LAS significantly outperforms FCFS for job size distributions with the high variability property. Moreover, as opposed to FCFS, we prove that LAS can service some jobs even at overload.

We then use simulations to investigate the performance of LAS in *packet* networks for various network configurations. We also study the interaction of LAS with TCP. We find that LAS has quite interesting features that cause it to reduce the transfer time for short TCP flows without penalizing large flows a lot. In addition, we show through simulations that LAS allocates bandwidth *equally* among competing TCP connections in congested heterogeneous networks.

The simulation results show that LAS penalizes *long-lived* TCP and UDP flows at high network load above 75%. To address this issue, we propose a family of new LAS-based scheduling policies that differentiate the service in order to improve the performance of priority flows. To evaluate these policies, we develop analytic models to compute the mean flow transfer time for TCP flows that share a single bottleneck link that uses LAS or one of new LAS-based policies . Over a wide range of settings , we show that the analytic models agree with the ns2 simulation of the policies in packet networks when loss rates are low. Evaluation of the proposed policies shows that they notably improve the performance of priority flows in terms of reducing their mean transfer time, average jitter, and loss rate while affecting ordinary flows in a minor way. Finally, we evaluate a hybrid policy called LAS-FCFS that services small jobs using LAS and large jobs using FCFS. This policy can be useful when the penalty for the largest jobs under LAS is not acceptable. Analytical results show that LAS-FCFS improves the performance of the largest jobs when a job size distribution exhibits the high variability property. We also analyze

and evaluate variants of LAS-FCFS that offer service differentiation.

# Résumé

Cette thèse porte sur l'étude de la discipline de service LAS (*least attained service*) et de variantes de cette discipline, dans le but d'offrir de la différentiation de service dans l'Internet. LAS donne la priorité au client qui a reçu le moins de service à un instant donné. Elle est connue, en théorie des files d'attente, depuis une trentaine d'années, mais n'a jamais été considérée dans les réseaux de paquets à cause de certains résultats (sur une file M/M/1 par exemple) montrant que LAS pénalise trop les plus "gros" clients.

Des études récentes ont montré que le trafic Internet, au niveau flot, se caractérise par une forte variabilité : de nombreux flots sont très courts alors que les flots les plus longs représentent plus de 50% de la charge totale. Motivé par ce nouveau résultat, nous avons analysé LAS dans le cas d'une file M/G/1 avec une discipline de service G ayant une variabilité plus ou moins marquée. Cette analyse a montré que les temps moyens de réponse conditionels de LAS dépendent fortement de la variabilité de G et que la fraction de flots pénalisés dans le cas où G a une forte variabilité est très faible. Nous avons comparé LAS avec SRPT (*Shortest Remaining Processing Time*), car, parmis les disciplines de service fonctions de la taille des clients, SRPT est optimale en terme de nombre moyen de clients actifs. Nous avons montré que LAS offrait des performances très similaires à celles de SRPT lorsque G est très variable. Nous avons aussi comparé LAS à FCFS (*First Come First Serve*) qui est la discipline de service des routeurs de l'Internet. Nous avons montré que LAS surpassait FCFS lorsque G est très variable. De plus LAS offre une fonctionnement dégradé en cas de surcharge, alors que FCFS est totalement instable.

Nous avons ensuite mené une étude à base de simulations pour déterminer les performances de LAS dans diverses configurations de réseaux paquets. Notamment, nous avons étudié les effets de LAS sur TCP. Nous avons trouvé que LAS interagit très bien avec TCP car il réduit le temps de transfert des flots courts et ne pénalise pas notablement les flots longs. De plus, nous avons montré que LAS ne souffrait pas des problèmes d'inéquité dans le partage de la bande passante (entre flots TCP et UDP par exemple) rencontrés avec FCFS.

Nos résultats de simulation montrent aussi que LAS pénalise trop les flots extrêmement longs lorsque la charge est trop élevée. Pour corriger ce problème, nous avons proposé une famille de politiques de service basées sur LAS et capables d'offir un meilleur service aux flots dits prioritaires. Pour évaluer ces politiques, nous avons développé des modèles analytiques de LAS et de ces politiques au niveau flot. Nous les avons validés par des simulations ns2 pour une large gamme de paramètres et un taux de perte faible. L'évaluation des politiques de service proposées montre qu'elles améliorent notablement la performance des flots prioritaires,

en terme de temps moyen de transfert, gigue et taux de perte, sans pour autant trop affecter les flots dits ordinaires. Finalement, nous avons évalué une politique de service LAS-FCFS qui sert les flots courts comme LAS et les flots longs comme FCFS. Cette politique peut être utile lorsque la pénalité infligée aux flots longs avec LAS est jugée inacceptable. Nous avons aussi proposé et évalué des variantes de LAS-FCFS capables d'offrir un service différentié.

# Contents

# List of Figures

# List of Tables

# Acronyms

Here are the main acronyms used in this document. The meaning of an acronym is usually indicated once, when it first occurs in the text.

| | |
|---|---|
| ADSL | Asymmetric Digital Subscriber Line |
| AF | Assured Forwarding |
| BB | Bandwidth Broker |
| BP | Bounded Pareto |
| CA | Congestion Avoidance |
| CATINT | Context Aware Transport/Network Internet Protocol |
| CBR | Constant Bit Rate |
| CDN | Content Distribution Network |
| cfmfv | continuous, finite mean, and finite variance |
| CoS | Class of Service |
| CR | Constraint-based Routing |
| DACS | Diffserv Admission Control Service |
| DF-LAS-FCFS | Differential hybrid of LAS and FCFS policies |
| EF | Expedited Forwarding |
| FB | Foreground Background |
| FCFS | First Come First Out |
| FIFO | First In First Out |
| FP-LAS-FCFS | Fixed Priority hybrid of LAS and FCFS policies |
| FTP | File Transfer Protocol |
| HTTP | Hypertext Transfer Protocol |
| IS-IS | Intermediate System-Intermediate System- |
| ISP | Internet Service Provider |
| ITO | Initial Retransmission TimeOut |
| NPP | Nonpreemptive Policies |
| LAS | Least Attained Service |
| LAS-FCFS | Hybrid of LAS and FCFS policies |
| LAS-fixed | LAS policy with constant priority function for priority flows |
| LAS-linear | LAS policy with linear priority function for priority flows |
| LAS-log | LAS policy with logarithmic priority function for priority flows |
| LAS-PS | A hybrid of LAS and PS |
| LIFO | Last In First Out |
| LSP | Label Switch Protocol |

| | |
|---|---|
| LSR | Label Switch Router |
| MPLS | MultiProtocol Label Switching |
| OSPF | Open Shortest Path First |
| P2P | Peer-to-Peer |
| pdf | probability density function |
| PHB | Per-Hop Behaviors |
| PS | Processor Sharing |
| QoS | Quality of Service |
| RED | Random Early Detection |
| RIP | Routing Information Protocol |
| RSVP | Resource ReserVation Protocol |
| RTO | Retransmission Timeout |
| RTT | Round Trip Time |
| SET | Shortest Elapsed Time |
| SS | Slow Start |
| SRPT | Shortest Remaining Processing Time |
| TCP | Transmission Control Protocol |
| TE | Traffic Engineering |
| UDP | User Datagram Protocol |
| VPN | Virtual Private Network |
| VoD | Video on Demand |

# Notations

Here is a list of the main notations and symbols used in this document. We have tried to keep consistent notations throughout the document, but some measures are denoted by more than one symbol depending on when they occur in the text.

| | |
|---|---|
| $x$ | Flow size or job size |
| $\lambda$ | Mean arrival rate |
| $\rho$ | Network load |
| $C$ | Coefficient of variation |
| $P$ | Maximum job or flow size for bounded Pareto distribution |
| $q$ | Probability of that a new arrival belongs to priority class |
| $k$ | Minimum job or flow size for bounded Pareto distribution |
| $\alpha$ | Slope parameter of the Pareto and Bounded Pareto distributions |
| $T(x)$ | Mean transfer time of a job or a flow of size $x$ |
| $S(x)$ | Mean slowdown of a job or a flow of size $x$ |
| $T$ | Mean transfer time |
| $S$ | Mean slowdown |
| $m_n$ | The nth moment of a distribution |
| $RTT$ | Round trip time |
| $L$ | Packet length (bits) |
| $R$ | Link speed (Mb/s) |
| $\overline{S}$ | Mean packet transmission time in a bottlenck link |

# Chapter 1

# Introduction

## 1.1 The Need for QoS Support in Internet

The existing Internet has its roots from ARPANET, an experimental data network funded by the U.S. Defense Advanced Research Projects Agency (DARPA) in the early 1960s. The ARPANET was built on the datagram model, where each packet carries the complete source and destination address and is individually delivered to its destination through the network. TCP/IP protocol stack in mid-1980s made it possible for different networks in the world to inter-connect and renders a universal service. A collection of interconnected networks world-wide is known as the Internet. The Internet was originally used by scientists for networking research and for exchange of information. Remote access, file transfer, and e-mail were popular applications and the datagram model works well for these applications. The service that the current Internet provides is often referred to as a *best effort*. Best-effort service represents the simplest type of service that a network can offer; it does not provide any form of guarantees to traffic flows. When a link is congested, packets are simply dropped when the queue overflows and TCP protocol assures that dropped packets are re-transmitted. Since the network treats all packets equally, any flows could get hit by the congestion. Although the best-effort service is adequate for some applications that can tolerate large delay variation and packet losses, it clearly does not satisfy the needs of many new applications such as multimedia applications. New architectures for resource allocation that support resource assurance and different levels of services are needed for the Internet to evolve into a multiservice network.

In the late 1980s, the International Telecommunication Union (ITU) selected a new networking paradigm known as Asynchronous Transfer Mode (ATM) for the Broadband-Integrated Service Digital Network (B-ISDN) [70, 101, 92]. The goal of ATM was to provide a universal multiservice network capable of supporting a wide range of applications with diverse networking requirements. ATM is a high performance, cell-oriented switching and multiplexing technology that utilizes fixed length packets to carry different types of traffic. ATM was designed to support data applications from Internet, voice from circuit-switching networks, and emerging multimedia applications in a single network. However, ATM was not universally deployed and

thus it failed to offer the services it was envisioned for on a truly global scale.

The 1990s witnessed an explosive use of Internet due to the emergence of the Web, which brought the Internet into homes and businesses of millions of people worldwide. The Web has fundamentally changed the Internet making it the world's largest public network. The Web also served as a platform for the deployment of new applications, such as online banking, streamed multimedia services, voice over IP (VoIP), tele-teaching, peer-to-peer (P2P), etc.

The successful evolution of Internet has brought new challenges. Many of the new applications have requirements different from those for which the Internet was originally designed. The datagram like model of the Internet has very limited resource management capabilities inside the network and cannot provide any resource guarantees to users. This implies that when some part of the Internet is overloaded the performance of all flows that traversing that part will be equally affected. Since Internet has become indispensable in our daily life and work, the lack of predictable performance is an issue that needs to be addressed. As a result, researchers have been trying to re-design the Internet so that it can simultaneously support different application types and satisfy their service requirements. This gave birth to the term *quality of service* (QoS) for Internet.

Internet QoS is defined as the management of available network resources to deliver consistent and predictable performance over the Internet in terms of latency, jitter, loss, throughput, and availability according to users service requirements. Internet QoS is a concept by which applications may indicate or negotiate their specific requirements to the network. Fundamentally, QoS enables either to guarantee the required service of all applications or to provide better services to certain applications to satisfy their minimum service requirements. This is accomplished by setting up *QoS mechanisms* in the network devices.

A QoS mechanism is a set of protocols designed to dictate network devices to service the competing applications in Internet differently by following a set of predefined *policies*. The task of the policies in a QoS mechanism is to determine when a network device should service packets of a particular application in the presence of packets from other contending applications. QoS mechanisms in Internet require changing either a part or all of the Internet. Thus, in order to provide Internet QoS it is necessary to provision or configure network devices to enable them to *classify flows* and to apply *buffer managements* and *packet scheduling algorithms*. In general, QoS mechanisms provide a set of tools to manage the use of network resources in a controlled and efficient manner.

There are a number of proposals for Internet QoS that differ in the QoS mechanisms they use. One is to use particular scheduling disciplines to serve packets in network routers. Scheduling algorithms have been studied for decades and many scheduling policies were first developed in the context of job scheduling in operating systems. The books by Conway [25], Coffman and Denning [23], and Kleinrock [64] describe and analyze a host of scheduling polices. In the last two decades, *packet* scheduling in routers has been an active area of research and most of the attention has been focused on *processor sharing* (PS) type of scheduling algorithms [76, 32, 109]. Processor Sharing has many interesting properties: It assures a max-min fair allocation of bandwidth among competing flows; provides protection against misbehaving flows [63]; and allows

to establish end-to-end delay bounds for individual flows [82]. Despite all these nice properties, PS scheduling has not been deployed. Instead, routers still implement *first-in first-out* (FIFO) (a.k.a. *first-come first-serve* (FCFS)) scheduling and possibly active queue management such as *random early detection* (RED).

Other examples of the proposed QoS solutions include *intergrated services* (Intserv) [59], *differentiated services* (Diffserv) [33], *multiprotocol label switching* (MPLS) [73], *constraint-based routing* (CR), *traffic engineering* (TE), and hybrids of two or more. In the rest of this chapter, we will first present the architectures of these previously proposed QoS solutions and discuss their main features that limit their deployment. We will then discuss motivation of the QoS solution proposed in this thesis work. Finally, we will conclude this section by briefly mentioning the thesis contributions and the outline the rest of the thesis.

## 1.2 QoS Architectures

### 1.2.1 Overprovisioning

One of the most obvious and the simplest QoS solution is to over-engineer the network by deploying ample of bandwidth to stay ahead of the demand curve. This approach is problematic since each time the bandwidth has increased, the demand has grown to consume it. Furthermore, this approach is not cost effective as it requires to install new infrastructure whenever an upgrade is necessary. Overprovisioning is commonly adopted by Internet Service Providers (ISPs) for one major reason: The lack of a reliable Internet traffic model to correctly estimate link utilization. Since the Internet traffic exhibits a large variation, overprovisioning to keep the average link utilization low is the widely used design solution at the moment to handle traffic variability incurred by the absence of access control and to protect the network against multiple failures [44].

### 1.2.2 Integrated Service (IntServ)

In Intserv [59], traffic is serviced in per-flow basis in accordance with each flow's absolute service request. Each network device must reserve resources for each flow and isolate each flow from another. In Intserv, *resource reservation protocol* (RSVP) signaling protocol is used by a host to request specific qualities of service from the network for particular application data streams or flows. RSVP is also used by routers to deliver quality-of-service (QoS) requests to all nodes along the path(s) of the flows and to establish and maintain state to provide the requested service. RSVP requests will generally result in resources being reserved in each node along the data path. A flow is either admitted to the network or rejected depending on the availability of enough network resources along the whole path to satisfy its service requirements. An admitted flow is guaranteed the service throughout its holding time on end-to-end basis. Once a flow of data is admitted to the network, RSVP uses soft state approach to manage resources in routers

and hosts. RSVP soft state is created and periodically refreshed using RSVP messages to avoid an explicit termination procedure to release resources used by a flow [17].

Intserv has a number of drawbacks that limit its deployment. With the explosive growth of internet, the number of flows whose state must be managed at a given instant by each core router is very large (tens of thousands). Isolation of flows in Intserv (having one logical buffer per-flow) may be very expensive when the number of flows is very large, which is normally the case in core routers. The other problem that hinders deployment of Intserv is backward incompatibility; Network devices are required to perform and understand a signaling protocol (RSVP) that is not yet widely deployed. Moreover, the "mice and elephants" paradigm of Internet flows means that most flows are very small. It is obvious that performing RSVP signaling on per-flow basis for this traffic will be a very complex task. In general, Intserv does not scale particularly in core routers. The failure of Intserv resulted to another QoS architecture known as differentiated service (Diffserv).

### 1.2.3 Differentiated Service (Diffserv)

Diffserv is a QoS framework that aggregates individual flows with similar traffic behaviors into a small number of classes and provides relative services among these classes [33]. Diffserv alleviates the complexity in core routers by assigning all complex processing such as flows classification and traffic dimensioning to edge routers. In the backbone, forwarding *per-hop behaviors* (PHB) are defined, namely, *expedited forwarding* (EF) and *assured-forwarding* (AF) [57, 61]. EF-PHB has a higher priority than AF-PHB. EF per-hop behavior is proposed for traffic that requires zero loss and zero jitter or queueing delay guarantees. EF-PHB is proposed to emulate leased lines connections. There are four AF-PHBs each with three drop precedences. Differentiated services are managed along the path by scheduling algorithms and buffer management mechanisms in routers. Priority queueing assures EF traffic is not delayed and buffer mechanisms choose the AF traffic from which to drop packets that are marked to have low priorities during congestion. No admission control or end-to-end complex signaling protocol are required in the Diffserv framework. Instead, in Diffserv, a flow that arrives at the edge router is classified and conditioned according to its traffic contract and is always accepted to the network.

Diffserv has a number of drawbacks too despite being scalable and not requiring a signaling protocol. When a single flow in a class violates its traffic contract the services of all other flows of the same class degrade equally. Diffserv does not provide absolute guarantees and during congestion, a high priority AF class may be demoted to lower priority AF class. Another drawback of Diffserv is that it does not provide end-to-end guarantees, which may be required by some applications.

An architecture proposed to enable Diffserv to support end-to-end QoS services is called *bandwidth broker* (BB) [111]. BB is a centralized network manager that makes it possible to manage network state on a domain basis rather that router basis. BB has a policy database that tracks user behaviors and allows admission control [77]. BB also maintains service agreements and negotiations among different adjacent domains. The work on BB is still new and its deployment is highly doubted because it is against the internet philosophy as being a distributed

system.

Another proposed scalable QoS architecture aimed at supporting end-to-end services is the hybrid of Intserv and Diffserv [15]. The scalability of Diffserv makes it suitable for large networks. In this framework, Intserv and Diffserv are used together to meet the needs of large ISPs who manage the backbones on the Internet and the need of the QoS to end users. In this hybrid framework, it is envisioned that Intserv will be deployed in stub networks and Diffserv in core networks. The RSVP signaling protocol is proposed to reserve resources in stub networks and some *Diffserv admission control service* (DACS) is proposed at the edge or boundary routers to control admission of flows to the Diffserv domain.

For Intserv, Diffserv, or a hybrid of both architectures to work, they need to be deployed within all ISP domains. Such investment can only be driven by content providers that would target new markets like *voice on demand* (VoD) or high quality TV. At the moment, content providers use ad-hoc solutions such as *content delivery networks* (CDNs) [20, 4] and ADSL to reach customers. Using ADSL implies that video and TV delivery does not use servers deployed in the network. Content providers are currently satisfied with these ad-hoc solutions and thus, they are not willing to invest for the wide deployment of Intserv or Diffserv like solutions within the core.

### 1.2.4 Multiprotocol Label Switching (MPLS)

*Multiprotocol label switching* (MPLS) was motivated by the need to reduce the processing complexity in legacy core routers. It is basically a simple forwarding scheme, where a fixed sized MPLS header is inserted between link layer header and network layer header in an IP packet [73, 102]. A *label switching protocol* (LSP) or extended RSVP is the signaling protocol used to setup an MPLS path. After path setup and if a flow is admitted, *label switch routers* (LSRs) only swap labels in packets headers to forward the packets through a completely determined path (tunnel) in an MPLS domain. MPLS is useful to provide both faster packet forwarding and efficient tunneling mechanisms. MPLS is needed for large networks and service providers to provide backup routers and load balancing. As opposed to Intserv and Diffserv, MPLS is actually used by ISPs to manage traffic instead of passively relying on the routing protocols [11], and also to deploy *virtual private networks* (VPN) [74, 85, 110]. For these applications though, MPLS is not directly targeted to offer QoS to end users even if QoS features exist.

The QoS support features of MPLS are its 3-bit class of service (CoS) field in MPLS header and its ability to determine path and reserve resources using LSR signaling protocol [106].

The CoS field enables a hybrid QoS architecture from Diffserv and MPLS architectures known as *Diffserv-aware* MPLS. In Diffserv-aware MPLS architecture, MPLS is proposed in transit networks where it further aggregates Diffserv classes into a smaller number of classes called *class types* [37]. The operations of routers in this architecture are basically the same as in Diffserv architecture with a few differences: At the ingress router an MPLS header is inserted into the packet; core routers process the packet based on its label and CoS field; and at the egress the MPLS header is removed from the packet. In this scheme, MPLS effects are confined within

the backbone networks. Allocation and management of resources in this architecture are also proposed to be handled by bandwidth broker [106].

### 1.2.5 Traffic Engineering and Constraint-Based Routing

One of the causes of network congestion may be due to uneven traffic distribution resulting from the shortest path based routing protocols (OSPF, RIP, and IS-IS). When traffic is unevenly distributed, some links remain underutilized while others are overloaded. MPLS provides the mechanisms to control traffic distribution in the Internet. More advanced techniques called traffic engineering (TE) and constraint-based routing (CR) are also proposed [98]. Traffic engineering is the process by which network congestion is avoided by arranging how traffic should flow through the network. Constraint-based routing is the major tool that makes TE automatic [106]. Given QoS requests, CR returns a route that will most likely be able to meet the QoS requirements. In addition to QoS requirements, CR also takes into account ISP policies such as increasing network utilization by evenly distributing traffic in the network. Towards this end, the CR may find a longer path but lightly loaded and better than the heavily loaded shortest path.

### 1.2.6 Measurement Based Admission Control

The measurement admission control approach is yet another mechanism that can enable the Internet to support QoS. In this approach, network state in terms of congestion level is measured and a flow request is accepted based on the results of the measurements. Examples of the measurement based approach include *probing* in which the source, edge devices, or bandwidth broker send probe packets to the network before sending data packets and the receiver sends probe records (e.g., the number of successfully received probe packets) back to the prober. The flow is admitted or rejected based on the probe records. This scenario is one of the simplest and can be easily deployed in the current internet because it makes use of the existing infrastructure. It does not cause more processing in the network devices neither does it require any extra support from the network. The problem with this method is that it is shown to support a very limited range of applications [19, 35]. It is also unreliable due to the bursty and dynamic nature of Internet traffic.

## 1.3 Motivation of our QoS Solution

The fact that the proposed QoS solutions either have not been implemented or are not implementable in the current Internet and the continuing need of QoS are the main motivations for our thesis work. Furthermore, recent studies in network tomography show that most of Internet contention today is observed at the edges of the network, and traffic at the core of the network experiences very small loss rates and negligible queueing delays [80]. This is because access

links normally are low speed and most of the links at the network backbone are overprovisioned to very high speeds. This motivated us to look for new QoS mechanisms that require deployment only at network edges. Our QoS solutions are motivated by recent observations of characteristics of Internet traffic in terms of their flow size distributions that we discuss in detail in the next section. The goal of QoS in this thesis is to improve the overall user perceived performance in Internet. The solutions proposed in this thesis achieves low average loss rates and transfer times for flows, assures a fair bandwidth allocation among competing flows, and differentiates the service of defined classes of flows in order to ensure that priority flows satisfy their service requirements.

### 1.3.1 Internet Traffic Characteristics

Measurement studies have shown that Internet traffic exhibits the mass disparity phenomenon: many flows are short with 10-20 packets on average and about 1% of the largest flows, contain a large fraction of the mass (bytes). This phenomenon is sometimes referred to as *mice and elephants* and has been observed in various aspects of Internet traffic specifically flow sizes, session sizes, session durations, FTP or P2P data transfers, and telnet packet interarrival times [84, 22] and in Web servers [14, 26]. The short flows in Internet traffic are mainly Web data transfers triggered by user interaction. As the use of Web remains popular, Web traffic will continue to make up a significant fraction of the Internet traffic. On the other hand, most of the long flows are known to originate from peer-to-peer applications [99].

This Internet traffic characteristic has been referred to as the *heavy-tailed property* [55, 13, 28]. Downey shows in [34] that the Internet data set complies better to a lognormal distribution than to a heavy-tailed distribution. In particular, the results in [47] reveal that it is not easy to decide whether the Internet traffic does fit a heavy-tailed distributions or not. Instead the work in [47] shows that the Internet traffic model is not necessarily heavy-tailed; rather, it fits many distributions that have high *coefficient of variation* ($C$). Defined as the ratio of the standard deviation to the mean of a distribution, $C$ is a common metric to measure the variability of a distribution, and the higher the $C$ value of a distribution the higher the variability of the distribution. Thus, we call this Internet traffic characteristic our thesis the *high variability property*. A number of distributions such as Pareto, lognormal, hyper-exponential, Weilbull [26], inverse Gaussian, and hybrid of lognormal have been shown to model different Internet traffic traces with a high variability property when their coefficient of variation is significantly greater than 1. Typical examples of C as observed in the internet traffic range between 5–20 [8, 26, 41].

Different researchers tried to exploit the high variability property of Internet traffic to improve systems performance. Examples include load balancing in distributed systems [52, 30, 29], switching and routing in the Internet [93, 39], and scheduling in Web servers [51, 27]. In contrast, when not accounted for during system design, the high variability property of Internet traffic can significantly affect the performance of the system as we will see in the next section.

### 1.3.2 The Impact of TCP and FIFO scheduling on the Performance of Short Flows

Web traffic, which contributes to a large fraction of short flows in Internet, is transferred using the TCP transport protocol. A look at the behavior of TCP reveals the factors that affect the performance of short flows: Regardless of the available bandwidth in the network, TCP **conservatively** initializes its congestion window (cwnd) to one packet and then doubles cwnd for every window worth of ACKs received until congestion is detected (this behavior is referred to as slow start (SS)). Thus, even if the network has sufficient bandwidth, the duration of a transfer depends very much on the round trip time (RTT) of the connection; Short TCP flows also have poor loss recovery performance [12] since they often do not have enough packets in transit to trigger the triple duplicate ACK mechanism (fast retransmission). Instead, they must rely on the retransmission timeout (RTO) to detect losses. Packet retransmissions that are triggered by a RTO can significantly prolong the transmission time of a short TCP flow; TCP uses its own packets to sample the round trip times and to estimate the appropriate RTO value. When a flow starts the initial RTO value is usually set to 3 seconds. Thus, losing packets in the SS phase can significantly prolong the transmission time of short flows.

FIFO is the scheduling discipline that is currently widely used in the Internet. An arriving packet to a FIFO queue with droptail or RED buffer management joins at the tail of the queue and has to wait for all packets it sees in the queue upon its arrival to be serviced before it can leave the queue. Thus, FIFO with droptail and RED don't discriminate arriving packets to a full buffer according to the sizes of their flows. Therefore, short flows can experience packet losses under FIFO with droptail or RED buffer management, which significantly affects their transfer times. Thus, in addition to TCP algorithm, FIFO scheduling that uses droptail or RED buffer management penalize short flows. As a consequence, the overall user perceived performance under FIFO is reduced since short flows make up a very large fraction of all flows in the Internet today.

The goal of this work is to look for an alternative scheduling policy to FIFO scheduling that can be used in packet networks in order to improve the overall user perceived performance by favoring short flows without penalizing the performance of long flows too much. Examples of such policies are size-based scheduling policies that give high service priorities to short flows and low service priorities to long flows. We discuss size-based scheduling in the next section.

## 1.4 Size-based Scheduling

### 1.4.1 Related Work

Size-based scheduling policies use a notion of size to decide which job to service next. Examples of size-based scheduling policies are *shortest job first* (SJF), *shortest remaining processing time* (SRPT), *least attained service* (LAS) [64], and *fair sojourn protocol* (FSP) [46]. These policies (except the FSP) have been known for several decades, but they have not been pro-

posed before for packet networks. The reason being that when analyzed under a M/M/1 queue, where the service times don't exhibit high variability property these policies are known to favor short jobs but penalize the largest jobs a lot. The observation of high variability property has made reseachers to revisit size-based policies and study their performances under a M/G/1 queue, where G is a general job size distribution with varying degree of variability [13].

SJF is a non-preemptive scheduling policy that gives service to the shortest job. A newly arriving shortest job must wait for the job in service to complete before it can receive the service. SRPT is a preemptive version of SJF, which favors short jobs by giving service to a job that has the shortest remaining processing time. In order to know the remaining processing time though, one needs to know the total service required by the job. This requirement limits implementation of SRPT in packet networks to environments where flow sizes are known a priori, such as in Web servers [53]. Unfairness analysis of SRPT with respect to variability of job size distribution is presented in [13]. The analytical results in [13] show that the degree of unfairness under SRPT is very small and in general it depends on the variability of a job size distribution. FSP is a fairly recent scheduling discipline that is made up from a hybrid of PS and SRPT [46]. FSP services a job with the shortest completion time under PS discipline, therefore it has similar deployment limitations in packet networks as have PS and SRPT .

Motivated by the high variability property of the Internet traffic, a number of new two-queue based policies that give service priority to short flows have also been proposed [48, 21, 69, 108, 10]. In these policies, a threshold of service (say in number of packets) is set. The first threshold number of packets of a flow are enqueued in the first queue and the remaining packets of a flow, if exist, are enqueued in the second queue. Fixed priority scheduling is used to service packets between the queues so that packets in the second queue receive service only if the first queue is idle. Packets in individual queues are serviced in FIFO order. In addition to priority scheduling, some of these policies use preferential dropping mechanisms among packets in the two queues where packets in the second queue have a higher probability to be dropped than packets in the first queue when the buffer is full. These disciplines are known to slightly reduce the mean transfer times of the short flows compared to FIFO scheduling, but their main challenge is the threshold that is difficult to tune to capture all short flows. This thesis proposes to use the LAS scheduling discipline in packet networks to improve the user perceived performance in terms of mean transfer times and loss rates.

### 1.4.2 LAS Scheduling

In queueing theory, LAS is known as a multi-level preemptive scheduling policy that gives service to the job in the system that has received the least service. In the event of ties, the set of jobs having received the least service shares the processor in a processor-sharing (PS) mode [64]. A newly arriving job always preempts the job (or jobs) currently in service and retains the processor until it departs, or until the next arrival appears, or until it has obtained an amount of service equal to that received by the job(s) preempted on arrival, whichever occurs first. An implementation of LAS needs to know the amount of service delivered to each job, which can be easily obtained from the server. LAS is also known in the literature under the names of

*foreground-background* (FB) [24, 64] or *shortest elapsed time* (SET) first scheduling [23]. In this work, we investigate LAS scheduling in packet networks as a means to reduce the transfer times of short flows and to improve the overall user perceived performance.

In packet networks, we consider the performance of a flow of packets, which differs from a job. The term job is commonly used in scheduling theory to denote a piece of workload that arrives to the system all at once. Given this definition, a flow in a packet network cannot be considered as a job since a flow does not arrive at the system at once. Instead, the source transmits a flow as a sequence of packets, possibly spaced out in time, that are in turn statistically multiplexed with packets from other flows. Also, when TCP is used as a transport protocol, due to the closed-loop nature of TCP, the treatment given to an individual packet may affect the temporal behavior of the remaining packets in the flow. For these reasons, it is not clear if the analytical results on LAS derived for jobs can be directly be applied to flows.

In the context of flows, the next packet serviced by LAS is the one that belongs to the flow that has received the least service. By this definition, LAS gives all the available bandwidth to a newly arriving flow until it receives an amount of service equal to the least amount of service received by any flow in the system before its arrival. If two or more flows have received an equal amount of service, they share the system resources fairly in a round robin fashion.

In packet networks, we can expect LAS to interact well with TCP in order to favor short flows. That is, by inserting the first packets of a flow at the head of the queue, LAS reduces the RTTs of the flow in slow start phase. This will help reduce the transfer times of short flows. Similarly, by giving service to a flow that has received the least amount of service, LAS can actually prevent any flow in the network to unfairly hog link bandwidth. Moreover, under LAS a packet arriving to a full buffer is first inserted according to its priority in the queue and then the packet with the lowest priority is dropped from the queue. This can avoid packet losses seen by short flows and can prevent the use of RTO to recover packet losses. Note that the flows that experience losses under LAS are likely to be flows that have sent many packets. These flows are likely to be in the CA phase, and they can use first re-transmission mechanisms to recover losses. These features are likely to make LAS a suitable scheduling discipline in packet networks to favor short flows without affecting too much the few largest flows.

## 1.5 Thesis Contributions and Outline

This thesis makes several contributions. The first contribution is the analysis of a LAS scheduling under the M/G/1 queue, where G is a job size distribution with a high variability property. When analyzed under the M/M/1 queue, LAS is known to favor short jobs and to penalize the largest jobs a lot. Perhaps this was the first factor that prevented considering LAS disciplines in packet networks. In Chapter 2 we analyze the performance of LAS scheduling under the M/G/1 queue. We compare LAS to a wide range of other policies such as PS, SRPT, and non-preemptive policies (NPP) such as FIFO, LIFO, etc. We evaluate the performance of LAS under job size distributions with varying coefficient of variations. We also establish the performance of LAS at overload $\rho \geq 1$. In general, we show that LAS reduces the response time of short

jobs without highly penalizing the largest jobs when the job size distribution exhibits a high variability property.

The second contribution is the performance analysis of LAS in packet networks. We study the interaction of LAS to TCP protocols, and evaluate the performance of LAS for TCP and UDP flows of varying sizes in Chapter 3. We simulate LAS in network simulator [1] and compare the performance of LAS to that of FIFO scheduling for flows that share a single common bottleneck link using generated traffic that exhibits the high variability property. Our simulation results show that under LAS scheduling, as compared to FIFO, the transmission time of short flows is significantly reduced, and most of short flows under LAS experience no loss. We also show that the improvement seen by short TCP flows under LAS is mainly due to the way LAS interacts with TCP algorithm in the slow start (SS) phase, which results in shorter round-trip times and zero packet loss rates for short flows. The results show that while reducing the transfer times of short flows, LAS offers similar transfer times for large flows as does FIFO. We also investigate the performance of long-lived TCP and UDP flows under LAS and compare the results to FIFO scheduling. Simulation results show that LAS offers similar performance to FIFO for long-lived flows at moderate network load. The performance of long-lived flows under LAS can deteriorate only at high load above 75%.

The third contribution of this thesis is the performance evaluation of LAS in heterogeneous networks, which is conducted in Chapter 4. Heterogeneous TCP networks with FIFO scheduling are known to allow some flows to unfairly utilize a large fraction of the available bandwidth. These networks include TCP networks with flows that have varying RTTs, networks that support TCP and UDP applications, and networks with flows that traverse multiple links. For the networks considered in Chapter 4, simulation results show that LAS is more effective than FIFO to offer close to fair sharing of network bandwidth among competing connections.

The fourth contribution of this thesis is the modeling, design, and analysis of new class-based scheduling disciplines that favor short flows and differentiate the service of some long-lived flows. Long-lived flows may carry important data that can not tolerate the performance offered to it by LAS scheduling. The main goal of these new policies is to guarantee an improved and accepted service to long-lived flows that are classified as priority flows.

The fifth contribution is the validation of analytical models of FCFS, LAS, and proposed new class-based policies for packet networks. The simulation results show that the analytic expressions for mean flow transfer time vs. flow size agree with the simulation results of the policies in packet networks. This contribution shows that the analytical models, instead of simulations, can directly be used to easily evaluate the performance of the policies in packet networks. Analytically evaluating models is less time consuming and can give results to a wider range of experiments than simulations of the models do. Results of the proposed models show that they can improve the performance of long priority flows without hurting the performance of short flows. The fourth and fifth contributions of this thesis are presented in Chapter 5

The final contribution, which is found in Chapter 6, is the performance analysis and modeling of a hybrid policy that services short flows using LAS and long flows using FCFS. We denote this policy as **LAS-FCFS**, and its model in packet networks as **LAS-PS**. LAS-FCFS is

useful to improve the performance of transfer times of all long flows while keeping the transfer times of short flows as under LAS. We show that the performance of LAS-FCFS improves with the increase in variability of a flow size distribution. We also analyzed variants of LAS-FCFS policies that offer service differentiation.

# Chapter 2

# Analysis of *LAS* Scheduling for Job Size Distributions with High Variance

Recent studies of Internet traffic have shown that flow size distributions often exhibit a high variability property in the sense that most of the flows are short and more than half of the total load is constituted by a small percentage of the largest flows[84, 22]. In the light of this observation, it is interesting to revisit scheduling policies that are known to favor small jobs in order to quantify the benefit for small jobs and the penalty for large jobs. Among all scheduling policies that do not require knowledge of job size, LAS scheduling policy is known to favor small jobs the most.

When analyzed under M/M/1 queue, LAS is known to reduce the response times of short jobs at the expense of significantly penalizing long jobs. In this chapter, we investigate the M/G/1/LAS queue for both, load $\rho < 1$ and $\rho \geq 1$, when $G$ is either a general job size distribution or a job size distribution with a wide range of variability. Our analysis shows that for job size distributions with a high variability property, LAS favors short jobs with a negligible penalty to the few largest jobs, and that LAS achieves a mean response time over all jobs that is close to the mean response time achieved by SRPT. SRPT is the optimal policy in terms of reducing the mean response time.

## 2.1 Introduction

Internet traffic studies revealed that Internet traffic consists of many short flows and that a tiny fraction of the largest flows constitutes more than half of the total load [26, 84]. It is interesting to study the performance of scheduling policies in the light of these findings to see if one can improve the performance of short jobs without penalizing too much the long jobs. We say that a job size distribution exhibits the high variability property if less than 1% of the largest jobs account for more than half of the load. This property has also been referred to as a heavy-tail property [27, 13], but it is not restricted to heavy-tail distributions [47]. In this paper, we analyze

the impact of variability of job size distributions on the performance of LAS scheduling policy.

*Response time* and *slowdown* are commonly used as performance metrics to analyze scheduling policies. Response time is the overall time a job spends in the system. Slowdown is the normalized response time, i.e., the ratio of the response time of a job to the size of that job. In particular, we use the *conditional mean response time* and the *conditional mean slowdown*, which (for a job of size $x$) are defined as $T(x) \triangleq E[T|X = x]$ and $S(x) \triangleq E[S|X = x]$ respectively. We also use the *mean response time* and the *mean slowdown* defined as $T \triangleq \int_0^\infty T(x)f(x)dx$ and $S \triangleq \int_0^\infty S(x)f(x)dx$ respectively. Slowdown is a useful metric to analyze fairness of a scheduling policy [13, 103]. Processor Sharing (PS) is known to be a fair policy since it offers the same slowdown to all jobs.

The shortest remaining processing time (SRPT) scheduling policy is proven [75, 90] to be the optimal policy for minimizing the mean response time. It has been known for a long time that for negative exponentially distributed job sizes, SRPT severely penalizes large jobs. However, Bansal and Harchol-Balter [13] have recently shown that the performance of SRPT and the penalty experienced by the largest jobs very much depend on the characteristics of the job size distribution. In particular, for some load values and for job size distributions with the high variability property, SPRT does not increase the mean response time of long jobs at all as compared to PS, i.e., all jobs have a lower conditional mean slowdown under SRPT than under PS. Nevertheless, SRPT has the drawback that it needs to know the size of the jobs. While the job size is known, for instance, in a Web server [53], it is generally not known in environments such as in Internet routers. An implementation of LAS scheduling does not need to know the jobs jobs sizes. Instead, it only needs to track the amount of service each job has received, an information that can be easily obtained from the server.

The expression for the conditional mean response time $T(x)$ for an M/G/1 queue served using LAS has been originally derived in [89], and was re-derived in [105, 23, 65] as well. However, the expression for $T(x)$ is complex and difficult to evaluate numerically, and therefore very little has been done to evaluate the performance of LAS with respect to the variability of the job size distribution. The variability of a job size distribution can be expressed in terms of its *coefficient of variation* ($C$), which is defined as the ratio of the standard deviation to the mean of the distribution. For a given mean, the higher the variance, the higher the $C$ value the higher the variability of the distribution. Coffman ([23], pp. 188-187), gives an intuitive result that compares the mean response times for LAS and PS: The mean response time of LAS is lower than the mean response time of PS for job size distributions with $C$ greater than 1, and it is higher for job size distributions with $C$ less than 1. Kleinrock ([64], pp. 196-197) compares the mean waiting times of LAS and FIFO scheduling for an M/M/1 queue and observes that LAS offers lower waiting times to short jobs while large jobs see significantly higher waiting times. The analysis of $M/G/1/LAS$ queue in steady state was done by Schassberger [87, 88]. However, Schassberger did not evaluate the impact of the variance of the service time distribution. Nuijens recently analyzed several performance aspects of LAS queue such as stationary queue length, maximum queue length, departure process, and sojourn time [78]. Nuijens showed that as compared to FIFO, LAS outperforms FIFO when a job size distribution exhibits the high variability property. In a recent work, Balter et al. [54] show that for an M/G/1 queue, the conditional mean slowdown of all work-conserving scheduling policies, which includes LAS, asymptoti-

cally converges to the conditional mean slowdown of PS, i.e. $\lim_{x\to\infty} S(x)_{LAS} = 1/(1-\rho)$.

This chapter investigates the performance of LAS considering the variability of the job size distribution. We first compare LAS to PS to analyze its unfairness. We derive an upper bound on the unfairness of LAS for a general job size distribution (Theorem 2.3.1). In particular, for some job size distributions that exhibit the high variability property, we observe that more than 99% of the jobs see their mean slowdown significantly reduced under LAS and less than 1% of the largest jobs experience a small increase of their mean slowdown under LAS as compared to PS. We also derive an upper bound that compares the mean response times of LAS and nonpreemptive policies (NPP) (Lemma 2.3.2).

While both, LAS and SRPT, favor short jobs, it is not known how the performance of LAS compares to the optimal policy SRPT. We provide mathematical expressions that compare these two policies and show that for some job size distributions with the high variability property, both policies offer very similar mean response times to all jobs (Theorem 2.3.5).

We also consider the overload case, i.e. $\rho \geq 1$, and prove the existence of a job size $x_{LAS}(\lambda)$ such that all jobs strictly smaller in size than $x_{LAS}(\lambda)$ will have a finite mean response time under LAS (Theorem 2.4.1). We then present a closed form expression for the conditional mean response time of LAS under overload (Theorem 2.4.2). We evaluate this expression by comparing the performance of LAS with PS and SRPT for $\rho \geq 1$. PS is known to exhibit an infinite mean response time under overload as opposed to LAS, which offers finite mean response time for job sizes less than $x_{LAS}(\lambda)$. The results are intriguing; for the job size distribution with the high variability property considered in this chapter, at overload of $\rho = 2$, jobs of size up to the 99th percentile[1] have a finite mean response time under LAS.

The rest of the chapter is organized as follows: In the next section we discuss the mathematical background necessary for the analysis carried out in the rest of the chapter. We analytically compare LAS to PS, to nonpreemptive (NPP) policies, and to SRPT in Section 2.3. In Section 2.4, we investigate the performance of LAS under overload. We conclude the chapter in Section 2.5.

## 2.2 Mathematical Background

In this section we first present mathematical expressions of the conditional mean response time and the conditional mean slowdown for different scheduling policies. Then we discuss the reasons for the choice of the particular job size distributions used in this chapter.

[1]The $p$th percentile is the flow size $x_p$ such that $F_x(x_p) = p/100$, where $F_x(x)$ is the cumulative distribution function of $x$.

### 2.2.1 Expressions for the Performance Metrics

Let the average job arrival rate be $\lambda$. Assume a job size distribution $X$ with a pdf (probability density function) denoted as $f(x)$. The abbreviation c.f.m.f.v is used to denote continuous, finite mean, and finite variance. Given the cumulative distribution function as $F(x) \triangleq \int_0^x f(t)dt$, we denote the survivor function of $X$ as $F^c(x) \triangleq 1 - F(x)$. We define $m_n(x)$ as $m_n(x) \triangleq \int_0^x t^n f(t)dt$. Thus $m_1 \triangleq m_1(\infty)$ is the mean and $m_2 \triangleq m_2(\infty)$ is the second moment of the job size distribution. The load of jobs with size less than or equal to $x$ is given as $\rho(x) \triangleq \lambda \int_0^x t f(t)dt$, and $\rho \triangleq \rho(\infty)$ is the total load in the system.

We assume an M/G/1 queue in this chapter, where $G$ is a c.f.m.f.v distribution and $M$ stands for memoryless distribution of the interarrival times. The Poisson distribution is known to closely model Internet flows interarrival times [45]. The expression of $T(x)$ for M/G/1/SRPT [91] can be decomposed into the conditional mean waiting time $W(x)$ (the time between the instant when a job arrives at the system until it receives the service for the first time) and the conditional mean residence time $R(x)$ (the time a job takes to complete service once the server has started serving it). Hence,

$$T(x)_{SRPT} = W(x)_{SRPT} + R(x)_{SRPT}$$

$$W(x)_{SRPT} = \frac{\lambda(m_2(x) + x^2 F^c(x))}{2(1-\rho(x))^2} \tag{2.1}$$

$$R(x)_{SRPT} = \int_0^x \frac{1}{1-\rho(t)} d(t) \tag{2.2}$$

Similarly, we decompose the expression of $T(x)$ for $M/G/1/LAS$ into two terms. We denote the first term as $\tilde{W}(x)_{LAS}$ and the second term as $\tilde{R}(x)_{LAS}$. Note that these terms do not denote the conditional mean waiting and residence times for LAS, rather they are introduced to ease the analysis in Section 2.3.3 where we compare LAS to SRPT. The formulas for $T(x)_{LAS}$ are given in [89],

$$T(x)_{LAS} = \tilde{W}(x)_{LAS} + \tilde{R}(x)_{LAS}$$

$$\tilde{W}(x)_{LAS} = \frac{\lambda(m_2(x) + x^2 F^c(x))}{2(1-\rho(x) - \lambda x F^c(x))^2} \tag{2.3}$$

$$\tilde{R}(x)_{LAS} = \frac{x}{1-\rho(x) - \lambda x F^c(x)} \tag{2.4}$$

Finally, the formulas of $T(x)$ for the M/G/1/PS and M/G/1/NPP queues [25], where NPP stands for any nonpreemptive policy are:

$$T(x)_{PS} = \frac{x}{1-\rho} \tag{2.5}$$

$$T(x)_{NPP} = \frac{\lambda m_2}{2(1-\rho)} + x \tag{2.6}$$

Examples of NPP policies include FIFO, *last-in first-out* (LIFO), RANDOM, etc. The definition of $S(x)$ reveals that for two scheduling policies A and B, the ratio $\frac{T(x)_A}{T(x)_B} = \frac{T(x)_A/x}{T(x)_B/x} = \frac{S(x)_A}{S(x)_B}$.

### 2.2.2 The Choice of the Job Size Distribution

We will analyze LAS by using a general c.f.m.f.v job size distribution and some specific job size distributions. The choice of specific job size distributions is motivated by the characteristics of the Internet traffic where *most of the flows are short and more than half of the load is due to a tiny fraction of the largest flows*. Different distributions have been shown to model the empirical Internet traffic given their coefficient of variation is larger than 1 [8, 26, 41]. These distributions include Pareto, bounded Pareto, lognormal distributions, hyper-exponential, Weibull, inverse Gaussian, and hybrid of lognormal and Pareto distributions. Let $X$ be a job size distribution, the pdf of the Pareto distribution is given as:

$$f(x) = \alpha k^{\alpha} x^{-\alpha-1}, \qquad x \geq k, 0 \leq \alpha \leq 2 \tag{2.7}$$

where $k$ is the minimum job size and $\alpha$ is the exponent of the power law. The cumulative distribution function of Pareto distribution $F(x)$ is thus:

$$F(x) = 1 - \left(\frac{a}{x}\right)^{\alpha}, \qquad x \geq k. \tag{2.8}$$

In this thesis, we use variants of the Pareto distribution, Pareto distribution of second kind and bounded Pareto distribution, for job size or flow size distributions that exhibits the high variability property. The Pareto distribution of second kind is a shifted variant of Pareto distribution with samples starting from 0 instead of $k$. The probability density function of the Pareto distribution of the second kind is given as:

$$f(x) = \alpha a^{\alpha}(x+a)^{-\alpha-1}, \qquad x \geq 0, 0 \leq \alpha \leq 2, a > 0 \tag{2.9}$$

where is the $a$ is the scale factor . We will often use this distribution in simulations to generate flows of varying sizes starting from flows of one packet. The distribution function of Pareto distribution of the second kind is given as:

$$F(x) = 1 - \left(\frac{a}{x+a}\right)^{\alpha}, \qquad x \geq 0. \tag{2.10}$$

Bounded Pareto (BP) distribution is commonly used in analysis because it can have a high variance and thus it can exhibit the high variability property as observed in the Internet traffic and also because the maximum job size can be set to mimic the largest observed Internet flow sizes [107, 27, 13]. In this chapter, we also use the bounded Pareto job size distribution for the same reasons. We denote the bounded Pareto distribution by $BP(k, P, \alpha)$, where $k$ and $P$ are the minimum and the maximum job sizes and $\alpha$ is the exponent of the power law. The pdf of the Pareto is given as

$$f(x) = \frac{\alpha k^{\alpha}}{1-(k/P)^{\alpha}} x^{-\alpha-1}, \qquad k \leq x \leq P, 0 \leq \alpha \leq 2 \tag{2.11}$$

The cumulative distribution function $F(x)$ and the n-th moment $m_n$ of the $BP(k, P, \alpha)$ distribution are:

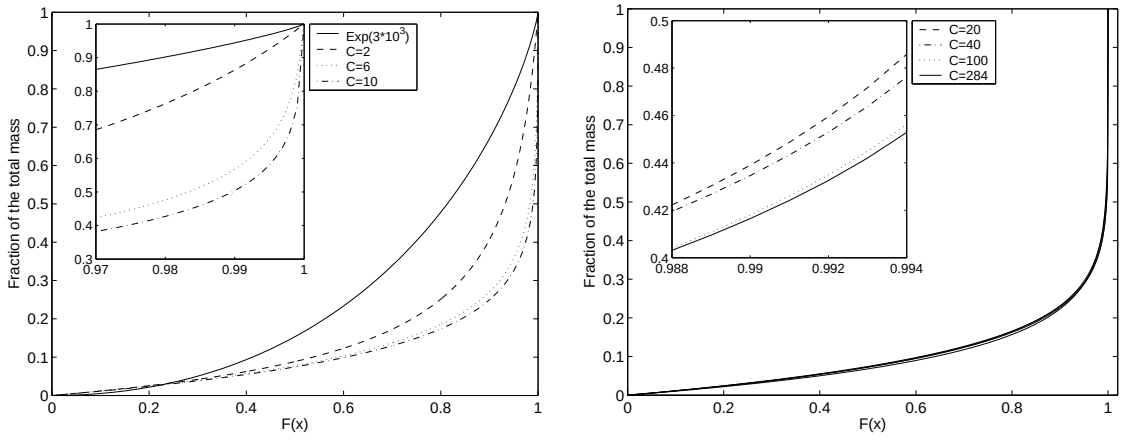
$$\begin{aligned} F(x) &= \frac{1}{1-(k/P)^{\alpha}}[1-(k/x)^{\alpha}], \quad k \leq x \leq P, 0 \leq \alpha \leq 2 \\ m_n &= \frac{\alpha}{(n-\alpha)(P^{\alpha}-k^{\alpha})}(P^n k^{\alpha} - k^n P^{\alpha}) \end{aligned}$$

Hence, the expression for coefficient of variation is $C \triangleq \frac{\sqrt{m_2 - m_1^2}}{m_1}$. To obtain different values of $C$ for $BP(k, P, \alpha)$, one needs to change one or more of the parameters of the BP distribution, i.e., $k$, $\alpha$, and $P$.

We will also consider exponentially distributed job sizes to see the performance difference when LAS is analyzed under M/M/1 queue model. The $C$ value of an exponential distribution is 1. We denote the exponential distribution with mean of $1/\mu$ by $Exp(1/\mu)$. The pdf of the exponential distribution is given as:

$$f(x)_{Exp} = \mu e^{-\mu x}, \qquad x \geq 0, \mu \geq 0$$

The variability of a distribution can be determined using the *mass-weighted function* ($M_w(x)$) [28], which (for a job of size $x$) is defined as the fraction of the total load constituted by jobs of size less than or equal to $x$ or $M_w(x) \triangleq \frac{\rho(x)}{\rho}$. We plot $M_w(x)$ as a function of $F(x)$ to see the fraction of total mass constituted by jobs of size less than or equal to $x$. The mass-weighted function will help us to see if a job size distribution exhibits the high variability property.




(a) $Exp(3 \times 10^3)$ *and BP with* $C = 2, 6, 10$ (b) *BP with* $C = 20, 40, 100, 284$

Figure 2.1: *Mass-weighted functions for* $BP(k, P, \alpha)$ *and* $Exp(3 \times 10^3)$ *job size distributions.*

Figure 2.1 shows the mass-weighted functions for the BP distributed job sizes with different $C$ values and the exponential job size distribution, each with the same mean of $3 \times 10^3$. Figure 2.1(a) shows that for the exponential distribution, the largest 1% of the jobs constitute only about 5% of the total load. We observe that the load constituted by the largest 1% of the jobs increases with increasing $C$ from 15% of the total load for $C = 2$ to close to 50% for $C = 10$. On the other hand, Figure 2.1(b) shows that all BP distributions with $C \geq 20$ exhibit the high variability property since 50% of the total load is constituted by less than 1% of the largest jobs.

In this chapter, we shall use the BP job size distribution $BP(332, 10^{10}, 1.1)$ with $C = 284$ as an example of the BP distribution that exhibits the high variability property. This distribution was also used in [13] to analyze the unfairness of SRPT. BP distributions with similar parameters are also used in [107, 27]. In addition to the $BP$ with $C = 284$, we also use other BP

distributions with $C < 284$ and the exponential distribution $Exp(3 \times 10^3)$ to see the impact of degree of variability on the performance of LAS.

## 2.3 Analytical Results

Previously, no analysis of M/G/1/LAS has been carried out for different values of C. The analysis for M/G/1/LAS is tedious as it involves nested integrals of complicated functions. Here, we provide elements to compare M/G/1/LAS to M/G/1/PS, and M/G/1/SRPT systems.

### 2.3.1 Comparison for a general job size distribution

Processor sharing is a well known fair scheduling policy that for any load $\rho < 1$ gives the same mean slowdown to all jobs, i.e., $S(x)_{PS} = \frac{1}{(1-\rho)}\ \forall x > 0$. In this section, we assume a general c.f.m.f.v job size distribution to investigate the unfairness of the LAS policy by comparing its slowdown to the slowdown of PS. In this section, we also compare LAS with a set nonpreemptive (NPP) policies, which include RANDOM scheduling, last-come first-serve (LCFS), and FCFS. We investigate the performance benefits offered under LAS compared to NPP policies in terms of mean response time for varying $C$ and $\rho$ values.

#### Comparison between LAS and PS

We define the function $\phi$ of $x$ as $\phi(x) \triangleq \lambda \int_0^x t f(t)dt + \lambda x F^c(x)$. If we integrate $\int_0^x t f(t)dt$ by parts in the definition of $\phi$ we obtain $\phi(x) = \lambda \int_0^x F^c(t)dt$. The following result holds for the mean response time of a job of size $x$:

**Theorem 2.3.1** *For all c.f.m.f.v job size distributions and load $\rho < 1$,*

$$T(x)_{LAS} \leq (1-\rho)\frac{2-\phi(x)}{2(1-\phi(x))^2}T(x)_{PS} \tag{2.12}$$

$$S(x)_{LAS} \leq (1-\rho)\frac{2-\phi(x)}{2(1-\phi(x))^2}S(x)_{PS} \tag{2.13}$$

Proof:

$$\begin{aligned} T(x)_{LAS} &= \frac{\lambda \int_0^x t^2 f(t)dt + \lambda x^2 F^c(x))}{2(1-\phi(x))^2} + \frac{x}{1-\phi(x)} \\ &\leq \frac{\lambda x \int_0^x t f(t)dt + \lambda x^2 F^c(x))}{2(1-\phi(x))^2} + \frac{x}{1-\phi(x)} \\ &\qquad \text{since} \int_0^x t^2 f(t)dt \leq x \int_0^x t f(t)dt \\ &= T(x)_{PS} \frac{1-\rho}{1-\phi(x)} \Big( \frac{\lambda \int_0^x t f(t)dt + \lambda x F^c(x)}{2(1-\phi(x))} + 1 \Big) \\ &\qquad \text{since } T(x)_{PS} = \frac{x}{1-\rho} \end{aligned}$$

$$= \quad T(x)_{PS} \frac{1-\rho}{1-\phi(x)} \Big( \frac{\phi(x)}{2(1-\phi(x))} + 1 \Big)$$

By definition of $\phi(x)$

$$= \quad (1-\rho) \frac{2-\phi(x)}{2(1-\phi(x))^2} T(x)_{PS} \tag{2.14}$$

Equation (2.13) follows directly by dividing both sides of Equation (2.14) by $x$. □

We use Theorem 2.3.1 to elaborate on the comparison between the mean slowdown of LAS and PS. We first introduce a lemma that we need in the proof of Theorem 2.3.2.

**Lemma 2.3.1** *For any load $\rho < 1$,*

$$\frac{2-\phi(x)}{2(1-\phi(x))^2} \leq \frac{2-\rho}{2(1-\rho)^2} \tag{2.15}$$

Proof: Function $\phi(x)$ is an increasing function for $x \in [0, \infty]$ since $\frac{d\phi(x)}{dx} = \lambda F^c(x) \geq 0$. Also, function $\frac{2-u}{2(1-u)^2}$ is an increasing function for $u \in [0,1]$. Hence, $\frac{2-\phi(x)}{2(1-\phi(x))^2}$ is an increasing function of $x$ and upper bounded by $\frac{2-\rho}{2(1-\rho)^2}$ since $\phi(x) \in [0, \rho] \subset [0,1]$. □

**Theorem 2.3.2** *For all c.f.m.f.v job size distributions and load $\rho < 1$,*

$$S_{LAS} \leq \frac{2-\rho}{2(1-\rho)} S_{PS} \tag{2.16}$$

Proof:

$$\begin{aligned} S_{LAS} &= \int_0^{+\infty} \frac{T(x)_{LAS}}{x} f(x)dx \\ &\leq \int_0^{+\infty} \frac{2-\phi(x)}{2(1-\phi(x))^2} f(x)dx \quad \text{Theorem 2.3.1} \qquad (2.17) \\ &\leq \frac{2-\rho}{2(1-\rho)^2} \int_0^{+\infty} f(x)dx \qquad \text{Lemma 2.3.1} \\ &= \frac{2-\rho}{2(1-\rho)^2} \\ &= \frac{2-\rho}{2(1-\rho)} S_{PS} \quad \text{Since } S_{PS} = \frac{1}{1-\rho} \qquad (2.18) \end{aligned}$$

□

**Corollary 2.3.1** *For all c.f.m.f.v job size distributions and load $\rho < 1$,*

$$T_{LAS} \leq \frac{2-\rho}{2(1-\rho)} T_{PS} \qquad (2.19)$$

Proof: The proof is similar to the proof of Theorem 2.3.2. □

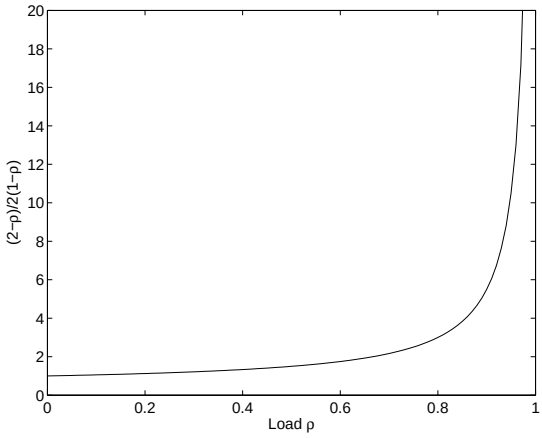



Figure 2.2: *Upper bound on the mean slowdown ratio $\frac{S_{LAS}}{S_{PS}}$.*

We illustrate the result of Theorem 2.3.2 in Figure 2.2. We can see that the ratio between the mean slowdown of LAS and PS is bounded by a value that is less than 2 for $\rho \leq 0.66$ and less than 6 for $\rho \leq 0.9$. The result indicates that for low and medium load the mean slowdown of LAS remains close to the mean slowdown for PS. This result clearly supports the fact that using LAS instead of PS results in a better slowdown for small jobs since LAS is known to favor small

jobs and the average slowdown over small and large jobs under PS and LAS remains fairly close for most system loads. Note that the real ratio between the average slowdown of LAS and PS is below the value of $\frac{2-\rho}{2(1-\rho)}$ used in Figure 2.2 due to the rather crude majorization of $\phi(x)$ applied in Lemma 2.3.1.

**Comparison between LAS and Nonpreemptive policies (NPP)**

We now compare the performance of LAS to NPP policies in terms of mean response time for job size distributions with varying $C$ values and for different values of load $\rho < 1$. Examples of nonpreemptive policies include FIFO, LIFO, and RANDOM. We derive an upper bound of the mean response time that is a function of $C$ and $\rho$ values. The bound shows that the mean response time offered by LAS improves in increasing $C$ values. From Equations (2.5) and (2.6), we get the expressions of mean response times for PS and NPP policies as:

$$T_{PS} = \frac{m_1}{(1-\rho)} \tag{2.20}$$

$$T_{NPP} = \frac{\lambda m_2}{2(1-\rho)} + m_1 \tag{2.21}$$

Equation (2.21) can also be expressed as:

$$\begin{aligned} T_{NPP} &= \frac{m_1}{(1-\rho)} + \frac{\lambda m_2}{2(1-\rho)} + m_1 - \frac{m_1}{(1-\rho)} \\ &= \frac{m_1}{(1-\rho)} + \frac{\lambda m_2}{2(1-\rho)} - \frac{\rho m_1}{(1-\rho)} \\ &= \frac{m_1}{(1-\rho)} + \frac{\lambda m_2}{2(1-\rho)} - \frac{\lambda m_1^2}{(1-\rho)} \\ &\quad \text{from } \rho = \lambda m_1 \\ &= \frac{m_1}{(1-\rho)} + \frac{\rho[C^2-1]}{2(1-\rho)} m_1, \\ &\quad \text{from } m_1^2 C^2 = m_2 - m_1^2 \text{ and } \lambda = \rho/m_1 \\ &= T_{PS} + \frac{\rho[C^2-1]}{2(1-\rho)} m_1, \end{aligned} \tag{2.22}$$

**Lemma 2.3.2** *For a c.f.m.f.v. job size distribution with mean $m_1$ and coefficient of variation $C$, the mean response time under LAS at load $\rho$ ($T_{LAS}$) is upper bounded as:*

$$T_{LAS} \leq \frac{(2-\rho)}{2(1-\rho)} T_{NPP} - \frac{\rho(2-\rho)[C^2-1]}{4(1-\rho)^2} m_1 \tag{2.23}$$

Proof: The proof follows directly when substituting $T_{PS}$ obtained from Equation (2.22) in Corollary 2.3.1. □

The bound on the mean response time of LAS (Equation (2.23)) is a function of $C$ $\rho$, since $T_{NPP}$ itself is a function of $C$ and $\rho$ (see Equation 2.22) for a given value of $m_1$. This bound is

interesting since it enables us to compare the performance of LAS relative to that of any nonpreemptive policy for job size distributions with any $C$ value. Figure 2.3 shows the upper bound

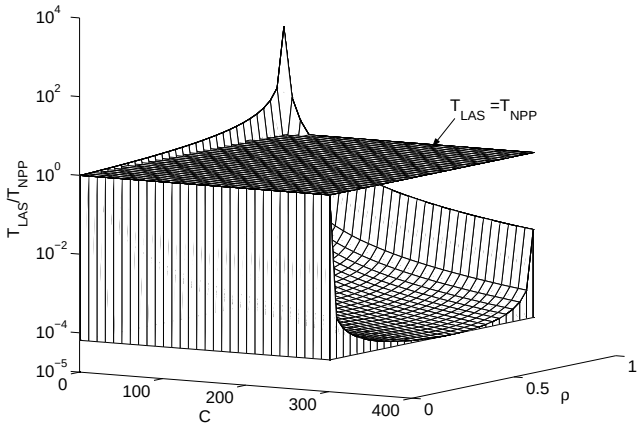



Figure 2.3: *Upper bound on the mean response time ratio $\frac{T_{LAS}}{T_{NPP}}$ as a function of load $\rho$ and coefficient of variation $C$.*

on the ratio of the mean response time of LAS to the mean response time of a nonpreemptive policy as a function of $C \geq 1$ and load $\rho < 1$. Observe that LAS offers significantly higher mean response times than nonpreemptive policies only for distributions with $C$ values close to 1. In this case, we also observe that the mean response time ratio increases to large values with increasing load. On the other hand, the mean response time of LAS is lower than that of nonpreemptive policies for distributions with higher $C$ values at all load values $\rho < 1$. Generally, for a given load $\rho$, the ratio $\frac{T_{LAS}}{T_{NPP}}$ decreases with increasing $C$.

In summary, we see that LAS achieves lower mean response time than any NPP policy, such as the FIFO policy, for job size distributions with a high variance. This is to be expected because in contrast to LAS, the service of a job under FIFO is not interrupted until the job leaves the system and so large jobs are favored over small jobs.

### 2.3.2 Distribution Dependent Comparison

In this section, we compare LAS to PS for job size distributions with varying $C$ values. A comparison of LAS with PS helps to analyze the degree of unfairness (penalty) seen by the largest jobs under LAS. The results in this section will also reflect the performance comparison between FIFO router and LAS router (see Chapter 5). We start with general results for exponentially distributed job sizes. We show that for the case of the exponential distribution, the average slowdown of jobs under LAS is always better than the average slowdown of the jobs under PS.

**Theorem 2.3.3** *For an exponential job size distribution and load* $\rho < 1$,

$$S_{LAS} \leq S_{PS} \tag{2.24}$$

Proof: Following the same reasoning as in Theorem 2.3.2, one obtains:

$$S_{LAS} = \int_0^{+\infty} \frac{T(x)_{LAS}}{x} f(x)dx$$

Using Equation (2.17) in Equation (2.25), we get,

$$S_{LAS} \leq \int_0^{+\infty} \frac{x(2-\phi(x))}{2(1-\phi(x))^2} \frac{f(x)}{x} dx \tag{2.25}$$

For exponential job sizes, we have $\phi(x) = \int_0^x \lambda F^c(t)dt = \rho(1-e^{-\mu x}) = \rho F(x)$ and $f(x) = \mu e^{-\mu x}$. Using these facts and plugging $h(\phi(x)) \triangleq \frac{2-\phi(x)}{2(1-\phi(x))^2}$ in Equation (2.25) we obtain:

$$S_{LAS} \leq \frac{1}{\rho} \int_0^{+\infty} h(\rho F(t)) \rho f(t) dt$$

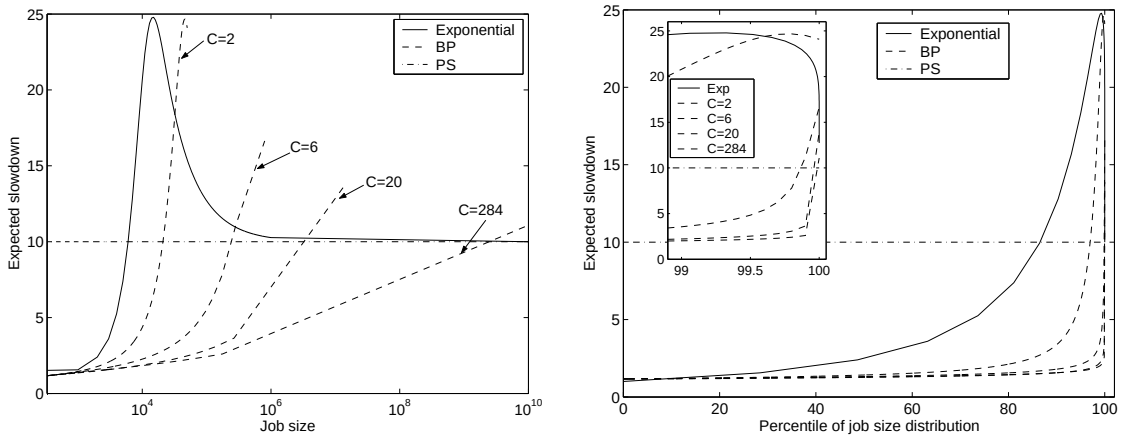
Replacing $\rho F(t)$ by $u$ in the above equation we get:

$$\begin{aligned} S_{LAS} &\leq \frac{1}{\rho} \int_0^{+\rho} h(u)du \\ &= \frac{1}{2\rho}\Big( -\ln(1-\rho) + \frac{\rho}{1-\rho}\Big) \\ &= \frac{1}{2\rho}\Big( -(1-\rho)\ln(1-\rho) + \rho\Big) S_{PS} \\ &\quad \text{since } S_{PS} = \frac{1}{1-\rho} \end{aligned} \tag{2.26}$$

A straightforward study of the derivative of $\psi$ defined as $\psi(\rho) \triangleq \frac{(\rho-(-1-\rho)\ln(1-\rho))}{2\rho}$ indicates that $\psi$ is a decreasing function and $\psi(\rho) \sim_{\rho\to 0} \frac{2\rho-\rho^2}{2\rho} \to 1$. Thus, $\forall \rho < 1, \psi(\rho) < 1$. □

This result shows that even for the case of exponentially distributed job sizes, where the high variability property does not hold, the mean slowdown is at most $\frac{1}{(1-\rho)}$. We could not derive bound similar to the one of theorem 2.3.3 for the BP distribution. However, we expect LAS to perform better in terms of the mean slowdown for job size distributions with $C > 1$ than for exponential job size distribution.

Next, we investigate the unfairness of LAS for job size distributions with varying $C$ values. We use the exponential distribution (with $C = 1$) and the BP distributions with $C$ values 2, 6, 20, and 284 all with the same mean value of $3 \times 10^3$. To obtain BP job size distributions with different $C$ values and the same mean we varied the values of $\alpha$ and $p$.




(a) *Job size* (b) *Percentile of job size distribution*

Figure 2.4: *Expected slowdown under LAS and PS for different job size distributions and different $C$ values.*

Figure 2.4(a) and 2.4(b) show the conditional mean slowdown as a function of job size and percentile respectively. We see from both figures that the percentage of the largest jobs that experience a penalty under LAS (i.e., have higher mean slowdown under LAS than under PS) and the degree of penalty decreases in increasing $C$ value. For the BP distribution with $C \geq 6$, less than 0.5% of the largest jobs suffer a small penalty and the performance difference between $C = 20$ and $C = 284$ is minor. We observe that as the $C$ value increases the penalty experienced by the largest jobs and the percentage of these largest jobs decrease. In the remainder of this chapter we will only consider $Exp(3 \times 10^3)$ and the $BP(332, 10^{10}, 1.1)$ distribution when we numerically analyze LAS. A similar theorem to the Theorem 2.3.3 that compares the mean response time $T$ of LAS and PS for exponential job sizes is obtained in the following.

**Theorem 2.3.4** *For an exponential job size distribution and load $\rho < 1$,*

$$T_{LAS} = T_{PS} \tag{2.27}$$

Proof: See Appendix A □

### 2.3.3 Quantitative comparison between LAS and SRPT

It is stated in [13] that for any job size distribution and at any load $\rho < 1$ every job with size $x$ has a higher conditional mean response time $T(x)$ under LAS than under SRPT. However, it was not elaborated how much higher the conditional mean slowdown of a job under LAS can be as compared to SRPT nor whether this depends on the variability of a job size distribution or load. In this section, we compare LAS with SRPT quantitatively and show that at low load and when a job size distribution has the high variability property, the conditional mean response time of a job under LAS is quite close to the one under SRPT.

**Theorem 2.3.5** *Let $\phi(x) \triangleq \rho(x) + x\lambda F^c(x)$, then for all c.f.m.f.v job size distributions and at load $\rho < 1$,*

$$T(x)_{SRPT} \leq T(x)_{LAS} \leq \Big(\frac{1-\rho(x)}{1-\phi(x)}\Big)^2 T(x)_{SRPT} \tag{2.28}$$

Proof: See Appendix B. □

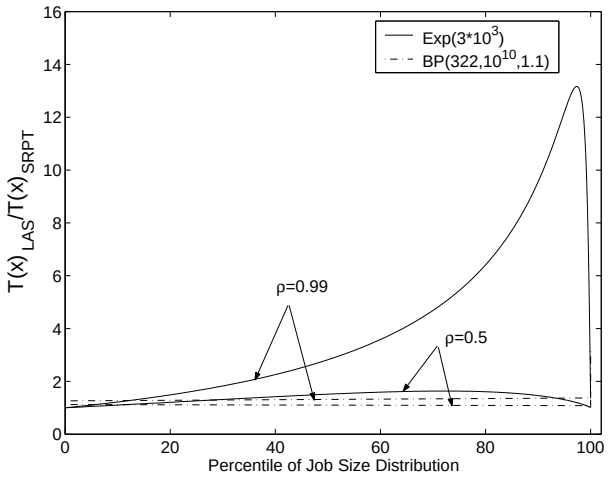



Figure 2.5: $\frac{T(x)_{LAS}}{T(x)_{SRPT}}$ *as a function of percentile of job size distribution.*

Figure 2.5 illustrates the result of Theorem 2.3.5 for the $BP$ distribution with $C = 284$ and the exponential distribution $Exp(3 \times 10^3)$ as a function of percentiles of job sizes to obtain results for a job size distribution that exhibits the high variability property and for a job size distribution that does not exhibit the high variability property. We observe from Figure 2.5 that the ratio between $T(x)_{LAS}$ and $T(x)_{SRPT}$ highly depends on the variability of the job size distribution. We see that for the bounded Pareto distribution, the ratio $\frac{T(x)_{LAS}}{T(x)_{SRPT}}$ is about $1.25$ for all jobs even under very high load, which shows that the conditional mean response times $T(x)$ of jobs under LAS are quite close to their response times under SRPT. For BP distributions with lower values of $284 > C \geq 20$, we observed that the value of $\frac{T(x)_{LAS}}{T(x)_{SRPT}}$ is quite close to the value for BP with $C = 284$ for all load values.

Jobs whose size is exponentially distributed experience under LAS a slightly higher conditional mean response time $T(x)_{LAS}$ than under SRPT for medium load $\rho = 0.5$. However, for very high load $\rho = 0.99$, the difference is much more pronounced and the ratio $\frac{T(x)_{LAS}}{T(x)_{SRPT}}$ approaches $13$ for some large jobs. Note also that as $x \to 100th$ percentile job size, the ratio $\frac{T(x)_{LAS}}{T(x)_{SRPT}} \to 1$ because $\lim_{x\to\infty} \phi(x) = \rho(x)$.

## 2.4 LAS under Overload

In real systems, it may happen that jobs arrive to the system at a higher rate than the rate at which they are serviced. This situation is referred to as overload condition where $\rho \geq 1$. To the best of our knowledge, no analysis of LAS under overload has been conducted. Here, we show that at load $\rho \geq 1$, LAS can service all small jobs up to a certain job size and we derive the formulas of the conditional mean response time for LAS under overload. SRPT was also proven to be stable under overload in [13] for a range of short jobs. Hence in this section, we also compare LAS to SRPT under overload.

**Theorem 2.4.1** *Let $\theta(x) \triangleq m_1(x) + xF^c(x)$, $\lambda$ be the job mean arrival rate, and $f(t)$ be a service time distribution with mean $m_1$. Then, for any load $\rho = \lambda m_1 \geq 1$, every job size $x < x_{LAS}(\lambda)$ with $x_{LAS}(\lambda) \triangleq \max\{x \mid \theta(x) \leq \frac{1}{\lambda}\}$ has a finite conditional mean response time under LAS, whereas every job of size $x \geq x_{LAS}(\lambda)$ experiences an infinite response time.*

Proof: The load offered to a server running the LAS scheduling policy under overload is $\rho = \lambda m_1 \geq 1$. However, the effective load $\rho_{\text{effective}}$ that corresponds to the work serviced by the server is equal to 1. Let $\theta_{\text{effective}} \triangleq \frac{\rho_{\text{effective}}}{\lambda} = \frac{1}{\lambda}$. Then, $\theta_{\text{effective}}$ is the expected service offered to a set of jobs that access the server under overload. This set depends on the scheduling policy. With LAS, every newly arriving job in the system gets an immediate access to the server. On the average a job receives a service of $\theta_{\text{effective}}$. But, since some jobs require less than $\theta_{\text{effective}}$, the jobs of size strictly smaller than $x_{\text{LAS}}(\lambda)$ may receive more service than $\theta_{\text{effective}}$, where $x_{\text{LAS}}(\lambda)$ is defined as:

$$x_{\text{LAS}}(\lambda) \triangleq \max\{x \mid \theta(x) \leq \frac{1}{\lambda}\} \tag{2.29}$$

Hence, for LAS under overload, one obtains $\rho_{\text{effective}} = \lambda\theta(x_{\text{LAS}}) = 1$. □

**Corollary 2.4.1** *For SRPT under overload, every job of size $x < x_{SRPT}(\lambda)$ with $x_{SRPT}(\lambda) \triangleq \max\{x \mid m_1(x) \leq \frac{1}{\lambda}\}$ has a finite conditional mean response time, whereas every job of size $x \geq x_{SRPT}(\lambda)$ experiences an infinite response time.*

Proof: The reasoning for SRPT is different from the one for LAS. Note that the service of a job under SRPT is not affected by the arrival of jobs with larger sizes. Hence, a set of jobs with size less than or equal to $x$ contributes a system load of $\rho(x) = \lambda \int_0^x tf(t)dt$. Thus under overload, only jobs with size strictly less than $x_{\text{SRPT}}(\lambda)$ receive service, where $x_{\text{SRPT}}(\lambda) = \max\{x \mid \rho(x) \leq \rho_{\text{effective}} = 1\}$, which is equivalent to $x_{\text{SRPT}}(\lambda) = \max\{x \mid m_1(x) \leq \frac{1}{\lambda}\}$. The jobs with size greater than or equal to $x_{\text{SRPT}}(\lambda)$ can not access the server since the jobs with size less than $x_{\text{SRPT}}(\lambda)$ always preempt them to maintain $\rho_{\text{effective}} = 1$. □

Note that $x_{\text{SRPT}}(\lambda) \geq x_{\text{LAS}}(\lambda)$ since with SRPT under overload, a job accesses the server only if its service can be entirely completed, which may not the case with LAS. We observe in Figure

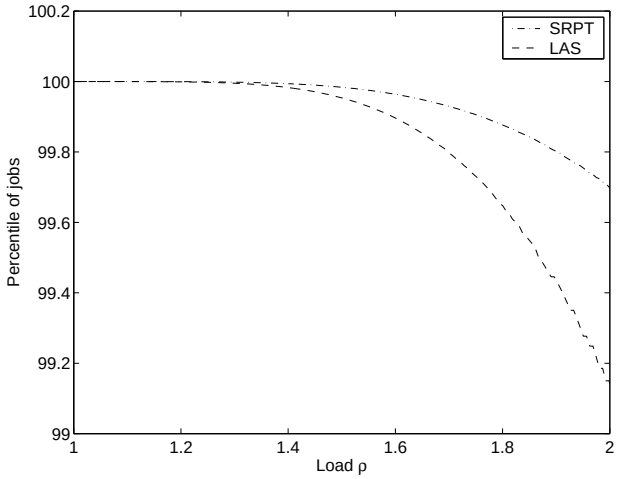



Figure 2.6: *Percentile of the largest job sizes that can be serviced under overload for* $BP(332, 10^{10}, 1.1)$ *job size distribution.*

2.6 that for $BP(332, 10^{10}, 1.1)$, the maximum job size that SRPT can service under overload is always larger than or equal to the maximum job size that LAS can service at a given load. It is worth mentioning that for the overload values considered, the maximum job that is serviced by both policies is above the 99th percentile of the job size distribution. This result is to be expected since for the job size distribution considered, more than 99% of the jobs are small and the fraction of load that these small jobs contribute to the system load is less than 50% of the total load.

We now compute the formulas for the mean response time of the jobs under overload. For the SRPT policy, the formulas in case of underload and case of overload are different [13]. This may be explained by the fact that all the jobs of size $x < x_{\text{SRPT}}(\lambda)$ with $\rho(x_{\text{SRPT}}(\lambda)) \triangleq \int_0^{x_{\text{SRPT}}(\lambda)} tf(t)dt = 1$ have a finite response time, and all the jobs of size $x \geq x_{\text{SRPT}}(\lambda)$ receive no service at all. Therefore, under overload, the SRPT system works as if there are no jobs of size $x \geq x_{\text{SRPT}}(\lambda)$. Hence, the moment[2] $m_2(x) + x^2F^c(x)$ that appears in the formulas of $T(x)_{SRPT}$ is computed only for the jobs of size $x < x_{\text{SRPT}}(\lambda)$, whereas in underload, $T(x)_{SRPT}$ is computed considering all job sizes.

For LAS under overload, all jobs can receive up to $x^-_{\text{LAS}}(\lambda) \triangleq \max\{x \mid \theta(x) < \frac{1}{\lambda}\}$. Thus, if the initial service requirement of a job is larger than $x^-_{\text{LAS}}(\lambda)$, it receives only $x^-_{\text{LAS}}(\lambda)$). Therefore, the moments[2] $m_2(x) + x^2F^c(x)$ and $m_1(x) + xF^c(x)$ that appear in the formulas for $T(x)_{LAS}$ must be computed considering all job sizes. As a consequence, the formulas for the mean response time in overload and underload for job sizes less than $x_{\text{LAS}}(\lambda)$ are identical:

[2] $m_2(x)+x^2F^c(x)$ and $m_1(x)+xF^c(x)$ are the moments of a truncated distribution $f_x(y)$ (with $f_x(y) = f(y)$ if $y < x$, $f_x(y) = F^c(x)$ if $y = x$, and $f_x(y) = 0$ if $y > x$) that account for the contribution of all jobs of size $y$ to the response time of the job of size $x$ (see ([64], pp. 173).

**Theorem 2.4.2** *The conditional mean response time of a job size $x$ for LAS under overload is:,*

$$T(x)_{LAS} = \begin{cases} \frac{\lambda(m_2(x)+x^2(1-F(x)))}{2(1-\rho(x)-\lambda x(1-F(x)))^2} + \frac{x}{1-\rho(x)-\lambda x(1-F(x))} & \text{if } x < x_{LAS}(\lambda) \\ +\infty & \text{if } x \geq x_{LAS}(\lambda) \end{cases}$$

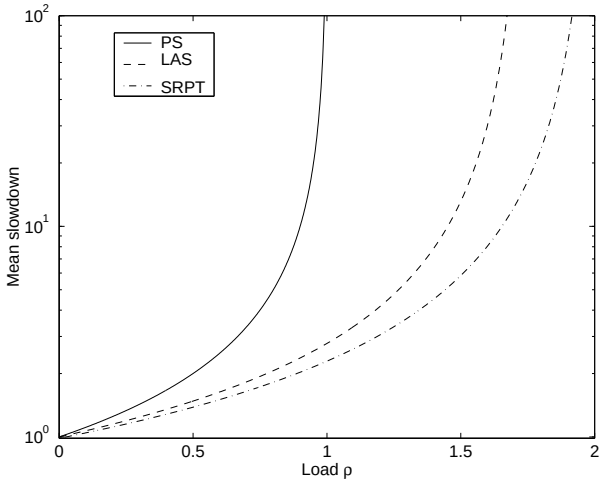



Figure 2.7: *Mean slowdown for the 99th percentile job for $BP(322, 10^{10}, 1.1)$ as a function of load.*

Figure 2.7 shows the mean slowdown of the 99th percentile job of the $BP(332, 10^{10}, 1.1)$ job size distribution under PS, LAS, and SRPT under overload. At load $\rho = 1.4$, the mean slowdown of the job under PS is infinity, whereas it is 10 and 4 under LAS and under SRPT respectively. In general, it is well known that the conditional mean response times $T(x)_{PS}$ of all jobs under PS is undefined for load $\rho \geq 1$. It is obvious from the figure that LAS becomes unstable at lower overload values than SRPT. However, we see that LAS is quite close to the optimal policy SRPT in terms of stability under overload.

## 2.5 Conclusion

In this chapter, we analyzed the $M/G/1/LAS$ queue to evaluate the performance of LAS scheduling for job size distributions with different degrees of variability. We showed through analysis and numerical evaluation that the variability of the job size distribution is important in determining the performance of LAS in terms of flows mean response time. In particular, we saw that the percentage of jobs that have higher slowdowns under LAS than under PS is smaller for job size distributions with the high variability property (have large $C$ values) than for job size distributions with values of $C$ close to 1. For the case of the exponential job size distribution ($C = 1$), we proved that the mean response time under LAS and PS are the same, and the mean slowdown for LAS is always less than or equal to the mean slowdown for PS.

Even for the case of a general job size distribution, we showed that for moderate load values, the mean slowdown of LAS remains fairly close to the mean slowdown of PS.

The comparison of LAS to nonpreemptive policies (NPP) reveals that the benefits of LAS over NPP in terms of reducing the mean response time improves with increasing variance of the job size distribution. Similarly, we compared LAS to the optimal policy SRPT and demonstrated that $T(x)_{LAS}$ is close to $T(x)_{SRPT}$ for job size distributions with high variability property. We also proved that LAS is stable under overload for a subset of small jobs and obtained the expression for the conditional mean response time $T(x)$ for LAS under overload. The ability of LAS scheduling to schedule jobs under overload does not apply to PS and FIFO scheduling.

We analyzed the M/G/1/LAS queue using a notion of a job that must arrive the queue all at once. However, The goal of this work is to propose LAS for packet networks where the widely used packets transport protocol is TCP. However, we are not interested in the performance (response time and slowdown) of a packet but the performance of a *flow*. A flow, as opposed to a job, consists of a sequence of packets that are spaced in time and are statistically multiplexed with packets from other flows. Thus, the focus of the next chapter is to study the performance of LAS in packet networks and the interaction of LAS to the TCP mechanisms.

# Chapter 3

# Analyzing the Performance of TCP Flows in Packet Networks with *LAS* Schedulers

This thesis work proposes using LAS scheduling in packet networks. TCP is a transport protocol that supports most of traffic in Internet today. Thus, this chapter investigates the interaction of LAS with TCP protocol to establish its performance for network flows in detail. Using network simulation we show that LAS scheduling, as compared to FIFO, avoids packet losses and significantly reduces the transfer times for short flows with a negligible increase in transfer time for the largest flows. We show that the improvement seen by short TCP flows under LAS is mainly due to the way LAS interacts with TCP algorithm in the slow start (SS) phase, which results in shorter round-trip times and very low packet loss rates for short flows. We also investigate the performance of *long-lived* TCP and UDP flows under LAS and compare the results to FIFO scheduling.

## 3.1 Introduction

Today, the Internet still provides only a best-effort service, TCP remains the most widely used transport protocol, not only for data transfer, but also for media streaming [100], and most of the TCP flows are short, while more than 50% of the bytes are carried by less than 1% of the largest flows [84, 28]. Because of these facts, there is still a need for new QoS mechanisms that will improve Internet services by prioritizing the Internet flows in a way that will trigger a good interaction with the TCP algorithms. Empirical studies have shown that these many short flows have 10-20 packets on average [22]. A recent Internet traffic measurement study shows that the largest flows originate from peer-to-peer file sharing services [99]. The short flows, on the other hand, are mainly Web data transfers triggered by user interaction. As the use of the Web remains popular, Web traffic will continue to make up a significant fraction of the Internet traffic. Web traffic is transferred using the TCP transport protocol.

The main task of TCP is to adjust the source rate according to the network congestion. Thus,

a TCP source starts sending packets at a low but exponentially increasing rate. This behavior is referred to as the *slow start* (SS). When the flow experiences packet losses, the TCP source first reduces its rate and then sends the data at a rate that increases at a slower rate than during the SS phase. This phase of TCP is known as *congestion avoidance* (CA). When a packet is lost, TCP can use *fast retransmit* mechanisms to quickly detect the loss and re-sends the packet. The fast re-transmit mechanisms is possible only after the source has received three duplicate ACK packets from the destination. Looking at this TCP mechanisms, one wonders if TCP was not created having in mind that Internet has only large flows that have enough packets to reach at CA steady state phase. A closer look at the behavior of TCP in a network reveals that it negatively affects the performance of *short* flows.

Short flows are transferred during the SS phase of the TCP. During this phase, regardless of the available bandwidth in the network, TCP conservatively initializes its sending window to one packet and then doubles the window for every window worth of ACKs received until congestion is detected . Thus, even if the network has sufficient bandwidth, the duration of a transfer at SS phase is dominated by the *round trip time* (RTT) of the connection, which can be much longer than the transfer time of short flows. Similarly, since short flows often do not have enough packets in transit to trigger the triple duplicate ACK mechanism followed by a fast retransmission, they must rely on the *retransmission timeout* (RTO) to detect loss. Packet retransmissions triggered by a RTO significantly prolong the transfer time of a short TCP flow. TCP uses its own packets to sample the round trip times and to estimate the appropriate RTO value. For the first control packets and the first data packet, the initial RTO value is usually set to 3 seconds. Thus, losing any of these packets can significantly prolong the transfer time of short flows.

Similarly, FIFO scheduling with drop-tail worsens the performance of short flows: Packets of short flows under FIFO are likely to experience long queueing delays and are equally likely to be dropped as packets from long flows when the buffer is full. The impact of FIFO to the poor performance of short flows is more pronounced at bottleneck link. The poor performance of short flows in TCP networks with FIFO schedulers prompted us to study LAS scheduling in conjunction with their interaction with TCP flows. In Chapter 2, we saw that LAS is very efficient to reducing the transfer times of jobs with a negligible penalty to the largest jobs when the job size distribution exhibits a high variability property observed to the Internet traffic. However, so far we cannot generalize the analytical results obtained for jobs in Chapter 2 to flows in packet networks. This is because job is defined to be a piece of workload that arrives to the system all *at once*. Given this definition, a flow in a packet network cannot be considered as a job since a flow does not arrive at the system at once. Instead, the source transmits a flow as a sequence of packets, possibly spaced out in time, that are in turn statistically multiplexed with packets from other flows. Also, when TCP is used as transport protocol, due to the closed-loop nature of TCP, the treatment given to an individual packet may affect the temporal behavior of the remaining packets in the flow.

To investigate the impact of LAS on the performance of TCP flows of different sizes (short and large) in network routers we use the network simulator ns-2 [1]. The implementation of LAS in ns-2 involves maintaining a single priority queue and inserting each incoming packet at its appropriate position in the queue. The less service a flow has received so far, the closer to

the head of the queue its arriving packets will be inserted. When a packet arrives to a full queue, LAS first inserts the arriving packet at its appropriate position in the queue and then drops the packet that is at the end of the queue.

This chapter is organized as follows; after presenting the related work in the next section, we present preliminary concepts that show how LAS interacts with TCP's congestion control and retransmission mechanisms in Section 3.3. We then discuss the simulation setting and results for short flows in Section 3.4. In Section 3.5 we investigate the performance of long-lived TCP and long-lived UDP flows that compete with Web traffic. We discuss implementation of LAS in Section 3.6, and finally we finally summarize the chapter in Section 3.7

## 3.2 Related Work

A large number of papers propose to improve the performance of short flows by changing TCP. For example [6, 81, 112, 5] suggest to use a larger initial window value to reduce the time short flows spend in the SS phase. Other papers [2, 62] propose rate based pacing of packets to reduce the burstiness of TCP transmissions. Reducing the burstiness reduces the likelihood of multiple packet losses in the same congestion window, which generally degrades the performance of a TCP flow. Other proposals to improve the startup and recovery behavior of TCP include [7, 12, 18, 36, 58]. The work in this chapter does not propose to modify TCP. Instead, we propose to use LAS scheduling to resolve the problems short flows experience in the Internet today.

Hence, closely related to our work are network-based approaches that are proposed to improve the performance of short flows. In [48, 21, 69, 108] Diffserv-like architectures are proposed in routers to isolate short flows from large flows. In these approaches, edge routers classify and mark incoming packets as belonging to a short or to a large flow (using a predefined threshold). The core routers use the packet marking to separate different flows in order to give preferential treatment to short flows using either selective packet discard mechanisms or priority queueing schemes. Simulation results show that the proposed mechanisms reduce the mean transfer time of short flows by about 30% [21] and by about 80% for medium sized flows of sizes between 50-200 packets [49] compared to FIFO with RED. In [48], a modified active queue management mechanism is proposed that reduces the packet drop rate of short flows and helps to reduce the transfer time of short and medium flows by 25-30% compared to RED.

In a recent work by Avrachenkov et. al. [10], a threshold based scheduling known as RuN2C, which is similar to the policy proposed in [21], is studied to differentiate between short and long flows in order to improve the response times of short flows. A RuN2C router maintains two queues; up to *Thr* packets are serviced in the first queue and if a flow has more than *Thr* packets the remaining packets are serviced in the second queue. RuN2C schedules packets in the second queue only if there are no packets in the first queue. Packets in each queue are serviced in FIFO order. In general, transfer times of short flows under RuN2C should be greater or equal to their transfer times under LAS. In particular, the performance of short flows under RuN2C depends on the threshold value *Thr*. For small *Thr* values short flows under RuN2C

experience low transfer time close to under LAS, whereas for very large *Thr* values, RuN2C offers the same performance to flows as FIFO scheduling.

The main difficulty with threshold-based schemes, which is not addressed in most of the previous work, is how to tune the threshold value to capture all short flows. In addition, flows that are marked to belong to the same class (short or large flows) are serviced indiscriminately in FIFO order. This may explain why the transfer time of short flows under these approaches is only slightly improved as compared to FIFO. LAS, in contrast, offers service priority at a much finer level of granularity. For example, the first packets of a flow receive the service under LAS (almost) immediately upon their arrival, whereas under FIFO, they must wait for all packets that are ahead of them in the queue to complete service before being served. Hence, LAS significantly reduces the mean transfer time and loss rate for short flows without the need to set a threshold or the need to use active queue management schemes.

Williamson and Wu [104] recently proposed a Context Aware Transport/Network Internet Protocol (CATNIP) for Web documents. This work is motivated mainly by the fact that loss of the first packets of a TCP flow has a much more negative impact on the transfer time of a flow than loss of other packets. CATNIP uses application layer information, namely the Web document size, to provide explicit context information to the TCP and IP protocols; Using CATNIP, IP routers identify and mark packets of a flow differently to avoid that the most important packets get dropped. Experiments with CATNIP in routers show that it can reduce the mean Web document retrieval time by 20-80%. In contrast to CATNIP, LAS does not require the knowledge of flow sizes and our simulation results indicate that LAS scheduling significantly outperforms CATNIP.

Finally, the *Shortest Remaining Processing Time* (SRPT) scheduling policy is known to be the optimal policy in the sense that it *minimizes* the mean transfer time of jobs [90]. An experiment where SRPT was implemented in the network interface of a Web server [53] showed that SRPT substantially reduces the mean transfer times of short Web objects. However, unlike LAS, SRPT must know the flow sizes to decide what flow to schedule next.

## 3.3 Background

In this section, we highlight the interactions of LAS with a TCP flow that is in slow start to explain how LAS scheduler improves the performance of short TCP flows. We will assume a Reno version of TCP. It is worth mentioning that we compare LAS scheduling to FIFO scheduling only, because FIFO is the most widely deployed policy in networks.

Short flows, e.g., most of Web transfers, may be completely transferred during the slow start phase. Thus, the performance of these flows is dominated by startup effects such as connection establishment and the conservative nature of the slow start phase. However, the TCP slow start phase is quite a slow process for short flows because the RTTs during this phase contribute significantly to the transmission times of short flows. In case of packet losses, short flows are most likely to timeout and make use of RTO to recover the losses because short flows

generally do not have enough packets to use fast retransmit mechanism to recover lost packets, which we will assume in this section. Using RTO to recover lost packets further deteriorates the transmission time of the flows. Hence, there is a need to pay special attention to short flows from the time they send their very first packets so as to avoid these negative effects. We elaborate on the ways LAS scheduler solves these problems for short flows in the next section.

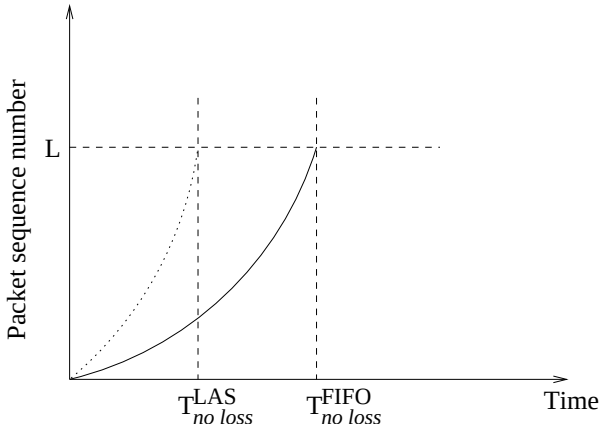



Figure 3.1: *Slow start behavior for short TCP flows with no packet loss under both polices.*

### 3.3.1 The impact of LAS on Short TCP Flows

There are a number of reasons that make LAS a better candidate than FIFO when it comes to improving the performance of short TCP flows. By giving highest service priority to the first packets in a flow, LAS prevents the first packets from waiting in the queue, and they will see little or no queuing delay, which significantly reduces the RTT of the first packets of each flow. Since the RTT dominates the transmission time of short flows, reducing the RTT means reducing the transmission time of short flows. Figure 3.1 shows the sequence number of received packets versus time that elaborates this behavior under both LAS and FIFO assuming that there is no congestion in the network. The figure shows that a short flow of length $L$ packets transmits its packets faster under LAS than under FIFO, due to lower RTT under LAS than under FIFO. As a result, the transmission time of the same flow under LAS ($T^{LAS}_{no\ loss}$) is lower than the transmission time under FIFO ($T^{FIFO}_{no\ loss}$).

The packet discard mechanism under LAS is such that when the buffer is full, LAS first inserts an arriving packet into the queue and then drops the packet that is at the tail of the queue, i.e., LAS gives buffer space priority to short flows. This significantly reduces packet losses during the initial slow start phase for all flows, regardless of their sizes. In fact, LAS mainly drops packets from large flows that have already transmitted a certain amount of data (have low service priority). Hence under LAS there is hardly a need for packet error recovery to rely on the RTO to detect packet loss. This is an important feature of LAS for short flows. Figure 3.2 shows the sequence number versus time plot for a short flow of size $L$ packets that does not experience losses under LAS but loses packet $(L_l + 1)$ under FIFO. The short flow under FIFO

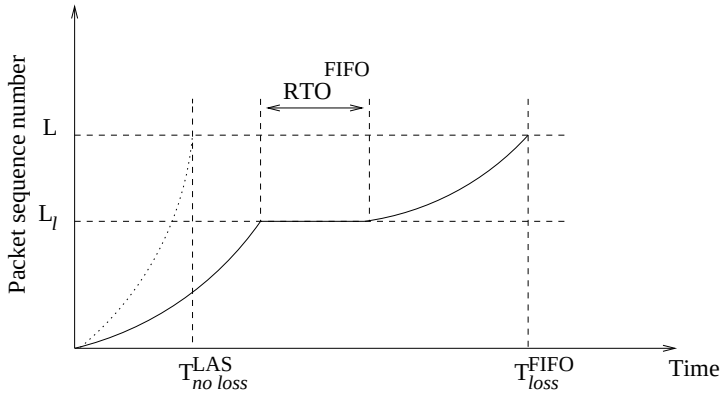



Figure 3.2: *Slow start behavior for short TCP flows with packet loss under FIFO.*

retransmits the packet after time $RTO^{FIFO}$, which significantly prolongs the transmission time $T^{FIFO}_{loss}$ of the flow.

Consider also the case when the short flow under both policies losses packet $(L_l + 1)$. Figure 3.3 shows that the response time under LAS is still lower than the response time of the flow under FIFO. It is important to note here that $RTO^{FIFO}$ is bigger than the $RTO^{LAS}$. This is because the computations of RTO in TCP at any time depends directly on previous RTT sample values [83], which are smaller under LAS than under FIFO. In practice, packet losses are likely

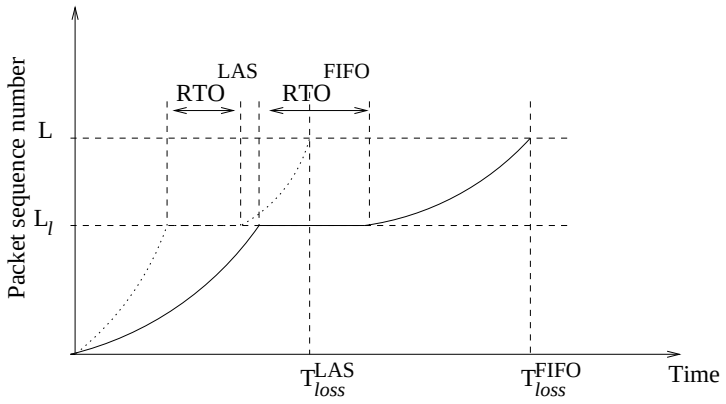


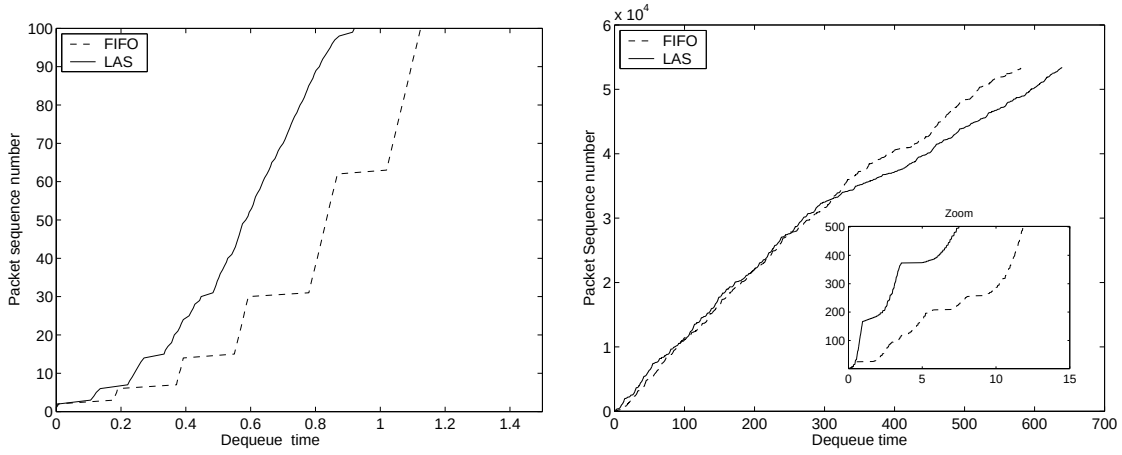

Figure 3.3: *Slow start behavior for short TCP flows with packet loss under both policies.*

to occur sooner under FIFO than under LAS. In this case, FIFO is highly likely not to have had enough RTT samples to obtain a short RTO. That is, the TCP has to either use the RTO that is close a conservatively set *Initial Timeout* (ITO) or use the ITO value as the RTO. ITO is often set to the high value of 3 seconds. This again lengthens the transmission time of short flows under FIFO. Given the same load conditions for both policies, it is unlikely that a short flow loses packets only under LAS but not under FIFO. So we do not elaborate this case. To summarize, we saw that the two main factors that make LAS improve the performance of short TCP flows are its inherent features of reducing the RTT during slow start phase and reducing losses of first packets of each flow. Note that the arguments presented in this section apply not only to short flows but also to long TCP flows during their first slow start phase. This means

that LAS enables all flows to reach *congestion avoidance* (CA) phase.

### 3.3.2 Illustration of LAS at Packet Level

Now, we illustrate the performance of flows under LAS at packet level. These plots are derived from traces obtained by simulating LAS in the network simulator (ns2) using the Web traffic parameters and network topology as given in Section 3.4.1. For flows of equal size, we compare the absolute time each packet leaves the queue (*dequeue time*) when served under LAS and FIFO respectively assuming that the first packets in both cases leave the queue at the same time. The results nicely illustrate the impact of LAS, as compared to FIFO, on short and long TCP flows.




(a) *Web object of size 100 packets* (b) *Web object of size 53,080 packets*

Figure 3.4: *Packet level analysis of LAS, load* $\rho = 0.7$.

Figures 3.4(a) and 3.4(b) show the times when the packets of the flows leave the queue of the router R1 (see Fig. 3.5) under LAS and under FIFO as a function of packet sequence number for flows with 100 packets and for a single very large flow respectively. Figure 3.4(a) shows that all packets leave the queue under LAS sooner than under FIFO, which shows that LAS reduces the time that each packet of these flows spends in the queue. Figure 3.4(b) shows the same tendency up until 3500 packets are sent, after that, packets under FIFO leave the queue sooner than under LAS as the priority of a flow decreases with an increasing amount of data the flow has sent. Both figures illustrate how LAS speeds up the transfer of the first packets of all flows, which makes LAS a very efficient mechanism to improve the performance of short flows. We also observe from Figure 3.4(a) that LAS scheduler also has the advantage of smoothing the bursts of TCP packets, which we see not to be the case for FIFO scheduler.

## 3.4 Simulation Results for Short Flows

### 3.4.1 Network topology and Web traffic model

To evaluate LAS, we use the network topology shown in Figure 3.5, where C1-C5 are clients initiating a series of Web sessions. During each session, a client requests several Web pages, each containing several objects from a randomly chosen server in the pool S1-S5 of servers. This Web model was first introduced in [38]. Each time an object is requested, a new TCP connection is established[1]. Thus the model emulates the HTTP 1.0 protocol. We refer to the packets that belong to one TCP connection as a flow.

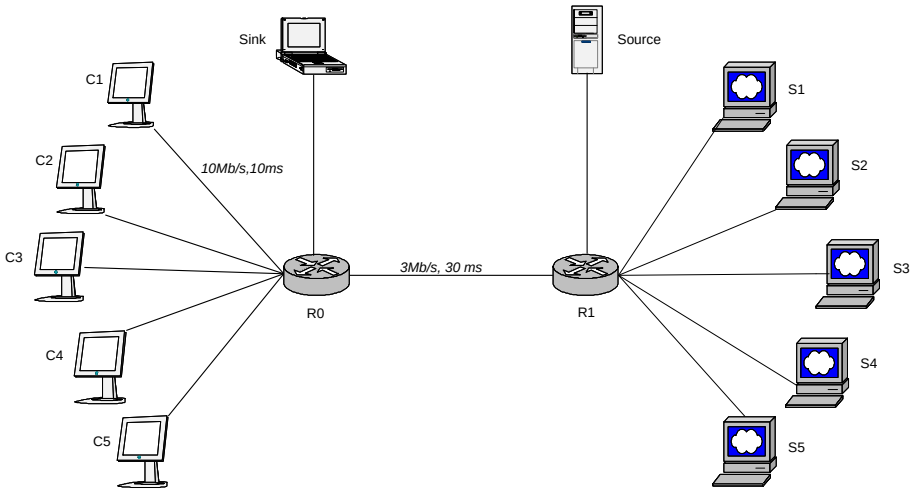



Figure 3.5: *Simulated network topology.*

The simulated Web traffic profile is obtained by setting the distributions of inter-session, inter-page and inter-object time (all in seconds), session size (in web pages), page size (in objects), and object size (in packets) as seen in Table 3.1. These parameter values result to a total load for Web traffic of 0.73. We consider a Pareto *object size* distribution of the second kind that we denote as $PII(a, \alpha, \bar{x})$, where $a$ is the minimum possible object size, $\alpha$ is the shape parameter, and $\bar{x}$ is the mean object size. We also use $Exp(1/\mu)$ to denote exponential distribution with mean $1/\mu$. Our values shown in Table 3.1, are also used in [21], and are based on the findings of [96].

| pages/session | objs/page | inter-session | inter-page | inter-obj | obj size |
|---|---|---|---|---|---|
| *Exp(60)* | *Exp(3)* | *Exp(8)* | *Exp(5)* | *Exp(0.5)* | *PII(1,1.2,12)* |

Table 3.1: *Web traffic profile.*

The generated traffic exhibits a high variability property: More than 80% of the flows have less than 10 packets and that flow sizes of more than 100 packets constitute less than 1% of all

[1]For this reason, object size and flow size are the same, and we use both terms interchangeably.

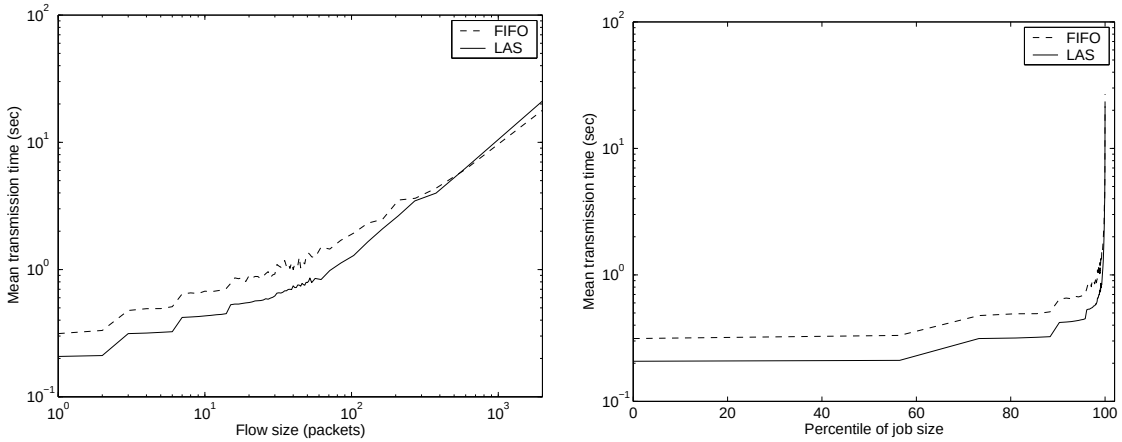
flows. In Figure 3.5, *Source* is either an infinite TCP or UDP source (*long-lived flow*) sending data to the destination *Sink*. The bottleneck link in the network topology is R1–R0 with its queue size limited to 60 packets. We analyze and compare the performances of LAS and FIFO schedulers when placed in the bottleneck link. For all simulations, packets that carry Web traffic or long-lived FTP data have a size of 1040 bytes, and packets of the long-lived UDP flow are 1000 bytes in size.

Each simulation is run for 6000 seconds, and the statistics are collected after a warm up period of 2000 seconds. Similarly, if *Source* is active, it will start sending data after 2000 seconds. In the next section, we discuss simulation results that compare the performance of LAS to FIFO in terms of mean transfer time and packet losses for short flows.

### 3.4.2 Transfer Time Performance

In this section, we discuss simulation results that show how LAS reduces the mean transfer time of short flows. Here, the transfer time of flow is the time interval starting when the first packet of a flow leaves a server and ending when the last packet of the flow is received by the corresponding client.

Figure 3.6(a) and 3.6(b) show the mean transfer time for LAS and FIFO as a function of object size and percentile of object size respectively. It is evident from the figures that LAS significantly reduces the mean transfer time with a negligible penalty to large flows. We note that more than 99% of the short flows benefit from LAS. We also observe that the mean transfer time of large flows under LAS shows only a small increase compared to FIFO. It is worth recalling, however, that based on the distribution of generated flow sizes, these large flows (of size greater than 100 packets) constitute less than 1% of all flows.




(a) *Flow sizes* (b) *Percentiles*

Figure 3.6: *Mean transfer time of ParetoII Web flow sizes, load* $\rho = 0.73$.

Table 3.1 indicates that on average, there are three objects in a page and the average object size is 12 packets. This implies that the average page size is 36 packets, which is about 37KB.

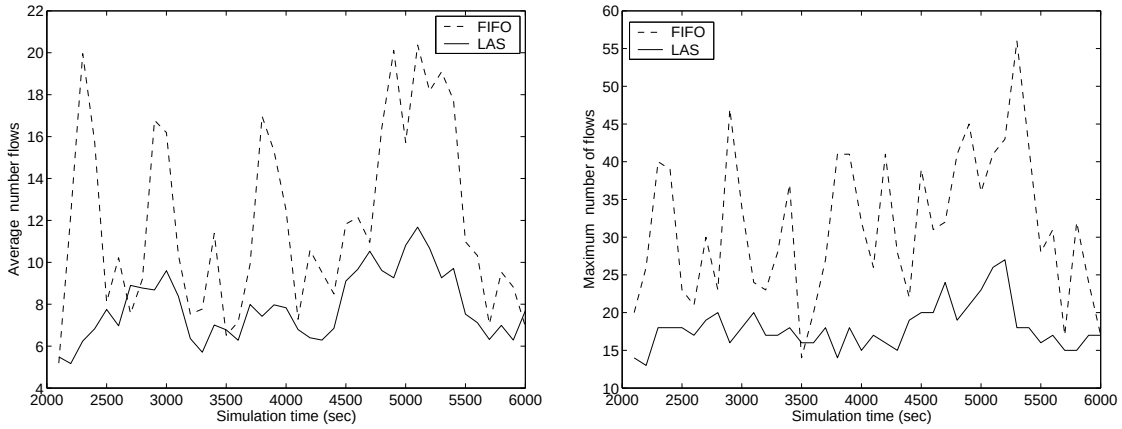
This number complies with the findings in [68] that the average size of all data on a Web page was 26-32KB. We see from Figures 3.6 that the mean transfer time of the Web transfer with 36 packets is lower under LAS than under FIFO. Thus, even if all objects in a page are transmitted using a single persistent HTTP connection, as the case for HTTP1.1, they will still experience a lower average transfer time under LAS than under FIFO.

In this section, we presented the results only for a single load value of $\rho = 0.73$. We also looked at other loads and observed that short flows also benefit under LAS at lower load values like load $\rho = 0.5$ and the performance of short flows under LAS as compared to FIFO improves with increasing load values .

### 3.4.3 Number of Simultaneously Active Flows

In this section we use simulation to compare the number of simultaneously active flows in a network with LAS to the network with FIFO schedulers. We use the network topology shown in Figure 3.5.

Figure 3.7(a) shows the average number of active Web flows in the network for the case when the bottleneck link R1–R0 uses LAS and when it uses FIFO. The results are obtained by considering the flows that have started sending data but have not yet terminated in every non-overlapping interval of 100 seconds. Figure 3.7(a) clearly shows that the average number of flows, in a considered time window, is lower under LAS than under FIFO. The overall mean number of flows under LAS is 7.8 flows, which is lower than 12.0 flows under FIFO scheduling. This result strongly indicates that Little's law may well apply for flows in packet networks.




(a) *Mean number of flows* (b) *Maximum number of flows*

Figure 3.7: *Number of flows as a function of simulation time, load* $\rho = 0.73$.

Figure 3.7(b), shows the maximum number of flows in the network for both, LAS and FIFO schedulers, in every non-overlapping time interval of 100 seconds. The results show the significant difference between the two policies; the maximum number of active flows under FIFO is close to twice the maximum number of active flows under LAS.

These results also show that LAS scheduler actually needs to maintain states for only a small number of simultaneous active flows. This is important when implementing LAS in routers (discussed in Section 3.6) as the number of flows to keep state information in LAS routers is reduced by the properties of the scheduler itself.

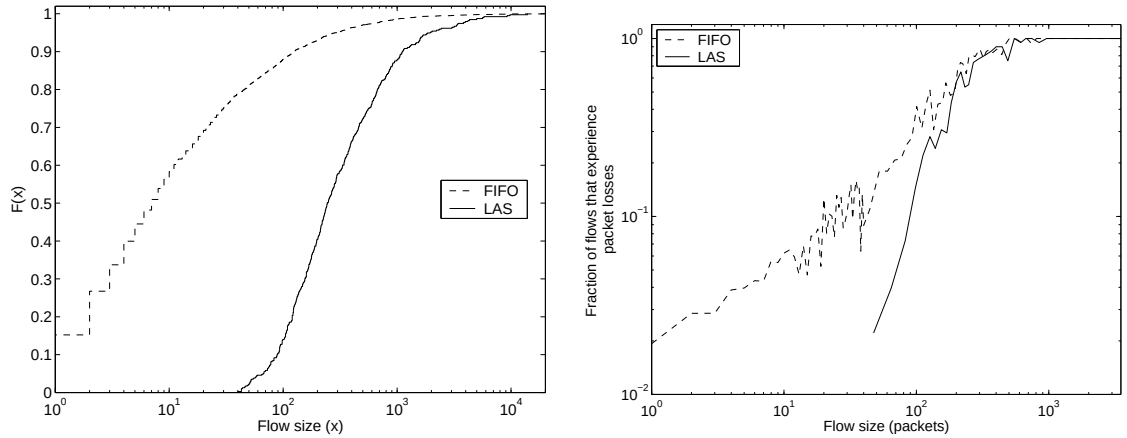
### 3.4.4 Loss Rate Performance

The overall performance of TCP depends to a certain degree on how it can efficiently detect and reduce network congestion. In general, short TCP connections are not capable of controlling congestion. When a short flow with very few packets experiences loss(es), the congestion window is too small that reducing its size will have no impact on reducing congestion. On the other hand, the network congestion is alleviated when a source with a large congestion window detects loss. In this case, reducing the congestion window reduces the overall network load and therefore the congestion. Hence, reducing the congestion windows of a few flows with large congestion windows can be sufficient to react to network congestion.

LAS speeds up the slow start phase of all flows (including largest ones) and enables sources to reach the CA phase faster. This improves the throughput performance of TCP since TCP flows under LAS, especially large flows, are likely to detect packet losses during the CA phase. Perhaps this is the reason why long flows under LAS receive similar transfer times as under FIFO. In routers with FIFO scheduling, short flows are equally likely as large flows to see their packets dropped, which deteriorates their throughput performance without helping to alleviate congestion. When the queue is full, LAS first inserts an incoming packet at its position in the queue (based on its priority value), and then drops the packet that is at the tail of the queue. It turns out that LAS very much protects flows from losing packets during the initial slow start phase and most likely drops packets from large flows. Apparently, simulation results also assert that speeding up the slow start phase and avoiding packet losses in this phase make LAS such an effective policy for short flows. In this section, we discuss the simulation results that illustrate the positive impact of LAS on the packet losses seen by short flows.

The simulation is performed with the parameters of Table 3.1 for Pareto distributed flow sizes. The overall loss rate for LAS is about 3.3% whereas for FIFO it is about 3.9%. The number of flows that experience one or more packet loss under FIFO is 12212, almost twice the number of flows that experience loss under LAS, which is 6763. Figure 3.8(a) shows the distribution of flows that experience packet losses under LAS and FIFO. We observe that short flows of size less than 40 packets do not experience any packet loss under LAS, whereas these short flows make about 80% of flows that experience packet losses under FIFO. In general, we observe that it is essentially the short flows that benefit from LAS in terms of significantly reduced packet losses.

Furthermore, Figure 3.8(b) shows the ratio of flows that experience packet losses over all flows of a given size. Again, we note a significant difference between LAS and FIFO for short flows; the percentage of flows that suffer losses is always higher under FIFO than under LAS for all flow sizes. Finally, it is also useful to analyze the average packet loss rate to understand if LAS reduces the loss rate of short flows by increasing the loss rate for large flows. Figure 3.9




(a) *CDF of flows that experience loss* (b) *Flows loss ratio*

Figure 3.8: *Loss analysis for Pareto distributed Web flow sizes, load* $\rho = 0.73$.

shows the average packet loss rate as a function of flow size. It is evident from Figure 3.9 that

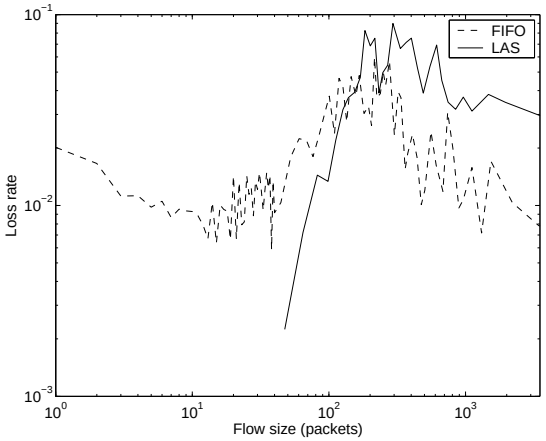



Figure 3.9: *Packet loss rate for Pareto distributed Web flow sizes,* $\rho = 0.73$

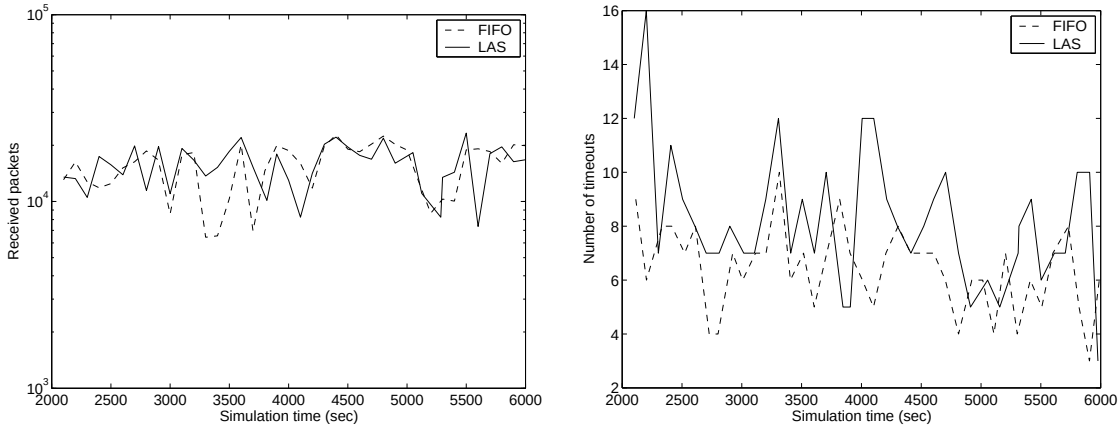
LAS eliminates loss rate for short flows less than 40 packets and approximately doubles the loss rates of large flows as compared to their loss rates under FIFO. Recall that these large flows are only the 1% largest flows in the traffic. Therefore, the significant reduction of loss rate of short flows under LAS does not come at the expense of a too high loss rate for the largest flows. In summary, it seems that the substantial packet losses reduction offered by LAS for short flows is the most important factor that improves the performance of short TCP flows under LAS as compared to under FIFO.

## 3.5 Single Long-lived Flow Competing with Web Background Traffic

We can see from Figure 3.6 that the mean transfer times of the largest generated Web flows under LAS do not differ much from their mean transfer times under FIFO. Similarly, we observed in Section 3.4.4 that the loss rates of large Web flows under LAS and FIFO are comparable. These results allow us to conclude that the performance of LAS for short flows comes with a negligible penalty in terms of increase of loss rates and mean transfer times of large flows. In this section, we want to investigate the performance of even larger (long-lived) TCP and UDP flows than the largest transmitted Web flow under LAS and compare the results under FIFO.

### 3.5.1 Single Long-lived FTP Flow

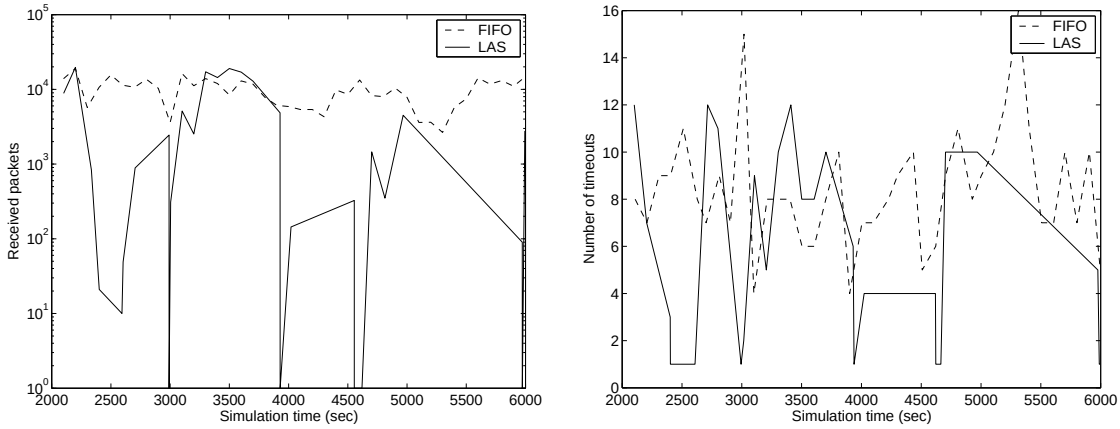
We use again the network topology shown in Figure 3.5. The long-lived FTP flow is represented by *Source* that starts sending data packets to *Sink* after $2000$ simulation seconds. The FTP source uses TCP Reno. The background Web traffic is generated using the parameters from Table 3.1 where Web flow sizes are distributed by Pareto distribution of second kind. We consider three load conditions: a total load of $\rho = 0.75$, $\rho = 0.9$, and $\rho = 0.98$. At load $\rho = 0.9$, the background Web traffic is generated using parameters shown in Table 3.1, whereas to obtain a total load of $\rho = 0.75$ and $\rho = 0.98$, we adjust the distribution of inter-session time in Table 3.1 to $Exp(11)$ and $Exp(6)$ respectively. We are interested in timeout and throughput behavior of a long-lived FTP transfer that shares the bottleneck link with the Web short flows.




(a) *Throughput* (b) *Timeouts*

Figure 3.10: *FTP flow at load* $\rho = 0.75$.

Intuitively, one expects the FTP source under LAS to starve as a result of LAS favoring short Web flows. The simulation results show that the starvation does not occur except at very high load values of $\rho > 0.9$. Figures 3.10 and 3.11 show the number of timeouts and the number of packets received in non-overlapping time intervals of 100 seconds as a function of simulation



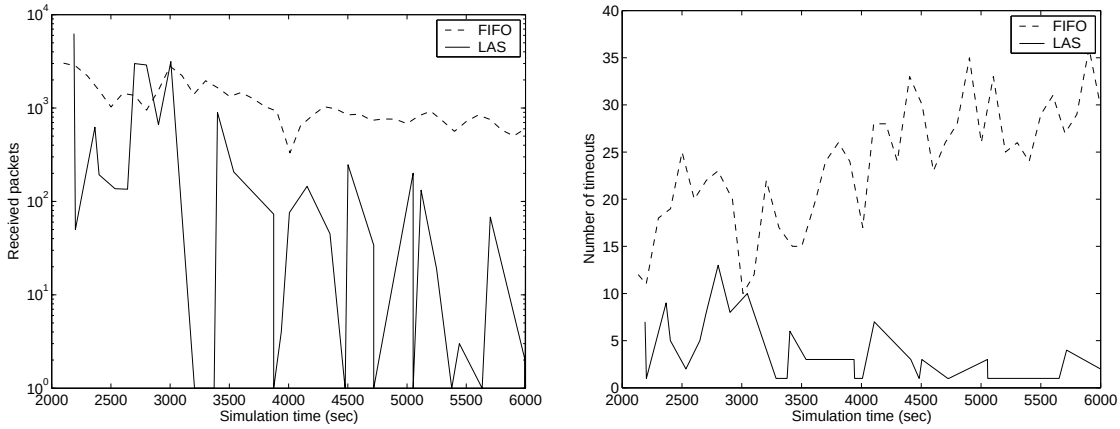

(a) *Throughput* (b) *Timeouts*

Figure 3.11: *FTP flow at load* $\rho = 0.9$.

time for load $\rho = 0.75$ and $\rho = 0.9$, respectively. For the case of $\rho = 0.9$, we observe only a few time instants when the long-lived flow is completely starved (has a zero throughput). These are possibly the epochs when the generated Web short flows have high transient load. The number of timeouts for the FTP flow at these epochs are also zero, which only means that the flow stops sending data.

The situation is different for a very high total load close to overload. At $\rho = 0.98$, LAS is observed to starve the FTP flow after some time instance (Figure 3.12). For LAS, both the number of timeouts and the throughput widely oscillate and decrease over time. During some periods, throughput values drop all the way to zero, when the FTP flow completely shuts off as the underlying TCP source stops sending packets for some time. Note that this occurs after a notable simulation duration of 3000 seconds, which is equivalent to about $10^3$ packets or about 1 Mbytes of data transfer of the FTP flow. Recall that the bottleneck link at high load gets very congested and LAS starves the long-lived flow as a result of favoring many short ones. Under FIFO however, starvation does not occur and the FTP source continues sending the data until the simulation terminates. We observe from the figure for the case of FIFO that the number of timeouts slightly increases and the throughput of the flow slightly decreases as the simulation time increases. FIFO however does not favor short flows, and this result is consistent with the well known lock-out phenomenon of long-lived TCP flows against short flows under FIFO.

Similarly, Table 3.2 shows the mean and variance of the throughput of the ftp flow under LAS and under FIFO at different load values. The simulation results in Table 3.2 are obtained by first calculating the throughput of the flow (in packets/sec) in each window of 100sec, and then by computing the mean and variance of the throughput samples obtained. At load 0.75, we observe that LAS offers similar performance to FIFO. At load 0.92, the throughput of the FTP flow deteriorates under LAS to a small value of 33.93 packets/sec as compared to 97.55 packets/sec under FIFO. The long-lived flow under LAS when the network is overloaded or close to 1 is almost shut down by the many Web flows in the network with high priorities. The variance of the throughput in Table 3.2 increases under LAS as load increases because there are epochs when the FTP flow is completely shut down by short flows, which is not the case under




(a) *Throughput* (b) *Timeouts*

Figure 3.12: *Performance of FTP flow at load* $\rho = 0.98$.

FIFO.

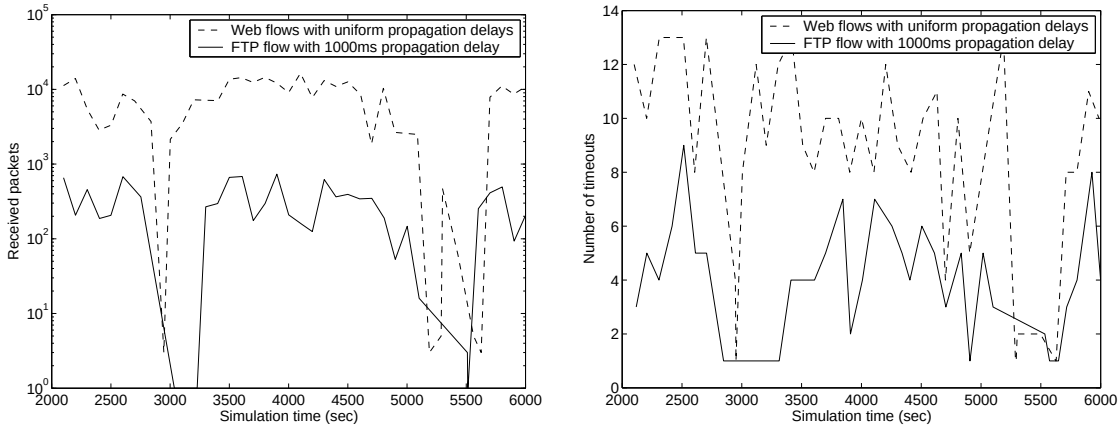
| | LAS | | FIFO | |
|---|---|---|---|---|
| Load $\rho$ | mean | variance | mean | variance |
| 0.75 | 157.8 | 1.03 | 156.30 | 1.31 |
| 0.92 | 33.93 | 2.27 | 97.55 | 1.01 |
| 0.98 | 4.82 | 151.35 | 11.86 | 43.68 |

Table 3.2: *Mean and variance of throughput (packets/sec) under LAS and FIFO.*

### 3.5.2 Impact of Varying Propagation Delays

In the previous analysis, we considered the topology when all servers-client pairs have the same one way propagation delay of 50ms. When flows have the same RTT values and maximum advertized window values, they also have equal maximum source rates. This increases burstiness and aggregation of the traffic at the bottleneck and under LAS may cause the performance of the long-lived flow to deteriorate. In this section, we show simulation results when the propagation delays in Figure 3.5 are changed.

We consider two cases: the case when only the propagation delay of the FTP source is changed to 1000 seconds and the case when propagation delays of links connecting servers and clients are uniformly distributed between 50ms and 400ms. Web parameters are kept the same as those used in Section 3.5.1 and resulted to total load of $\rho = 0.9$. Simulation results showing the performance of the FTP flow for both cases are shown in Figure 3.13. We observe that compared to the results shown in Section 3.5.1, LAS offers a higher throughput when propagation delays of links are not the equal. Note that the throughput is higher when propagation delays are uniformly distributed than when only the propagation delay of the FTP flow is increased. We observe also that the FTP experiences more timeouts for the case of different propagation delays but in most cases they are higher than 1, which shows that the FTP flow hardly completely




(a) *Throughput* (b) *Timeouts*

Figure 3.13: *Performance of FTP flow for topology links with varying propagation delays.*

shuts off.

### 3.5.3 Single Long-lived UDP Flow

When FIFO scheduling is used in routers, open-loop traffic can pose a danger to TCP traffic because it does not respond to network congestion as do TCP-based applications. It is well known that when UDP-based applications share a bottleneck link with TCP-based applications in a FIFO router, they tend to grab as much bandwidth as corresponds to their source rates, while TCP flows will back-off and see their throughput decreased. LAS, by definition, prevents any flow to consume a larger share of bandwidth than any competing flow, regardless of the underlying transport protocol. In particular, LAS fairly services all flows that have received the same amount of service in the system.

Here we only consider long-lived UDP flows but it is obvious that short UDP flows can benefit from LAS more than short TCP flows; for short UDP and TCP flows of equal sizes that arrive at the LAS router at the same time, the UDP flow (particularly with CBR source rate) can complete the service earlier than the TCP flow that is subject to a closed loop control mechanisms. In the next section, we investigate the performance of a long-lived UDP flow when competing with short TCP flows under the LAS scheduling policy. We compare the simulation result for the case of FIFO scheduling.

#### Loss Analysis of a Long-lived UDP Flow

In long run, the long-lived UDP flow under LAS has a lower priority than (most of) Web background traffic. As the UDP flow is open-loop and does not adapt its rate to packet losses, the UDP flow can suffer a significant loss during periods of congestion as seen in Figure 3.14, which shows the number of packets lost in every block of 100 packets sent. Compared to un-

der FIFO, the long-lived UDP flows under LAS experiences pretty bad loss performance. To solve this problem, we propose new LAS-based differential policies that give service priority to long-lived flows that are classified as priority flows in Chapter

5.

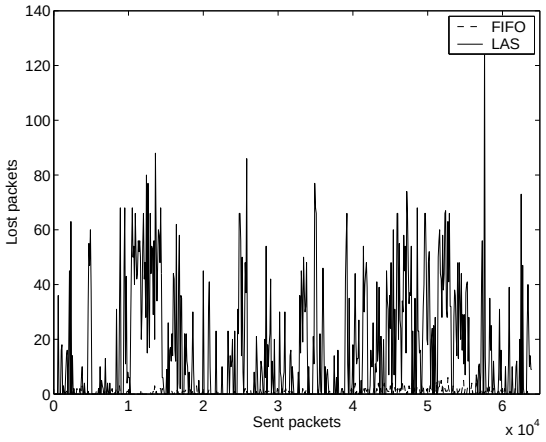


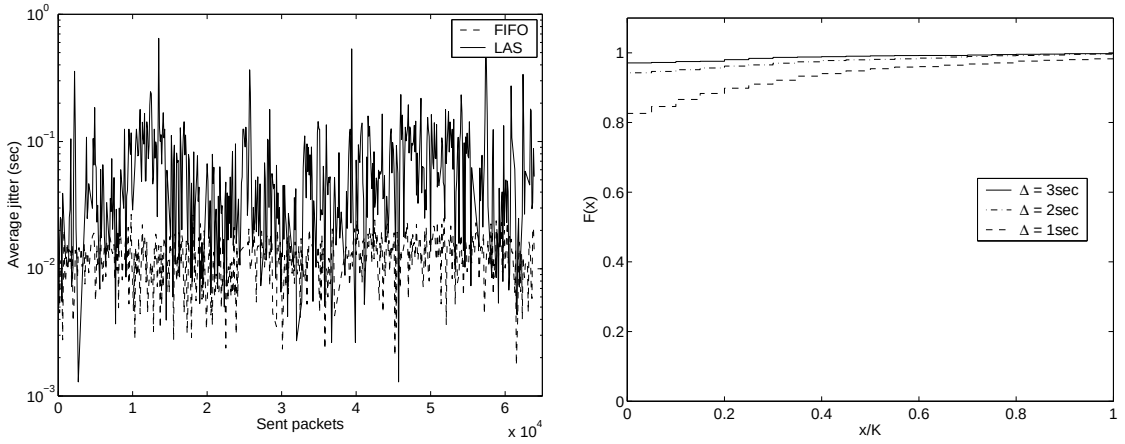

Figure 3.14: *Loss performance for a UDP flow at load* $\rho = 0.75$.

The scheduling policy has a big impact on how closed-loop TCP flows and open-loop UDP flows interact in a bottleneck router; (i) Under a LAS scheduler, the TCP flows of Web traffic have buffer space and service priority over the long-lived UDP flow. As a consequence, in case of congestion, the long-lived UDP flow suffers high jitter and an increased loss rate. (ii) Under FIFO scheduler it is the opposite as UDP flows will degrade the performance of the TCP flows: As the open-loop UDP flow does not react to congestion, it will be the TCP flows that suffer loss and back-off, seeing their share of the bottleneck bandwidth reduced.

### Jitter Analysis

We use the network topology shown in Figure 3.5 where *Source* is a UDP source that starts sending data to *Sink* at a Constant Bit Rate (CBR) of 16 packets/sec after an initial warm-up period of 2000 seconds. Thus, the UDP source generates a long-lived flow of $16 \times 10^4$ packets when the simulation terminates 4000 seconds later, which makes $160$ Mbytes of transferred data. The Web background traffic is generated using the parameters shown in Table 3.1.

The jitter performance is an important performance metric for multimedia applications, since these multimedia applications have timeliness requirements. We define jitter as the absolute difference between the differences in inter-departure times and in inter-arrival times of two consecutive packets of a flow. Figure 3.15(a) shows the jitter averaged over non-overlapping groups of 100 packets sent. It is evident that the UDP flow experiences a much higher average jitter under LAS than under FIFO. The UDP flow under LAS is always pre-empted by new TCP flows, which delays its packets in the system and so increases the jitter.




(a) *Average jitter*

(b) *The CDF of the ratio of the packets in a block of $K = 20$ packets that miss deadline*

Figure 3.15: *UDP flow performance, load $\rho = 0.75$.*

While the simulation results show high jitter for the long-lived UDP flow under LAS, we argue that a small *playout buffer* of no more than 4 seconds for this simulation scenario guarantees that no packet misses the *deadline*. A packet is said to miss its deadline if the time difference between its send time $t_s$ and the receive time $t_r$ is larger than or equal to the sum of the playout buffer size $\Delta$ and the average transfer time $\bar{T}$; i.e., $t_r - t_s > \Delta + \bar{T}$. Figure 3.15(b) shows the cumulative distribution of the ratio of the packets in a block of $K = 20$ packets that miss deadline, where $\bar{T} = 0.5$ seconds. We observe from the figure that the number of packets that do not miss their deadlines increases in increasing playout buffer size. For $\Delta = 3$ seconds about 98% of blocks have all packets arriving at the destination in time, and all packets arrive in time for $\Delta = 4$ seconds.

## 3.6 Implementation Issues

Implementing LAS scheduler involves two main tasks: Maintaining a priority queue that implements the highest-priority-first queue, and keeping state of all active flows in the network at any time. There are a number of methods proposed to implement a priority queue for high speed packet networks. Moon et. al. present a detailed comparison of different existing hardware approaches used to build priority queues [72]. The work in [16] presents a hardware implementation called *pipelined priority queue architecture* for a priority queue that implements the highest-priority-first schedulers (like LAS). The architecture presented in [16] is shown to be fast and to scale well with respect to the number of priority levels and the queue size. In particular, the work shows that the implementation can support connections with up to 10Gb/s rates and over 4 billion priority levels.

A network flow is identified by its source and destination addresses and ports. For each flow of packets, a router with LAS scheduling must (a) identify the first packet and subsequent packets in a flow, (b) use the sequence number of packets in the flow to obtain the amount of

bytes ($L$) carried by each arriving packet, and (c) use $L$ value of each packet to compute the the amount of bytes of the flow served so far. We propose a *priority assignment unit* in LAS scheduling whose functions include computing the priority of each arriving packet and inserting the packet at its appropriate position in the queue. The priority assignment unit then tracks for each flow $f$ the amount of bytes $S_f$ served so far. $S_f$ is initialized with 0. The updated value of $S_f$ after each packet arrival determines the priority of the next packet of the flow $f$ and so is used to determine where to insert it into the priority queue. Recall that the lower the priority value the higher the priority, and the closer to the head of the queue the packet is inserted. When a packet of size $L$ for flow $f$ arrives, the priority assignment unit executes the following operations:

– Use $S_f$ to insert packet into priority queue

– Update $S_f$ value to $S_f := S_f + L$

LAS at a link will be non-preemptive and the packet to be served next will be taken from the head of the queue. Packets having the same priority ($S_f$ values) are serviced in first-come-first-serve order. This implies that the packets from flows that have received the same amount of service are served in an approximate round-robin fashion, which is an approximation of the processor-sharing service of equal priority jobs in the idealized LAS policy.

Keeping per-flow state is a challenging task for routers in the Internet backbone links due to the large number of simultaneously active flows. However, congestion and packet losses occur most often at the edges and the access links of the Internet [80], where the number of active flows is moderate. Moreover, it is found in [60] that only less than 10% of the links in the backbone experience utilization higher than 50% as a consequence of bandwidth over-provisioning. Since bottleneck queues have a significant impact on end-to-end performance, deploying LAS in routers at the access links that transmit packets over bottleneck links should reap the benefits of LAS in terms of reducing the transfer time of short TCP flows.

Recent developments in network processors allow per-flow states of more than million concurrent flows to be processed by a router interface at a line speed [86]. In addition, the analysis of traffic data collected from Sprint IP backbone shows that the average number of active flows per second in backbone is less than half a million [44]. We thus articulate that deployment of LAS in backbone bottleneck links will be feasible in the near future.

We are aware that the deployment of a new scheduling policy in routers will face many obstacles and wide deployment will be an elusive goal. However, we expect that a limited deployment of LAS in certain links at well identified bottlenecks such as access links allows to improve the perceived user performance. Such bottlenecks can be the access links, edge links, or the transition from the wired to the wireless Internet.

In case LAS needs to be implemented at all links of the network, the Diffserv like model, which pushes per-flow state complexity to access links, can be adopted. In this case, priority assignment units in edge routers compute and stamp priority value of each each packet in a priority field to be created in packets headers (See [16] for possible parameters of a packet priority field). Routers in core links then do not need to compute priorities and to keep per-flow state. Instead, they use the priority field value of each incoming packet to insert the packet into its appropriate position in the queue.

## 3.7 Conclusion

LAS scheduling for jobs has been known for over thirty years. The work in in this chapter is the first attempt to evaluate the impact of LAS link scheduling in routers to TCP as well as UDP flows. Using a realistic traffic scenario that captures the high variation of the flow sizes observed in the Internet, we saw that LAS scheduler in the bottleneck links (i) significantly reduces the mean transfer time and loss rate of short TCP flows, (ii) does not starve long flows at all, and long-lived TCP flows under LAS are penalized only at high load values close to overload. Since LAS gives space and service priority to newly arriving flows, it degrades the jitter and loss performance of long-lived UDP flows. However, a small playout buffer at the destination allows to compensate for the jitter.

Various authors (see Section 3.2) have proposed to improve the performance of short TCP flows by changing the mechanisms deployed in the routers. However, our proposal has several distinctive features:

- LAS is conceptually very simple as there is not a single parameter that needs to be set,
- The priority queue of LAS improves the transfer time of short flows much more than schemes that propose to use *separate FIFO* queues for short and long flows,
- LAS protects TCP flows against open-loop UDP flows, which FIFO scheduling does not.

There remain a number of questions regarding the performance benefits of LAS for short and long TCP flows. TCP in FIFO networks is known to be unfair against some flows in heterogeneous networks environments. Examples of these networks include networks shared by UDP and TCP applications, networks with heterogeneous propagation delays, and TCP networks with asymetric flows that traverse multiple congested gateways. By definition, the LAS scheduler should enforce fair bandwidth sharing among competing flows since LAS gives service to flows based on the amount of data sent. Hence, intuitively no flow under LAS, regardless of its transport protocol, can hog the link bandwidth and lock-out other flows. We will evaluate how LAS tackles this problem in the next chapter.

# Chapter 4

# *LAS* Scheduling to Avoid Bandwidth Hogging in Heterogeneous TCP Networks

We define a TCP network as the network where source rates are controlled by TCP. In this chapter, we consider the performance of heterogeneous TCP networks such as networks with connections that have heterogeneous propagation delays, networks that support TCP and UDP applications, and networks with multiple congested links. These heterogeneous networks are known to cause unequal bandwidth allocation among competing connections to the extent that some connections occupy all or a large fraction of network bandwidth. This phenomenon is sometimes known as *bandwidth hogging*. Most of the previous work identified TCP algorithms as the main causes of bandwidth hogging. Thus, they proposed to prevent bandwidth hogging by changing the TCP. In this chapter, we show that scheduling policy in a router can also be a cause of bandwidth hogging or can be used to prevent it. In particular, we show in this chapter that LAS link scheduling avoids bandwidth hogging because a connection under LAS scheduler does not receive service if there is another active connection with less attained service. Simulation results in this chapter show that in contrast to FIFO scheduling, LAS scheduling allocates equal bandwidth shares among competing connections in congested heterogeneous networks.

## 4.1 Introduction

One of the main tasks of TCP is to fairly allocate bandwidth to competing users by controlling their sources transmission rates. Despite its popularity and its efforts to fairly allocate bandwidth, TCP in networks with FIFO scheduling is known to allow unequal bandwidth sharing among competing connections to the extent that some connections occupy much larger fraction of bandwidth than other connections. This phenomenon is also called *bandwidth hogging*. The term unfairness will sometimes be used in this chapter to imply bandwidth hogging. Bandwidth hogging problem has been particularly observed in *heterogeneous networks* such as networks with heterogeneous propagation delays, networks with applications using TCP and UDP proto-

cols, and networks with multiple congested links.

Connections in networks follow paths with different propagation delays, which is a function of RTT. In addition, network measurements have shown that RTTs experienced by TCP connections are widely varying (Figure 1 in [3]). TCP inherently causes each TCP flow to receive a bandwidth that is inversely proportional to its *round trip time* (RTT) [79]. Hence, TCP connections with low RTT may unfairly receive a large allocation of network bandwidth compared to other TCP connections in the network with a high RTT. This also explains the problem of TCP in networks with multiple congested links. In these networks, a connection that traverses more links also has a longer RTT than connections that cross fewer links. It is shown in [42] that in networks with multiple congested links TCP bias against connections with long RTT.

UDP-based applications are oblivious to congestion and make no attempt to reduce their sources transmission rates when packets are lost by the network. Instead, they attempt to use as much bandwidth as their sources can send. Thus, UDP-based streaming applications are likely to cause congestion and unfair bandwidth allocations against TCP-based applications. Most of the proposed solutions to avoid bandwidth hogging suggest modifying the TCP protocol itself, e.g., [18] or propose TCP-friendly algorithms for UDP applications [95] and only a few propose solutions inside the network. The work in this chapter is more related to network-based solutions that propose to use different buffer management and scheduling algorithms in network routers.

Random early detection (RED) [43] is a buffer management scheme that distributes packet loss rates to a small number of connections to prevent oscillation problem that occurs in FIFO routers with drop tail queues when multiple connections repeatedly experience packet losses. It is well known that connections get the same throughput if they have the same per-connection queue length. RED does not try to control per-connection buffer occupancy; therefore, it is shown that it doesn't always avoid unfairness [67]. Bandwidth hogging problems in RED networks still exist, since it is possible for other connections to occupy a large fraction of network bandwidth.

Flow RED [67], on the other hand, improves the fairness by sharing the buffer occupancy fairly among active TCP connections and also when TCP connections are mixed with UDP connections. Deficit Round Robin (DRR) [94] approximates PS scheduling and maintains a sub-queue for each flow in the router buffer. DRR uses a deficit counter for each sub-queue to make sure that each connection utilizes no more than its pre-defined service rate in a round manner. Through this, DRR provides reasonable fair service among connections. Using simulations, [56] shows that DRR is fair for heterogeneous networks where connections have different propagation delays and capacities.

This chapter proposes to use LAS scheduling to prevent bandwidth hogging in heterogeneous networks. Recall that LAS-based routers maintain a single priority queue and insert each incoming packet at its appropriate position in that queue. The less service a connection has received so far, the closer to the head of the queue its arriving packets will be inserted. When a packet arrives and the queue is full, LAS first "inserts" the arriving packet at its appropriate position in the queue and then drops the packet that is at the end of the queue. Thus, LAS

should avoid bandwidth hogging since it inserts packets of a connection that has received the most service at the tail of the queue to make sure that the connection does not receive service until other active connections have received equal amount of service or they are idle. Also, packets at the tail of the queue (belonging to connections that have received the most service under LAS) are dropped in case the queue being full. Recall that dropping these packets in TCP networks, makes corresponding sources to reduce their rates.

We simulate LAS by using the network simulator ns-2 [1].The simulation results presented in this chapter show that, unlike FIFO scheduling, LAS scheduling at links is efficient to prevent any connection, regardless of its propagation delay or its transport protocol, to occupy an unfairly large fraction of network bandwidth throughout the active duration of the connection. These results further show that, as opposed to a common belief, the bandwidth hogging problem in TCP networks is not caused by TCP algorithms only but the scheduling discipline may as well contribute to the problem.

## 4.2 LAS in Heterogeneous Networks with a Single Congested Link

In this section, we analyze the performance of LAS in simple heterogeneous networks with a single congested link. We consider networks with varying propagation delays and networks with TCP-based applications competing against UDP applications. We simulate a simple net-

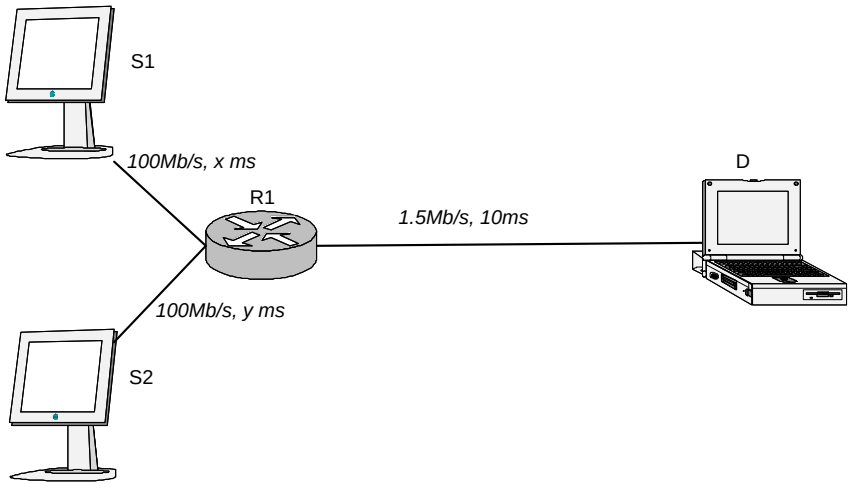
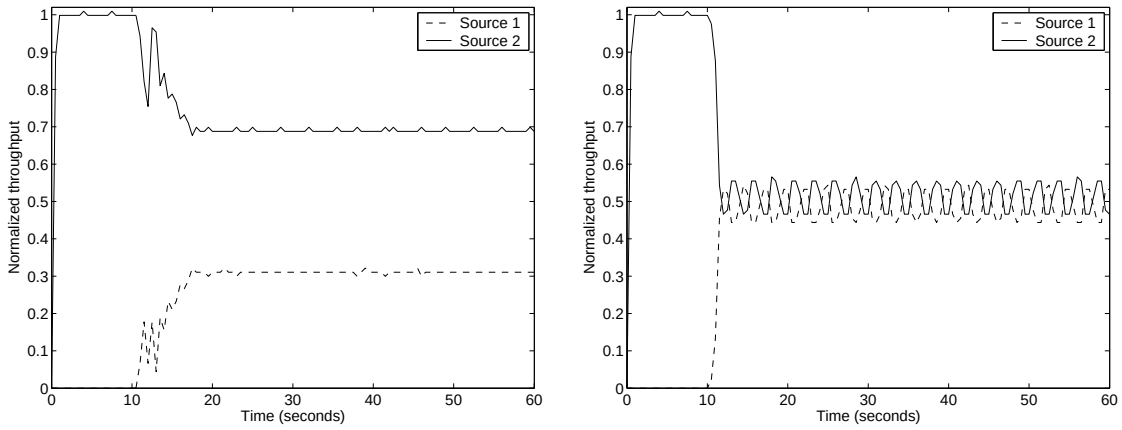



Figure 4.1: *A network with a single shared link.*

work topology shown in Figure 4.1, where two sources S1 and S2, send data to the same destination D through a bottleneck link R1-D, where LAS or FIFO scheduler is deployed. We vary the propagation delay values $x$ and $y$ and we set source types to TCP or UDP.

### 4.2.1 TCP Sources with Heterogeneous Propagation Delays

Internet traffic is a mixture of traffic traversing different paths, with links of different capacities and propagation delays e.g., asymmetric links such as those that use both terrestrial and satellite links, ADSL or cable modem. In this section we study the impact of LAS to the throughput performance of TCP connections with varying RTTs. The fact that the offered throughput under TCP is inversely proportional to connection's RTT, is widely known to cause unequal bandwidth allocation among competing flows. That is, TCP gives a higher share of bandwidth to connections with low RTTs than to connections with high RTTs. FIFO schedulers, on the other hand, service packets in accordance of their arrivals: it takes no action to avoid a flow monopolizing all the service. In this section, we present simulation results that show that LAS scheduling is a suitable policy that avoids this problem. LAS schedules connections taking into account their received service, and schedules first the packets that belong to a connection that has received the least service. Thus, LAS either drops or buffers packets of a connection that has attained the most service. In doing so, LAS increases the queuing delay of the connection, which stretches its RTT and so reduces its throughput.




(a) *FIFO* (b) *LAS*

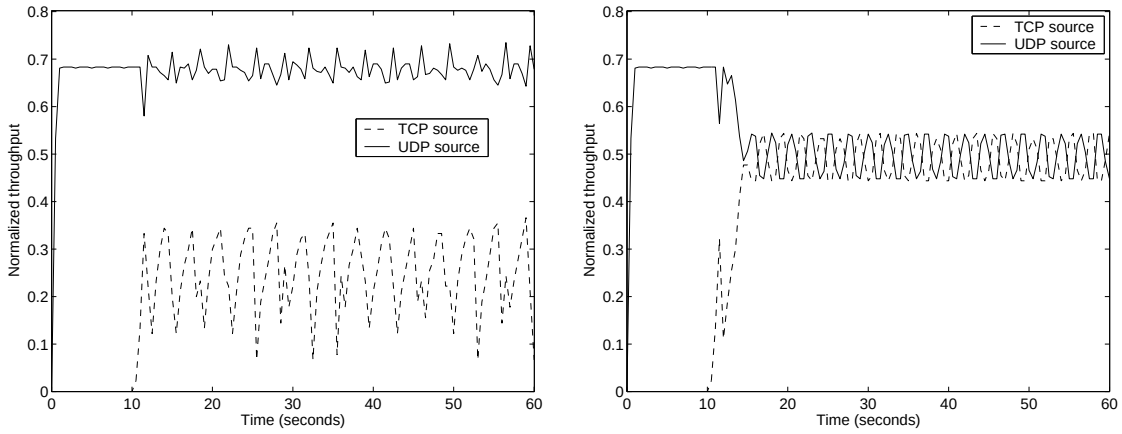
Figure 4.2: *Throughput obtained by TCP flows with different RTTs.*

We study the network shown in Figure 4.1, where both sources use TCP and the propagation delays for links S1-R1 and S2-R2 are $x = 100ms$ and $y = 1ms$ respectively. S2 starts transmitting data at time 0 and S1 starts 10 seconds later. Figure 4.2(a) shows simulation results when the scheduling discipline at link R1-D is FIFO with droptail bugger management and Figure 4.2(b) shows the results for LAS. We clearly note the benefits of LAS over FIFO; S2 attains an equal share of bandwidth as S1 under LAS throughout the duration of the simulation. For the case of FIFO, however, the source transmitting over low RTT link (S2), receives a significantly larger share of bandwidth than does source S1.

### 4.2.2 Competing UDP and TCP Sources

Supporting UDP-based applications in the Internet is known to be difficult mainly because they do not respond to network congestion like TCP-based applications. As a result, when transmitted over the same link with TCP-based applications, UDP applications tend to take as much bandwidth as their source rates. In this section, we present simulation results that first illustrate this problem in the current Internet architecture (with FIFO schedulers) and then show that, under the same network, LAS can fairly allocate bandwidth among competing UDP and TCP sources.

We again use the network shown in Figure 4.1, where S1 is a TCP source and S2 is a UDP source transmitting constant bit rate (CBR) application at a rate of 1024Kb/s. The propagation delays $x$ and $y$ are both set to 100ms.




(a) *FIFO* (b) *LAS*

Figure 4.3: *Throughput obtained by UDP and TCP connections.*

Figure 4.3(a) shows the simulation results in terms of the throughput obtained by connections under FIFO and Figure 4.3(b) shows the result under LAS. We note the unfairness effect for the case of FIFO, where the UDP application occupies the same amount of bandwidth in the presence of a TCP connection as when it is alone in the network. We also observe for the case of FIFO that the throughput of the TCP source shows oscillations with high amplitudes. This is due to frequent packet losses of the TCP connection. In contrast, the TCP connection acquires the same throughput as the UDP connection in the case of LAS. This is achieved by dropping some packets of the UDP connection. These results demonstrate that LAS fairly allocates bandwidth among competing flows with different protocols.

## 4.3 LAS in Networks with Multiple Congested Links

The performance of TCP in a network with multiple congested links was studied by Floyd [42]. The work in [42] showed that a network with multiple links and FIFO schedulers biases against

connections that traverses multiple links (also have long RTT values) to the extent that they may receive a very low throughput. In this section we consider the effect of multiple congested links on the throughput of connections when LAS is implemented in the routers. We compare the results obtained for the case of LAS routers to FIFO routers at bottleneck links.

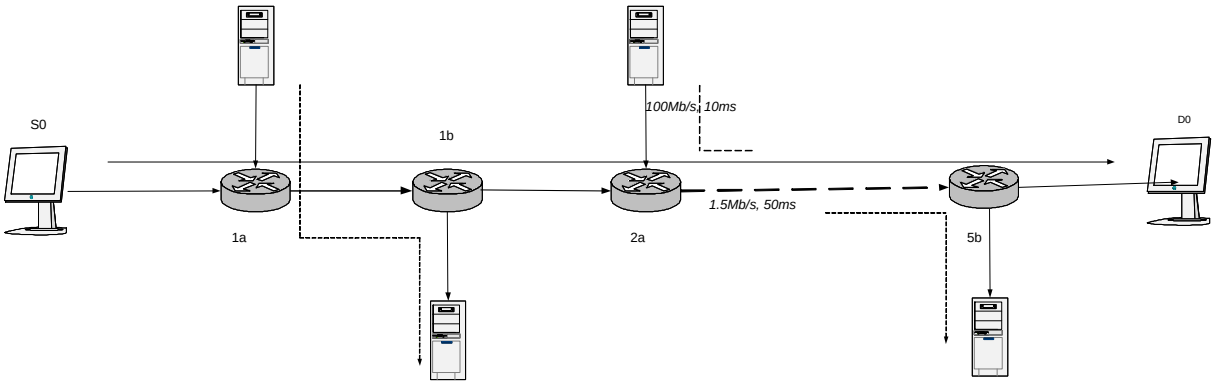



Figure 4.4: *Simulated network topology.*

### 4.3.1 All FTP Connections

We first study LAS under the topology used in [42], also shown on Figure 4.4. All buffer sizes are limited to 60 packets, and the maximum window size is 100 packets. In Figure 4.4, connections 1-5 are FTP connections that traverse a single congested link and have a *low* propagation delay of 70ms each, whereas connection 0 is an FTP connection that traverses multiple congested links and its propagation delay is *high* at 470ms. Thus, connections 1-5 have short RTTs and connection 0 has a long RTT. We analyze the throughput performance of both types of connections when schedulers at bottleneck links, i.e., links 1a-1b, 2a-2b, ..., and, 5a-5b, are either all FIFO or all LAS. All connections send packets during the whole simulation duration.

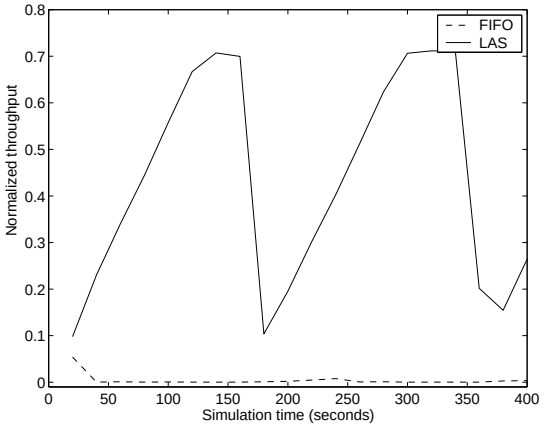
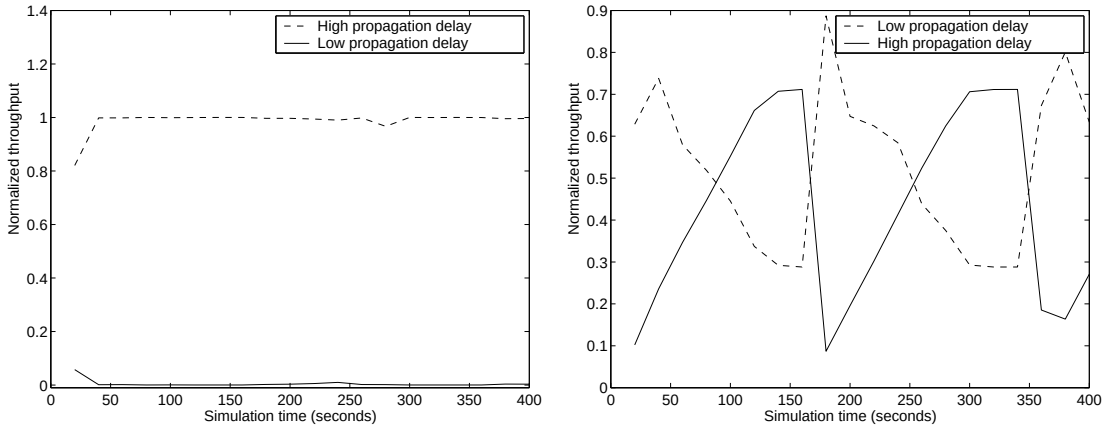



Figure 4.5: *Throughput of a connection with high propagation delay.*

We first analyze the overall throughput obtained by the connection with the high propagation delay (connection 0) under LAS and under FIFO. Figure 4.5 shows the throughput of connection 0 for the network of congested links with FIFO and LAS schedulers. The throughput of connection 0 under FIFO is very low staying at zero during almost the whole simulation, and all the network bandwidth is taken by connections with short RTTs. However, we observe that the throughput of connection 0 under LAS schedulers is higher than under FIFO. The reason for a sharp decrease in the middle of the curve is discussed below.

Figure 4.6 shows the throughput of connection 1 and connection 0 as observed at link 1a-1b. This figure illustrates how the two connections share the bandwidth at bottleneck link 1a-1b. We observe that for the case of LAS (Figure 4.6(b)), the two connections at the bottleneck almost evenly share the link bandwidth as opposed to the case of FIFO (Figure 4.6(a)), where connection 1 occupies all the bandwidth and completely starves connection 0. The results at other congested links were observed to be the same as the results at link 1a-1b.




(a) *FIFO* (b) *LAS*

Figure 4.6: *Throughput at link 1a-1b for connections with high and low propagation delays.*

The throughput of both connections under LAS (see Figure 4.6(b)) is observed to widely oscillate in time. This is primarily caused by the way LAS scheduling works, but also depends on the network parameters used (see next section). We consider connections at link 1a-1b to illustrate the effect of LAS scheduling on the oscillations in throughput. At time time 0, both connections have the same priority under LAS. Connection 1 initially has a higher source rate due to its short RTT and so it rapidly occupies the available bandwidth. As connection 1 sends more packets, the priorities of its packets decrease rapidly, and packets of connection 0 attain a higher service priority in the router. However, the rate of connection 0 increases very slowly due to its long RTT. These are the epochs during which we observe the slow increase and slowly decrease in the throughput of connection 0 and connection 1 respectively. This continues until both connections have sent an equal amount of data, at the simulation time slightly later than 150 sec (also slightly before 350 sec, see Figure 4.7). After this point, we observe a sharp increase in the throughput of connection 1 and a sharp decrease in the throughput of connection 0. These are the times when packets of connection 1 have a higher priority again than those of connection 0. Here, connection 1 rapidly increases its sending rate (congestion window)

whereas connections 0 possibly temporarily backs off with timeouts. Observe in Figure 4.7 that connection 1 tends to send a larger amount of data than connection 0 in almost all times. This is again due to the different RTTs of the connections, which determine their speed of adjustment.

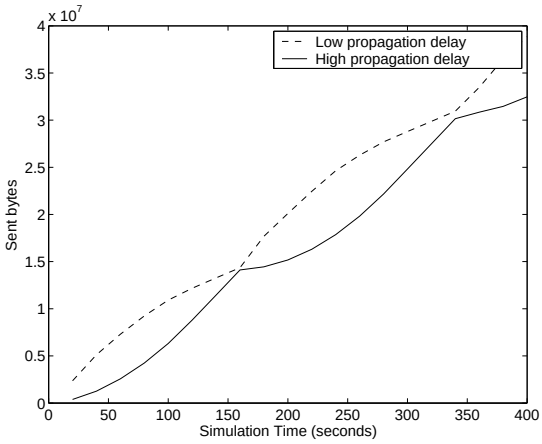



Figure 4.7: *Illustration of LAS at link 1a-1b.*

Finally, Table 4.1 shows the number of lost packets for networks with congested links with FIFO and LAS schedulers. The table shows that LAS looses more packets from short connection than FIFO. The reason for a smaller number of lost packets in connections 1-5 under FIFO than under LAS is that the source of connection 0 under FIFO schedulers completely backs off and does not send packets to the network for a long duration (has zero throughput). Apart from giving an acceptable throughput to connection 0, LAS also maintains as approximately equal number of lost packets as FIFO for this connection.

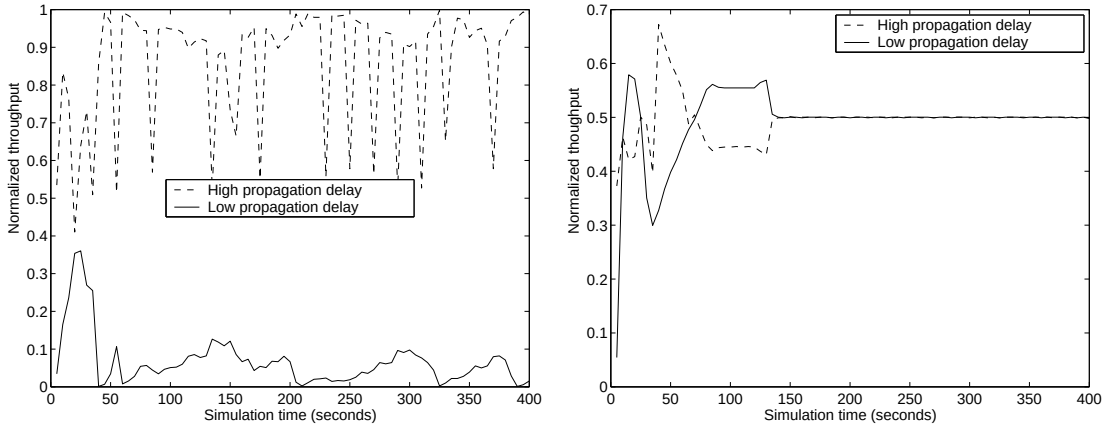
| | All | Connections 1-5 | Connection 0 |
|---|---|---|---|
| LAS | 1013 | 926 | 87 |
| FIFO | 649 | 569 | 80 |

Table 4.1: *Number of lost packets of connections.*

### 4.3.2 Sensitivity of LAS to network parameters

We simulated LAS scheduling for the network topology shown in Figure 4.4 using slightly different parameters to investigate the sensitivity of the LAS scheduling to network parameters. We consider changing either link capacities, the maximum advertized window size, or the buffer size from the parameter set used in Section 4.3.1. Each of these parameters has an impact on the throughput of the connections. For example, TCP source rate increases when increasing the maximum window size and decreasing only the buffer sizes increases the packet loss rate (and thus the TCP source rate decreases).

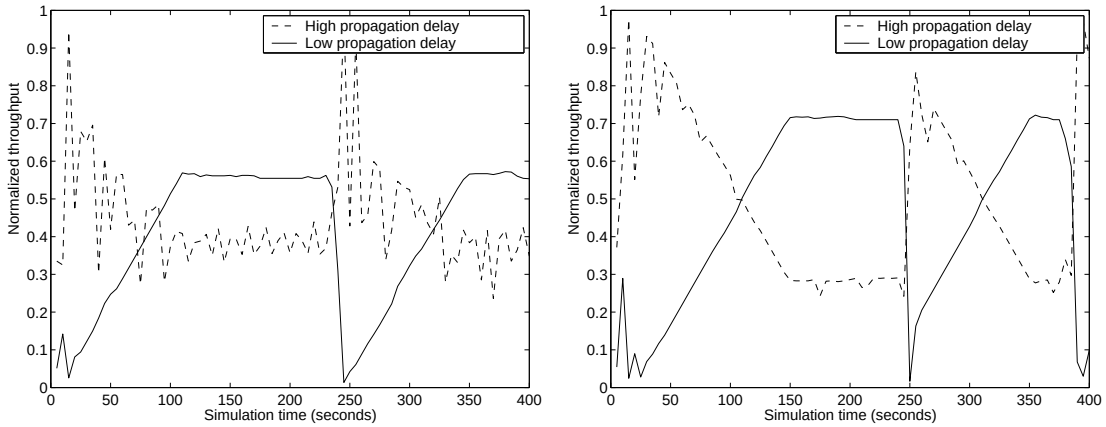
Figure 4.8 shows the throughput results of connection 1 and connection 0 obtained at link 1a-1b when the network access links speeds are changed from 100Mb/s to 10Mb/s while keeping all




(a) *FIFO* (b) *LAS*

Figure 4.8: *Throughput of connections with high and low propagation delays at link 1a-1b for network access link speeds of 10Mb/s.*

other parameters the same as in Section 4.3.1. Observe that connections attain equal throughput under LAS after a short time interval. The throughput of the connection under FIFO remains similar to before where connection 1 occupies almost all the link bandwidth.




(a) *Buffer sizes are 30 packets* (b) *Maximum window is 128 packets*

Figure 4.9: *Throughput under LAS scheduling at link 1a-1b for connections with high and low propagation delays.*

Figures 4.9 show the throughput of connections at link 1a-1b when buffer sizes are changed from 60 to 30 packets (Figure 4.9(a)), and when the maximum advertized window size is changed from 100 to 128 packets (Figure 4.9(b)), all other parameters being the same. We observe that the performance of both connections under LAS scheduling are similar to the results obtained in Section 4.3.1. Hence, we conclude that the performance of LAS is sensitive to network parameters. However, the results shown in this section indicate that the sensitivity of parameters only changes the evolution of throughput of connections and the average throughput of connections is observed to be similar. Hence, we confirm that the LAS scheduler tends to

fairly distribute available bandwidth among all active connections.

## 4.4 Network with Multiple Routers and *Web* Connections

In this section, the connections with short RTTs are Web file transfers with their size distribution exhibiting a high variability property. Thus, the Web transfers consist of many short transfers and a few very large transfers. This flow size distribution agrees with the traffic distribution observed in the Internet today. We consider a Web model with a pool of Web clients that request files from a pool of servers. We simulate the topology (Figure 4.10) with network parameters as shown in the figure. C1-C5 denotes a pool of five clients and S1-S5 denotes a server pool of five servers. Thus, Web files can traverse at least one router (have low RTTs of 140ms), whereas the FTP connection traverses all links (has long RTT 470ms). All buffer sizes are limited to 60 packets, and the maximum window size is 64 packets.

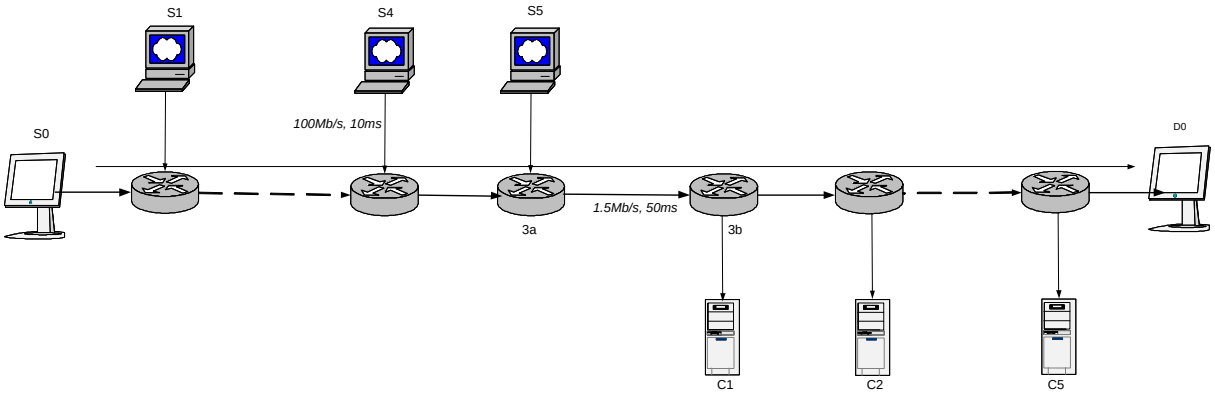
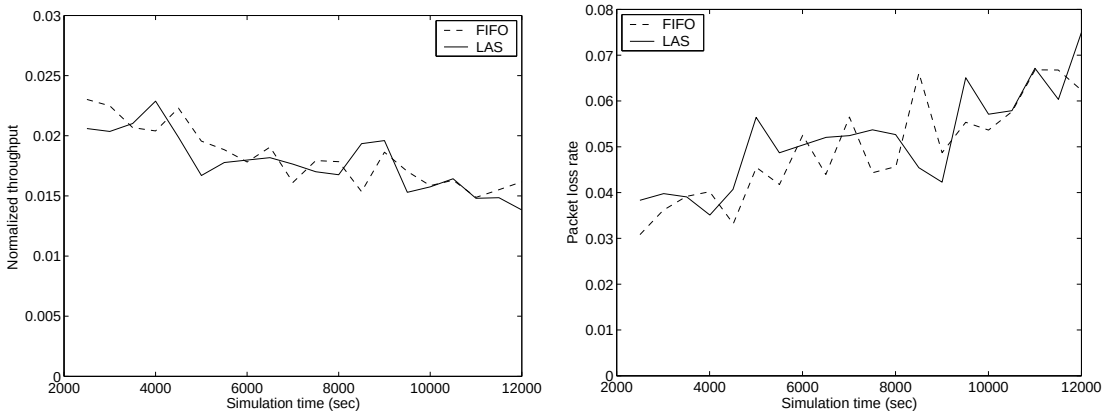



Figure 4.10: *Simulated network topology.*

The topology in Figure 4.10 emulates a network scenario where clients from different autonomous systems (AS) access files stored on Web servers located in the same AS. In this case, the peering link of the AS where servers are located is likely to be the bottleneck. We will consider the case when LAS is implemented only at this bottleneck access link (link 3a-3b) and all other routers use FIFO. The generated Web traffic is expected to show a different impact to the performance of the FTP connection with long RTT under LAS compared to when connections with short RTTs are FTP file transfers. Since each time a client requests a file, it receives it using a new connection that has the highest priority under LAS since it has sent no data to the network. Thus, all arriving connections of Web objects in the system are likely to maintain higher priorities than the FTP connection until they complete. The goal is to examine the performance of a FTP connection with long RTT that must traverse a number of links serving Web objects with short RTTs.

Figure 4.11(a) shows the throughput of the connection with long RTT under LAS and FIFO when the load due to Web traffic is about 0.7. We observe that the throughput of the FTP connection under LAS and FIFO is about the same at all times, and the FTP connection under both




(a) *Throughput* (b) *Loss rate*

Figure 4.11: *Performance of an FTP connection with high propagation delay under LAS and FIFO.*

policies has low throughput. This shows that the throughput of a connection in this topology is more limited by the low data rate of the source, which is a result of its long RTT. Similarly, Figure 4.11(b) shows that packet loss rates of the FTP connection under LAS and FIFO are also the same. These results show that the fact that the FTP connection has a lower priority than Web connections during their transfer times under LAS has no negative impact on the throughput of the FTP connection for moderate load of Web traffic (i.e., load $\rho < 0.9$). Instead, the high RTT value (470ms) of the FTP connection is a dominant factor affecting its throughput. The results shown in this section also support the results in Chapter 3 that LAS in packet networks favors short flows while only negligibly penalizing large flows. We also simulated the topology of Figure 4.10 when LAS is implemented in all links, the results is observed to be similar to results shown in this section when LAS is deployed only at the bottleneck link. This shows that it implementing LAS at only bottleneck links can be enough to reap its benefits.

## 4.5 Conclusion

In this chapter we showed that the bandwidth hogging problem, which is commonly caused by some connections in heterogeneous TCP networks, is alleviated when packets at a link are scheduled according to the LAS scheduling. LAS schedules connections by giving service to a connection that has attained the least amount of service. As a result, LAS prevents any connection from occupying all or a large fraction of network bandwidth regardless of the difference or variation in propagation delays or varying transport protocols of competing connections as is the case for FIFO scheduling.

The simulation results presented in this chapter refute the common belief that bandwidth hogging problem in TCP networks is only due to the inherent TCP mechanisms. In particular, we showed that for a network with a single bottleneck link, LAS maintains the same throughput between connections with long and short RTTs and between competing connections that

use UDP and TCP transport protocols in the same network. When LAS is simulated at the bottleneck of multiple links, the results indicate that connections with short RTTs do not starve connections with long RTTs. The performance of a connection with long RTT in a network with multiple bottlenecks is observed to be the same under LAS and under FIFO when Web transfers with short RTTs and with a realistic flow size distribution are used. This shows that while LAS is well known to favor short connections, it does not penalize long flows when implemented at selected bottlenecks and the main factor that limits the throughput of the long flow in the topology used is its long RTT.

# Chapter 5

# Performance Models and Analysis for *LAS*-based Scheduling Disciplines in a Packet Switched Network

In Chapter 3, we saw that when LAS is used for scheduling packets over the bottleneck link of an Internet path, it can greatly reduce the average flow times for short flows while it only slightly increases the average flow transfer times for the long flows that share the same bottleneck. However, we also saw that at high load, long-lived flows under LAS experience large jitter and low throughput. This chapter proposes and evaluates new differentiated LAS scheduling policies that improve the performance of long and long-lived flows that are identified as "priority" flows.

To evaluate the new policies, we develop analytic models to estimate the average flow transfer time as a function of flow size, and the average packet transmission time as a function of position in the flow, for the single-bottleneck "dumbbell topology" used in many *ns* simulation studies. Models are developed for FCFS scheduling, LAS scheduling, and each of the new differentiated LAS scheduling policies at the bottleneck link. Over a wide range of configurations, the analytic estimates agree very closely with the *ns* estimates. Thus, the analytic models can be used instead of simulation for comparing the policies with respect to mean flow transfer time (as a function of flow size) and mean packet transfer time. Furthermore, an initial discrepancy between the analytic and simulation estimates revealed errors in the parameter values that are often specified in the widely used ns Web workload generator. We develop an improved Web workload specification, which is used to estimate the transfer time and packet jitter for long-lived flows.

Simulation and analytical results for the scheduling policies show that the proposed policies can greatly improve the mean flow transfer time, throughput, and average jitter in packet delivery times for long priority flows while providing performance similar to LAS for ordinary flows.

# 5.1 Introduction

## 5.1.1 Modified LAS Policies: Motivation

The results in Chapter 2 show that the mean response time of large jobs under LAS highly depends on the job size distribution. In particular, for job size distributions with a high coefficient of variation ($C$), the small mean response times seen by the short jobs comes at a cost of only a very small increase in mean response times for the largest jobs. The simulation studies conducted in Chapter 3 were partly motivated by the uncertainty on whether the analytical results obtained for jobs equally or approximately apply to flows in packet networks. This is one of the questions we answer in this chapter.

Similarly, in Chapter 3 we showed that although LAS scheduling on a bottleneck link can greatly improve the average flow transfer time for short flows without appreciably degrading the average total transfer time of long flows. At high load values, we also saw that the long-lived flows can experience significantly lower throughput and higher variance in their individual packets transfer times (jitter) as compared with FCFS scheduling. For download applications, and for streaming applications that buffer data at the destination to mask the variability in packet transfer times, the jitter in packet delivery does not impact application performance. On the other hand, since a typical implementation of a LAS scheduler already maintains a small amount of state for each flow, extensions to the LAS scheduler can provide a differentiated service to flows that require low jitter or high throughput, as well as to other flows that may be willing to pay for improved service.

To illustrate this point, we consider the following two-class policies, in which *ordinary* (class 2) flows have packet priority value equal to the byte sequence number, $x$ (i.e., as under plain LAS), while *priority* (class 1) flows have modified packet priority values that are a function of the sequence number, $P(x)$, which is defined for each policy as follows:

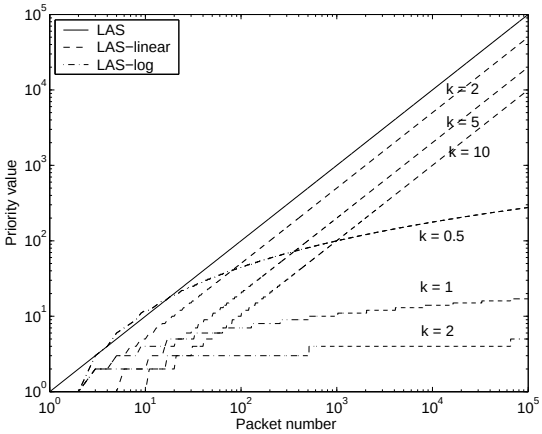



Figure 5.1: *Priority value as a function of attained service.*

- *LAS-fixed(k)*: Packets $x$ in class 1 flows have a constant priority value $P(x) = k$, for a

specified integer $k > 0$.

- *LAS-linear (k)*: Packet $x$ in class 1 has priority value $P(x) = x/k$ for a specified integer $k > 0$.
- *LAS-log (k)*: Packet $x$ in class 1 has priority value $P(x) = (\log_2(x))^{\frac{1}{k}}$, for a specified $k > 0$.

Observe that under plain LAS, $P(x) = x$ for packets of all flows. Note that smaller priority values have higher priority, with priority value 1 representing the *highest* priority. In each policy, the function $P(x)$ determines the relative priority value of each packet in a class 1 flow. If the priority values of the consecutive packets in a class 1 flow increase more slowly than linearly, a class 1 flow will interfere with a lower fraction of load from class 2 flows than under plain LAS. As a result, class 1 flows will have a lower expected transfer times than class 2 flows with the same size. Refinements in the priority assignment function to more than two classes of flows are also possible. We use the above two-class policies to obtain some initial insights into the relative advantages and disadvantages of the different types of priority value functions.

A flow of a given size is said to be *penalized* by a policy if it has higher mean transfer time under the policy than with standard FCFS scheduler. The differentiated policies proposed in this chapter reduce the penalty for differentiated (class 1) flows that require better service without hurting the performance of short class 2 flows. This is because class 2 flows under these differentiated policies still receive their service under LAS scheduling. If the long flows of class 1 represent a small fraction of all flows, the class 2 flows will experience minimal penalty.

### 5.1.2 Related Work

The performance of size-based scheduling policies for Internet routers has been analyzed using ns simulations of flows that share a common bottleneck (e.g., [107, 50, 40]). In other prior work, analytic models have been developed and used to compare the theoretical properties of size-based scheduling policies. Examples of these analyses include SRPT [51, 13], LAS [40, 64], a comparison of the asymptotic performance of scheduling policies [54], and a new policy called *Fair Sojourn Protocol* (FSP) [46]. In most cases, these papers investigate unfairness or optimality properties of the policies. For example, Wierman and Harchol-Balter [103] classify scheduling policies based on their unfairness.

### 5.1.3 Modeling Approach

In this chapter of the thesis, we take a different approach. We derive analytic models that estimate, for each scheduling policy we consider, the average flow transfer time as a function of flow size for TCP flows that share a given bottleneck link. The models also estimate the average packet transfer time as a function of sequence number (i.e., position in the flow). One goal is

to produce the same estimates analytically as are obtained in an ns2 simulation of the dumbbell topology with the scheduling policy, and TCP flows generated by the ns Web workload model. At first this may seem like a daunting task, since the TCP congestion control policy and the router scheduling policy interact in complex ways that may be difficult to capture analytically. For example, previous analytic models of scheduling policies assume that jobs arrive at random in time and a job arrives to the server all at once, whereas the packets belonging to a single flow arrive at a router at disparate points in time that are partially determined by the TCP congestion avoidance protocol.

We make three key observations that enable the development of the analytic models. First, due to the nature of a system bottleneck, at high link utilizations (e.g., 70%) where differences in scheduling policy performance are observable, each flow is likely to have at least one packet queued at the bottleneck link a high fraction of the time. Second, for LAS-based scheduling of two flows that have so far received the same amount of service, if the next several packets from one of the flows arrive after the corresponding packets from the other flow have been served, the late packets will have priority for service after they arrive and thus the flow with late packets will "catch up" to the flow whose packets arrived earlier. Third, for FCFS scheduling of the packets waiting for the bottleneck link, moderate to large flows that have similar propagation delays and share the common bottleneck will have their packets transmitted in an interleaved fashion; the "quantum" for the interleaving varies due to the dynamic TCP window size and unpredictable packet arrival times, but is small compared to most flow sizes. Thus, we conjecture that the flow transfer time for each router scheduling policy can be analyzed assuming each flow approximately always has at least one packet in the queue for the bottleneck link. Using this conjecture, we show how previous models of scheduling policies, with *customizations* to model connection establishment and the initial TCP slow-start phase, can be used to obtain the average flow transfer time as a function of flow size and average packet transmission time as a function of sequence number for FCFS as well as LAS routers. We also use the conjecture to derive *new models* of the average flow and packet transfer times for each of the new differentiated service policies at the bottleneck link. We validate the conjecture by implementing LAS and the differentiated service policies in the ns2 simulator and comparing the analytic results against the simulation results for the "dumbbell" bottleneck topology used in many previous ns simulation studies.

A key result of this chapter is that the analytic results for the widely implemented FCFS policy, LAS policy, and the proposed differentiated service policies, are in excellent agreement with the ns simulation estimates. This result could motivate development of analytic models for many other types of design questions that have commonly been evaluated using ns simulations. Key advantages of the analytic models include very quick solution time, accurate modeling of the heavy tailed flow size distributions, and key insight into the system features that have a direct impact on the computed performance. Furthermore, a detailed analysis of initial discrepancies between the analytic results and simulation results, for FCFS scheduling, revealed an error in the ns2 workload parameters often specified for a popular ns workload generator [21]. The error in these workload parameters was not obvious from the simulation results alone, because simulation estimates of expected flow transfer times appeared plausible for the flow sizes with a high variability property that are found in the Internet. However, the analytic models predicted

lower average transfer time as a function of flow size for the same distribution of the flow sizes. As discussed in section 5.4, a detailed examination of the simulation flow times revealed that the ns2 workload parameter values were unrealistic. This illustrates a common experience in analytic modeling, which is that analytic models can be very useful in validating and correcting the simulation.

This chapter uses the validated analytical models to compare the performance of the LAS-based differentiated service policies, the single-class or plain LAS policy, and the FCFS scheduling policy. The validated simulator is also used to evaluate jitter, in more detail than possible analytically, for the differentiated service policies and for the LAS policy. The results show that, compared to FCFS policy, the proposed policies can simultaneously achieve low mean transfer time for short flows, a mean transfer time for the ordinary long flows similar to LAS, and improved jitter in the packet transfer times for long priority flows.

We first discuss the models in Sections 5.2 and 5.3. Then, we validate the new policies in Section 5.4 and compare their performances in terms of mean transfer time, loss, jitter, and unfairness in Section 5.5. Section 5.6 summarizes this chapter.

## 5.2 FCFS and LAS Scheduling

In this section we provide two analytic models for the average flow transfer time as a function of flow size, $T(x)$, for flows that (a) share a common bottleneck link and (b) experience negligible contention at other points in their respective transmission paths. One of the two models provides results for the widely deployed FCFS scheduling of packets on the bottleneck link; the other provides results for the single-class LAS scheduling policy. The goal is to estimate average transfer time for such flows as accurately as estimated by ns2 simulations for the simple dumbbell topology with a given flow arrival rate, flow size distribution, bottleneck link speed, and bottleneck link scheduling policy.

### 5.2.1 Notation and Assumptions

We assume that (a) flow sizes are expressed in units of packets of a specified size, and (b) the unit of time for all measures that require a time unit, is the time it takes to transmit one packet on the bottleneck link. The overall distribution of flow size is given by $F(x)$, and the complement of the distribution is given by $F^c(x)$. Key measures for modeling LAS policies include (1) the n$th$ moments of the flow size distribution, $\overline{x^n}$, (2) the link load, $\rho = \lambda\overline{x}$, (3) the $n$th moments of the flow size distribution in which the flows are truncated at a given value $x$, which are given by

$$\overline{x_x^n} = \int_0^x y^n dF(y) + x^n F^c(x), \tag{5.1}$$

and (4) the load due to the truncated flows,

$$\rho_x = \lambda\overline{x_x}. \tag{5.2}$$

For each scheduling policy, we derive $T(x)$, the *mean end-to-end flow transfer time* as a function of flow length $x$. For a flow of a given size $x$, the average time to transfer the first $j$ packets of the flow is equal to $T(j)$. Thus, the average time between when packets $j$ and $j+1$ complete transmission is equal to $T(j+1) - T(j)$.

An important goal is to create the simplest models that contain the essential details for producing the same results as a (correct) detailed ns2 simulation for key workloads and system configurations of interest. One important advantage of such models is that they clarify which system details have a principal impact on observed policy performance. The analytic models can be used in place of simulations to quickly compare policy performance with respect to $T(x)$ for a variety of workloads and system configurations, such as varying relative load of high priority and low priority flows, and for a range of bottleneck link speeds. Simulation can then be used to evaluate the most promising policies using more detailed performance measures and for system configurations that violate the assumptions in the analytic model. To these ends, we make the simplifying assumptions outlined below.

The workloads of interest involve TCP flows, with flow size distributions that have a high variability property, that share a common bottleneck link that has utilization in the range of 70% or higher. This is the range of link utilizations for which the choice of link scheduling policy has a discernable impact on system performance. We assume that *most* flows through the bottleneck have modest disparity in their respective round trip propagation delay. We also assume, unless otherwise stated, that each flow has a window of packets in flight (i.e. at least one packet in the bottleneck queue) most of the time.

We derive the analytic models for the case of low packet loss rates (i.e., the link buffer is large enough to achieve a loss rate under, for example, 2%). Simulations can be used to study the impact of configurations with higher loss rates on the most promising policies that are identified in the low-loss analytic comparisons. Alternatively, the analytic models can be extended to estimate and account for packet loss; we defer such extensions to future work.

Below we derive the mean transfer time, assuming zero propagation delay between the flow endpoints and the bottleneck, for a flow of size *x*. We then add two times the average round trip propagation delay to estimate the average delay for connection establishment, which is non-negligible for short flows. Finally, for each of the first few windows we add another round trip propagation delay to model the initial TCP slow start. This yields an approximate method that accounts for the impact of TCP algorithms in the model. The exact expression is computed in Section 5.3.5. We assume that the propagation delay for other packets is fully overlapped with the bottleneck queueing delay and service time of at least one other packet in the same flow.

For simplicity in the presentation of the models, we assume that all packets have same size (e.g., one kilobyte). This is also a common assumption in ns2 simulations. The models can easily be extended to have a distribution of average packet size for each flow, where the distribution could depend on the class of the flow as well as the flow size. However, such extensions are beyond the scope of this thesis.

### 5.2.2 FCFS Model

If the packets are scheduled on the link in FCFS order, transmissions of packets from different flows will be interleaved since the packets in each flow arrive at variable times. The "quantum" for interleaving will typically be non-uniform because a variable number of packets in a particular flow will arrive between two consecutive packets in another active flow. However, we assume roughly similar flow rates for moderate to large flows that each have moderate propagation delay, share the common bottleneck, and experience negligible contention in the rest of their transmission path. Thus, we assume that the scheduling of the flows on the link is approximately round-robin with an average quantum size that is small for moderate to large flows. Ignoring the delays for connection establishment and retransmitting lost packets, we estimate the average transfer time for a flow of size $x$ using the processor-sharing approximation for round-robing scheduling [64]:

$$T(x)_{PS} = \frac{x}{1-\rho} \tag{5.3}$$

This approximation is likely to underestimate the mean transfer time for short flows, since the model does not include the impact of the variable and non-negligible quantum size on the average delay seen by short flows. The impact of the approximation will be evaluated in the model validations.

### 5.2.3 LAS Model

Assuming each flow has at least one packet in the queue for the bottleneck link most of the time, the mean transfer time for a flow of size $x$ under the LAS policy, expressed in units of the time to transmit a packet on the bottleneck link, can be modeled by the mean total time to serve a job of size $x$ at a LAS server, which is [64]:

$$T(x)_{LAS} = \frac{W_o(x) + x}{(1-\rho_x)}, \tag{5.4}$$

where $W_o(x)$ is the average backlog of packets with sequence numbers less than or equal to $x$ at a random point in time,

$$W_o(x) = \frac{\lambda \overline{x_x{}^2}}{2(1-\rho_x)}.$$

Although each flow does not actually have a packet in the queue at every point in time, as would be required for the above model to be exact, a packet that arrives a little later than the time at which it would receive service in the exact model will be served earlier than all packets with higher sequence numbers, and thus will cause one unit of interference in the total transfer time of the other flows that have so far received more service. Thus, we anticipate that this model of the LAS router may be more accurate than the model of the FCFS router.

## 5.3 Differentiated LAS Scheduling

Under the single-class LAS scheduling policy, packets later in a flow have lower priority and thus longer expected transmission delays. In this section we consider three alternative two-class differentiated service scheduling policies that are each simple extensions of the single-class LAS policy. The goals of these differentiated service disciplines are to reduce the expected total transfer time of (moderate to long) class 1 flows and to reduce the differential between the average transfer time for earlier and later packets in the class 1 flows, without appreciably increasing the expected transfer time of short class 2 flows. The policies are LAS-fixed(k), LAS-linear (k) and LAS-log (k), which have packet priorities for class 1 that are *fixed*, *linear* and *logarithmic* in the packet sequence number, respectively. Packets in class 2 flows have priority equal to their sequence number, as under the single-class LAS policy. Scheduling of the packets on the outgoing link is otherwise the same as in the single-class LAS policy. Thus, packets in a class 1 flow have on average higher priority than the packets in a class 2 flow of the same size, but the long class 1 flows do not significantly interfere with the large number of very short class 2 flows.

Flows in class 1 will be called the *priority* flows and will be denoted by subscript or superscript $p$, while the flows in class 2 will be called the *ordinary* flows and denoted by subscript or superscript $r$. In these differentiated service policies, a fraction $q$ of the flows are class 1 flows. Each class of flows has a possibly different distribution of flow sizes, denoted by $F_i(x)$ for class $i$. The analytical models of the two-class LAS policies presented in this section estimate average flow transfer time as a function of flow size. The models adopt the technique used to evaluate the LAS policy, which is to view each flow as a job with service requirement $x$ that arrives to the bottleneck queue at a random point in time.

### 5.3.1 LAS-fixed(k) Model

In the LAS-fixed(k) policy, each packet in a class 1 (priority) flow has priority equal to the $k$th packet in a class 2 flow, where $k$ is a parameter of the policy. The motivation for this policy is to provide the same average delay in the bottleneck link queue for each packet in a class 1 flow. Clearly flows that are shorter than $2k$ will receive lower average priority as class 1 flows than as class 2 flows. Thus, we anticipate that flows shorter than $2k$ will typically select class 2.

We first derive the average flow transfer time for the first $x < k$ packets in an ordinary class 2 flow, $T_r(x)$. These packets do not see any interference from class 1 flows. Thus, the average flow transfer time is given by the time in a single-class LAS system that serves only class 2 flows that are shorter than size $k$. Expressed in units of the time to transmit a packet on the bottleneck link,

$$T_{r,fixed}(x) = \frac{W_{x,r} + x}{1 - \rho_{x,r}}, \quad x < k, \tag{5.5}$$

where $\rho_{x,r}$ is the load due to class 2 flows with service requirements less than $x$, and $W_{x,r}$ is the

corresponding average number of backlogged class 2 packets. That is,

$$W_{x,r} = \frac{\lambda_r \overline{x_{x,r}^2}}{2(1-\rho_{x,r})} \tag{5.6}$$

with $x_{x,i}^n$ defined similarly to Equation (5.1):

$$\overline{x_{x,i}^n} \triangleq \int_0^x y^n dF_i(y) + x^n F_i^c(x). \tag{5.7}$$

For a priority class 1 flow, let $R_p(j), j > 1$, denote the average time between when packets $j-1$ and $j$ complete service at the bottleneck link. We can express $T_{p,fixed}(j)$, the average total flow transfer time for the first $j$ packets in a class 1 flow, as follows:

$$T_{p,fixed}(j) = T_{p,fixed}(j-1) + R_p(j). \tag{5.8}$$

Similarly, the average transfer time for an ordinary class 2 flow of size $k$ is given by

$$T_{r,fixed}(k) = T_{r,fixed}(k-1) + R_r(k). \tag{5.9}$$

To derive expressions for $T_{p,fixed}(1)$, $R_p(j)$, and $R_r(k)$, we need the following two quantities: $\overline{r_p}$, the average size of the priority flows not including the first packet in the flow:

$$\overline{r_p} = \overline{x_p} - F_p(1), \tag{5.10}$$

and $\overline{R_p}$, the average service time of class 1 packets that have sequence number greater than 1:

$$\overline{R_p} = \frac{\sum_{j=1}^{\infty} R_p(j+1)[1-F_p(j)]}{\overline{x_p} - 1}. \tag{5.11}$$

The average time for the first packet in a class 1 flow to complete service at the bottleneck link, $T_{p,fixed}(1)$, expressed in units of the time to transmit a (fixed size) packet on the bottleneck link and assuming the packet arrives at a random point in time, is given by

$$\begin{aligned} T_{p,fixed}(1) &= W_{k-1,r} + \lambda_r F_r^c(k) R_r(k) + \lambda_p R_p(1) + \lambda_p \overline{r_p}\,\overline{R_p} + \lambda_r [T_{p,fixed}(1) - 1]\overline{x_{k-1,r}} + 1, \\ &= \frac{W_{k-1,r} + \lambda_r F_r^c(k) R_r(k) + \lambda_p R_p(1) + \lambda_p \overline{r_p}\,\overline{R_p} - \lambda_r \overline{x_{k-1,r}} + 1}{(1 - \lambda_r \overline{x_{k-1,r}})}. \end{aligned} \tag{5.12}$$

The first two terms in the above sum are the the average number of backlogged class 2 packets that have sequence number less than $k$ and equal to $k$, respectively. The third and fourth terms are the average number of backlogged class 1 packets that have sequence number equal to 1 and sequence number greater than 1, respectively. (Recall that all of these class 1 packets have equal priority $k$.) The fifth term is the average number of class 2 packets with sequence number less than $k$ that arrive while the first packet in a class 1 flow is waiting in the queue. Finally, the last term represents the transmission time of the first packet in the class 1 flow.

Let the time between when packets $j-1$ and $j$ in a class 1 flow complete service at the bottleneck link be the time during which packet $j$ is the next packet in the flow to be transmitted on the link. This time has average value $R_p(j), j > 1$, given in units of a packet transmission time on the link, as follows:

$$\begin{aligned} R_p(j) &= \lambda_p \overline{x}_p R_p(j-1) + \lambda_r F_r^c(k) R_p(j-1) + \lambda_r [R_p(j)-1]\overline{x_{k,r}} + 1, \\ &= \frac{\lambda_p \overline{x}_p R_p(j-1) + \lambda_r F_r^c(k) R_p(j-1) - \lambda_r \overline{x_{k,r}} + 1}{(1-\lambda_r \overline{x_{k,r}})}, \quad j > 1. \end{aligned} \tag{5.13}$$

The first two terms in the above sum are the average number of class 1 priority packets, and the average number of ordinary packets with sequence number $x = k$, respectively, that arrive after packet $j-2$ is transmitted on the bottleneck link and before the transmission of packet $j-1$ is complete. These packets will be transmitted on the link after packet $j-1$ is transmitted and before packet $j$ is transmitted. The third term in the sum is the average number of class 2 packets with sequence number less than $k$ that arrive while packet $j$ in a class 1 flow is the next packet in the flow to be transmitted on the bottleneck link. Finally, the fourth term represents the transmission time of packet $j$ in the class 1 flow.

To derive the average time between when packet $k-1$ and packet $k$ in a class 2 flow are transmitted on the bottleneck link, $R_r(k)$, we assume the first packet of the class 2 flow arrives at a random point in time. Note that the priority $k$ packets from class 1 and class 2 that are in the queue when the first packet in the class 2 flow arrives are still in the queue when the $k-1$st packet completes its transmission, assuming the flow continuously has at least one packet in the queue. The average backlog of such packets, estimated by the first three terms in the sum below, is summed together with the priority $k$ packets that arrive from other flows during the time to transmit the first $k-1$ packets in the class 2 flow, which are the next two terms in the sum below, to obtain the total number of priority $k$ packets from other flows that must be transmitted after the class 2 packet with sequence number $k-1$, and before the class 2 packet with sequence number $k$. Thus,

$$\begin{aligned} R_r(k) &= \lambda_p R_p(1) + \lambda_p \overline{r_p}\,\overline{R_p} + \lambda_r F_r^c(k) R_r(k) + \\ &\quad [\lambda_p \overline{x_p} + \lambda_r F_r^c(k)] T_{r,fixed}(k-1) + \lambda_r R_r(k) \overline{x_{k,r}} + 1, \\ &= \frac{\lambda_p R_p(1) + \lambda_p \overline{r_p}\,\overline{R_p} + [\lambda_p \overline{x_p} + \lambda_r F_r^c(k)] T_{r,fixed}(k-1) + 1}{(1 - \lambda_r F_r^c(k) - \lambda_r \overline{x_{k,r}})}. \end{aligned} \tag{5.14}$$

Note that the last two terms in the above sum are the average number of class 2 packets with sequence number less than $k$ that arrive while packet $k$ in a class 2 flow is the next packet in the flow to be transmitted on the bottleneck link, and the time to transmit packet $k$ in the class 2 flow, respectively. Finally, we derive the average transfer time for class 2 flows with size $x > k$,

$$\begin{aligned} T_{r,fixed}(x) &= \widehat{W_x} + \lambda_r T_{r,fixed}(x) \overline{x_{x,r}} + \lambda_p T_{r,fixed}(x) \overline{x}_p + x, \\ &= \frac{\widehat{W_x} + x}{(1 - \lambda_r \overline{x_{x,r}} - \lambda_p \overline{x}_p)}, \quad x > k. \end{aligned} \tag{5.15}$$

where $\widehat{W_x}$ is the average backlog, at a random point in time, of all packets from class 1 and class 2 flows that have priority less than $x$, given by

$$\widehat{W_x} = \frac{\lambda[q\overline{x_p^2} + (1-q)\overline{x_{x,r}^2}]}{2(1-\lambda[q\overline{x_p} + (1-q)\overline{x_{x,r}}])}. \tag{5.16}$$

### 5.3.2 LAS-linear(k) Model

The LAS-fixed(k) policy has the key property that each packet in a priority class 1 flow has the same expected delay in the bottleneck queue. However, this policy also has the drawback that *all* packets in the priority flows have higher priority than ordinary class 2 packets that have sequence number greater than $k$. Thus, depending on the fraction of flows that are identified as priority class 1 flows and the distribution of class 1 flow sizes, class 2 flows that are longer than $k$ may have high expected total transfer time. The LAS-fixed(k) model can be used to explore the quantitative impact of the workload parameters on the average flow and packet transfer times. In this section we develop an alternative differentiated service policy.

LAS and LAS-fixed(k) represent two extremes in the growth rate of the priority function, $P(x)$, for class 1 packets, namely $P(x) = x$, and $P(x) = k$. Here we consider an intermediate policy, namely the LAS-linear(k) policy in which $P(x) = x/k$. Figure 5.1 illustrates this priority function, which grows more slowly for larger values of $k$.

To compute the mean flow transfer times for class 1 priority flows and class 2 ordinary flows, we first need to compute the moments of the flow size distribution for flows that interfere with a class 1 flow of size $x$. An arriving priority flow that requires $x$ units of service in a LAS-linear scheduler will be delayed by other priority flows that have received less than $x$ units of service and by ordinary flows that have received less than $x/k$ units of service. The moments of the flow size distribution for priority flows truncated at $x$ and ordinary flows truncated at $x/k$, are given by

$$\overline{x_{x,linear(k)}^n} = q\overline{x_{x,p}^n} + (1-q)\overline{x_{x/k,r}^n}, \tag{5.17}$$

where $\overline{x_{x,i}^n}, i = p, r$ is defined in equations (5.7). The bottleneck link load for these flows, $\rho_{x,linear(k)} = \lambda\overline{x_{x,linear(k)}}$.

To compute the mean transfer time for a priority class 1 flow of size $x$, $T_{p,linear}(x)$, we note that the flow size defines which packets from other flows will delay the flow, and otherwise the packet queue is a single-class LAS queue. Thus,

$$T_{p,linear}(x) = \frac{W_{x,linear(k)} + x}{(1-\rho_{x,linear(k)})}, \tag{5.18}$$

where

$$W_{x,linear(k)} = \frac{\lambda\overline{x_{x,linear(k)}^2}}{2(1-\rho_{x,linear(k)})} \tag{5.19}$$

An ordinary class 2 flow of size $x$ will have a sequence of priorities for each of its packets that corresponds with the sequence of packet priorities in a class 1 flow of size $kx$. Thus,

$$T_{r,linear}(x) = \frac{W_{kx,linear(k)} + x}{(1 - \rho_{kx,linear(k)})}. \tag{5.20}$$

### 5.3.3 LAS-log Model

The LAS-linear(k) policy provides a slower decrease in consecutive packet priority for class 1 flows than the single-class LAS policy, but the decrease still has the same form, namely linear. Initial results from the model for LAS-linear(k) showed that later packets in the long class 1 flows have high expected delay in the bottleneck queue for workloads of interest. This illustrates the use of the models to quickly assess the properties of a policy. We thus propose the LAS-log(k) policy, which has a much slower growth of the packet priority function, $P(x)$, as illustrated in Figure 5.1.

A high priority flow that requires $x$ service units under *LAS-log* is preempted by all high priority flows that have received less than $x$ units of service and all ordinary class 2 flows that have received less than $(log_2 x)^{(1/k)}$ units of service. The moments of the distribution of flow sizes for priority and ordinary flows truncated at $x$ and $(log_2 x)^{(1/k)}$, respectively, are given by

$$\overline{x^n_{x,log(k)}} = q\overline{x^n_{x,p}} + (1-q)\overline{x^n_{(log_2 x)^{1/k},r}} \tag{5.21}$$

We define the load that these flows place on the bottleneck link, $\rho_{x,log} = \lambda\overline{x_{x,log}}$.

Since this policy is similar to LAS-linear(k), except for the priority function, the equations for $T_{p,log}$ and $W_{x,log(k)}$ are the same as equations (5.18) and (5.19) for the LAS-linear(k) policy, with subscripts "linear" replaced by "log". The equation for $T_{r,log}$ is the same as equation (5.20) for the LAS-linear(k) policy, with the same subscript replacement and with subscript $kx$ replaced by $2^{x^k}$.

### 5.3.4 LAS-fixed(k) Model: UDP Priority Flows

The above models were developed assuming that both the priority class 1 flows and the ordinary class 2 flows are TCP flows. A key feature of such flows is that most of the time there is a window of packets in flight from the source. Another key workload of interest is one in which the priority flows are long-lived UDP media streams, in which packet sending is paced so the client can display the media stream as it arrives. Media streaming can also be performed using TCP, but scalable streaming protocols benefit from slower pacing under UDP, since more clients can share portions of the stream if the data is not transmitted too quickly.

In this section, we consider the case that each class 1 priority flow is a UDP stream that sends a packet every $r$ units of time, where the unit of time is the time to transfer one packet on the bottleneck link. In this case, $T_{p,fixed}(x) = x/r$.

We illustrate that the models can be modified to compute the average transfer time as a function of flow size for ordinary class 2 flows by showing how this is done for the LAS-fixed(k) policy. Specifically, the first two terms in equation (5.14) estimate the average number of priority packets in the queue at a random point in time. These terms are replaced by $\frac{1}{r}R_{p,UDP}$, where $R_{p,UDP}$ is the mean transmission time for a class 1 UDP packet. Thus,

$$\begin{aligned} R_r(k) &= \frac{1}{r}R_{p,UDP}(k) + \lambda_r F_r^c(k)R_r(k) + [\frac{1}{r} + \lambda_r F_r^c(k)]T_{r,fixed}(k-1) + \lambda_r R_r(k)\overline{x_{k,r}} + 1, \\ &= \frac{\frac{1}{r}R_{p,UDP}(k) + [\frac{1}{r} + \lambda_r F_r^c(k)]T_{r,fixed}(k-1) + 1}{(1 - \lambda_r F_r^c(k) - \lambda_r \overline{x_{k,r}})}. \end{aligned} \tag{5.22}$$

To compute $R_{p,UDP}$, we estimate the average number of class 2 and class 1 packets in the queue when the UDP packet arrives (first three terms in the sum below), and the average number of ordinary class 2 packets with sequence number less than $k$ that arrive while the UDP packet is in the queue (the fourth term below), to obtain

$$\begin{aligned} R_{p,\text{UDP}} &= W_{k-1,r} + \lambda_r F_r^c(k)R_r(k) + \frac{1}{r}R_{p,UDP} + \lambda_r(R_{p,UDP} - 1)\overline{x_{k-1,r}} + 1, \\ &= \frac{W_{k-1,r} + \lambda_r F_r^c(k)R_r(k) - \lambda_r\overline{x_{k-1,r}} + 1}{(1 - \frac{1}{r} - \lambda_r\overline{x_{k-1,r}})}. \end{aligned} \tag{5.23}$$

As in the LAS-fixed(k) model for TCP class 1 flows, $T_{r,fixed}(k+1) = T_{r,fixed}(k) + R_r(k)$ and $T_{r,fixed}(x)$ for $x > k$ is as given in equation (5.15).

### 5.3.5 Accounting for the TCP algorithm in the models

The expressions for the expected transfer times of flows $T(x)$ for the models derived in Sections 5.2 and 5.3 have units of the time to transfer a single packet across bottleneck link. Let $\overline{S}$ be the average packet transmission time across a bottleneck link in seconds. To obtain the expression for flow transfer time in seconds we multiply $T(x)$ by $\overline{S}$, i.e., $T(x)\overline{S}$. In addition, the total transfer time of a TCP flow must also account for the time due to connection setup phase or a 3-way handshake and the window based congestion control that limits the amount of bytes sent during one RTT. We denote this time as $T_{idle}$. To derive the expression of $T_{idle}$ we consider the model for a Web server sending packets to a client taking TCP's dynamic congestion window. First note that in TCP, one round trip time ($RTT$) is required to initiate the TCP connection. After one $RTT$, the client sends a request for the object. After the total of two RTTs, the client begins to receive data from the server.

In a TCP congestion control window, the server starts with a congestion window of one segment and sends one packet to the client. When the server receives an acknowledgement for the packet, it increases its congestion window to two segments and sends two packets to the clients. As it receives the acknowledgement for the two packets, it increases the congestion window to four segments and sends four packets to the client. The process continues with the congestion window doubling in every one $RTT$. Having transmitted a window's worth of data, the server may stop transmitting while it waits for an acknowledgment from the client.

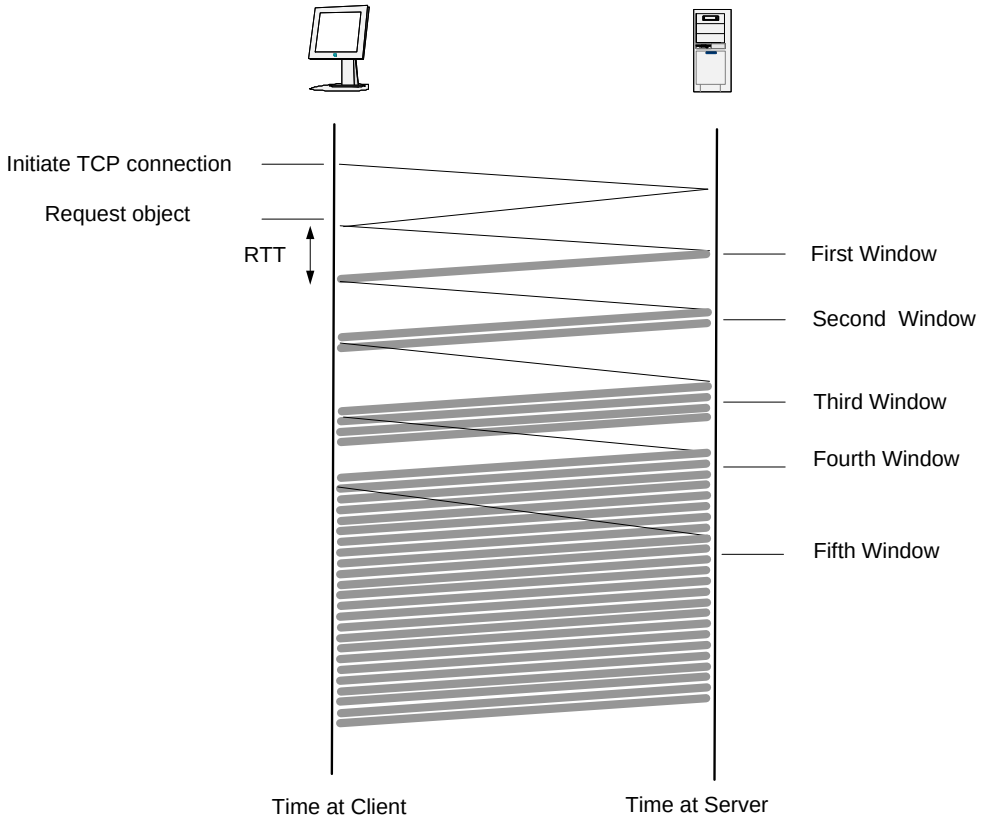



Figure 5.2: *TCP delay during slow start phase.*

Referring to Figure 5.2, the server stops after transmitting the first, second, and third windows but not after transmitting the fourth window. $T_{idle}$ affects the transfer times of flows, particularly short flows. In this section, we want to compute this time that must be taken into account in the analytical models derived in this chapter.

The amount of time the server has stopped after transmitting the $k$th window is derived in [45, 66] as follows. From the time the server begins to transmit the first $k$th window until the time when the server receives an acknowledgement for the first packet in the window is $RTT + L/R$, where $R$ is the link speed and $L$ is the packet size. The transmission time of the $k$th window is $(L/R)2^{k-1}$. Thus, the time that the server stops when sending the $k$th window is

$$L/R + RTT - 2^{k-1}L/R$$

Let $K$ be the number of windows required to transmit the object of size $O$. The server can stop transmitting after transmitting each of the windows $1, 2, \cdots, K-1$; the object is completely sent when the server sends the $K$th window. We can now calculate the total time that the server stops sending data. This time has two components: $2 \times RTT$ for setting up the TCP connection and requesting the file; and the sum of times the server stops after sending windows of data. Thus,

$$T_{idle} = 2RTT + \sum_{k=1}^{K-1}\{L/R + RTT - 2^{k-1}L/R\}$$

Let $Q$ be the number of times server would stop after sending a window of data if the object has an infinite number of packets. $Q$ can then be expresses as:

$$Q = \lfloor log_2(1 + \frac{RTT}{L/R}) \rfloor + 1$$

The actual number of times that the server stops is $J = min\{Q, K\text{-}1\}$. Combining the equations above gives the closed form expression of the stop time:

$$T_{idle} = 2RTT + J[RTT + \frac{L}{R}] - (2^J - 1)\frac{L}{R}$$

The complete expression of $T(x)$ for the analytic models is thus given as:

$$T(x) = T_{i,l}(x)\overline{S} + T_{idle} \tag{5.24}$$

where $T_i^l(x)$ is the mean transfer time (in units of $\overline{S}$) of class $i \in \{r, p\}$ flow of size $x$ under policy $l \in \{linear, fixed, log\}$ and under LAS and PS. The time that the server stops has more impact when sending packets during slow start (SS) phase, since TCP congestion windows at SS phase have few packets. The Equation (5.24) assumes that a flow does not experience packet losses and the source rate is only limited by the speed of a bottleneck link.

## 5.4 Model Validations

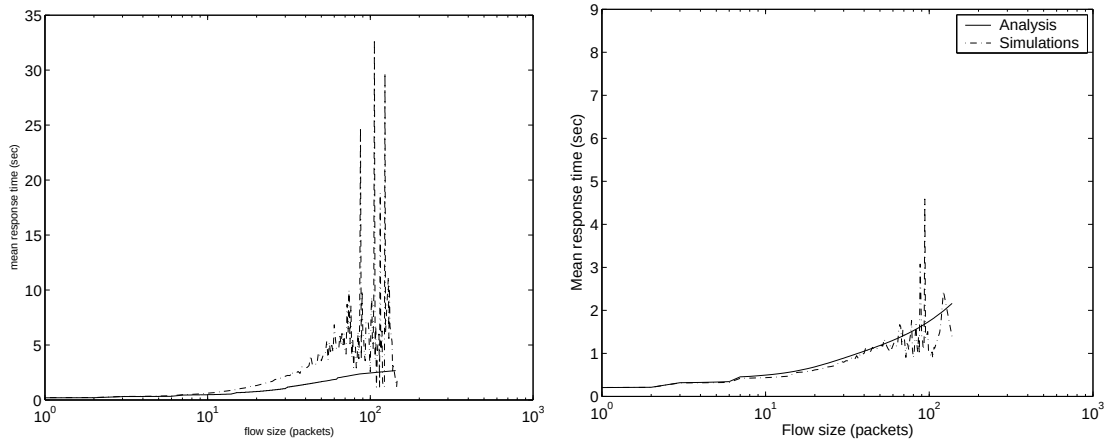
### 5.4.1 Methodology

To validate the analytical models we implement the scheduling policies for network links in the network simulator ns2 [1]. In the validation experiments, we use the dumbbell topology shown in Figure 3.5 in Chapter 3. We also consider the Web model proposed by Feldmann et al. [38] to generate a traffic profile with specified distributions. The particular combination of workload parameters used to generate Web traffic is shown in Table 3.1 in Chapter 3. This set of parameters results in a total load on the bottleneck link, $\rho$, equal to 0.73. Recall that the transfer of each object initiates a new TCP flow. For this reason, object size and flow size are the same, and we use both terms interchangeably. The *object sizes* are distributed by the Pareto distribution of second kind that has a high variability property.

Using the simulator, we can obtain performance measures such as the mean transfer time as a function of flow size, mean packet transfer time as a function of position in the flow, packet loss rate, and jitter in the inter-packet arrival times at the client. Simulations are run for 6000 seconds and the data is collected after a warm up period of 2000 seconds. We compare the results obtained from simulation with the analytical results for mean flow transfer time as a function of flow size. If the results are in agreement, we consider the analytical model (and the simulator) validated. The validated models are then used to compare the performance of the policies.

### 5.4.2 A Note on the Workload Model

The ns2 Web workload generator we used to validate our analytic models has been used extensively. The validation of our flow level models using this workload generator revealed that the choice of the workload model input parameters must be done carefully. Otherwise, while achieving a target average load such as 0.7, the generated workload may include some unrealistic long overload periods. We discovered this as a result of an initial discrepancy between analytic and simulation results for the FCFS link scheduling policy. The discrepancy was originally detected for heavy tailed flow sizes, in which both the simulation and analytic results appeared plausible. However, results for a hypothetical exponential flow size distribution with the same average flow size (i.e., 12 packets) immediately indicated that the simulation results for average flow transfer times were higher than could be expected in practice (e.g., *average* flow transfer time of more than 25 sec for flows with 100 packets see Figure 5.3(a)).




(a) *Workload 1: Inter-session* $Exp(15)$*, Inter-Page* $(Exp(10))$*, InterObject* $Exp(0.01)$*, PageSize* $Exp(3objects)$*, Session Size* $Exp(100pages)$*, object size* $Exp(12)$

(b) *Workload used corresponds to the one of Table 3.1 with object size* $Exp(12)$

Figure 5.3: *Validation of LAS for exponentially distributed flows of mean size 12 packets* $(Exp(12))$*.*

A closer examination of the number of timeouts and other measures in the ns2 simulations showed that the workload generator for the parameters given in the Figure 5.3(a) was generating an unrealistic number of parallel flows during substantial periods in the simulation. The reason behind this is that some parameters of the workload model (session interarrival time, number of pages per session, number of objects per page and average object size) control the average (macroscopic) load on the bottleneck link while other parameters (inter-page and inter-object times) affect the microscopic load of the model. If the latter are set to too large (e.g., inter-page time of 10 seconds, as has been used in the previous literature), sessions overlap unrealistically, generating transient, yet long, overload periods. The use of a better combination of parameter values for the workload generator, as given in Table 3.1, gave results that agree with the analytic model, shown below.

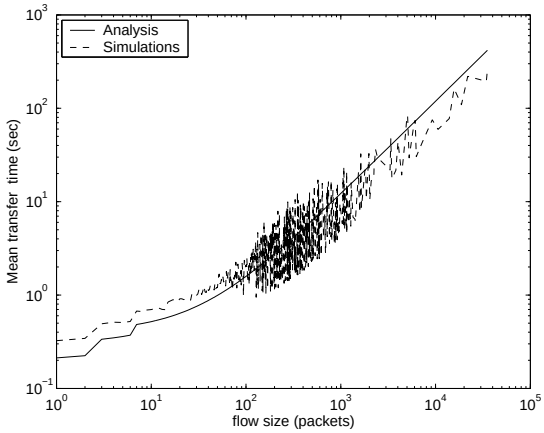



Figure 5.4: *Validation of FCFS model,* $\rho = 0.73$*, loss rate = 1.2%.*

### 5.4.3 FCFS Model Validation

Figure 5.4 presents the mean flow transfer time versus flow size for the analytical model of the dumbbell topology with FCFS link scheduling and for the corresponding detailed ns simulation. The results demonstrate that the analytic model has very good overall agreement with the ns2 simulation estimates. The mean flow transfer time is slightly underestimated for flows shorter than 100 packets (as anticipated in Section 5.2.2). The average transfer time is perhaps also slightly overestimated for flows larger than 1,000 packets. One reason may be that, due to practical bounds on the simulation run length, the simulated flow size distribution does not precisely match the Pareto distribution [31]. To partially mitigate the discrepancy, the analytic model uses a bounded Pareto distribution that approximately fits the measured simulated distribution. The overall agreement between the analytic and simulation estimates is excellent, and the small discrepancy for short flows that is due to the processor-sharing approximation can easily be taken into account when comparing analytic results for FCFS against results for other policies.

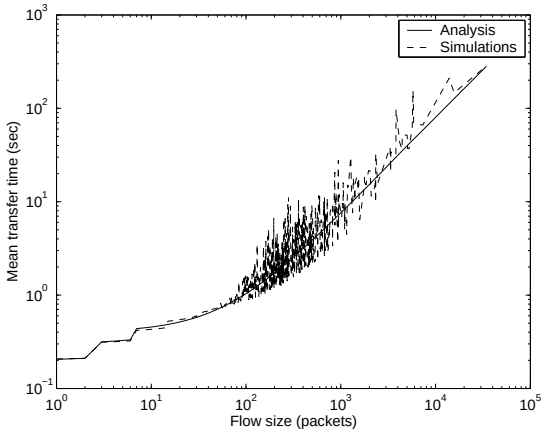



Figure 5.5: *Validation of LAS model,* $\rho = 0.73$*, loss rate = 1.2%.*

### 5.4.4 LAS Model Validation

To validate the analytical models for each scheduling policy, we varied the average intersession time, buffer size, and/or maximum TCP window size, to achieve different link loads and loss rates. In each figure we plot the mean transfer time versus flow size obtained from the analytic model as well as from the simulation of the TCP flows in the Web workload. Figure 5.5 shows representative validation results for configurations with low loss rate (i.e., 1.2%). We observe a near perfect agreement between the analytic model and the simulation.

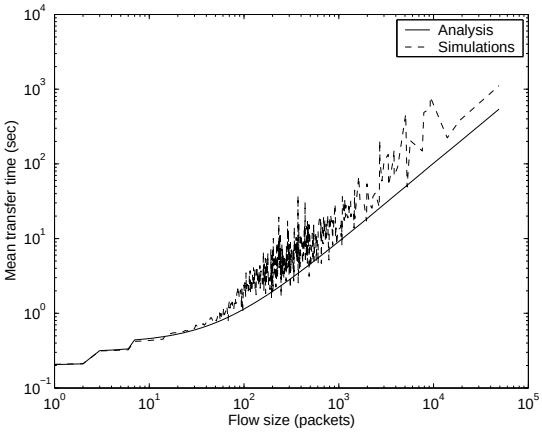
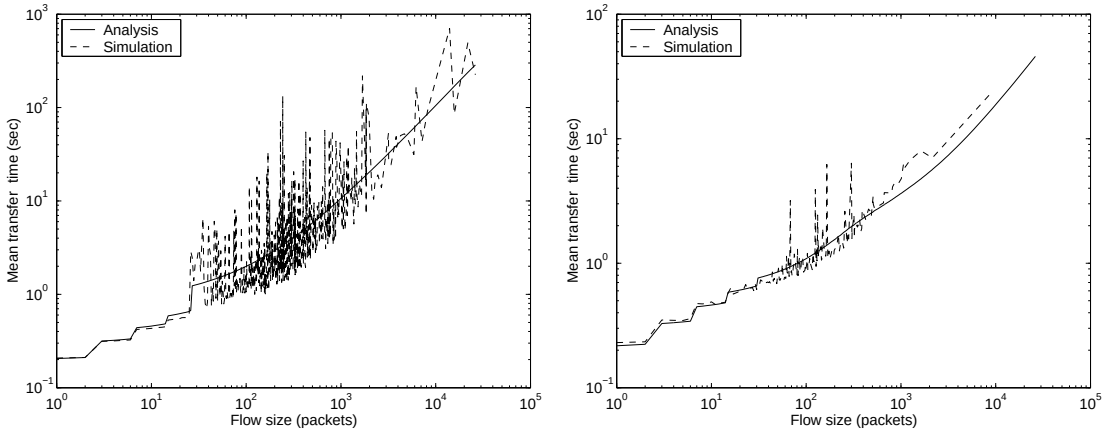



Figure 5.6: *Validation of LAS model,* $\rho = 0.73$ *and loss rate* = $3.6\%$.

It is well known that the throughput of the commonly deployed TCP protocols (e.g., TCP Reno) is inversely proportional to the square root of the loss rate [79]. Thus, an increase in loss rate will reduce the throughput and increase the average flow transfer time. The analytic models presented in Sections 5.2 and 5.3 do not consider the impact of packet loss, and thus will underestimate the mean flow transfer time when packet loss is significantly higher than 1%. This is illustrated in Figure 5.6, which shows the mean transfer time results in a system with LAS scheduling and loss rate of 3.6%. There is good agreement between the analytic and simulation results for small flow sizes, since packets in the short flows have a high priority under LAS and are rarely lost. Most of the loss is experienced by the long flows, which have mean transfer time higher than predicted by the analytic model. The analytic models can be extended to include the impact of packet loss on the mean flow transfer time, using a variation on the techniques in [97], but such extensions are out of scope of this thesis.

### 5.4.5 LAS-fixed(k) Model Validation

In the validations of the differentiated LAS policies, unless otherwise stated, the priority flows represent 10% of the Web TCP flows (i.e., $q = 0.1$), and thus comprise 10% of the total load on the bottleneck link. For validating the LAS-fixed(k) model for UDP priority flows, the priority flow is a single long-lived UDP flow that sends packets at a constant bit rate during the entire simulation. The rate of the flow is chosen such that it contributes 10% of the total load.




(a) *Ordinary Flows,* $k = 25$ (b) *Priority Flows,* $k = 25$

Figure 5.7: *Mean transfer time validation for LAS-fixed(k) TCP;* $\rho = 0.73$*, loss rate* = $1.2\%$*, and* $q = 0.1$.

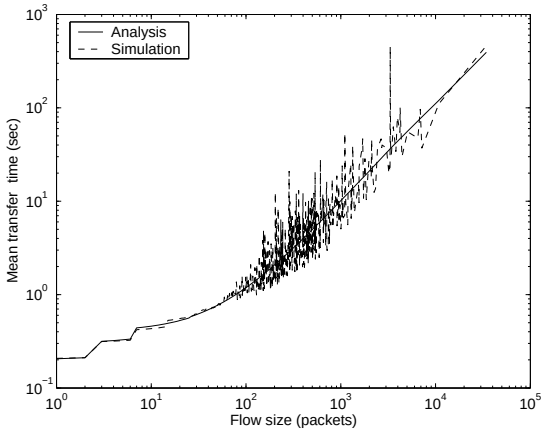
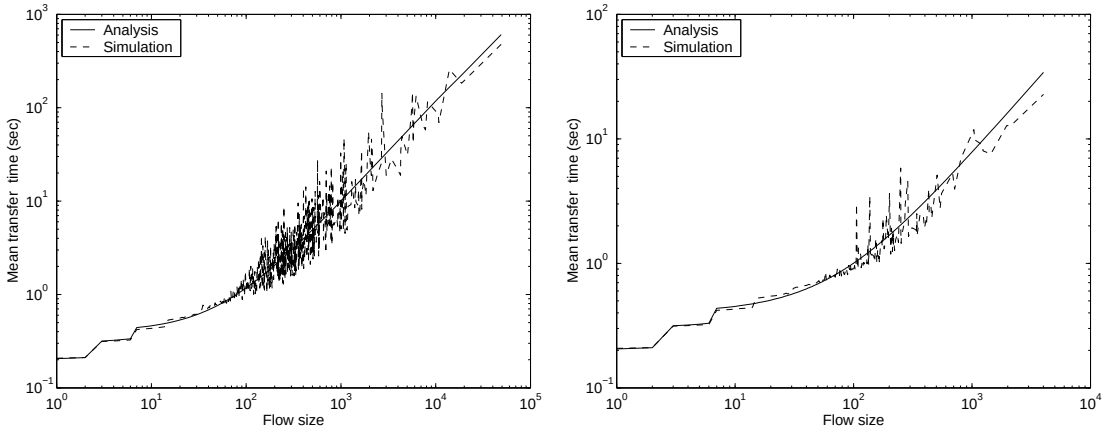



Figure 5.8: *Mean transfer time validation for LAS-fixed(k) UDP; load* $\rho = 0.73$*, loss rate* = $1.2\%$*, and* $q = 0.1$.

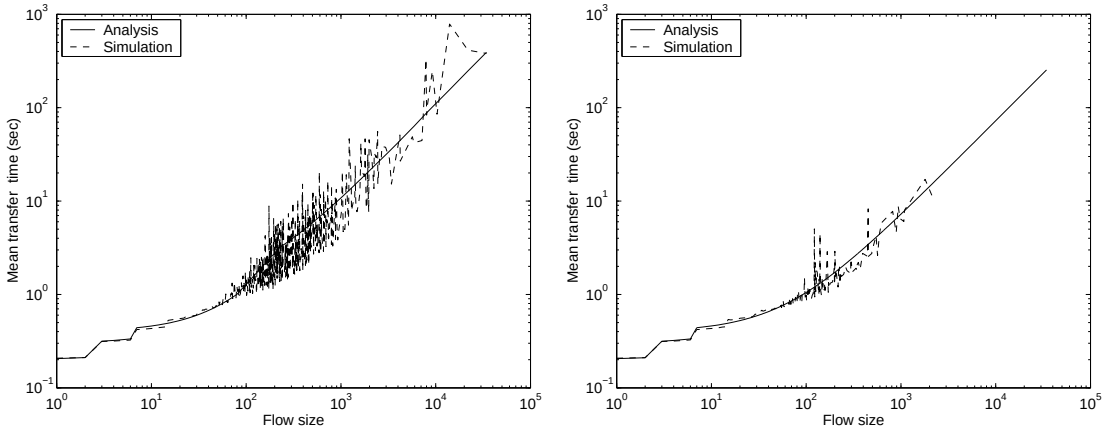
Figure 5.7 shows the validation results for LAS-fixed(k), with $k = 25$ and TCP priority flows. We observe that the simulation validates the model. Similarly, Figure 5.8 shows the results for ordinary TCP flows when $k = 25$ and the priority flow is the UDP flow. Observe again that the analytic mean flow transfer time as a function of flow size for the ordinary flows is in excellent agreement with the simulation results. Note also that the mean total transfer time of the priority flow is not validated, since the priority flow is at fixed rate and is active during the entire simulation.




(a) *Ordinary flows,* $k = 5$ (b) *Priority flows,* $k = 5$

Figure 5.9: *Mean transfer time validation for LAS-linear;* $\rho = 0.73$*, loss rate= 1.2%, and* $q = 0.1$.

### 5.4.6 LAS-linear(k) Model Validation



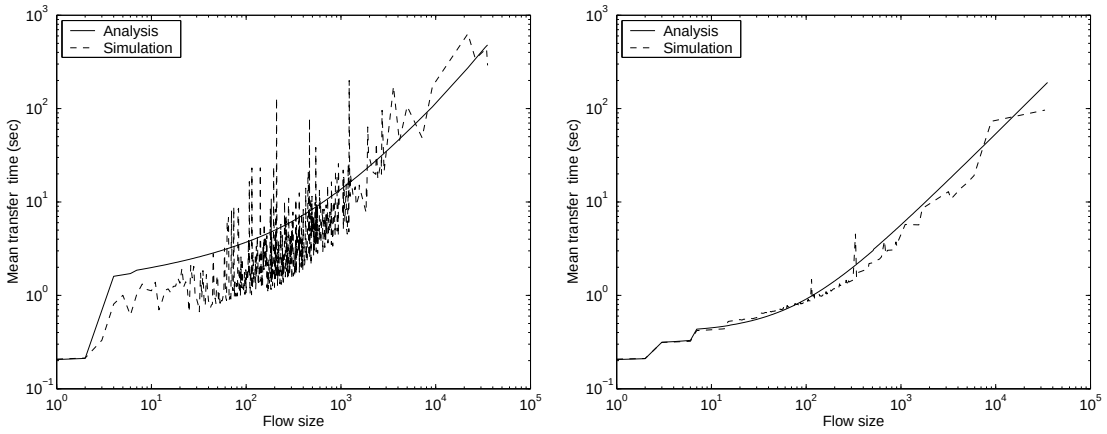

(a) *Ordinary flows,* $k = 0.5$ (b) *Priority flows,* $k = 0.5$

Figure 5.10: *Mean transfer time validation for LAS-log;* $\rho = 0.73$*, loss rate* = $1.2\%$*, and* $q = 0.1$.

Figures 5.9(a) and (b) show the mean transfer time as a function of flow size for ordinary and priority flows, respectively. The simulation results and analytical estimates are in excellent agreement. We have observed the same level of agreement for other values of $k$.

### 5.4.7 LAS-log(k) Model Validation

Figures 5.10 and 5.11 show the mean flow transfer time as a function of the flow size under LAS-log(k), for $k = 0.5$ and $k = 2$, respectively. For $k = 0.5$, the simulation results and the analytical estimates are in excellent agreement. For $k = 2$ the mean transfer times of low priority flows differ slightly. However, the analysis and simulation results are in good agreement for $k = 2$; in particular, both results exhibit the significant increase in transfer time for the ordinary flows of size 3 as compared to ordinary flows of size 2. Note that while ordinary flows of size 2 need to share the link with priority flows up to size $2^{2^2} = 16$, ordinary flows of size 3 need to share the link with priority flows up to size $2^{3^2} = 64$.




(a) *Ordinary flows,* $k = 2$ (b) *Priority flows,* $k = 2$

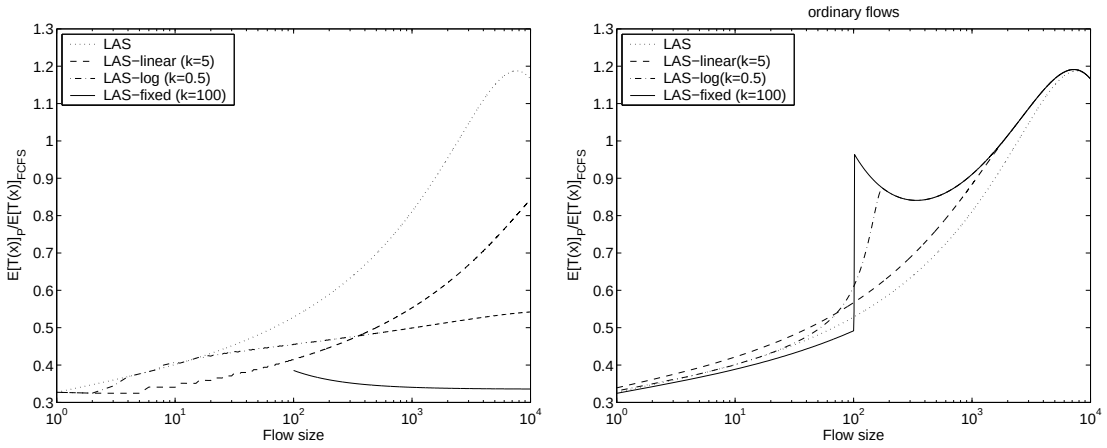
Figure 5.11: *Mean transfer time validation for LAS-log;* $\rho = 0.73$*, loss rate* = $1.2\%$*, and* $q = 0.1$.

In summary, in this section we have validated the analytical models of the FCFS, LAS, and differentiated LAS policies for a Web traffic profile with Pareto distributed object sizes. We presented representative results for particular values of $k$ and $q$; we have also obtained similar validation results for other parameter values and for exponentially distributed object sizes, for all policies. Given these validations, the analytical models can be used to evaluate the performance of LAS and differentiated LAS policies for flows that share a bottleneck link (e.g., at the edge of the network) and experience little other contention in the network. We apply the models in this way in the next section.

Extensions of the analytic models for multiple points of contention are possible, but are outside the scope of this thesis. Simulations of topologies that have multiple bottleneck links are quite complex to specify and very time-consuming to run, so analytic models for such networks could be quite advantageous.

## 5.5 Policy Comparisons

In two-class LAS, ordinary flows have packet priorities equal to the position of the packet in the flow, whereas priority flows have packet priorities determined by a priority function $P(x)$, where $x$ is the position in the flow. Since $P(x) < x$, for each of the differentiated service policies, an ordinary flow with attained service $x$ will experience equal interference from other ordinary flows as in the single-class LAS system, and greater interference from the priority flows. Conversely, a priority flow with attained service $x$ will experience equal interference due to other priority flows but less interference from ordinary flows in the differentiated LAS policy as compared with the single-class LAS system. In this section we provide results on the quantitative impact of the priority functions on mean flow transfer times, packet loss, and packet jitter. The results in this section are obtained by considering the models $T(x)$ for the proposed policies without taking into account the impact of the TCP algorithm. We note that using these simple models is enough to get qualitative performance comparisons among the proposed policies.




(a) *Priority flows* (b) *Ordinary flows*

Figure 5.12: *Policy Performance Comparisons: Ratio of Mean Flow Transfer Time for Policy P to FCFS (*$q = 0.1$, $\rho = 0.7$*).*

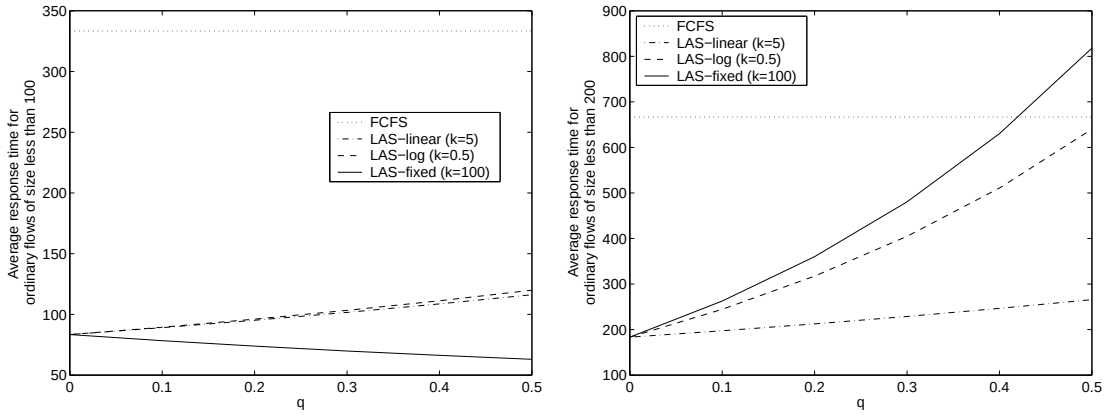
### 5.5.1 Mean Flow Transfer Time

Figure 5.12 shows, for LAS and each differentiated LAS policy, the ratio of the average flow transfer time under the policy to the average flow transfer time under FCFS, versus flow size. The results are shown for a system with $\rho = 0.7$ and 10% of the flows are priority flows ($q = 0.1$). The analytical results in this section are obtained using the Bounded Pareto flow size distribution, $BP(1, 10^4, 0.92)$, which has a mean of 12. The LAS-fixed(k) policy with $k = 100$ provides the best performance for the (large) priority flows, but at a significant performance penalty compared to LAS for ordinary flows larger than $k$. The LAS-linear(k) with $k = 5$ achieves better performance than LAS for priority flows, and similar performance for

the ordinary flows. The LAS-log(k) policy with $k = 0.5$ achieves significantly more uniform performance than LAS-linear(k) for priority flows, with some penalty compared with LAS or LAS-linear(k) for moderate-size ordinary flows.

Figures 5.13(a) and (b) show the impact of varying the percentage of priority flows on the average transfer time of ordinary flows of size less than 100 or 200 packets, respectively. For ordinary flows that have size up to $100$, the average transfer time under LAS-linear $(k = 5)$ and LAS-log $(k = 0.5)$ increases as the percentage of priority flows increases. However, for LAS-fixed$(k = 100)$ the priority flows do not interfere with the ordinary flows of size less than 100. Therefore, as the percentage of priority flows increases, the traffic volume due to ordinary flows of size less than $100$ decreases, and thus the mean transfer time for short ordinary flows also decreases.

For ordinary flows of size less than or equal to $200$, the average transfer time increases as the percentage of priority flows increases, for all of the differentiated service policies. For LAS-fixed $(k = 100)$, the increase is most pronounced, since all packets from priority flows will have priority over packets from ordinary flows that have attained a service of at least $100$. On the other hand, the increase is modest for the LAS-linear($k = 5$) policy, and moderate for the LAS-log($k = 0.5$) policy if $q < 0.3$.




(a) *Flow size* $x \leq 100$ (b) *Flow size* $x \leq 200$

Figure 5.13: *Average transfer time of short ordinary flows versus fraction of jobs in priority class 1 (q);* $\rho = 0.7$.

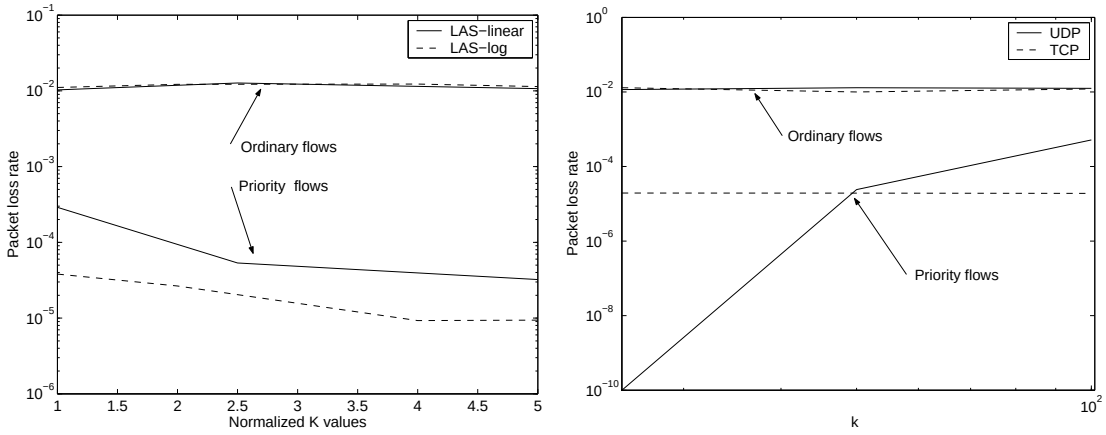
### 5.5.2 Packet Loss Rates

Figure 5.14(a) shows simulation results for the packet loss rate of ordinary and priority TCP flows under LAS-log and LAS-linear as a function of normalized values $k/min(k)$, where $min(k) = 0.5$ for LAS-log and $min(k) = 25$ for LAS-linear.

We observe that priority flows have a loss rate that is more than an order of magnitude lower than the loss rate for ordinary flows. For LAS-log, the reduction in loss rate for priority flows is higher than for LAS-linear as large flows see their priority decrease much more slowly under

LAS-log than LAS-linear (c.f. Figure 5.1). Increasing $k$ helps both policies further reduce the loss rate of priority flows.

Figure 5.14(b) shows, for LAS-fixed(k), the packet loss rate for priority and ordinary TCP and UDP constant bit rate flows. As already observed for LAS-log and LAS-linear, the loss rate of priority flows is reduced by more than one order of magnitude. While the loss rate for priority TCP flows is fairly insensitive to the choice of $k$ for $k \leq 100$, this is not the case for the fixed rate UDP flow. This difference is due to the underlying transmission protocol. The TCP source adjusts its window size in response to observed packet loss, whereas the UDP source transmits packets at a fixed rate. As the value of $k$ is increased the priority UDP flow will experience more interference due to the packets from ordinary flows.




(a) *LAS-linear and LAS-log* (b) *LAS-fixed*

Figure 5.14: *Packet loss rate as a function of normalized $k$ values for $\rho = 0.73$ and $q = 0.1$.*

### 5.5.3 Performance of Long-lived Flows

#### Packet Jitter for a Priority UDP Flow

We use simulation to study the jitter experienced by a long-lived UDP flow under plain LAS and under differentiated LAS policies, where the UDP flow is treated as priority flow. The UDP flow sends packets from source to sink at a constant rate of 128 Kb/s until the simulation terminates.

Figure 5.15 shows the average jitter of received packets, which are grouped in non-overlapping windows of 100 sent packets each. We observe that the LAS scheduling has the highest average jitter among the three policies and LAS-log has the lowest average jitter. This is due to the fact that, for $k = 2$, the priority of the packet at position 64000 in the UDP flow under the LAS-log policy is $(log_2(64000))^{0.5} \approx 4$. The priority value of the UDP flow is constant at $25$ for the LAS-fixed(k). Under LAS, simulation results show that the UDP flow experiences temporary loss rates up to $10\%$ or more (see Figure 3.14 Chapter 3), while the flow does not experience any appreciable packet loss under either LAS-log or LAS-fixed(k).

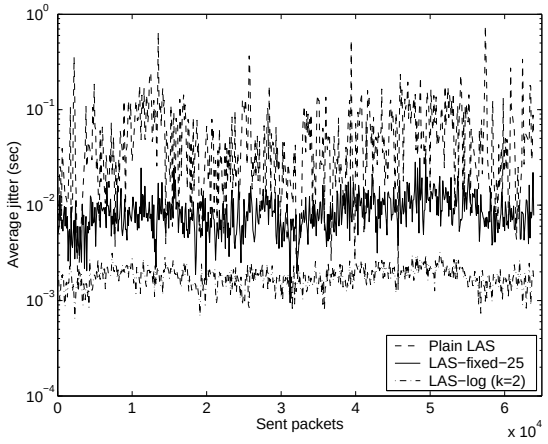



Figure 5.15: *Average jitter of UDP flow for non-overlapping windows of 100 sent packets,* $\rho = 0.7$.

### Throughput for a Priority FTP Flow

In Chapter 3, the throughput of a long-lived FTP flow was observed to deteriorate over time at high loads of about 0.9. A long-lived FTP flow should see good throughput performance under the differentiated policies proposed in this chapter depending on the $k$ parameter used. An as example, we simulated the same network topology used to analyze the performance of a long-lived FTP flow under LAS in Chapter 3 (Figure 3.5), except that the access link for the FTP source is changed to 1Mb/s to limit its source rate otherwise it would occupy all of the bottleneck link. We study the throughput performance of the long-lived FTP flow under *LAS-log* policy with $k = 0.5$ and $k = 1$. In addition, like for any priority based policy, to avoid starving ordinary flows, the fraction of the priority flows in the network should be reasonably small. We set the fraction of priority flows to 1/3 to be the maximum fraction it can possibly have. We set the long-lived flow as a priority flow and the priority values of its packets follow the logarithmic function as shown in Figure 5.1.

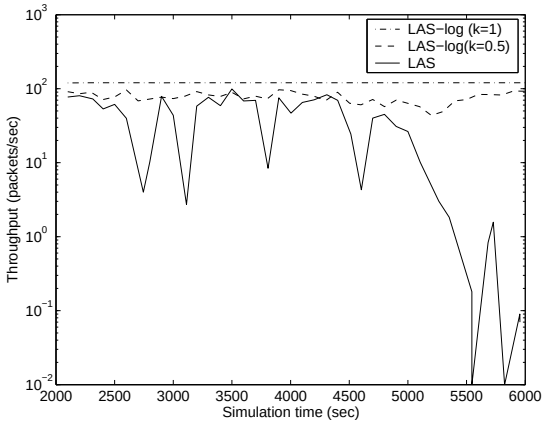
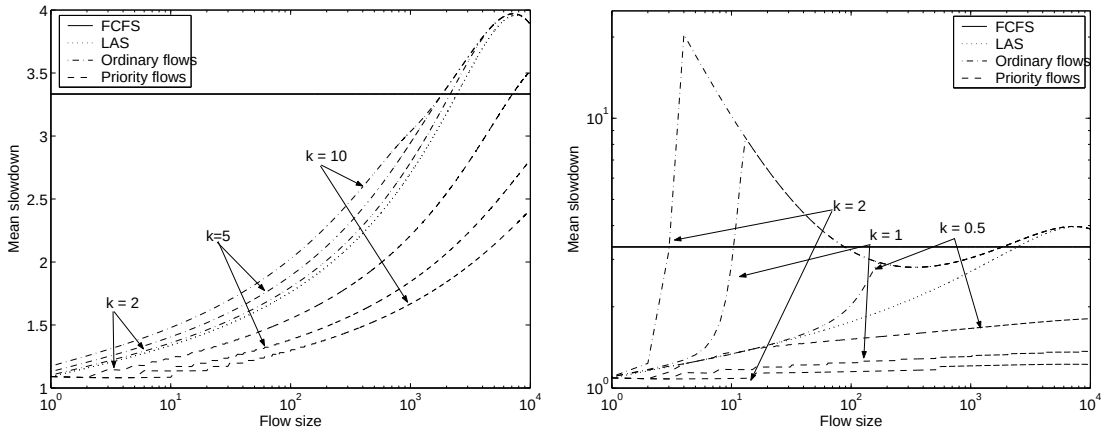



Figure 5.16: *Throughout of the long-lived FTP flow under LAS and LAS-log (k):* $k = 0.5$ *and* $k = 1$.

Figure 5.16 compares the throughput of the long-lived FTP flow under plain LAS and LAS-log with $k = 0.5$ and $k = 1$ as a function of simulation time. We first note that the throughput under plain LAS is highly varying, and decreases to zero after at the end of the simulation time. In contrast, the figure shows that LAS-log discipline improves the throughput of the flow. When $k$ value is set to $0.5$, the flow maintains almost constant throughput around 90 packets/sec. For $k = 1$, the performance of the long-lived flow reaches to about 120 packets/sec and does not deteriorate over time. The throughput of 120 packets/sec is the equivalent to the access link speed of 1Mb/s for packets of equal size of 1040 bytes. Thus, we can conclude that LAS-log can improve the service of long-lived priority FTP flows to the extent that the flows are not affected by Web traffic at all.

### 5.5.4 Unfairness Analysis

A job is said to be treated unfairly by a policy if its mean slowdown under the policy is greater than its mean slowdown under FCFS scheduling, and it is treated fairly by a policy if its mean slowdown under that policy is smaller than its mean slowdown under FCFS [103]. Similarly, a policy is said to be **always fair** if given any job distribution and any load condition, all jobs experience a mean slowdown strictly less than their mean slowdowns under FCFS. Based on these definitions, the work in [103] further establishes that LAS policy is *never fair*, which means that LAS unfairly treats at least one job.




(a) *LAS-linear* (b) *LAS-log*

Figure 5.17: *Mean slowdown as a function of flow size for* $q = 0.1$ *and load* $\rho = 0.7$*, for* $BP(1, 10^4, 0, 92)$ *for LAS-log and LAS-linear policies.*

When the goal of a policy is to offer service differentiation among classes of flows, as the case for the proposed policies in this chapter, the policy is analyzed based on its performance under different classes of flows. In this case, it is interesting to know if the policy is fair or unfair to one or more classes of flows. In other words, it is important to know if a policy is either *always*, *sometimes*, or *never* fairly treats a particular class of flows. In this section, we present some results that show the impact of the parameter $k$ on the unfairness of the proposed

policies. However, regardless of the parameters used by a policy, it is apparent that any LAS-based policy with service differentiation will treat ordinary priority classes of flows unfairly since plain LAS is an unfair policy and ordinary flows under differentiated LAS policies have higher transfer times than under plain LAS.

Figures 5.17(a) and 5.17(b) show the mean slowdown as a function of flow size and parameter $k$ for priority and ordinary flows under LAS-linear and LAS-log respectively. The plots also show the mean slowdown of LAS and FCFS. We note that, while LAS-linear and LAS-log are always unfair to ordinary flows and they are sometimes fair to priority flows. Unfairness for both policies depend on their $k$ values. In particular, for the flow size distribution used in this section, we observed that LAS-linear and LAS-log treat priority jobs fairly for $k \geq 3$ and $k \geq 0.32$ respectively.

In practice, it is desirable to choose the $k$ value for any policy such that its performance is improved with a minimum priority flows with minimum impact on the performance of ordinary flows. To achieve this, care should be taken in choosing the proper value of $k$ for all proposed policies, for example as the value of $k$ for LAS-log policy gets small, e.g., $k = 0.3$, packet priorities of the priority flows decrease faster than the ones for ordinary flows. This causes *priority inversion* and ordinary flows experience better performance than priority flows in this case (see Figure 5.18). For LAS-log, we note from Figure 5.17(b) that $k = 1$ is a good choice. On the other hand, LAS-linear does not have a priority inversion problem and any choice of $k$ value that reduces the transfer time and jitter of priority flows to the extent that their service requirements are satisfied with a small penalty for ordinary flows is applausable.

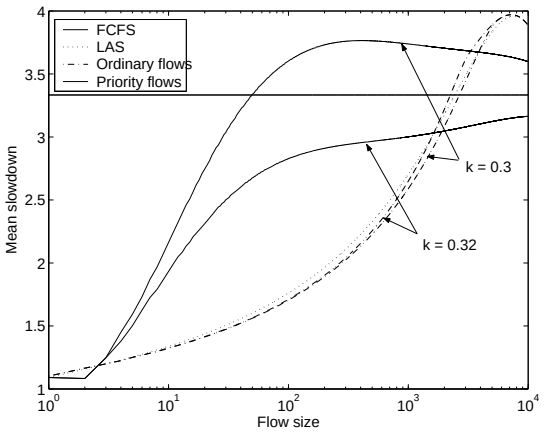
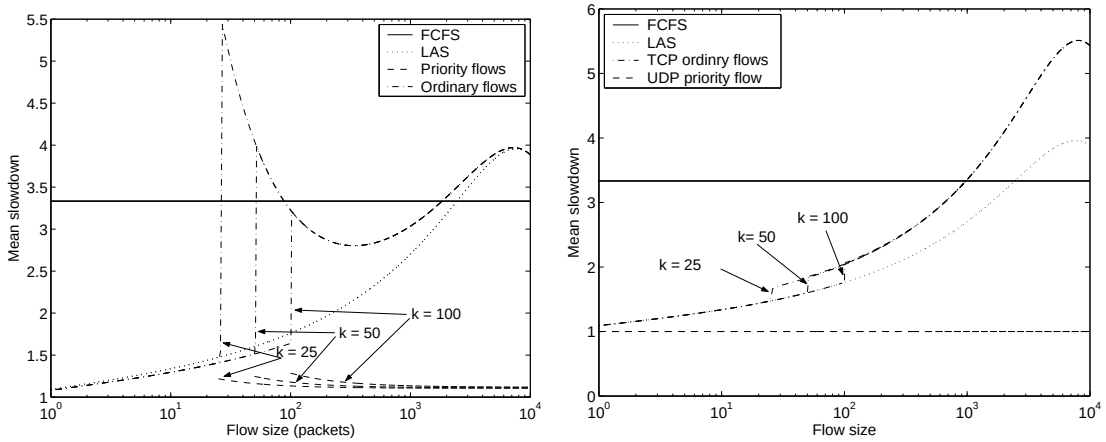



Figure 5.18: *Mean slowdown as a function of flow size for LAS-log, load* $\rho = 0.7$*, and* $BP(1, 10^4, 0.92)$.

Figures 5.19(a) and 5.19(b) show the mean slowdown as a function of flow size under LAS-fixed(k): TCP and UDP versions respectively. We observe from the figures that these policies can significantly reduce the transfer times of priority flows, but they can also penalize the ordinary flows a lot depending on the $k$ value used. For the TCP version of LAS-fixed, moderately large $k$ values don't cause heavy penalty for ordinary flows. As for UDP version of LAS-fixed, Figure 5.19(b) shows that different choices of $k$ values show similar impact on the penalty experienced ordinary flows, which is small. To summarize, we observe that the proposed differ-

entiated LAS policies are *always unfair* to ordinary flows and *sometimes fair* to priority flows.




(a) *LAS-fixed(k): TCP* (b) *LAS-fixed(k): UDP*

Figure 5.19: *Mean slowdown as a function of flow size for* $q = 0.1$ *and load* $\rho = 0.7$*, UDP flow of rate 2 packet/sec, and* $BP(1, 10^4, 0.92)$.

## 5.6 Conclusion

This chapter has examined how new variants of LAS scheduling in can improve the performance of long-lived TCP and UDP priority flows. While LAS significantly reduces the mean transmission time and loss rate of short flows, it also increases the average and variance in packet transmission time of the later packets in long flows and reduces the average throughput of long-lived flows at high load. To reduce this performance penalty, we have developed several alternative two-class LAS policies that provide service differentiation for long flows that require better performance.

For each policy, we have developed an analytical model that estimates the mean flow transfer time versus flow size as accurately as an ns2 simulation, for the dumbbell topology with loss rate below 2%. Advantages of the accurate analytic models include their efficiency for policy performance comparisons over a wide range of system configurations and workloads, the ease of obtaining measures such as the average packet transmission time as a function of position in the flow, and the insights into system behavior as well as which system features have principal impact on observed performance.

Simulation and analytical results in this chapter show that the proposed two-class policies can significantly improve the performance of priority long-lived flows. The performance of the differentiated LAS policies depends on the value of $k$ used. For well chosen values, we observe that all policies, i.e., LAS-log, LAS-linear, and LAS-fixed policies significantly improve the performance of priority flows while having a small negative impact on the performance of the

ordinary flows. In general however, depending on value of $k$, the performance improvement for priority flows can come at the expense of a significant performance degradation for ordinary short flows. This may not be desired, in particular when these short flows make a large fraction of all flows. In addition, the actual traffic distribution in a network may change and the chosen $k$ value may give different performance results for ordinary flows with different flow size distributions. This motivates us to analyze in the next chapter a variant of LAS scheduling that improves the performance of the largest flows while preserving for all short flow, a performance as good as under plain LAS.

# Chapter 6

# *LAS-FCFS* Model and its Variants to Improve the Performance of Large Flows and to Offer Differentiated Services

Differentiated LAS based scheduling policies proposed in the previous chapter significantly improve the performance of priority flows, but they can penalize short ordinary flows in terms of increasing their transfer times. Penalizing short flows may not be desired particularly when the flow size distribution exhibits high variability property. In this chapter, we present another variant of LAS scheduling that improves the performance of the largest jobs and keeps the performance of short jobs the same as under plain LAS. We call this policy **LAS-FCFS**.

LAS-FCFS services all jobs that have attained less than a specified *threshold* of service by using LAS scheduling and all jobs that have attained more than a threshold amount of service by using FCFS. Jobs of size less than the threshold are called *foreground jobs* and jobs with service requirement more than the threshold are known as *background jobs*. Background jobs under LAS-FCFS receive service only if there are no foreground jobs waiting in the queue. The advantage of using LAS-FCFS is to maintain the low response time for short jobs exactly as offered under LAS while using FCFS to reduce the mean response time of the largest jobs.

The numerical results show that the performance of LAS-FCFS in terms of reducing the mean response time of the largest jobs depends on the variability of a job distribution and is more pronounced for job sizes that exhibit the high variability property. While reduces the penalty experienced by the largest jobs, LAS-FCFS penalizes the jobs with moderate size. We propose variants of LAS-FCFS that offer differentiated services in order to prevent penalty to all identified high priority jobs. The numerical analysis reveals that *Fixed-Priority* LAS-FCFS achieves an ideal service differentiation at the expense of the high penalty for the low priority small jobs. While *Differential* LAS-FCFS offers acceptable service differentiation, it does not penalize small jobs with low priority at all. We finally present the integral expressions for analytic models that compute the mean flow transfer time in packet networks as a function of a flow size for LAS-FCFS and its variants.

# 6.1 Introduction

LAS is known to favor short jobs but it can penalize the largest jobs. Conversely, FCFS is known to penalize short jobs and to favor large jobs. In this chapter, we analyze another scheduling policy that captures the good features of both, LAS and FCFS, which we call LAS-FCFS. LAS-FCFS services all arriving jobs using LAS scheduling until they attain a threshold amount of service. If a job is smaller than the threshold it leaves the system otherwise LAS-FCFS services the remaining service jobs that are larger than the threshold using FCFS scheduling. Under LAS-FCFS, all short jobs smaller than the threshold complete their service requirements under LAS scheduling and large jobs complete their remaining service requirements under FCFS.

Before we proceed it is worth mentioning that we will perform analytical evaluation of the policies in this chapter at a **job** level and we will present closed form integral expressions for the corresponding analytic models in packet networks. The exact analysis of the policies in this chapter for packet networks is hard as it involves processor sharing with **batch** arrivals.

It is easier to model LAS-FCFS for jobs using a finite series of feedback queues as shown in Figure 6.1. In this case, LAS-FCFS is modeled by $N$ queues, which are classified as (*N-1*) *foreground queues* and one *background queue*. As a result, LAS-FCFS is also known by $FB_N$ [24, 65]. In fact, LAS-FCFS was first proposed and analyzed in [24] under the name of $FB_N$. LAS is sometimes known as $FB_\infty$ because its model for jobs can be derived when Figure 6.1 has an infinite series of feedback queues [24].

All queues in the LAS-FCFS model share a single server according to their priorities. Let the first foreground queue be indexed 1 and the background queue indexed $N$, then a job at the head of queue $j$ receives service if all queues $i < j$ are empty. In LAS-FCFS, all jobs arrive at the first (highest priority) foreground queue where they receive, in FCFS order, a fixed amount of service called *quantum*. Each queue represents service quanta from different jobs but of the same priority value. A job then either leaves the system if it has completed its service or is relayed to the subsequent queue. From each foreground queue, the job receives a service of one quantum if the previous queues are empty, until it is completely served. A job in service is preempted by an arriving job only if it has completed its service quantum.

Jobs that complete their service in one of the foreground queues are called *foreground jobs*, and jobs that have not completed their service in one of the foreground queues are called *background jobs*. Background jobs in the background queue are also serviced in FCFS order. A background job returns back at the *head* of the background queue after each quantum of service, where it is immediately taken back to service if all foreground queues are still empty. The process continues until the job leaves the system. Quantum values in LAS-FCFS can be the same for all queues or can be different for each queue. We will use this quantum based approach to derive mathematical expressions of all policies discussed in this chapter.

Note that the service received by foreground jobs is the same as their service under plain $LAS$. Hence, the benefits of $LAS$ in minimizing the mean response time for jobs can also be reaped by using a LAS-FCFS policy. The numerical results for LAS-FCFS show that it favors the largest jobs better when a job size distribution has a high variability property than those

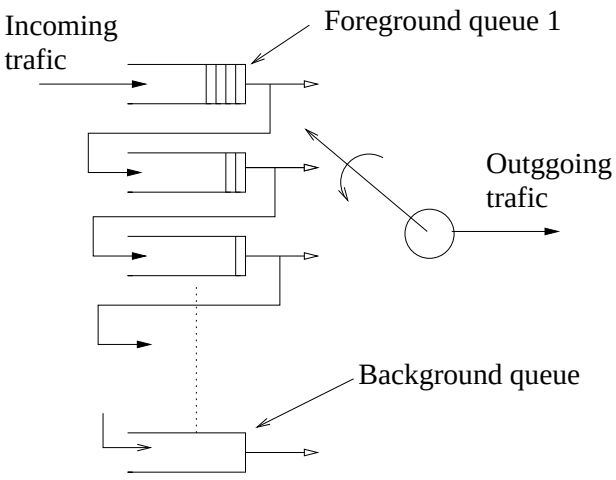



Figure 6.1: *LAS-FCFS model.*

that don't have. LAS-FCFS benefits the largest jobs of a job size with lower variability (exponentially distributed) when the threshold value is set small. Small threshold values however can cause penalty to short jobs. The results also show that the maximum mean slowdown of the background jobs that return to the background queue a few times (also have moderate size) depends on the threshold value and system load $\rho$: it increases in increasing load and for small threshold value. Just like LAS, LAS-FCFS can service some jobs at overload.

While the background jobs that are penalized under LAS-FCFS comprise a very tiny percentage of jobs for a job size distribution with high variability property (less than 1%), this penalty is not acceptable for critical jobs or users. LAS-FCFS however, cannot differentiate the service of jobs using any attribute other than their sizes. Hence, it cannot guarantee the service quality of the critical jobs. Moreover, since large jobs are serviced under FCFS, at high load or overloaded conditions, LAS-FCFS can penalize them despite its efforts to avoid the penalty. Thus, we propose variants of LAS-FCFS that differentiate the service among jobs by classifying them into *high priority jobs* and *low priority jobs* based on desired attributes. The objective is to guarantee low mean response time for **all** high priority jobs even at overloaded conditions.

We first propose *fixed-priority* LAS-FCFS (**FP-LAS-FCFS**) scheduling policy, which serves jobs of each priority in a separate LAS-FCFS system such that *low priority* jobs receive service only if there is no *high priority* job in the corresponding LAS-FCFS system. In fixed priority LAS-FCFS, the service of high priority jobs depends only on the load they constitute rather than the total system load. Therefore, fixed priority LAS-FCFS can guarantee the service of the high priority jobs even under overload, as long as they constitute load of less than 1, which should be the case in practice. A variant of fixed priority LAS-FCFS scheduling that uses simple FCFS instead of LAS-FCFS to service high priority jobs is also analyzed. Note that fixed priority LAS-FCFS can significantly improve the mean response time of high priority jobs at the expense of high response time to low priority small jobs. However, the mean slowdown of all small jobs under LAS-FCFS (without service differentiation) is already significantly low. Since these jobs constitute a large percentage of all jobs in a job size distribution that shows the high variability property, penalizing them, as in fixed priority LAS-FCFS, degrades the over-

all system performance. Thus, we propose a *differential* LAS-FCFS (**DF-LAS-FCFS**) policy, which also significantly reduces the mean response time of the high priority background jobs while maintaining the mean slowdown of low priority small jobs as low as under LAS-FCFS. This is achieved by differentiating only the service of the background jobs such that the low priority background jobs receive service in the background queue only when there are no high priority background (and foreground) jobs in the system.

Using the results of the models developed in Chapter 5, an analytic model for LAS-FCFS in packet networks will service packets in foreground flows using LAS and packets in background queue using approximate PS scheduling. We call the analytic model of LAS in packet networks a **LAS-PS**. Closely related to LAS-PS is FLIPS [40], which is another designated name for a model of LAS-FCFS in packet networks. In [40], FLIPS is simulated and the results show that FLIPS offers lower mean transfer times for the largest flows than does LAS. Similar scheduling policies to LAS-FCFS in packet networks are Run2C [10], *short flow differentiating* (SFD) [21], and ML-PS [50]. Run2C and ML-PS are threshold based policies proposed to schedule packets in order to favor short flows and to minimize the penalty for the largest flows. These policies are however different from LAS-FCFS because they use FCFS policy to service short flows instead of LAS as under LAS-FCFS. SFD and ML-PS use preferential dropping in addition to fixed priority scheduling between short and long flows.

The rest of the chapter is organized as follows: In the next section we derive the expressions of the conditional mean response time for LAS-FCFS using the notion of quantum. In Section 6.3, we discuss LAS-FCFS under overload. In Section 6.4 we present numerical results for LAS-FCFS. Variants of LAS-FCFS for service differentiation are proposed in Section 6.5. We then discuss analytic models of LAS-FCFS and its variants for flows in packet networks in Section 6.6 and finally we conclude the chapter in Section 6.7.

## 6.2 LAS-FCFS scheduling

The expression for the conditional mean waiting time for a fixed size quantum LAS-FCFS is derived in [105, 65]. In ([64], Chapter 4), different multilevel processor-sharing variants of LAS-FCFS are analyzed. In [65, 64] numerical evaluations for the conditional mean response time of jobs under LAS-FCFS are presented for the case of an M/M/1 system. We will derive the expressions for the conditional mean response time for LAS-FCFS and analyze it under the M/G/1 model.

In LAS-FCFS scheduling model, a job in a queue $j$ receives a service of one quantum ($s_j$) only if all queues $i, i < j$ are empty. Let $\tau_j$ be the service received by a job up to the foreground queue $j \leq N-1$, i.e., $\tau_j \triangleq \sum_{i=1}^{j} s_i$. A foreground job of size $\tau_j$ is delayed by the system backlog due to all the jobs in the system at the arrival instant ($W_o(\tau_j)$), truncated to a size $\tau_j$ if the job size is $x > \tau_j$. This workload is given by the Pollaczek-Khinchin (PK) mean value

formula [64] applied to a job of size $\tau_j$ as:

$$W_o(\tau_j) = \frac{\lambda \overline{x^2_{\tau_j}}}{2(1-\rho_{\tau_j})} \tag{6.1}$$

where $\overline{x_x}$ and $\rho_x$ are as defined in Equations (5.1) and (5.2) respectively. Note that $W_o \triangleq W_o(\infty)$ is the mean total work in the system seen by an arriving job. One can show that Equation (6.1) is the same as Equation (9) in [105], which is used to derive the expression for the conditional mean waiting time in LAS. Equation (6.1) is used in this chapter because of its compact form. The arriving job at queue $j$ will further be delayed by all new arrivals to queues $i < j$ before he completes the service. Thus, the jobs will receive service in queue $j$ after a total waiting time of:

$$\begin{aligned} W(\tau_j) &= W_o(\tau_j) + \lambda(W(\tau_j) + \tau_{j-1})\overline{x_{\tau_{j-1}}} \\ &= \frac{W_o(\tau_j) + \tau_{j-1}\rho_{\tau_{j-1}}}{(1-\rho_{\tau_{j-1}})} \end{aligned} \tag{6.2}$$

where $\rho_{\tau_{j-1}} = \lambda\overline{x_{\tau_{j-1}}}$. Since $T(\tau_j) \triangleq W(\tau_j) + \tau_j$, we obtain the expression for the conditional mean response time of a job of size $\tau_j$ under LAS-FCFS ($T(\tau_j|\tau_j \leq \tau_{N-1})$) as:

$$T(\tau_j|\tau_j \leq \tau_{N-1}) = \frac{W_o(\tau_j) - s_j\rho_{\tau_{j-1}}}{(1-\rho_{\tau_{j-1}})} + \frac{\tau_j}{(1-\rho_{\tau_{j-1}})}. \tag{6.3}$$

It is easy to see that as quantum sizes, $s_i \;\; i \in \{1, 2, \cdots, N\}$ approach to zero, Equation (6.3) becomes the same as the expression for $T_{LAS}(x)$ given in Equation (5.4).

Let us now look at background jobs. A background job returns at the head of the background queue $k_N \triangleq \lfloor \frac{x-\tau_{N-1}}{s_N} \rfloor$ times, and each time it is immediately taken back into service if all foreground queues are still empty. We denote the mean waiting time of a background job as $W_B(x)$, which is due to the total work that the background job finds in the system upon its arrival ($W_o$) and the delay due to service interruptions caused by new arrivals while the background job is in the system ($W_s(\tau_{N-1})$). The time during which these new arrivals may occur has a mean duration of $T(x) - s_N = W_B(x) + \tau_{N-1} + (k_N - 1)s_N$. The term $k_N - 1$ arises from the fact that a job is not interrupted once it begins its last quantum of service. The new arrivals have a mean arrival rate of $\lambda$. Hence, by Little's Law, the average number of these new arrivals is given as $\lambda(W_B(x) + [\tau_{N-1} + (k_N - 1)s_N]) = \lambda(T(x) - s_N)$, each of which delays the background job by an average time of $\overline{x}_{\tau_{N-1}}$. Therefore, the expression of $W_s(\tau_{N-1})$ is given as $(W_B(x) + [\tau_{N-1} + (k_N - 1)s_N])\rho_{\tau_{N-1}}$, where $\rho_{\tau_{N-1}} = \lambda\overline{x}_{\tau_{N-1}}$. Hence,

$$W_B(x) = W_o + W_s(\tau_{N-1}), \tag{6.4}$$

substituting the expression of $W_s(\tau_{N-1})$ in Equation (6.4) we obtain:

$$W_B(x) = W_o + (W_B(x) + [\tau_{N-1} + (k_N - 1)s_N])\rho_{\tau_{N-1}},$$

making $W_B(x)$ the subject, we obtain the expression of $W_B(x)$ as:

$$W_B(x) = \frac{W_o + [\tau_{N-1} + (k_N - 1)s_N]\rho_{\tau_{N-1}}}{(1-\rho_{\tau_{N-1}})}. \tag{6.5}$$

We will use the expression of $W_s(\tau_{N-1})$ repeatedly in this chapter, it is therefore convenient to provide its equation here as follows:

$$W_s(\tau_{N-1}) \triangleq (T(x) - s_N)\rho_{\tau_{N-1}} \tag{6.6}$$

Using the relation that $T(x) = W(x) + x$, we obtain the expression for the conditional mean response time of a background job of size $x > \tau_{N-1}$ as:

$$T(x|x > \tau_{N-1}) = \frac{W_o - s_N\rho_{\tau_{N-1}}}{(1 - \rho_{\tau_{N-1}})} + \frac{x}{(1 - \rho_{\tau_{N-1}})}. \tag{6.7}$$

Finally, we obtain the expression for the mean response time of a job under LAS-FCFS as:

$$T(x) = \begin{cases} \frac{W_o(x) - s_j\rho_x}{(1-\rho_x)} + \frac{x}{(1-\rho_x)} & \text{if } x \leq \tau_{N-1} \\ \frac{W_o - s_N\rho_{\tau_{N-1}}}{(1-\rho_{\tau_{N-1}})} + \frac{x}{(1-\rho_{\tau_{N-1}})} & \text{if } x > \tau_{N-1}. \end{cases} \tag{6.8}$$

Note from Equation (6.8) that the conditional mean response time of a job with size less than or equal to $\tau_{N-1}$ under LAS-FCFS is the same as its conditional mean response time under $LAS$ for small quantum $s_j$ values. Therefore, a good tuning of $\tau_{N-1}$ value for a job size distribution with a high variability property can guarantee that all small jobs receive the same service under LAS-FCFS as under $LAS$.

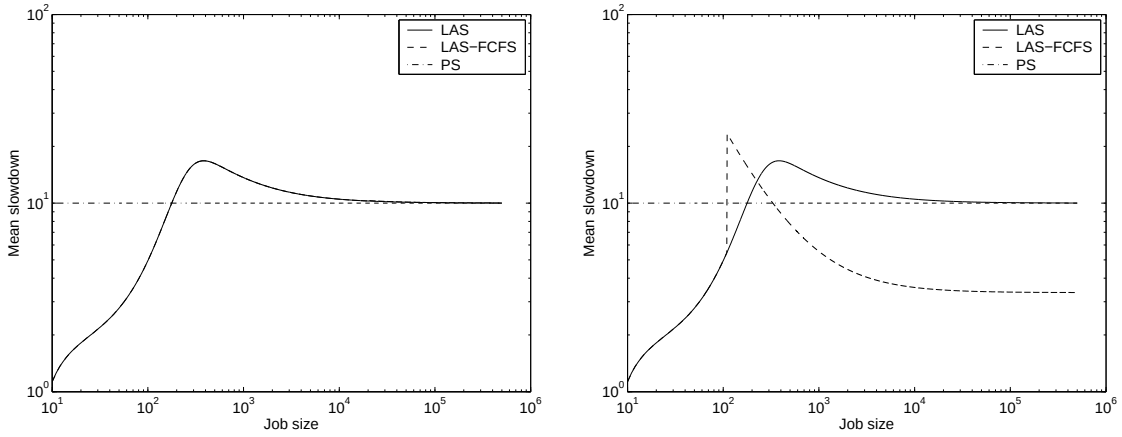
## 6.3 LAS-FCFS under overload

The partial stability of $LAS$ under overload was proven in Chapter 2. We showed that LAS is capable of providing some service to some jobs under overload. Since foreground jobs under LAS-FCFS have the same response times and slowdowns as under LAS, based on Theorem 2.4.1 in Chapter 2, we deduce that on average jobs in a foreground queue $j$ can receive service under overload if $\rho_{x_{\tau_j}} \leq 1$. On the other hand, jobs in the background queue experience, on average, an infinite response time even if all foreground jobs are serviced and $\rho_{\tau_{N-1}} \leq 1$. This is because the FCFS policy applied to the background queue becomes unstable. The instability of FCFS at background queue is due to the fact that when all foreground jobs receive service under overload condition, the load at the background queue $\rho - \rho_{\tau_{N-1}}$ is always greater than the remaining effective load $1 - \rho_{\tau_{N-1}}$ hence the instability of the background queue. When the background queue is unstable, its size keeps growing to infinity. This shows that the tuning of $\tau_{N-1}$ is important to make sure that small jobs receive service in the foreground queue so that they may as well completely receive service under LAS-FCFS in case of overload. The tuning may require a prior knowledge of the job size distribution. This may be a difficult task in practice and is out of scope of this thesis.

## 6.4 Numerical Evaluation of LAS-FCFS

In this section, we discuss numerical results of LAS-FCFS and compare it to $LAS$ and PS for empirical job sizes with high variability and low variability. Our objective is to evalu-

ate LAS-FCFS in terms of reducing the response time of the largest jobs while maintaining the response times of small jobs as low as under LAS scheduling. In particular we study the dependency of LAS-FCFS to the variability of a job size distribution in terms of minimizing the penalty received by the largest jobs under LAS. We use the bounded Pareto distribution $BP(10, 5{*}10^5, 1.1)$ as a typical example of high variability empirical job sizes and the exponential distribution to represent a job size distribution with no high variability property, both with a mean job size of 72.3. The $C$ value for the BP distribution is 19.8. $BP(10, 5*10^5, 1.1)$ distribution exhibits a high variability property as about 99% of the jobs have small sizes and less than 1% of the largest jobs constitute about 50% of the total load. The number of foreground queues used in numerical analysis is *N-1* $= 2000$ and the quantum size is $s_i = 10, \forall i \in \{1, 2, \cdots, N\}$. Hence, the amount of service required by a job that finishes service in the last foreground queue is $\tau_{N-1} = (N-1) \times s_i = 20000$.




(a) $\tau_{N-1} = 20000$ (b) $\tau_{N-1} = 100$

Figure 6.2: *Expected conditional slowdown for exponential job sizes as a function of job size at load* $\rho = 0.9$.

Figure 6.2 shows the mean slowdown of different job sizes at load $\rho = 0.9$ for the case of exponentially distributed job sizes for $\tau_{N-1}$ values of 20000 and 100. We first note that LAS-FCFS does not have any impact on the largest flows for large value of $\tau_{N-1} = 20000$. This is likely because the highest jobs with sizes greater than $\tau_{N-1}$ for exponential distribution constitute to a very small fraction of the total load. For smaller $\tau_{N-1}$, we see from Figure 6.2(b) that the largest jobs under LAS-FCFS experience lower mean slowdown than under PS. However, large values of $\tau_{N-1}$ will be preferred since small $\tau_{N-1}$ causes small jobs that receive no penalty under LAS to be penalized. Observe for the case of BP job sizes in Figure 6.3 that at large $\tau_{N-1}$ value of 20000, LAS-FCFS reduces the penalty for the largest jobs.

As noted in Equation (6.8), a foreground job under LAS-FCFS has the same response time (resp. slowdown) as under LAS. That is why, in the first part of the mean slowdown curves of LAS-FCFS shows identical mean slowdown as LAS. However, we note that all foreground jobs have a lower mean slowdown under LAS and LAS-FCFS than under PS for the BP distribution. For the exponential distribution, this is not the case particularly for large $\tau_{N-1}$ values.

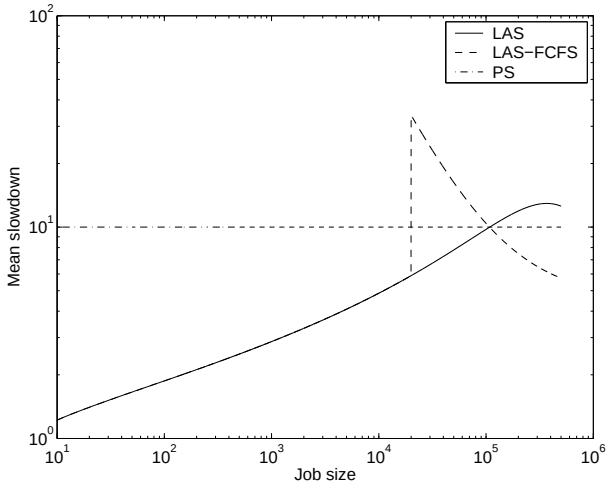
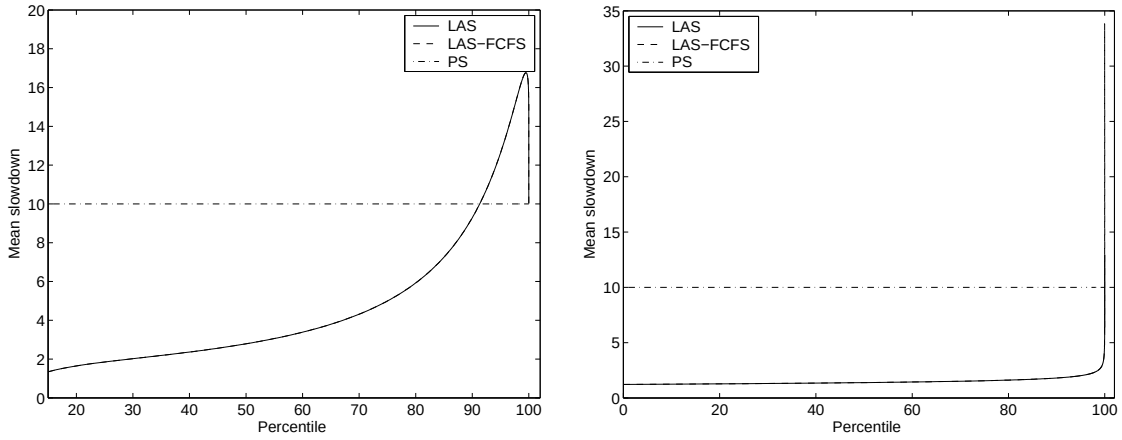



Figure 6.3: *Mean slowdown as a function of size for* $BP(10, 5 * 10^5, 1.1)$*,* $\rho = 0.9$*, and* $\tau_{N-1} = 20000$.

The curve for the mean slowdown of the background jobs shows two trends; where jobs return to the background queue a few times and jobs that return to the background queue many times. Depending on the size of $\tau_{N-1}$, the background jobs that return to the queue a few times have a moderate size and those that return many times are the largest jobs. We observe that the moderate jobs have a higher slowdown under LAS-FCFS than under LAS or PS, as they are no longer favored in the background queue. Instead, they are the smallest jobs in the background queue and thus are penalized by the FCFS policy in the background queue. The peak slowdown value experienced by jobs depends on the value of $\tau_{N-1}$ and on the load: the peak slowdown decreases for increasing $\tau_{N-1}$ or decreasing load $\rho$ values. If $\tau_{N-1}$ is small, the slowdowns (resp. response times) of the jobs in the background queue are affected by the service of a larger number of jobs in the background queue due to the FCFS scheduling order in the background queue. The dependence of the mean slowdown on the load may be explained by the physical system behavior where the interference among jobs in the background queue is smaller when the arrival rate is small than when it is high.

The jobs that return to the background queue many times are the largest jobs in the distribution. We observe from Figure 6.3 that for job sizes with a high variability property LAS-FCFS reduces the mean slowdown of these largest jobs. The performance of these jobs for the exponentially distributed job sizes highly depend on the value of $\tau_{N-1}$ and only undesirable small values of $\tau_{N-1}$ show impact.

We now analyze the percentage of the largest jobs that are penalized under LAS-FCFS and LAS for the BP and exponential job size distributions. Figures 6.4(a) and 6.4(b) show the slowdown of LAS, LAS-FCFS with $\tau_{N-1} = 20000$, and PS as a function of the percentiles of job size distributions considered . For the exponential distribution, we observe that about 15% of the jobs experience a higher slowdown under LAS and LAS-FCFS than under PS. In general, the percentage of jobs that receive a penalty for exponential job sizes is observed to be high and large values of $\tau_{N-1}$ in LAS-FCFS do not help to alleviate the penalty. Note that for the BP distribution, less than 1% of the largest jobs have a higher mean slowdown (but finite) under




(a) *Exponential distribution* (b) $BP(10, 5 * 10^5, 1.1)$

Figure 6.4: *Mean conditional slowdown as a function of percentile of job size distribution, at load* $\rho = 0.9$ *and* $\tau_{N-1} = 20000$.

LAS and LAS-FCFS than under PS. Figure 6.4(b) does not show these jobs as they are in the largest 1% of all jobs. Figure 6.5 shows a zoom of Figure 6.4(b) for the largest job sizes.

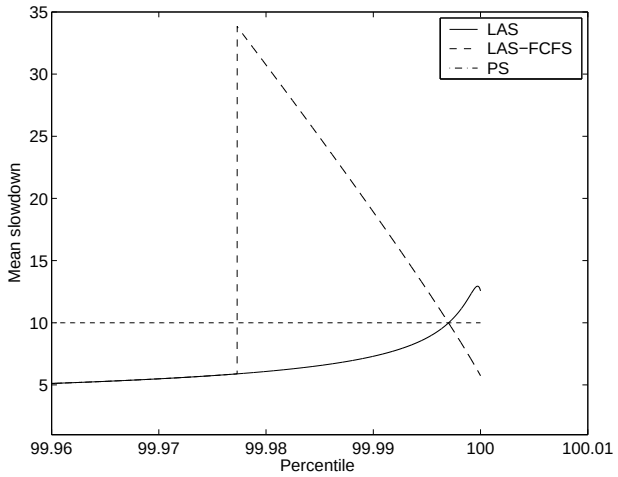



Figure 6.5: *Expected slowdown as a function of job size percentile for* $BP(10, 5 * 10^5, 1.1)$, $\tau_{N-1} = 20000$, *and load* $\rho = 0.9$.

To summarize this section, we observed that LAS-FCFS reduces the penalty to the largest jobs. For job size distributions that don't show a high variability property, e.g., exponential distribution, LAS-FCFS reduces the penalty of the largest jobs when $\tau_{N-1}$ is small. This however, leads to penalizing many intermediate size jobs, which is not desired.

Looking at Figure 6.5, we see that LAS-FCFS can prevent penalty of the largest jobs at the expense of penalizing a small percentage of other jobs of moderate size. This is true for any job size distribution with high variability property. Avoiding a penalty for the largest jobs is a good feature of LAS-FCFS because these jobs constitute a large fraction of the total load

for job size distributions with high variability property. Reader can argue whether improving the performance of very few largest jobs at the expense of penalizing moderate size jobs worths efforts. We see this as the setback of LAS-FCFS and since LAS is a work conserving policy, the conservation law (see [64]) states that improving the performance of some jobs (by changing their priority orders like in LAS-FCFS) yields to the performance of other jobs to deteriorate. This prompts us to proposing new variants of class-based LAS-FCFS policies to improve the performance of jobs of all sizes in a priority class.

## 6.5 Service Differentiation in LAS-FCFS

LAS-FCFS cannot differentiate jobs based on an attribute other than their size, and the jobs that enter the background queue a few times experience high response times under LAS-FCFS as load increases and receive no service at all under overload. In many networking environments, service differentiation to guarantee the QoS of important traffic is enforced by using some attributes such as protocol number, type of application, or user-assigned priorities. To achieve such a service differentiation, we propose variants of LAS-FCFS architecture for jobs that can classify the incoming jobs and differentiate their service.

These LAS-FCFS variants that we propose classify the incoming jobs into *high priority* and *low priority* jobs. Then, the high priority jobs are favored over the low priority jobs. The differential models proposed in this chapter differ from other differential models proposed for flows in Chapter 5 mainly because they don't use priority function to alter the priorities of priority jobs. We label the variables corresponding to high priority jobs by a subscript or superscript $H$ and the ones for the low priority jobs by a subscript or superscript $L$. The size of a high priority job is referred to $x_H$ and that of a low priority job to $x_L$. We assume that a high priority job arrives at the system with probability $q$ and a low priority job arrives with probability $1-q$. When $\lambda$ is the average arrival rate of jobs in the system, the average arrival rates of high and low priority jobs are then $\lambda_H = q\lambda$ and $\lambda_L = (1-q)\lambda$ respectively. We refer to Table 6.1 for a quick guidance on the used symbols and their meanings. That is, if $f(x)$ is the pdf of all job sizes, then $f(x) = f_L(x) = f_H(x)$. The load corresponding to high priority jobs of sizes less than or equal to $x$ and low priority jobs of sizes less than or equal to $x$ are $\rho_x^H = q\rho_x$ and $\rho_x^L = (1-q)\rho_x$ respectively. Similarly, we denote the system backlog due to high priority jobs as $W_o^H$ and the backlog due to low priority jobs as $W_o^L$. The expressions for $W_o^H$ and $W_o^L$ are the same as in Equation (6.1) with $\tau_{N-1} = \infty$, and with $\lambda$ and $\rho$ values corresponding to the class of the job. That is, $W_o^H = \frac{\lambda_H \overline{x^2}_\infty}{2(1-\rho_\infty^H)}$ and $W_o^L = \frac{\lambda_L \overline{x^2}_\infty}{2(1-\rho_\infty^L)}$.

In the next sections, we present two variants of LAS-FCFS. We derive the expressions for the conditional mean response times of foreground and background jobs with high priority and low priority. We present numerical results to compare the performance improvements in terms of reducing the mean response time for high priority jobs and evaluate the penalty for the low priority jobs. Finally we discuss the corresponding models for packet networks.

| Symbol | Meaning |
|---|---|
| $x_H$ | size of a high priority job |
| $x_L$ | size of a low priority job |
| $q$ | fraction of the high priority jobs |
| $\lambda$ | average jobs arrival rate |
| $\lambda_H$ | $q\lambda$ |
| $\lambda_L$ | $(1-q)\lambda$ |
| $\overline{x_x^n}$ | Equation (5.1) |
| $\overline{x_\infty}$ | first moment ($m_1$) |
| $\overline{x_\infty^2}$ | second moment ($m_2$) |
| $\rho_x$ | Equation (5.2) |
| $\rho_\infty$ | total load, $\rho$ |
| $\rho_x^H$ | $q\rho_x$ |
| $\rho_x^L$ | $(1-q)\rho_x$ |
| $W_o^H$ | backlog experienced by high priority jobs |
| $W_o^L$ | backlog experienced by low priority jobs |

Table 6.1: *Symbols used in differentiated LAS-FCFS policies*

### 6.5.1 Fixed Priory LAS-FCFS Architecture

We first propose a *fixed priority* LAS-FCFS scheduling policy (*FP-LAS-FCFS*), which classifies the incoming job to high priority and low priority, and then forwards each class to a separate LAS-FCFS. Thus, the jobs in each class are serviced in LAS-FCFS order. FP-LAS-FCFS gives a fixed priority to jobs in the high priority class and the low priority jobs are serviced only if there are no high priority jobs waiting, i.e., all queues in the LAS-FCFS model that corresponds to high priority class are empty. Moreover, the low priority service is *preempted* on the arrival of the high priority jobs. Figure 6.6 shows the fixed priority LAS-FCFS architecture. For simplicity in analysis, we assume that the number of queues, quantum values, and the values of $\tau_{N-1}$ used in either LAS-FCFS systems in FP-LAS-FCFS are the same.

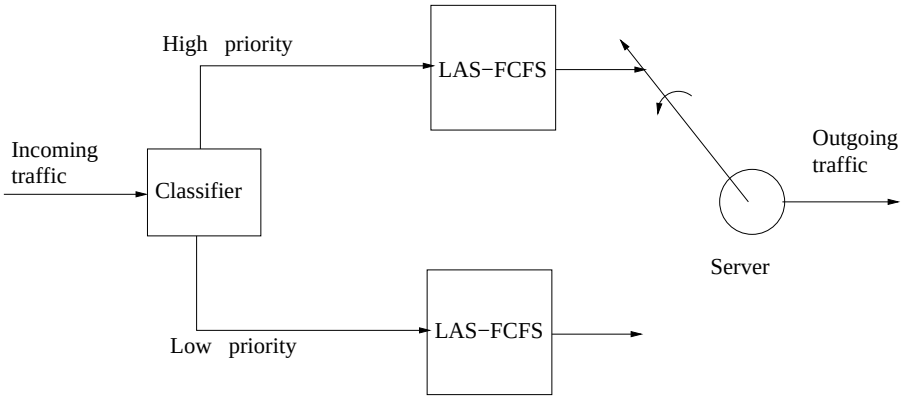



Figure 6.6: *Fixed priority LAS-FCFS architecture.*

The fixed priority LAS-FCFS scheduling policy improves the service of high priority jobs

by avoiding the interruptions of the service of the high priority background jobs due to low priority jobs in the foreground queue. In the following sections, we compute the expressions of the conditional mean response times for jobs with different priorities.

**High priority foreground jobs**

The conditional mean response time of high priority foreground jobs under fixed priority LAS-FCFS is the same as its conditional mean response time under LAS-FCFS with the mean arrival rate $\lambda = \lambda_H$ and load $\rho = \rho^H_\infty$. Hence, from Equation (6.3), we get the expression for conditional mean response time for a job size $\tau_j$, $\forall j \in \{1, ..., N-1\}$ under FP-LAS-FCFS ($T(x|\tau_j \leq \tau_{N-1})_{FP\text{-}LAS\text{-}FCFS}$) as:

$$T(\tau_j|\tau_j \leq \tau_{N-1}) = \frac{W^H_o(\tau_j) - s_j\rho^H_{\tau_{j-1}}}{(1-\rho^H_{\tau_{j-1}})} + \frac{\tau_j}{(1-\rho^H_{\tau_{j-1}})}, \tag{6.9}$$

where $W^H_o(\tau_j) = \frac{\lambda_H \overline{x^2_{\tau_j}}}{2(1-\rho^H_{\tau_j})}$, which is the system backlog due to the high priority jobs in the system that delays the high priority foreground job of size $\tau_j$.

**High priority background jobs**

The response time of a high priority background job under fixed priority LAS-FCFS is the same as the response time in an isolated LAS-FCFS policy with the mean arrival rate $\lambda_H$ at load $\rho^H_x$. Hence, the expression for the mean response time of the job size $x_H$ under fixed priority LAS-FCFS is easily derived from Equation (6.7) as:

$$T(x_H|x_H > \tau_{N-1}) = \frac{W^H_o - s_N\rho^H_{\tau_{N-1}}}{(1-\rho^H_{\tau_{N-1}})} + \frac{x_H}{(1-\rho^H_{\tau_{N-1}})}.$$

Figure 6.7 shows the mean slowdowns of the high priority jobs with mean arrival rate $\lambda_H = 0.3\lambda$ under fixed priority LAS-FCFS at load $\rho = 0.9$ and $\rho = 0.5$. We see from the figure that fixed priority LAS-FCFS significantly reduces the slowdown of the high priority jobs. We note that even at load $\rho = 0.9$, the mean slowdown of high priority jobs under fixed priority LAS-FCFS is far below their mean slowdown under PS. Observe also that a reasonable mean arrival rate of the high priority jobs also guarantees that the high priority jobs will continue to receive service under overload. In particular, the high priority jobs receive service under overload as along as $\rho^H_\infty = q\rho < 1$.

**Low priority foreground jobs**

Assume an isolated low priority LAS-FCFS system. The mean waiting time of the low priority background job ($x_L$) in this isolated system is the same as its mean waiting time in an LAS-FCFS system with mean arrival rate and load of $\lambda_L$ and $\rho^L_x$ respectively. We denote this waiting

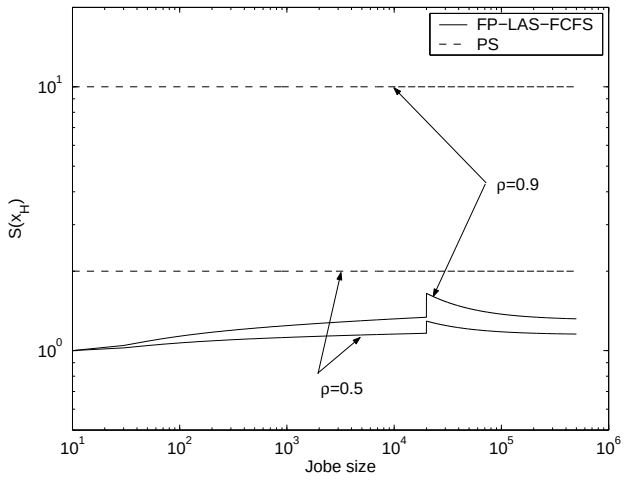



Figure 6.7: *Expected slowdown of high priority jobs under fixed priority LAS-FCFS as a function of job size for* $BP(10, 5 * 10^5, 1.1)$, $\tau_{N-1} = 20000$, *q = 0.3, and load* $\rho = 0.9$.

time by $W(x_L)$. In the fixed priority LAS-FCFS policy however, the low priority foreground job will be further delayed by the service of the backlog that it finds in the high priority LAS-FCFS system upon its arrival $W_o^H$, the service of new arrivals of the high priority jobs while the low priority job is in the system $W_s(x_H)$, and its service time $x_L$. Hence,

$$T(x_L|x_L \leq \tau_{N-1}) = W_o^H + W(x_L) + (W_s(x_H) + x_L).$$

The expressions for $W_o^H$ and $W(x_L)$ are given in Equation (6.1) for $\tau_j = \infty$ and $\rho_{\tau_j} = \rho_\infty^H$ and Equation (6.2) for $\lambda = \lambda_L$ and $\rho_{\tau_{j-1}} = \rho_{\tau_{j-1}}^L$ respectively. Similarly, $W_s(x_H)$ is given by Definition 6.6 as $W_s(x_H) = (T(x_L|x_L \leq \tau_{N-1}) - s_N)\rho_\infty^H)$. Then,

$$T(x_L|x_L \leq \tau_{N-1}) = W_o^H + W(x_L) + x_L + T(x_L|x_L \leq \tau_{N-1})\rho_\infty^H - s_N\rho_\infty^H,$$

after some algebra, we obtain $T(x_L|\tau_{N-1} \geq x_L)$ as:

$$T(x_L|x_L \leq \tau_{N-1}) = \frac{W_o^H + W(x_L)}{(1-\rho_\infty^H)} + \frac{x_L - s_N\rho_\infty^H}{(1-\rho_\infty^H)}. \tag{6.10}$$

**Low priority background jobs**

Finally, a low priority background job is delayed by the system backlog due to the high priority jobs that it finds in the system upon arrival, $W_o^H$, the service of new arrivals of high priority jobs $W_s(x_H)$, and its service time $x_L$. In addition, the job will wait in the system due to its waiting time in a low priority LAS-FCFS system assuming that it is isolated $W_B(x_L)$. That is,

$$T(x_L|x_L > \tau_{N-1}) = W_o^H + W_B(x_L) + W_s(x_H) + x_L.$$

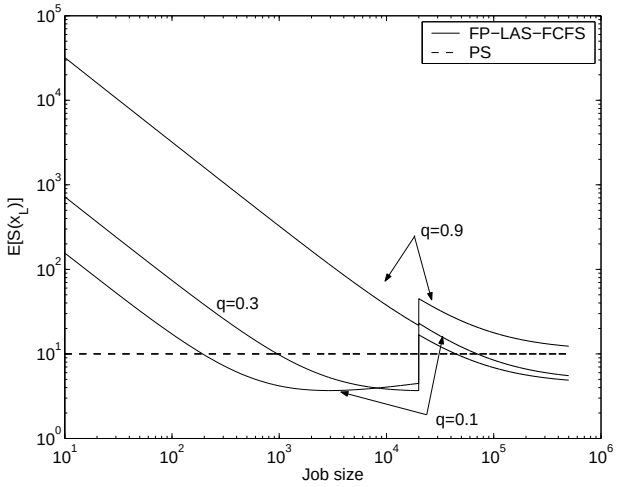



Figure 6.8: *Expected slowdown of low priority jobs under fixed priority LAS-FCFS as a function of job size for* $BP(10, 5 * 10^5, 1.1)$*,* $\tau_{N-1} = 20000$*, and load* $\rho = 0.9$

The expression for $W_B(x_L)$ is given in Equation (6.5) for $W_o = W_o^L$ and $\rho = \rho^L_{\tau_{N-1}}$ and $W_s(x_H)$ is given by Equation (6.6) as $(T(x_L|x_L > \tau_{N-1}) - s_N)\rho^H_\infty$. Hence, the expression for $T(x_L|x_L > \tau_{N-1})$ is given as:

$$T(x_L|x_L > \tau_{N-1}) = W_o^H + W_B(x_L) + T(x_L|x_L > \tau_{N-1})\rho^H_\infty - (s_N\rho^H_\infty - x_L),$$

after some algebra, we obtain:

$$T(x_L|x_L > \tau_{N-1}) = \frac{W_o^H + W_B(x_L)}{(1-\rho^H_\infty)} + \frac{x_L - s_N\rho^H_\infty}{(1-\rho^H_\infty)}. \tag{6.11}$$

Figure 6.8 shows the mean slowdown of the low priority jobs under fixed priority LAS-FCFS for different $q$ values at load $\rho = 0.9$. We observe from the figure that the mean slowdown of the low priority small jobs under fixed priority LAS-FCFS is quite high and increases in increasing the mean arrival rate of high priority jobs ($\lambda_H = q\lambda$). Note that the low priority jobs experience the minimum mean response time under fixed priority LAS-FCFS when $q \approx 0$, i.e., there are only low priority jobs in the system. Hence, the minimum possible mean response time of low priority jobs under fixed priority LAS-FCFS is the same as under LAS-FCFS. Thus, the service of the high priority jobs under fixed priority LAS-FCFS comes at the expense of a high penalty to small jobs with low priority.

The proposed fixed priority LAS-FCFS scheduling policy uses separate LAS-FCFS scheduling to service high priority and low priority jobs. LAS-FCFS is more useful if we want to improve the performance of the largest jobs in addition to reducing the response time for short jobs. In a practical system however, the fraction of the largest priority jobs must be reasonably small to avoid starvation for low priority jobs. Moreover, in most cases, users will choose to pay for a better service only for long jobs. In the light of these facts, one can easily see that their may not be a need to use a separate LAS-FCFS scheduling to service priority jobs. Thus, we propose the extension of fixed priority LAS-FCFS that uses FCFS to service high priority

jobs and LAS-FCFS to service low priority jobs, which are serviced only when the FCFS queue for priority jobs is empty. We simply call this policy the extension of FP-LAS-FCFS because like FP-LAS-FCFS, it uses fixed priority to service low priority jobs only if the queue of high priority jobs is idle. However, we sometimes call this policy a fixed priority **FCFS-LAS-FCFS**. The expression for the mean response time of high priority jobs in this fixed priority extension is thus simply their mean response time under FCFS policy, i.e.,

$$T(x_H) = Wo^H + x_H \tag{6.12}$$

The expression for the mean response times for low priority jobs remain the same as Equations (6.10) and (6.11).

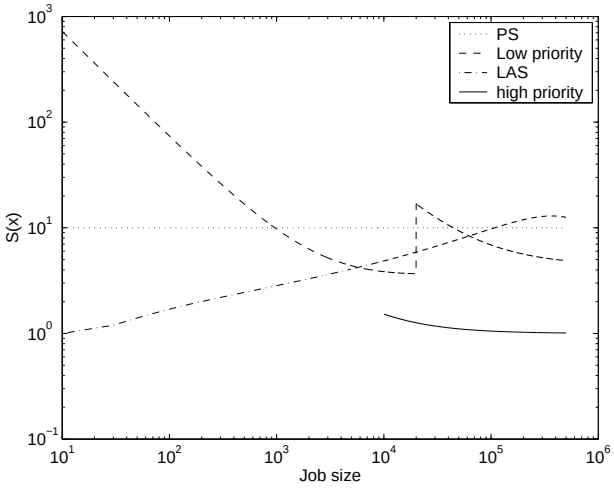



Figure 6.9: *The performance of extended FP-LAS-FCFS for* $BP(10, 5 * 10^5, 1.1)$, $q = 0.3$, $\tau_{N-1} = 20000$, *and load* $\rho = 0.9$.

Figure 6.9 shows the mean slowdown of high and low priority jobs under the extended fixed priority LAS-FCFS policy. Recall that we consider high priority jobs to be always large jobs. Observe the performance improvement for high priority jobs. Due to FCFS for background jobs of low priority, we can observe that the background low priority jobs experience very low slowdown than under plain LAS.

### 6.5.2 Differential LAS-FCFS

Figure 6.4(b) shows that if the job size distribution exhibits high variability, more than 99% of jobs have a lower slowdown under LAS-FCFS than under PS. Thus, without service differentiation, all small jobs under the LAS-FCFS policy experience a low mean response time or mean slowdown. It is mainly the background jobs that are penalized under plain LAS-FCFS. In previous section, we saw that fixed priority LAS-FCFS scheduling significantly improves the service of high priority jobs but can cause a heavy penalty to small low priority jobs. This may affect the overall performance if the job sizes exhibit the high variability property with majority of jobs being small. In this section, we propose and analyze another variant of LAS-FCFS that

we call *Differential* LAS-FCFS (*DF-LAS-FCFS*). The objective of this policy is to improve the mean response time of high priority background jobs while maintaining the response time of all small jobs as low as under LAS-FCFS. To this end, differential LAS-FCFS services all small jobs in foreground queues, and classifies background jobs into low priority and high priority background jobs that are serviced in separate background queues, see Figure 6.10. Low priority background jobs are serviced in a low priority background queue only if all foreground queues and the high priority background queue are empty, whereas the high priority background jobs receive service once all foreground queues are empty. In the following sections, we analyze the differential LAS-FCFS.

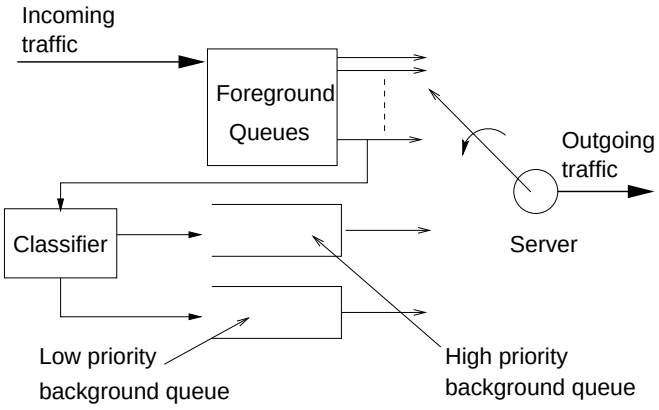



Figure 6.10: *Differential LAS-FCFS architecture.*

**Foreground jobs**

The conditional mean response time of a foreground job (high priority or low priority) under differential LAS-FCFS is the same as the conditional mean response time under LAS-FCFS with the same mean arrival rate. Hence, the formula for the mean response time of a foreground job that completes service in queue $j$, $j \in \{1, ..., N-1\}$ ($T(\tau_j|\tau_j \leq \tau_{N-1})$) is the same as Equation (6.3) with appropriate mean arrival rate $\lambda$ and load $\rho$ values.

**High priority background jobs**

Now, we compute the expression for the mean response time of high priority background jobs ($T(x_H|x_H > \tau_{N-1})$). A high priority background job is delayed in the queue due to the service of the system backlog of the high priority jobs that it finds in the system $W_o^H$, the service of the backlog of low priority jobs in the foreground queues $W_o^L(\tau_{N-1})$, and its own service $x_H$. In addition, the job is delayed by the service of the newly arriving jobs in foreground queues by $W_s(\tau_{N-1})$. That is,

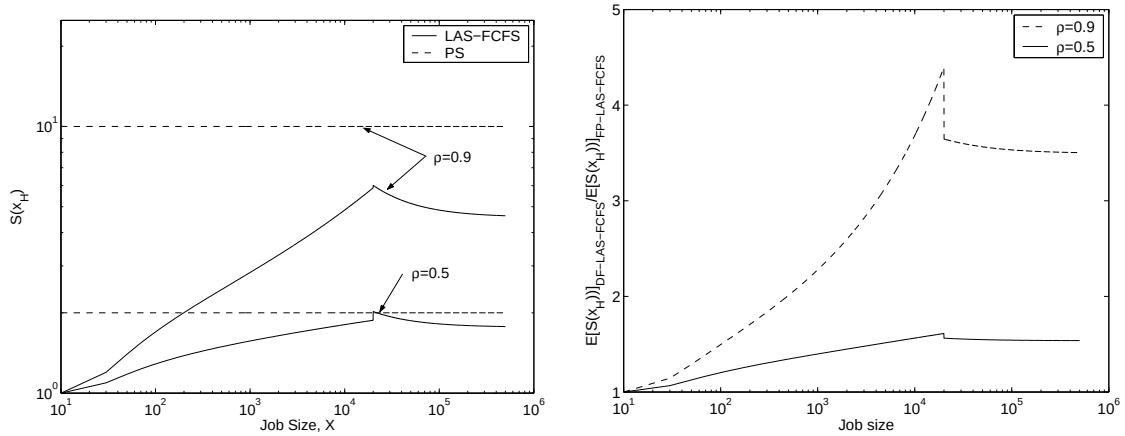
$$T(x_H|x_H > \tau_{N-1}) = W_o^H + W_o^L(\tau_{N-1}) + W_s(\tau_{N-1}) + x_H.$$

Definition 6.6 gives the expression of $W_s(\tau_{N-1})$ as $(T(x_H|x_H > \tau_{N-1}) - s_N)\rho_{\tau_{N-1}}$ and the expression for $W_o^L(\tau_{N-1})$ is given from Equation (6.1) for $\tau_j = \tau_{N-1}$ and $\lambda = \lambda_L$. Hence, $T(x_H|x_H > \tau_{N-1})$ is given as:

$$T(x_H|x_H > \tau_{N-1}) = W_o^H + W_o^L(\tau_{N-1}) + T(x_H|x_H > \tau_{N-1})\rho_{\tau_{N-1}} - s_N\rho_{\tau_{N-1}} + x_H,$$

after some algebra, we get:

$$T(x_H|x_H > \tau_{N-1}) = \frac{W_o^H + W_o^L(\tau_{N-1})}{(1-\rho_{\tau_{N-1}})} + \frac{x_H - s_N\rho_{\tau_{N-1}}}{(1-\rho_{\tau_{N-1}})}. \tag{6.13}$$




(a) $S(x_H)$ (b) $\frac{S(x_H)_{DF-LAS-FCFS}}{S(x_H)_{FP\text{-}LAS\text{-}FCFS}}$

Figure 6.11: *The performance of DF-LAS-FCFS for* $BP(10, 5 * 10^5, 1.1)$ *as a function of job size, at* $q = 0.3$ *and* $\tau_{N-1} = 20000$.

In the differential LAS-FCFS architecture, all arriving jobs to the foreground queues interrupt the service of high priority background jobs. However, when the job size distribution has a high variance, the load constituted by these small jobs is small, less than half of the total load. Hence, differentiating the service of only background jobs has a positive impact on the the mean response time of high priority background jobs. Figure 6.11(a) compares the mean slowdown of high priority jobs $S(x_H)$ under differential LAS-FCFS and PS for the BP distribution at load $\rho = 0.5$ and $\rho = 0.9$, and $q = 0.3$. We see that for both considered load values, differential LAS-FCFS offers a lower mean slowdown than PS for all high priority jobs. Figure 6.11(b) shows the ratio of the mean slowdown of high priority jobs under differential LAS-FCFS to their mean slowdown under LAS-FCFS. We observe that $\frac{S(x_H)_{DF\text{-}LAS\text{-}FCFS}}{S(x_H)_{FP\text{-}LAS\text{-}FCFS}}$ reaches as high as above 4 at load $\rho = 0.9$, which means that the maximum conditional mean slowdown under differential LAS-FCFS is 4 times higher than the conditional mean slowdown under fixed priority LAS-FCFS. The maximum value of the ratio is quite low, about 1.5 for load $\rho = 0.5$. The performance difference between differential LAS-FCFS and fixed priority LAS-FCFS in terms of reducing the mean response time of high priority jobs is not very significant. And this is accounted by the fact that fixed priority LAS-FCFS maintains twice as much as the number of queues as differential LAS-FCFS.

**Low priority background jobs**

Finally, the mean response time of a low priority background job $T(x_L|\tau_{N-1} < x_L)$ is a result of its waiting time due to the backlog that it finds in the system upon its arrival $W_o$ , its service time $x_L$, and the average waiting time due to service interruptions of newly arriving jobs. The service of these newly arriving jobs that affect the response time of the low priority background job are the service of the low priority jobs in the foreground queues $W_s^L(\tau_{N-1})$ and the service of all new arrivals of high priority jobs $W_s^H(\infty)$. That is,

$$T(x_L|\tau_{N-1} < x_L) = W_o + W_s^H(\infty) + W_s^L(\tau_{N-1}) + x_L.$$

Definition 6.6 gives the expressions for $W_s^H(\infty)$ and $W_s^L(\tau_{N-1})$ as $(T(x_L|x_L > \tau_{N-1}) - s_N)\rho_\infty^H$ and $(T(x_L|x_L > \tau_{N-1}) - s_N)\rho_{\tau_{N-1}}^L$ respectively. Hence,

$$\begin{aligned} T(x_L|\tau_{N-1} < x_L) &= W_o + x_L + T(x_L|x_L > \tau_{N-1})\rho_\infty^H + T(x_L|x_L > \tau_{N-1})\rho_{\tau_{N-1}}^L \\ &- (s_N\rho_\infty^H + s_N\rho_{\tau_{N-1}}^L), \end{aligned}$$

simplifying the above equation, we get

$$T(x_L|x_L > \tau_{N-1}) = \frac{W_o - s_N(\rho_\infty^H + \rho_{\tau_{N-1}}^L)}{(1 - \rho_\infty^H - \rho_{\tau_{N-1}}^L)} + \frac{x_L}{(1 - \rho_\infty^H - \rho_{\tau_{N-1}}^L)}.$$

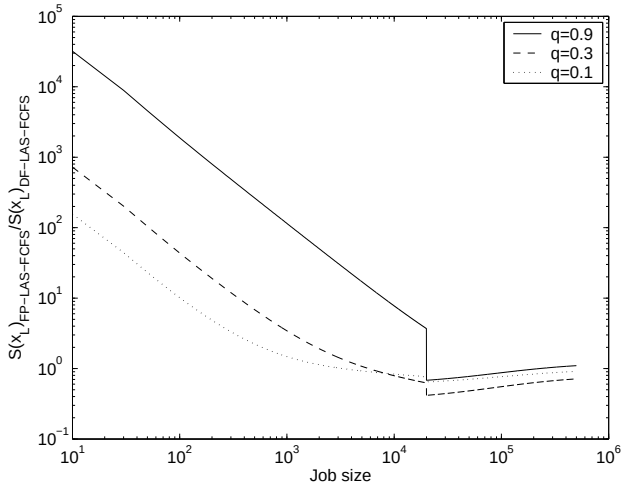



Figure 6.12: $\frac{S(x_L)_{FP-LAS-FCFS}}{S(x_L)_{DF-LAS-FCFS}}$ for $BP(10, 5*10^5, 1.1)$ as a function of job size, at load $\rho = 0.9$.

We now look at the improvement of differential LAS-FCFS over fixed priority LAS-FCFS in terms of reducing the mean response time of low priority jobs for the BP distribution considered. This is shown in Figure 6.12 where we plot the ratio of the mean slowdown between the policies. We see that, while there is no big difference between the mean slowdown of low priority background jobs offered by both policies, differential LAS-FCFS significantly reduces the mean slowdown of short jobs as compared to fixed priority LAS-FCFS. Looking at Figures

6.3, 6.8, and 6.12, we can see that the mean slowdown of the background low priority jobs under differential LAS-FCFS is also similar to under plain LAS-FCFS. Moreover, under overload, high priority background jobs under differential LAS-FCFS receive service as long as the load due to the service received in foreground queues is less than 1, which is the case by our definition of highly varying job sizes. Therefore, we conclude that differential LAS-FCFS is a more suitable policy than a fixed priority LAS-FCFS with respect to improving the mean response times of all jobs in the system.

## 6.6 Network Models

### 6.6.1 Model for LAS-FCFS

In Chapter 5, we showed that LAS model that estimates the transfer times of flows in packet networks is the same as the analytical model of LAS for jobs. Similarly, we showed that the model of FCFS in packet networks is well modeled by PS. The model for LAS-FCFS is thus derived using the models of LAS and FCFS taking into consideration the fixed priority for foreground flows under LAS-FCFS. Based on our findings in Chapter 5, the model for transfer times of foreground flows under LAS-FCFS is identical to the model of flows of similar sizes under LAS. On the other hand, the mean transfer times of background flows can be modeled by using the processor sharing (PS) in multilevel scheduling where PS is not the policy at the first level. In general, the analytical model of LAS-FCFS can be seen as a hybrid model of LAS and PS. For this reason, we denote the network model of LAS-FCFS in this thesis work as **LAS($\tau_{N-1}$)-PS**, where $\tau_{N-1}$ is the maximum service given for the foreground flows. Using the priority change notion used in Chapter 5, the network model of LAS($\tau_{N-1}$)-PS has a non-linear priority function $P(x)$ for a packet $x$ as:

$$P(x) = \begin{cases} x & \text{if } x \leq \tau_{N-1} \\ \\ \tau_{N-1} + 1 & \text{if } x > \tau_{N-1} \end{cases} \tag{6.14}$$

The integral expression of $T(x)$ for LAS($\tau_{N-1}$)-PS is obtained using Equations (4.27), (4.36), and (4.39) in [64] and is given in Equation (6.15) below. Let us denote the mean transfer times for a flow of size $x$ under LAS as $T_{LAS}(x)$. LAS($\tau_{N-1}$)-PS network model is then given as:

$$T_{\text{LAS-FCFS}}(x) = \begin{cases} T_{LAS}(x) & \text{if } x \leq \tau_{N-1}, \\ \\ T_{LAS}(\tau_{N-1}) + \frac{\alpha(x-\tau_{N-1})}{(1-\rho_{\tau_{N-1}})} & \text{if } x > \tau_{N-1}. \end{cases} \tag{6.15}$$

Note that $T_{LAS}(\tau_{N-1})$ is the mean duration a background job takes in the foreground queues in order to arrive in the background queue. Given that $b$ is the average number of flows that arrive in addition to the tagged job $b + 1 = E[\tau_{N-1}^2]/E[\tau_{N-1}]$, $\alpha(x)$ satisfies the following integral

equation:

$$\begin{aligned}\alpha'(x) &= \frac{\lambda}{(1-\rho_{\tau_{N-1}})}\int_0^x \alpha'(y)F^c(\tau_{N-1}+x-y)dy \\ &\quad + \frac{\lambda}{(1-\rho_{\tau_{N-1}})}\int_0^\infty \alpha'(y)F^c(\tau_{N-1}+x+y)dy + bF^c(x) + 1 \qquad (6.16)\end{aligned}$$

where $\alpha'(x) = \frac{d}{dx}\alpha(x)$. Equation (6.16) is known to be the conditional response time for a job of size $x$ in an M/G/1/PS system with batch arrivals. Note that $\frac{\alpha(x-\tau_{N-1})}{(1-\rho_{\tau_{N-1}})}$ in Equation (6.15) is the mean time elapsed when a flow of size $x$ is in service in the FCFS background queue.

In [9], it is shown that Equation (6.16) can iteratively be solved using the fixed point iteration method. However, this method is complex and time consuming in particular for flow size distributions with the high variability property. It is thus recommended in [9] to use the derived upper bounds, which are shown to have a good accuracy for the performance of the background jobs when the job size distribution has the high variability property, and in particular for large flows, The expression for the upper bound of $\alpha(x)$ is as follows:

$$\alpha(x) \leq \frac{x}{1-\rho} + \frac{b\overline{x}(2-\rho)}{2(1-\rho)^2}. \qquad (6.17)$$

Finally, the implementation of LAS-FCFS in packet networks is similar to the implementation of LAS with a single priority queue. The only difference is that under LAS-FCFS only packets from foreground flows need to be assigned priorities and to be inserted to their positions in the queue based on their priorities. On the other hand, arriving packets from background flows are simply at the tail of the queue. Thus, compared to LAS, the LAS-FCFS implementation requires to assign priorities to fewer packets, to maintain a smaller number of priority levels, and to keep track of smaller number of flows than plain LAS. In [72], it is reported that the number of priority levels is one of the bottlenecks in the implementation of a priority scheduling policy. Some priority scheduling architectures do not scale if the number of priority levels is very large. Using this fact, we can see that LAS-FCFS is more scalable than plain LAS since the number priority levels is limited, e.g., at most $\tau_{N-1}$ priority levels for each active flow if the threshold in LAS-FCFS is set to $\tau_{N-1}$ packets.

### 6.6.2 Models for Variants of LAS-FCFS

We next consider the models for fixed priority FCFS-LAS-FCFS policy and differential LAS-FCFS and present the integral expressions of the models. The network model of this fixed priority FCFS-LAS-FCFS is close to the model of LAS-fixed(k) when all packets from priority flows are given the highest priority possible, e.g., $k = 1$. The difference between the two models is that the priority of the largest low priority flows under the extended fixed priority LAS-FCFS model is fixed to $k = \tau_{N-1} + 1$. We also note that the extended FP-LAS-FCFS is a multilevel scheduling policy with three levels: the first level is for high priority flows serviced under FCFS, this level is easily modeled by simple PS; the second level is for low priority

foreground flows that are serviced using LAS, thus we can model this level by LAS; the third level can be modeled as a processor sharing with batch arrivals. In general, the expression of the mean response time model $T(x)$ for the extended fixed priority model is as follows:

$$T(x) = \begin{cases} \frac{x}{(1-\rho_{\infty}^{H})} & \forall\ x \in \{x_H\}, \\ T_{LAS}(x_L|x_L \leq \tau_{N-1}) & \{x : x = x_L \leq \tau_{N-1}\}, \\ T_{LAS}(x_L|x_L = \tau_{N-1}) + \frac{\alpha(x_L - \tau_{N-1})}{(1-\rho_{\tau_{N-1}}^{L} - \rho_{\infty}^{H})} & \{x : x = x_L > \tau_{N-1}\}. \end{cases} \quad (6.18)$$

where the second and the third lines of Equation (6.18) are obtained from Equation (6.10) and the expression for $\alpha(x)$ satisfies the integral equation of Equation (6.16) with $(1 - \rho_{\tau_{N-1}})$ replaced by $(1 - \rho_{\tau_{N-1}}^{L} - \rho_{\infty}^{H})$.

The network model for DF-LAS-FCFS is similar to LAS-fixed(k) where $k = \tau_{N-1} + 1$ for high priority background jobs. The difference is that the priorities of packets background low priority jobs is also fixed at $\tau_{N+1} + 2$. Differential LAS-FCFS is a multilevel scheduling policy with the first level using LAS scheduling for all jobs until they receive $\tau_{N-1}$, the second and the third levels use FCFS for high priority and low priority background flows, these levels can be modeled using separate bath processor sharing. The expression of mean response time $T(x)$ for the DF-LAS-FCFS model is given as:

$$T(x) = \begin{cases} T_{LAS}(x) & x \leq \tau_{N-1}, \\ T_{LAS}(\tau_{N-1}) + \frac{\alpha(x_H - \tau_{N-1})}{(1-\rho_{\tau_{N-1}})} & \{x : x = x_H > \tau_{N-1}\}, \\ T_{LAS}(x_H|x_H = P) + \frac{\alpha(x_L - \tau_{N-1})}{(1-\rho_{\tau_{N-1}}^{L} - \rho_{\infty}^{H})} & \{x : x = x_L > \tau_{N-1} + 1\}. \end{cases} \quad (6.19)$$

where the expression for $\alpha(x)$ satisfies the integral equation of Equation (6.16) with $(1 - \rho_{\tau - N})$ replaced by $(1 - \rho_{\tau_{N-1}}^{L} - \rho_{\infty}^{H})$ and $T(x_H|x_H = P)$, the mean duration it takes for a low priority background job to arrive at the background queue, can be obtained from Equation (6.13).

## 6.7 Conclusion

This chapter presents and numerically analyzes models of LAS-FCFS scheduling for job and network flows, and proposes variants of LAS-FCFS to offer service differentiation. LAS-FCFS is considered to reduce the penalty experienced by the largest flows under plain LAS scheduling. We observe that, like LAS, the performance of LAS-FCFS depends of the variability of a job size distribution. In particular, LAS-FCFS reduces the penalty of the largest jobs more when a job size distribution has high variability property.

Since LAS-FCFS cannot differentiate the services of jobs based on attributes other than their sizes and its ability to service jobs under overload is limited to foreground queues only, it can-

not guarantee low response time for important jobs or users (particularly with large job sizes). Thus, we propose and analyze two variants of LAS-FCFS that classify the incoming jobs, before servicing them, into high and low priority based on any desired attributes. These policies are referred to as *fixed priority* LAS-FCFS ( *FP-LAS-FCFS*) and *differential* LAS-FCFS (*DF-LAS-FCFS*). Numerical results conducted for empirical job sizes with a high variability show that fixed priority LAS-FCFS offers absolute guarantees to the response time of high priority jobs at the expense of a high penalty for low priority small or intermediate size jobs. Similarly, differential LAS-FCFS also guarantees the service of the high priority jobs. In addition, differential LAS-FCFS maintains the mean response time of the low priority foreground jobs as low as the mean response time under LAS-FCFS. In contrast to LAS-FCFS, both differential LAS-FCFS and fixed priority LAS-FCFS can also guarantee the service of the high priority jobs under overload at any reasonable mean arrival rate of the high priority jobs. Finally, the network models of LAS-FCFS and of its differentiated variants that are analyzed in this chapter have potential to reduce delay of long-lived flows and to offer service differentiation among flows of different priorities.

# Chapter 7

# Thesis Summary and Outlook

The existing Internet does not provide any mechanisms that allow for a service differentiation. However different applications may have different requirements such as low response time, guaranteed data rates, jitter, or loss rates. To this end, various QoS architectures have been proposed but most of them can not be deployed in the Internet today due to mainly complex signaling protocols required for their wide implementations. Instead, Internet still supports only a best-effort service and the need for QoS still persists. With this in mind, this thesis proposes a QoS solution to improve the user perceived performance in terms of delay, packet loss rate, jitter, throughput, etc. The proposed solution in this thesis differs from most of the previously proposed architectures in that it only needs a limited deployment in bottleneck links.

The QoS solution proposed in this thesis involves using the least attained service (LAS) scheduling in bottleneck nodes and servers (e.g. Web servers) to service flows of packets. LAS is a priority based discipline that services packets of a flow according to the amount of service received by the flow. The less the attained service by the flow the higher the priorities of its packets. The choice of LAS scheduling as a solution to QoS problem in Internet is motivated by the observed high variability property of flow sizes. Under this traffic characteristics, TCP networks with FIFO scheduling are known to degrade the overall network performance in terms of mean delay and loss rate. In what follows, we present a summary of the results established in this thesis.

We evaluated LAS scheduling analytically using job size distributions with varying degrees of variability. We compared the performance of LAS to a wide range of other scheduling policies to study its performance improvements for job sizes. The numerical results showed that the performance of LAS highly depends on the variability of a job size distribution. In particular, we saw that LAS significantly reduces the mean response time of short jobs for any job size distribution, and it negligibly penalizes a tiny fraction of the largest jobs for job sizes with a high variability property. Moreover, we proved that the penalty experienced by the largest jobs is moderate. The comparison of LAS to SRPT showed that LAS performs very close to SRPT and its comparison with FIFO showed that LAS yields a lower mean response time for job size distributions with high variability property.

We studied the interaction of LAS to network protocols such as TCP and UDP and evaluated its performance when scheduling flows in packet networks. We found that LAS has interesting features that make it very suitable to reduce the transfer time of short TCP flows without highly penalizing the largest flows. This is a result of inherent LAS behaviors such as reducing the RTT of the first packets of flows during slow start and avoiding losses to the early packets of flows that arrive to a full buffer. We simulated LAS in packet networks and compared it to FIFO scheduling. Simulation results revealed that compared to FIFO, LAS reduces the mean transfer time and loss rates of short flows, and the performance of long flows under LAS and under FIFO is similar. This improves the mean user perceived performance for traffic with high variability property. We evaluated the performance of long-lived flows that compete against many short flows under LAS. We saw that LAS can degrade the performance of long-lived flows only at very high load close to overload.

We also studied the performance of LAS in congested heterogeneous networks such as TCP networks with heterogeneous propagation delays, networks that simultaneously support UDP and TCP applications, and networks with multiple congested links. The TCP protocols in these heterogeneous networks with FIFO schedulers are known to permit some sources to occupy a large fraction of available bandwidth to the extent of starving other competing sources. This is also known as *bandwidth hogging*. We found that LAS in heterogeneous networks avoids unequal bandwidth sharing among competing connections. Simulation results showed that LAS can avoid bandwidth hogging in heterogeneous networks regardless of the RTTs or underlined transport protocols of sending sources. We also saw that in contrast to FIFO scheduling, LAS scheduling adds robustness against non-TCP applications.

We proposed new LAS-based scheduling policies that differentiate the services of flows to improve the performance of priority flows without hurting the performance of short ordinary flows too much. We modeled the proposed policies for network flows that share a single bottleneck link with low packet losses and validated them using simulation of the policies in packet networks. The mean transfer times vs. flow sizes for the proposed analytic models showed very good agreement to the simulation results. Thus the analytic models, instead of simulation, were then used to easily obtain performance measures that compare different proposed policies in terms of mean transfer times of flows, average jitter in packet delivery, and unfairness. Simulations were used to compute the measures that the analytic models cannot produce, such as loss rates. In general, the analytical and simulation results showed that all proposed policies improved the performance of priority flows. In particular, LAS-log ($k = 1$) policy was seen to significantly improve the performance of priority flows while maintaining the performance of short flows as low as under LAS scheduling.

We finally analyzed a variant of LAS scheduling that favors short jobs the same way as LAS does, and reduces the penalty experienced by the largest jobs. We denoted this policy as *LAS-FCFS*. We derived the expressions of the mean transfer times for analytic models of LAS-FCFS for both, jobs and network flows. Reducing the penalty for the largest flows is efficient when a job size distribution has a high variability property where the largest jobs constitute a large fraction of the mass of the distribution. While LAS-FCFS accomplishes this, it penalizes some jobs of intermediate sizes. This is due to the conservation law and LAS, from which LAS-FCFS was derived, is a work-conserving policy. We thus proposed new class-based policies that can

differentiate services of different priority classes and use LAS-FCFS to preserve the ability of LAS-FCFS to prevent the penalty for the largest jobs. The aim of the proposed new policies is to improve the performance of *all* jobs in high priority class, while reducing the penalty of the largest jobs from the low priority classes. We also derived the analytic models of the proposed variants of LAS-FCFS policy for M/G/1 queue for jobs and network flows in packet networks. Numerical results showed that a particular proposed policy known as *Differential* LAS-FCFS significantly reduces the mean response time of high priority jobs while maintaining the performance of short jobs the same as under plain LAS, and the performance of the largest low priority flows similar to plain LAS-FCFS.

We saw that the performance improvement of LAS as compared to FIFO is more pronounced at high link utilizations of 70% or more. Network links with high link utilizations are likely to be network bottlenecks that can cause frequent packet losses and long queuing delays. Apparently, work in network tomography found these bottlenecks in Internet to be the low speed access links at the edge of networks. Implementing LAS in edge routers is feasible because the number of flows in access links is moderate.

Modeling of LAS and its variants in packet networks was limited in this thesis only to networks with no or very low packet loss rates. Packet losses of a TCP flow increase the transfer time of the flow due to the closed loop feedback mechanisms of TCP. Modeling flow transfer times under high loss is our ongoing work. As a future work, we intend to implement LAS in packet networks to analyze its performance for flows and to further explore practical challenges. Examples of these challenges include the performance of different flows under LAS when their size distribution does not exhibit high variability property and the question of using packet sequence numbers to compute their priorities in LAS. The experimental router in which we will implement LAS is the access point between a wireless network and the wired Internet. Wireless networks normally have low bandwidth and see high utilization and TCP in wireless networks shows unpredictable behavior even when there is no congestion. We expect that implementing LAS at these access points will be very beneficial.

# Chapter 8

# Résumé Détaillé en Français

## 8.1 Introduction

La seule classe de service fournit par l'Internet est la classe *best effort*. Le service best effort représente le type de service le plus simple qu'un réseau puisse offrir; il ne fournit aucune sorte de garantie aux flots [1] qui le traversent.

Quand un lien est congestionné, les paquets sont perdus suite à un débordement des buffers et TCP assure la retransmission des paquets. Puisque le réseau traite tous les paquets de la même façon, tous les flots peuvent être indistinctement victimes de la congestion. Le service best-effort est adapté aux applications qui peuvent tolérer des délais variables et des pertes de paquets. En revanche, de nombreuses applications telles les applications multimédia ne peuvent s'en satisfaire. De fait, de nouvelles architectures pour l'allocation de ressources capables de fournir des services différenciées sont nécessaires pour que l'Internet puisse évoluer en un véritable réseau multi-services.

Les années 90 ont vu l'explosion de l'Internet avec l'émergence du Web qui a transformé l'Internet en réseau commercial et grand public. Le Web a également ouvert la voie pour le déploiement de nouvelles applications telles la banque en ligne, les services de diffusion multimédia, la voix sur IP, le télé-enseignement, le pair-à-pair, etc.

Cette évolution spectaculaire de l'Internet a amené de nouveaux challenges. Beaucoup d'applications ont des besoins différents de celles qu'on trouvait à l'origine de l'Internet. Puisque l'Internet est devenu indispensable à notre quotidien, son manque de prédictibilité est devenu un problème crucial. En conséquence, les chercheurs ont proposé de modifier l'Internet pour qu'il puisse supporter des applications avec des contraintes de qualité de service ou QoS (Quality of Service) très différents.

La qualité de service dans l'Internet est définie comme la gestion des ressources réseaux disponibles dans le but de fournir des performances prédictibles en terme de bande passante,

[1] Un flot est un groupe de paquets partageant un ensemble commun d'attributs

délai, gigue, perte, etc. Pour implanter différents niveaux de QoS, on utilise des mécanismes qui contrôlent l'usage des ressources du réseau au travers de règles. Ces règles s'appliquent à tout ou partie de l'Internet et permettent notamment : la classification des flots, la gestion des buffers et les politiques de services des paquets.

Il existe un grand nombre de propositions pour introduire de la QoS dans l'Internet. On citera notamment : *integrated services* (Intserv) [59], *differentiated services* (Diffserv) [33], *multiprotocol label switching* (MPLS) [73], *constraint-based routing* (CR). La plupart de ces architectures ne sont pas implantées à l'heure actuelle pour des raisons de problème lors du passage à l'échelle, de compatibilité descendante ou de fiabilité.

Ci-après, dans ce résumé, nous allons tout d'abord présenté la motivation de cette thèse puis ses différentes contributions.

## 8.2 Motivation pour une nouvelle solution au problème de la QoS

Le fait que de nombreuses solutions pour fournir de la QoS n'aient pas été déployées associé au besoin toujours plus fort de QoS constitue la motivation principale de cette thèse. De plus, de nombreuses mesures sur l'Internet ont montrées que la contention dans l'Internet avait lieu principalement aux accès au réseau. Ceci est dû au fait que les liens d'accès sont en général à bas débits alors que les liens de cur sont surdimensionnés. Cela a motivé notre choix de nous concentrer sur des solutions de fourniture de QoS au niveau des liens d'accès. Notre objectif principal, en terme de QoS, est d'améliorer les performances perçues par les utilisateurs. La solution que nous proposons permet d'avoir des taux de pertes et des temps de transfert la plupart du temps inférieur à ceux fournis par l'Internet actuel, de maintenir l'équité entre les flots et/ou d'offrir de la différentiation de service.

### 8.2.1 Caractéristiques du trafic Internet

Les mesures sur l'Internet ont montré que le trafic exhibe une forte disparité au niveau de la répartition de la masse au sens où il est constitué de nombreux flots très courts de 10 ou 20 paquets alors que 1% des flots les plus longs transportent 50% des octets. Pour parler de ce phénomène, on parle souvent de souris et d'éléphants. Les souris sont en général des transferts Web alors que la plupart des éléphants sont dus aux applications pair-à-pair [99].

De nombreuses distributions ont été proposées pour modéliser le trafic de l'Internet : Pareto, lognormal, hyper-exponentielle, Weilbull [26], Gaussienne inversée, etc., avec toujours des coefficient de variations ($C$) plus grands que 1 [8, 26, 41]. Le coefficient de variation est défini comme le rapport entre l'écart type et la moyenne d'une loi. $C$ est une métrique usuelle pour mesurer la variabilité d'une distribution (plus $C$ est grand, plus la variabilité est grande).

De nombreuses études ont tenté de tirer partie de la grande variabilité des distributions de l'Internet pour améliorer ses performances. Par exemple, [52, 30, 29] ont proposés des systèmes de répartition de charge dans les systèmes distribués, [93, 39] ont proposé des modifications au niveau du routage ou de la commutation, [51, 27] de nouvelles politiques de services pour le Web. Nous montrons dans la section suivante que si la forte de variabilité de l'Internet n'est pas prise en compte, les performances en pâtissent.

### 8.2.2 L'impact de TCP et FIFO sur les performances de petits flots

Le trafic Web, qui constitue une large fraction des flots courts dans l'Internet, est transféré au moyen du protocole de transport TCP. Une étude attentive de TCP souligne les facteurs qui affectent les performances des flots courts : indépendamment de la bande passante du réseau, TCP va de manière préventive, toujours initialiser sa fenêtre de congestion à un paquet et doubler ensuite pour chaque ACK reçu jusqu'au moment où il détecte une perte (phase de slow-start). Ainsi, même si le chemin choisi à suffisamment de ressource, la durée du transfert dépend fortement du RTT (round trip time ou temps d'aller-retour) de la connexion. Les flots courts TCP souffrent aussi de performances en terme de reprise sur erreur puisqu'ils n'ont souvent pas assez de paquets à émettre pour un fast-retransmit [12]. Au contraire, ils doivent faire une reprise sur timer qui prolonge de fait le temps total de transmission et ce d'autant plus que le timer de retransmission est initialisés avec des valeurs en général très grandes (3 seconds). Ainsi, perdre des paquets dans une phase de slow-start pour un flot court est très pénalisant.

FIFO (First In First Out) est la discipline de service utilisée couramment dans l'Internet. La théorie des files d'attente nous enseigne que FIFO favorise toujours les clients longs et pénalise les clients courts car un client arrivant dans une file d'attente doit attendre le service complet de tous les clients devant lui dans la file avant d'être servi. Un phénomène similaire est observable dans un réseau de paquets où l'entité atomique est le flot (et non le client) [43]. De plus, les files dans les routeurs de l'Internet sont gérées suivant la politique drop-tail qui ne différencie pas les flots longs des flots courts. Ainsi, un flot court est défavorisé par l'utilisation de FIFO/drop-tail alors qu'il est déjà défavorisé avec TCP. Puisque ces flots courts constituent la majorité des flots de l'Internet, on voit qu'une amélioration de leur service aurait un impact direct sur les performances du réseau vu des utilisateurs.

L'objectif de ce travail est de trouver une politique de service (et de gestion des buffers) alternative à FIFO qui soit utilisable dans les réseaux de paquets et améliore significativement le temps de réponse des flots courts sans trop pénaliser les flots fonds. Les politiques de services qui discriminent suivant la taille des clients sont des politiques dites basées sur la taille. Nous présentons celle que nous avons choisi, LAS (Least Attained Service) dans la section suivante.

## 8.3 La politique de service LAS

En théorie des files d'attente, LAS est une politique de service préemptive multi-niveaux qui donne la priorité au client dans le système qui a reçu le moins de service jusqu'à présent. Dans le cas d'égalité, les clients partagent le serveur en mode processor sharing (PS) [64]. Un client nouvellement arrivé préempte toujours le (ou les) client(s) en services et est servi jusqu'à son départ, une nouvelle arrivée ou jusqu'au moment où il a obtenu un service égal à celui des clients qu'il a préempté. Une implémentation de LAS requiert la connaissance de la quantité de service reçue par les clients, ce qui peut être obtenu facilement auprès du serveur. LAS est aussi connue dans la littérature sous les noms de *foreground-background* (FB) [24, 64] ou *shortest elapsed time* (SET) first scheduling [23]. Dans ce travail de thèse, nous évaluons l'utilisation de LAS dans des réseaux paquets pour réduire le temps de transfert des flots courts et réduire le temps de réponse global perçu par l'utilisateur. Dans les réseaux paquets, la métrique à considérer est la performance d'un flot de paquets, qui est une entité différente d'un client dans une file d'attente. Le terme de client est communément utilisé en théorie des files d'attente pour désigner une quantité de travail qui arrive instantanément dans le système. Avec cette définition, il est clair qu'un flot de paquets ne peut être assimilé à un client dans une file d'attente. Au contraire, une source dans un réseau paquet transmet un flot de paquet sous la forme de paquets espacés dans le temps qui se trouvent multiplexés avec les paquets d'autres flots. De plus, lorsque TCP est utilisé, l'envoie des paquets est fonction de l'historique (récent) de la connexion. Pour toutes ses raisons, il est clair que les modèles analytiques de LAS conçus pour les files d'attente ne peuvent pas s'appliquer tels quels dans les réseaux de paquets.

Dans le cas des flots, le prochain paquet servi par LAS est celui appartenant au flot qui a émis le moins d'octets jusqu'à présent. Par définition, LAS octroie toute la bande passante à un nouveau flot jusqu'à ce que celui-ci soit préempté ou ait reçu une quantité de service égale à celle des flots qu'il a préemptés. Si plusieurs flots ont reçu une quantité de service équivalente, ils partagent équitablement le serveur (politique round-robin).

### 8.3.1 Analyse de LAS

#### Métriques

Soit $\lambda$ le taux d'arrivée des clients. On considère une distribution $X$ avec une densité $f(x)$. L'abréviation c.m.f.v.f signifie dans la suite continue, à moyenne finie et à variance finie. Étant donnée une fonction de répartition $F(x) \triangleq \int_0^x f(t)dt$, on dénote par $F^c(x) \triangleq 1 - F(x)$ le complémentaire de cette fonction.

On définit $m_n(x)$ comme $m_n(x) \triangleq \int_0^x t^n f(t)dt$. Ainsi $m_1 \triangleq m_1(\infty)$ est la moyenne et $m_2 \triangleq m_2(\infty)$ le second moment de la distribution considérée. La charge des clients de taille inférieur ou égal à $x$ est donnée par $\rho(x) \triangleq \lambda \int_0^x tf(t)dt$, et $\rho \triangleq \rho(\infty)$ est la charge totale du système.

On considère une file M/G/1 dans cette section, òu $G$ est une distribution c.m.f.v.f et $M$

signifie des arrivées poissonnienes. Un processus de Poisson fournit une modélisation acceptable du processus d'arrivée des flots dans l'Internet [45]. L'expression du temps de réponse conditionnel des clients de tailles $x$ pour la politique de service M/G/1/SRPT [91] peut être décomposé en un temps d'attente conditionnel $W(x)$ (temps entre l'instant où un client arrive dans le système et temps au bout duquel il est servi pour la première fois) et un temps de résidence conditionnel défini $R(x)$ comme le temps entre le premier service et la fin du service du client :

$$T(x)_{SRPT} = W(x)_{SRPT} + R(x)_{SRPT}$$

$$W(x)_{SRPT} = \frac{\lambda(m_2(x) + x^2 F^c(x))}{2(1-\rho(x))^2} \tag{8.1}$$

$$R(x)_{SRPT} = \int_0^x \frac{1}{1-\rho(t)} d(t) \tag{8.2}$$

De manière similaire, on décompose le temps de réponse $T(x)_{LAS}$pour une file $M/G/1/LAS$ en deux termes. Le premier terme $\tilde{W}(x)_{LAS}$ et le second $\tilde{R}(x)_{LAS}$. Notons que ces termes ne correspondent pas aux temps conditionnels d'attente et de résidence définis pour SRPT. Ils sont introduits pour simplifier les comparaisons entre politiques dans la section 2.3.3. Le temps moyen conditionnel de réponse $T(x)_{LAS}$ pour LAS est donné par [89],

$$T(x)_{LAS} = \tilde{W}(x)_{LAS} + \tilde{R}(x)_{LAS}$$

$$\tilde{W}(x)_{LAS} = \frac{\lambda(m_2(x) + x^2 F^c(x))}{2(1-\rho(x) - \lambda x F^c(x))^2} \tag{8.3}$$

$$\tilde{R}(x)_{LAS} = \frac{x}{1-\rho(x) - \lambda x F^c(x)} \tag{8.4}$$

Finalement, les formules de $T(x)$ pour des files M/G/1/PS et M/G/1/NPP, où NPP signifie des politiques de service non préemptives sont [25] :

$$T(x)_{PS} = \frac{x}{1-\rho} \tag{8.5}$$

$$T(x)_{NPP} = \frac{\lambda m_2}{2(1-\rho)} + x \tag{8.6}$$

Les politiques NPP incluent FIFO, *last-in first-out* (LIFO), RANDOM, etc. On définit le temps de réponse conditionnel normalisé par $S(x) = \frac{T(x)}{x}$. La définition du temps de réponse normalisé $S(x)$ montre que pour deux politiques A et B, le ratio $\frac{T(x)_A}{T(x)_B} = \frac{T(x)_A/x}{T(x)_B/x} = \frac{S(x)_A}{S(x)_B}$.

**Choix de la distribution des tailles des clients**

Nous allons analyser LAS pour des distributions de taille de clients c.m.f.v.f. spécifiques. Le choix d'une distribution est motivé par les caractéristiques du trafic Internet où *la plupart des flots sont courts et plus de la moitié de la charge est due à une très faible fraction des flots les plus grands*. Plusieurs distributions ont été utilisée pour modéliser le trafic Internet avec toujours un coefficient de variation plus grand que 1 [8, 26, 41]. Parmi ces distributions, on peut citer : la distribution de Pareto, la distribution de Pareto bornée, la distribution lognormale, la distribution hyper-exponentielle, la distribution Gaussienne inversée et des mélange de lognormal et Pareto. Soit $X$ la taille des clients. La densité des clients pour une distribution de Pareto bornée est:

$$f(x) = \alpha k^{\alpha} x^{-\alpha-1}, \qquad x \geq k, 0 \leq \alpha \leq 2 \tag{8.7}$$

où $k$ est la taille minimale des clients et $\alpha$ l'exposant de la loi de puissance. La fonction de répartition $F(x)$ est donnée par :

$$F(x) = 1 - \left(\frac{a}{x}\right)^{\alpha}, \qquad x \geq k. \tag{8.8}$$

Dans cette these, nous avons utilisé des variantes de la distribution de Pareto (la distribution de Pareto de seconde espèce et la distribution de Pareto bornée) pour modéliser la forte variabilité des distributions observées sur l'Internet. La distribution de Pareto de seconde espèce est version translatée de la distribution de Pareto avec des tailles d'échantillon commençant à zéro et non à $k$. La densité d'une distribution de Pareto de seconde espèce est donnée par :

$$f(x) = \alpha a^{\alpha} (x+a)^{-\alpha-1}, \qquad x \geq 0, 0 \leq \alpha \leq 2, a > 0 \tag{8.9}$$

où $a$ est le facteur d'échelle. Nous avons souvent utilisé cette distribution dans nos simulations pour générer des flots de taille variable avec un minimum de un paquet. La fonction de répartition de Pareto de seconde espèce est donnée par :

$$F(x) = 1 - \left(\frac{a}{x+a}\right)^{\alpha}, \qquad x \geq 0. \tag{8.10}$$

La distribution de Pareto borne est utilisée communément en analyse car elle peut avoir une grande variance et modélise ainsi bien les distributions observées sur l'Internet. De plus la taille maximale bornée des flots permet de rendre compte de la taille maximale des flots observés [107, 27, 13]. Nous utilisons également la distribution de Pareto bornée pour cette

même raison. Nous la notons $BP(k, P, \alpha)$, où $k$ et $P$ sont les tailles minimales et maximales des clients et $\alpha$ l'exposant de la loi de puissance. La densité d'une loi de Pareto est donnée par :

$$f(x) = \frac{\alpha k^\alpha}{1-(k/P)^\alpha} x^{-\alpha-1}, \qquad k \leq x \geq P, 0 \leq \alpha \leq 2 \tag{8.11}$$

La function de repartition $F(x)$ et le nième moment $m_n$ pour une loi $BP(k, P, \alpha)$ sont donnés par :

$$\begin{aligned} F(x) &= \frac{1}{1-(k/P)^\alpha}[1-(k/x)^\alpha], \quad k \leq x \leq P, 0 \leq \alpha \leq 2 \\ m_n &= \frac{\alpha}{(n-\alpha)(P^\alpha - k^\alpha)}(P^n k^\alpha - k^n P^\alpha) \end{aligned}$$

Ainsi, l'expression du coefficient de variation d'une loi $BP(k, P, \alpha)$ est donné par $C \triangleq \frac{\sqrt{m_2 - m_1^2}}{m_1}$. Pour obtenir différentes valeurs de $C$ pour une loi $BP(k, P, \alpha)$, on ajuste un ou plusieurs paramètres de la distribution, i.e., $k$, $\alpha$, ou $P$.

Nous considérons également le cas des distributions exponentielles des taills de clients pour évaluer LAS dans le cas d'une file M/M/1. Pour une loi exponentielle, le coefficient de variation est 1. On dénote la loi exponentielle avec une moyenne de $1/\mu$ par $Exp(1/\mu)$. La densité de la distribution exponentielle est donnée par :

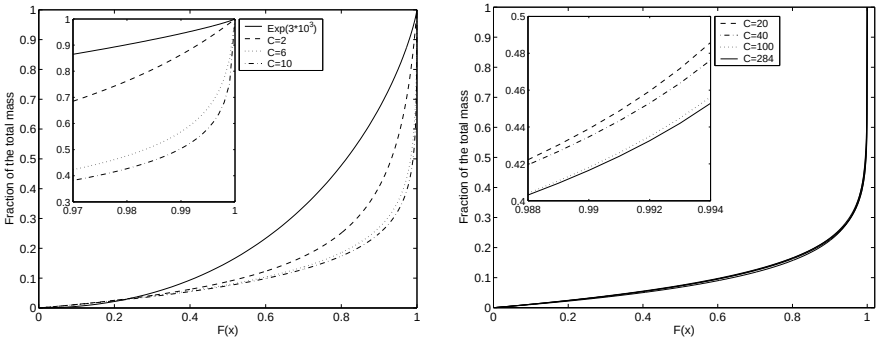
$$f(x)_{Exp} = \mu e^{-\mu x}, \qquad x \geq 0, \mu \geq 0$$

La variabilité de la distribution peut être estimé à partir de la fonction de masse pondérée ($M_w(x)$) [28], qui ( pour un client de taille $x$) est défini comme la fraction de la masse constituée par les clients de taille inférieure ou égale à $x$ : $M_w(x) \triangleq \frac{\rho(x)}{\rho}$. the high variability property.

La figure 8.1 représente la fonction de masse pondérée pour des distributions BP avec différentes valeurs de $C$ et pour la distribution exponentielle, toutes avec une même moyennes de $3 \times 10^3$. La figure 8.1(a) montre que pour la distribution exponentielle, 1% des clients les plus gros constituent environ 5% de la masse totale. Au contraire, pour une loi BP, on observe que 1% des plus gros clients concentrent une masse qui croît avec $C$ de 15% de la masse totale pour $C = 2$ à près de 50% pour $C = 10$. Par ailleurs, la figure 8.1(b) montre que pour les lois BP avec $C \geq 20$ exhibent toute une forte variabilité au sens où 50% de la masse totale est constitué par moins de 1% des plus grands clients.

### 8.3.2 Résultats analytiques

Nous avons comparé LAS avec plusieurs autres politiques de services incluant PS, SPRT et FIFO pour une distribution des durées des clients BP ou exponentielle. Nous présentons certains de ces résultats dans cette section.




(a) $Exp(3 \times 10^3)$ *and BP with* $C = 2, 6, 10$ (b) *BP with* $C = 20, 40, 100, 284$

Figure 8.1: *Fonction de masse pondérée pour des lois* $BP(k, P, \alpha)$ *et* $Exp(3 \times 10^3)$.

**Distribution générale**

Pour une distribution générale, nous avons établi le théorème 8.3.1 et son corollaire 8.3.1 qui montre la relation entre les temps de réponse conditionnels normalisés de LAS et PS :

**Theorem 8.3.1** *Pour toute distribution c.m.f.v.f et toute charge* $\rho < 1$,

$$S_{LAS} \leq \frac{2-\rho}{2(1-\rho)} S_{PS} \tag{8.12}$$

Proof:

$$\begin{aligned} S_{LAS} &= \int_0^{+\infty} \frac{T(x)_{LAS}}{x} f(x)dx \\ &\leq \int_0^{+\infty} \frac{2-\phi(x)}{2(1-\phi(x))^2} f(x)dx \quad \text{Théorème 2.3.1} \qquad (8.13) \\ &\leq \frac{2-\rho}{2(1-\rho)^2} \int_0^{+\infty} f(x)dx \qquad \text{Lemme 2.3.1} \\ &= \frac{2-\rho}{2(1-\rho)^2} \\ &= \frac{2-\rho}{2(1-\rho)} S_{PS} \quad \text{puisque } S_{PS} = \frac{1}{1-\rho} \qquad (8.14) \end{aligned}$$

□

**Corollary 8.3.1** *Pour toute distribution c.m.f.v.f et toute charge* $\rho < 1$,

$$T_{LAS} \leq \frac{2-\rho}{2(1-\rho)} T_{PS} \tag{8.15}$$

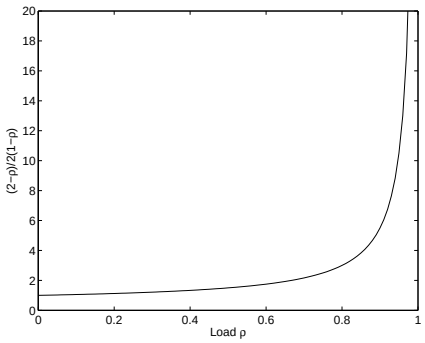



Figure 8.2: *Borne supérieure du* $\frac{S_{LAS}}{S_{PS}}$.

Proof: La preuve est similaire à celle du théorème 8.3.1. □

La figure 8.2 illustre le théorème 2.3.2. On peut voir que le ratio entre les temps de réponse normalisés est inférieur à 2 pour une charge $\rho \leq 0.66$ et inférieur à 6 pour $\rho \leq 0.9$. Ce résultat indique que pour des charges modérées, le temps de réponse normalisé de LAS reste proche de celui de PS. On peut donc espérer que la pénalité subie par les flots les plus longs sous LAS ne soit pas trop élevée (ce qui se traduirait par un temps de réponse normalisé moyen élevé). Notons également que la borne utilisée pour trouver ce résultat est en faite assez peu fine, due à la technique de majoration utilisée dans le lemme 2.3.1.

Nous avons aussi dérivé une borne supérieure qui compare le temps de réponse moyen de LAS avec la famille des lois non préemptivs NPP. Cette borne est fonction de coefficient de variation et de la charge :

**Lemma 8.3.1** *Pour toute distribution c.m.f.v.f avec une moyenne* $m_1$ *et un coefficient de variation* $C$*, le temps moyen de réponse avec LAS pour une charge* $\rho$ *(*$T_{LAS}$*) est borné supérieurement par :*

$$T_{LAS} \leq \frac{(2-\rho)}{2(1-\rho)} T_{NPP} - \frac{\rho(2-\rho)[C^2-1]}{4(1-\rho)^2} m_1 \tag{8.16}$$

Proof: Le résultat s'obtient en substituant $T_{PS}$ dans l'équation (2.22) au chapitre 2 dans le Corollary 8.3.1. □

Cette borne est intéressante car elle permet de comparer LAS avec les politiques NPP pour toute distribution et toute valeur de $C$.

La figure 8.3 représente que le ratio du temps de réponse de LAS au temps de réponse des politiques NPP pour $C \geq 1$ et $\rho < 1$. On observe que LAS a un temps de réponse moyen plus élevé que les politiques NPP seulement dans le case de où $C$ est proche de 1. Dans ce cas, on observe que le ratio augmente avec la charge. A l'opposé, le temps de réponse moyen de

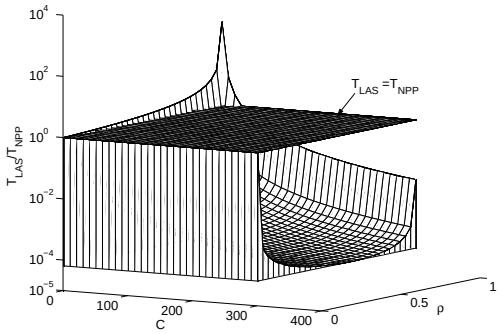
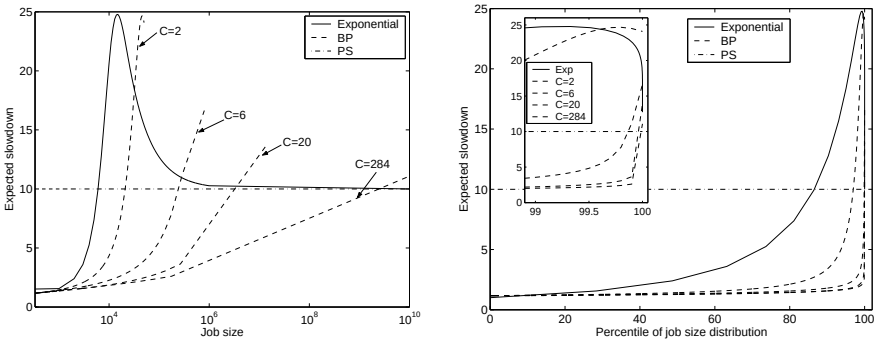



Figure 8.3: *Borne supérieure sur $\frac{T_{LAS}}{T_{NPP}}$ comme une function de la charge $\rho$ et du coefficient de variation $C$.*

LAS est inférieur à celui des lois NPP pour toute distribution avec un coefficient de distribution supérieur à 1 et pour toute charge $\rho < 1$. De manière générale, on observe que pour une charge $\rho$ donnée, le ratio $\frac{T_{LAS}}{T_{NPP}}$ décroît avec $C$.

**Distributions spécifiques**

Nous présentons maintenant des résultats obtenus pour des distributions BP avec différentes valeurs de $C$ et pour la distribution exponentielle. Un client est dit traité inéquitablement si son temps moyen de réponse conditionnel avec une politique de service donnée (par exemple LAS) est inférieure à ce même temps moyen conditionnel avec PS. Nous étudions ici l'inéquité de LAS pour des distributions avec différentes valeurs de $C$. Nous utilisons la distribution exponentielle ($C = 1$) et des distributions BP avec des valeurs de $C$ de 2, 6, 20, et 284. Toutes les distributions ont la même moyenne de $3 \times 10^3$. Pour obtenir des distributions BP avec différentes valeurs de $C$ et une même moyenne, on fait varier $\alpha$ et $p$.




(a) *Job size*

(b) *Quantile de la distribution des tailles de clients.*

Figure 8.4: *Temps moyen de réponse normalize sous LAS et sous PS pour différentes valeurs de $C$.*

Les figure 8.4(a) et 8.4(b) représentent les temps moyens de réponse conditionnels en fonc-

tion des quantiles de la distribution des tailles de clients. On voit, à partir de ces deux figures, que le pourcentage des flots les plus gros qui sont pénalisés avec LAS (i.e. ont un temps de réponse normalisé supérieur avec LAS qu'avec PS) ainsi que le niveau de la pénalité décroissent lorsque $C$ croît. Pour la distribution BP avec $C \geq 6$, on observe que moins de 0.5% des flots les plus longs souffrent d'une pénalité et que la différence de performance entre $C = 20$ et $C = 284$ est minime.

Dans le cas de la distribution exponentielle, on peut obtenir des résultats plus précis et notamment le théorème suivant :

**Theorem 8.3.2** *Pour une distribution exponentielle et une charge $\rho < 1$,*

$$\begin{aligned} S_{LAS} &\leq S_{PS} \\ T_{LAS} &= T_{PS} \end{aligned} \tag{8.17}$$

Proof: Voir la prevue du théorème 2.3.3 au chapitre 2 et dans l'annexe A □

Notons que LAS fonctionne mieux pour des distributions avec des valeurs de $C$ élevées. La distribution exponentielle, elle, a un coefficient de variation égal à 1, mais, malgré cela, le théorème 8.3.2 montre que les performances moyennes de LAS restent meilleure que celle de PS.

### 8.3.3 LAS and SRPT

SRPT est une politique optimale au sens où elle minimise le temps de réponse moyen. Il est donc important de comparer LAS à SRPT pour évaluer de combien elles différent et voir si la différence est fonction de la variabilité de la loi. Nous avons tout d'abord comparé le temps moyen conditionnel de réponse de LAS et SRPT :

**Theorem 8.3.3** *Soit $\phi(x) \triangleq \rho(x) + x\lambda F^c(x)$. Alors, pour toute distribution c.m.f.v.f et pour toute charge $\rho < 1$,*

$$T(x)_{SRPT} \leq T(x)_{LAS} \leq \Big(\frac{1-\rho(x)}{1-\phi(x)}\Big)^2 T(x)_{SRPT} \tag{8.18}$$

Proof: Voir annexe B. □

Les résultats numériques dérivés à partir du théorème précédent montre que LAS offre un temps de réponse similaire à SRPT quand la distribution de la taille des clients est fortement variable. Pour la distribution exponentielle, LAS offre des temps de réponse notablement plus élevés que SRPT.

Nous avons aussi analysés LAS en surcharge et comparer les résultats obtenus avec ceux de SRPT. Nous avons tout d'abord prouvé que Las peut servir des clients même en cas de surcharge ; puis, nous avons dérivé le temps de réponse conditionnels moyens sous LAS en surcharge :

$$T(x)_{\text{LAS}} = \begin{cases} \frac{\lambda(m_2(x)+x^2(1-F(x)))}{2(1-\rho(x)-\lambda x(1-F(x)))^2} + \frac{x}{1-\rho(x)-\lambda x(1-F(x))} & \text{si } x < x_{\text{LAS}}(\lambda) \\ +\infty & \text{si } x \geq x_{\text{LAS}}(\lambda) \end{cases}$$

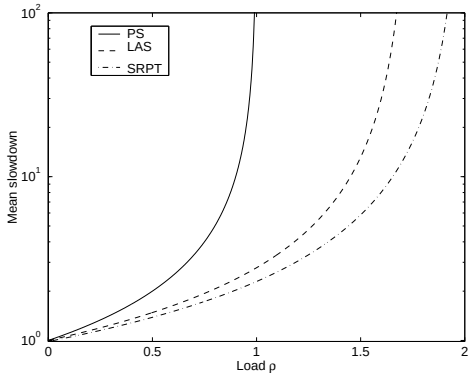



Figure 8.5: *Temps moyen de réponse normalize pour le 99ème quantile pour la distribution* $BP(322, 10^{10}, 1.1)$ *en function de la charge.*

Le temps de réponse de PS est non borné lorsque $\rho > 1$. Ce n'est pas le cas pour SRPT comme prouvé dans [13]. La figure 8.5 montre qu'au contraire de PS, LAS et SRPT sont stables en surcharge pour certaines tailles de clients. On observe également que LAS devient instable avant SRPT.

### 8.3.4 Modèle analytique de LAS dans un réseau de paquet

Supposons que chaque flot ait au moins un paquet dans la file d'attente du goulot d'étranglement la quasi-totalité du temps. Alors, le temps moyen de service du client de taille $x$ avec LAS exprimé en unité de temps de transfert d'un paquet au niveau du goulot d'étranglement peut se modéliser par le temps de réponse d'un client de taille $x$ dans un file d'attente gérée avec LAS, c'est-à-dire [64]:

$$T(x)_{LAS} = \frac{W_o(x) + x}{(1 - \rho_x)}, \tag{8.19}$$

où $W_o(x)$ est le nombre moyen de paquets en attente avec un numéro de séquence inférieur ou égal à $x$ à un instant aléatoire :

$$W_o(x) = \frac{\lambda \overline{x_x{}^2}}{2(1 - \rho_x)}.$$

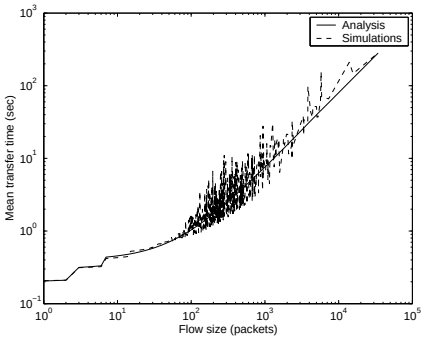



Figure 8.6: *Validation du modèle paquet de LAS, avec* $\rho = 0.73$ *et un taud de perte* = $1.2\%$.

Bien que chaque flot n'ait pas nécessairement un paquet dans la file d'attente tout le temps, comme le demande le modèle ci-dessus pour être exact, un paquet qui arrive un peu plus tard que prévu (dans le modèle de file d'attente) est tout de même capable de récupérer son retard car sa priorité plus grande le fait passer devant les paquets devant lui. C'est pourquoi nous pensons que, globalement, ce modèle est plus précis que le modèle paquet pour FIFO présenté à la section 5.2.2.

La figure 8.6 représente le temps moyen conditionnel de transfert en fonction de la taille du flot obtenu par le modèle analytique ainsi que par simulation avec des flots TCP en ns-2. La figure 8.6 est similaire à la figure 8.6 pour un taux de perte plus faible (i.e., 1.2%). On observe une très bonne prédiction du modèle analytique. Cela montre que le modèle paquet de LAS est très précis.

## 8.4 LAS dans les réseaux paquet

Cette thèse propose d'utiliser LAS dans les réseaux paquets, et notamment dans l'Internet où TCP domine. Dans cette section, nous étudions l'interaction de LAS avec TCP pour examiner en détail les performances au niveau flot avec LAS. A l'aide de simulation, nous montrons que LAS, comparé à FIFO, limite les pertes de paquets et réduit significativement les temps de réponse des flots courts au prix d'une pénalité négligeable pour les flots longs. Nous montrons également que le gain en performance avec LAS est dû principalement à la façon dont LAS interagit avec TCP dans la phase de slow-start (SS) en réduisant le RTT et minimisant les pertes. Nous étudions également les performances des flots TCP et UDP illimités (ou extrêmement longs) en comparant les résultats obtenus avec ceux de FIFO

### 8.4.1 Résultat de simulation pour LAS

Dans les réseaux paquets, la politique LAS interagit bien avec les flots TCP courts car elle insère les premiers paquets d'un flot court en tête de buffer. Le RTT des flots courts se trouve ainsi réduit, ce qui réduit directement le temps de réponse de ces flots. De plus, avec LAS, un paquet qui arrive au serveur est d'abord inséré dans la queue, puis le paquet qui a la plus basse priorité est éliminé. Cela limite les pertes pour les flots courts. Les flots qui subissent des pertes sont les flots longs a priori. Or ceux-ci ayant suffisamment de paquets de transmettre sont en général capable de récupérer leurs pertes à l'aide d'un fast retransmit qui n'affecte que modérément le débit de la connexion par opposition à une expiration de timer. Au contraire, les flots courts ont plus tendance à récupérer leurs pertes par timer, ce qui affecte d'autant leurs performances. La figure 8.7 montre par exemple le taux de perte paquet moyen en fonction de la taille des flots. On voit clairement que pour cette simulation, les flots de moins de 40 paquets ne subissent aucune perte avec LAS alors qu'ils subissent beaucoup de perte avec FIFO.

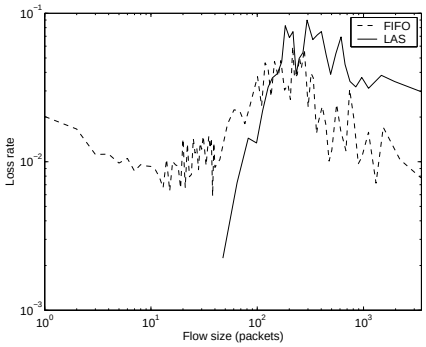



Figure 8.7: *Taux de perte de paquet avec une distribution de Pareto des tailles de flots (Web) pour* $\rho = 0.73$

La figure 8.7 montre que LAS réduit le taux de perte de pour les flots courts et multiplie approximativement par deux le taux de perte des flots longs comparé à FIFO. Il faut néanmoins garder à l'esprit que les flots longs ne représente que 1% des flots du trafic. En résumé, Las offre un gain de performance très important pour les flots courts dans un réseau paquet.

De manière similaire, le tableau 8.1 montre la moyenne et la variance de la bande passante d'un transfert FTP très long (illimité) sous LAS et sous FIFO pour différentes valeurs de charge. Les résultats de simulation du tableau 8.1 sont obtenus en calculant la bande passante obtenue par le flot (en paquets/s) pour des fenêtres de 100 secondes sur lesquels on calcule ensuite la moyenne et la variance de processus. Pour une charge de 0.75, LAS se comporte de manière similaire à FIFO. Ce n'est que pour des charges très élevées, au dessus de 0.90 que les performances du transfert FTP se dégradent notablement par rapport à FIFO. Notons également que la variance augmente également de manière très importante pour des charges élevés car avec LAS,il y a des périodes où le flot FTP ne reçoit aucun service.

| | LAS | | FIFO | |
|---|---|---|---|---|
| Charge $\rho$ | moyenne | variance | moyenne | variance |
| 0.75 | 157.8 | 1.03 | 156.30 | 1.31 |
| 0.92 | 33.93 | 2.27 | 97.55 | 1.01 |
| 0.98 | 4.82 | 151.35 | 11.86 | 43.68 |

Table 8.1: *Moyenne et Variance du débit (en paquets/s) avec LAS et FIFO.*

## 8.5 LAS dans les réseaux hétérogènes

Nous étudions ici les performances de LAS dans des réseaux TCP hétérogènes. Par réseaux TCP hétérogènes, on entend des réseaux où les connections ont des temps de propagation très différents, des réseaux où les protocoles de transport sont hétérogènes (TCP et UDP), ou des réseaux avec des goulots d'étranglement multiples. Ces réseaux hétérogènes sont connus pour causer une allocation inégale de la bande passante entre les connexions. Les travaux précédents sur ces problèmes ont conclu en des faiblesses de TCP et ont donc proposer certaines modifications de TCP. Dans cette section, nous montrons que LAS peut éliminer ces effets indésirables.

### 8.5.1 Résultats de simulation

Dans cette section, nous analysons les performances de LAS dans des réseaux hétérogènes simples, i.e. avec un seul routeur congestionné. Nous considérons des réseaux avec des temps de latence différents ou des protocoles de transport différents.

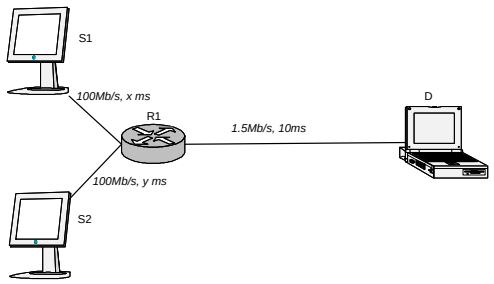
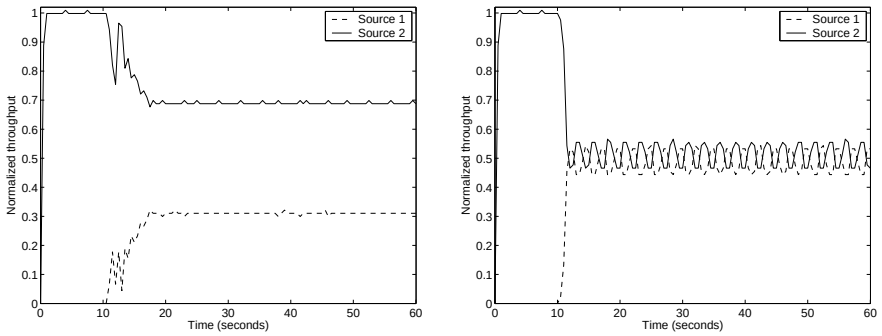



Figure 8.8: *Un réseau avec un seul lien partagé.*

Nous simulons la topologie de la figure 8.8, où les deux sources S1 et S2 envoient des données vers la destination D au travers du goulot d'étranglement R1-D, où nous déployons LAS ou FIFO. Nous faisons varier les temps de propagation des $x$ et $y$ et le type de la source, TCP ou UDP.

Nous étudions tout d'abord le réseau de la figure 8.8, où les deux sources sont de type TCP et les temps de propagation des liens S1-R1 et S2-R2 sont $x = 100ms$ et $y = 1ms$ respectivement. S2 commence à émettre à $t = 0$ alors que S1 commence 10 secondes plus tard. La figure 8.9(a) montre les résultats de simulation quand on utilise la politique FIFO/droptail et

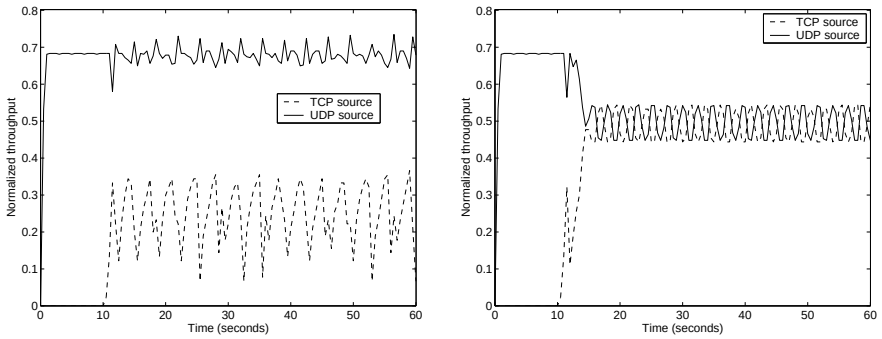



(a) *FIFO* (b) *LAS*

Figure 8.9: *Débit des flots TCP avec des RTTs différents.*

la figure 8.9(b) lorsqu'on utilise LAS. On voit clairement que LAS, élimine l'inéquité observée avec FIFO puisque les deux sources se partagent équitablement la bande passante.

Nous étudions maintenant le réseau de la figure 8.8 où S1 est une source TCP et S2 une source UDP qui transmet avec un débit constant de 1024Kb/s (les temps de propagation sont les mêmes).




(a) *FIFO* (b) *LAS*

Figure 8.10: *Throughput obtained by UDP and TCP connections.*

La figure 8.10(a) présente les résultats obtenus par simulation dans le cas de FIFO et la figure 8.10(b) dans le cas de LAS. On note l'inéquité entre les sources dans le cas de FIFO puisque la source UDP utilise une bande passante égale à son débit d'émission. Les grandes oscillations du débit de la connexion TCP sont dues aux fréquentes pertes subies par la connexion. Au contraire, lorsqu'on utilise LAS, les deux sources se partagent équitablement la bande passante. Cette équité est obtenue notamment en éliminant plus de paquets de la connexion UDP.

Les performances de TCP dans un réseau avec de multiples points de congestion ont été étudiées pour la première foir par S. Floyd [42]. Dans [42], S. Floyd montre un biais du partage de la bande passante en défaveur des connexions qui traversent l'entièreté du réseau. Encore une fois, nos simulations montrent que LAS est capable de rétablir l'équité entre les connexions (longues et courtes en terme de RTT), comme le montre les figures 8.12(a) et 8.12(b) qui représentent les bandes passantes d'une connexion courte et longue à chaque fois. La topologie

utilisée est présentée figure 8.11.

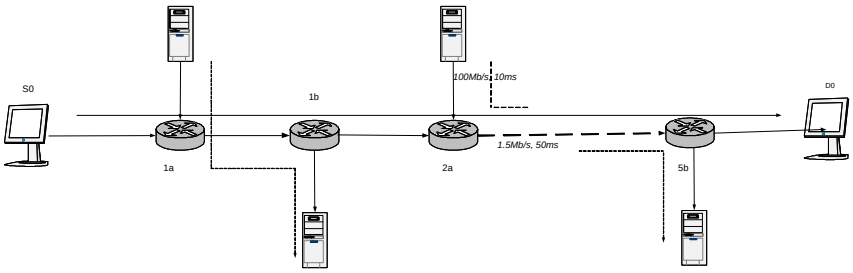
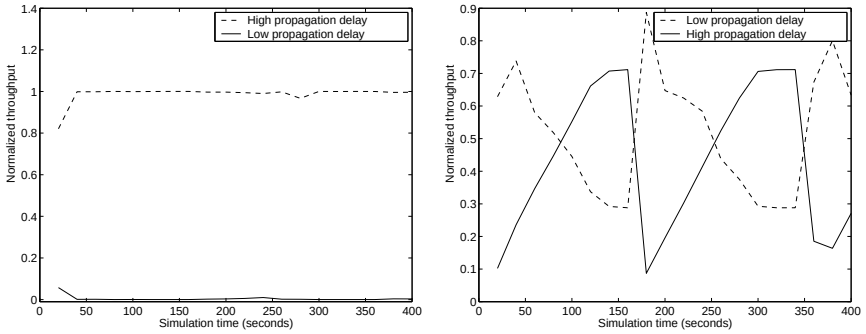



Figure 8.11: *Topologie de réseau simulée.*




(a) *FIFO* (b) *LAS*

Figure 8.12: *Débit au lien 1a-1b.*

Les résultats obtenus montrent que les performances relatives des connections courtes et longues sous LAS varient en fonction des paramètres du réseau. Néanmoins, même si l'on peut observer des oscillations plus marquées du débit de la connexion longue dans certains cas, le débit moyen à long terme des connections longues et courtes avec LAS convergent toujours vers la même valeur.

## 8.6 Modèles pour les politiques LAS différenciées

### 8.6.1 Modèles au niveau flots

Avec LAS, la priorité des paquets d'un flot décroît avec le nombre de paquets envoyés. Dans cette section nous considérons 3 politiques de services basées sur LAS capables de fournir un service différencié en fonction de la classe d'un flot. Ces politiques sont basées sur LAS car on désire garder les bonnes caractéristiques de LAS (temps de réponse faible pour les petits flots, etc.). Les politiques de service proposées fonctionnent avec deux classes : la classe des flots prioritaires et la classe des flots ordinaires. L'objectif de ces politiques de service différenciées est de réduire le temps moyen de réponse des flots de la classe prioritaire sans trop affecter le temps de réponse des flots ordinaires et c'est pour cela que ces politiques sont dérivées de LAS.

Ces trois politiques sont dénommées LAS-fixed(k), LAS-linear (k) and LAS-log (k), car la priorité des paquets des flots de la classe prioritaire décroît de façon fixe, linéaire ou logarithmique. Les paquets des flots de classe 2 (classe ordinaire) ont une décroissance de leur priorité identique à LAS. Hormis ce changement, le traitement des paquets dans les files de LAS-fixed(k), LAS-linear (k) and LAS-log (k) est similaire à celui de LAS.

Les flots de classes 1 (prioritaires) seront désignées avec l'indice ou l'exposant $p$, alors que les flots de classe 2 seront désignés avec l'indice ou l'exposant $p$. Dans l'analyse des politiques de services différenciées, on note $q$ la fraction des flots de classe 1. Chaque classe peut avoir sa propre distribution de taille de flots avec une fonction de répartition $F_i(x)$ pour la classe $i$. L'analyse (temps de réponse conditionnels) des politiques LAS à deux classes (LAS-fixed(k), LAS-linear (k) and LAS-log (k)) est présentée dans cette section. Le type d'analyse est similaire à l'étude de LAS au niveau paquet, puisque l'on suppose que le premier paquet d'un flot de taille $x$ arrive au goulot d'étranglement à un instant aléatoire.

De manière plus formelle, on peut présenter les politiques LAS à deux classes en disant que les flots ordinaires ont une priorité à un instant donné égale au nombre de paquets envoyés (en supposant que tous les paquets ont la même taille - on pourrait aussi raisonner au niveau octets, mais la présentation au niveau paquet est plus intuitive). Au contraire, les flots prioritaires voient leur priorité croître suivant la fonction $P(x)$ définie ci-après :

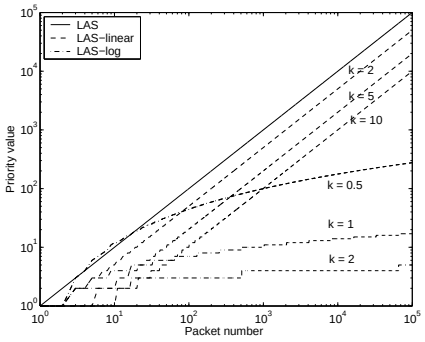



Figure 8.13: *Priority value as a function of attained service.*

- *LAS-fixed(k)*: le $x$ième paquet de la classe 1 a une priorité $P(x) = k$, avec $k > 0$ fixe.
- *LAS-linear (k)*: le $x$ième paquet de la classe 1 a une priorité $P(x) = x/k$, avec $k > 0$ fixe.
- *LAS-log (k)*: le $x$ième paquet de la classe 1 a une priorité $P(x) = (\log_2(x))^{\frac{1}{k}}$, avec $k > 0$ fixe.

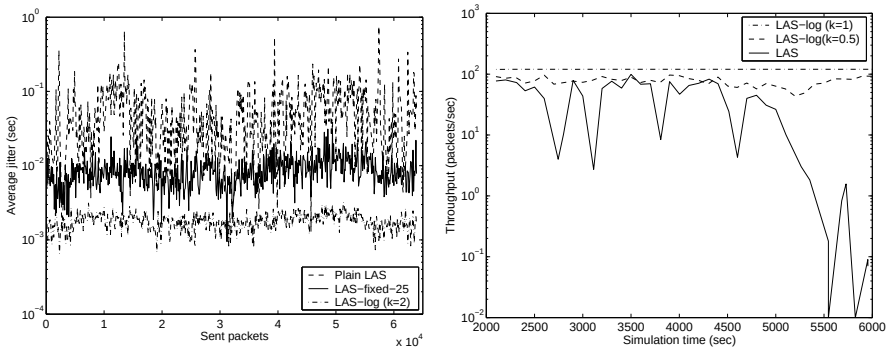
On peut observer que dans le cas de LAS, $P(x) = x$ pour le paquet numéro $x$. Pour chaque politique, la fonction $P(x)$ détermine la priorité relative de chaque paquet pour un flot donné. Puisque $P(x) \leq x$ pour LAS-fixed(k) (pour $x \geq k$), LAS-linear (k) and LAS-log (k), on comprend qu'un flot prioritaire de classe 1 et de taille $x$ pour l'une quelconque de ces politique interfèrent avec les flots ordinaire de taille inférieure à $x$. En conséquence, leur temps de réponse

doit être inférieur au cas de LAS (sans classe). Dans cette section, nous étudions LAS-fixed(k), LAS-linear (k) and LAS-log (k) pour obtenir de premières intuitions sur les avantages et les inconvénients de ces modèles à deux classes.

Pour chacune de ces politiques, nous avons développé un modèle analytique qui estime le temps de réponse conditionnel moyen et nous avons validé chaque modèle analytique avec des simulations ns-2 pour une topologie en étoile (avec un goulot d'étranglement central) et des taux de pertes inférieurs à 2%.

### 8.6.2 Résultats de simulation des politiques de services LAS à deux classes

Nous avons tout d'abord validé les modèles analytiques des politiques LAS-fixed(k), LAS-linear (k) and LAS-log (k) dans le chapitre 5. Nous avons ainsi montré que tous ces modèles estiment avec une grande précision les temps de réponse conditionnels dans le cas de taux de perte modérés. Nous avons ensuite analysés les performances relatives des différentes politiques pour des flots longs prioritaires au travers de simulations. Nous avons considéré le cas de longs flots TCP et le cas de longs flots UDP. Pour les flots UDP, la métrique retenue est la gigue, alors que pour le cas de flot TCP, la métrique utilisée est le débit moyen.




(a) *Gigue moyenne d'un flot UDP cal-* (b) *Débit d'un long flot TCP sous LAS et culée sur des fenêtres sautantes de 100 LAS-log (k), avec* $k = 0.5$ *et* $k = 1$. *paquets avec* $\rho = 0.7$.

Figure 8.14: *Performance des longs flots avec LAS et avec les politiques LAS à deux classes.*

Considérons tout d'abord le cas d'une source UDP qui transmet à un débit constant de 128 kb/s. La figure 8.14(a) représente la gigue moyenne des paquets groupés en fenêtre sautantes de 100 paquets. On observe que LAS offre la gigue la plus grande alors de LAS-log(2) offre la gigue la plus faible. Cela est dû au fait que, pour $k = 2$, la priorité du paquet numéro 64000 avec LAS-log(2) $(log_2(64000))^{0.5} \approx 4$ alors que la valeur de la priorité pour LAS-fixed est fixée à $25$. De plus, avec LAS, les simulations ont montré que les taux de pertes pouvaient atteindre $10\%$ ou plus (voir figure 3.14 au chapitre 3), alors que le même flot sous LAS-log et LAS-fixed(k) ne subit quasiment aucune perte.

La figure 8.14(b) compare le débit d'un long flot FTP avec LAS et LAS-log pour $k = 0.5$ et $k = 1$ en fonction du temps de simulation. On note tout d'abord que l débit avec LAS

varie beaucoup et tombe à zéro à la fin de la simulation. LAS-log améliore significativement les performances du transfert FTP. Lorsque $k = 0.5$ le transfert FTP maintient un débit de 90 paquets/s, qui ne se détériore pas avec temps. Avec $k = 1$, le débit se maintient au niveau de 120 paquets/s. On peut donc conclure que LAS-log peut significativement améliorer le service reçu par les flots TCP longs jugés prioritaires.

### 8.6.3 Modèle au niveau clients

LAS est connu pour favoriser les petits flots mais peut pénaliser les grands flots. Au contraire, FCFS favorise les grands flots au dépend des petits. Dans cette section nous analysons une politique de service hybride, que nous appelons LAS-FCFS qui sert les petits flots suivant la politique LAS et les grands flots suivant la politique FCFS. Plus précisément, LAS-FCFS sert tous les flots qui ont reçu un service inférieur à un seuil suivant la politique LAS puis les sert en mode FCFS lorsqu'ils dépassent le seuil.

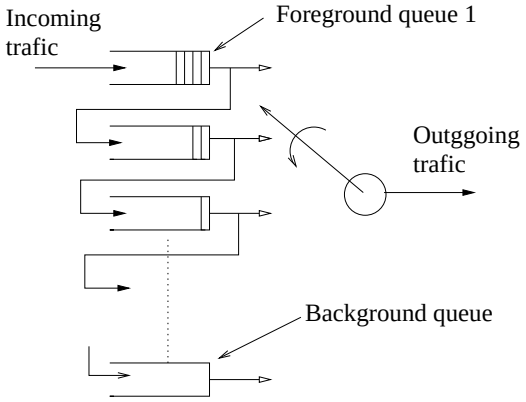



Figure 8.15: *LAS-FCFS model.*

LAS-FCFS au niveau client (i.e. en théorie des files d'attente) est modélisé en utilisant un serveur avec $N$ files d'attente (voir figure 8.15). Les files sont classées en 2 catégories :(*N-1*) *files foreground* et une *file background*. Les clients qui finissent leur service dans l'une des files foreground sont appelés clients foreground ; sinon, ils sont appelés clients background. Le serveur sert en priorité les files foreground et sert la file background si les files foreground sont vides. LAS-FCFS est connu dans la littérature sous le nom de $FB_N$ [24, 65]. En fait, LAS-FCFS a été proposée et analysée pour la première fois dans [24] sous le nom de $FB_N$. La politique LAS est par extension appelé $FB_\infty$ car son modèle au niveau client peut être dérivé de celui de la figure 8.15 en prenant un nombre infini de files [24].

Nous avons tout d'abord étudié numériquement LAS-FCFS pour des flots à forte variabilité et des flots de taille distribuée exponentiellement. Nous avons montré que LAS-FCFS pouvait réduire significativement le temps de réponse dans le cas où la distribution était fortement variable. Pour obtenir le même type de résultat pour le cas exponentiel, nous avons montré qu'il fallait significativement réduire le temps de service total dans les files foreground, ce qui signifie plus de flots courts pénalisés.

Nous avons également proposé et analysé deux politiques de services différenciées dérivées de LAS-FCFS. Comme dans le chapitre précédent, nous avons supposé l'existence de classes de clients, les clients prioritaires et les clients ordinaires. Les deux politiques proposées sont :

- *fixed-priority* LAS-FCFS (**FP-LAS-FCFS**) présentée dans la figure 8.16 où l'on sépare d'abord les flux prioritaires des flux ordinaires. Puis on utilise un ordonnanceur LAS-FCFS pour chaque classe avec priorité à l'ordonnanceur correspondant à la classe prioritaire.

- {*differential* LAS-FCFS (**DF-LAS-FCFS**) présentée dans la figure 8.17 où l'on ne distingue le traitement des flots prioritaires de celui des flots ordinaires qu'au-delà d'un certain seuil.

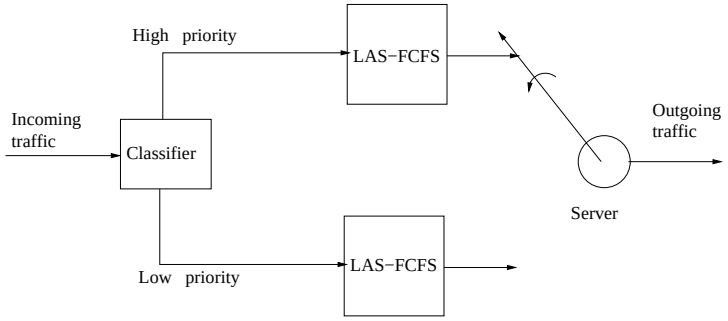



Figure 8.16: *Fixed priority LAS-FCFS architecture.*

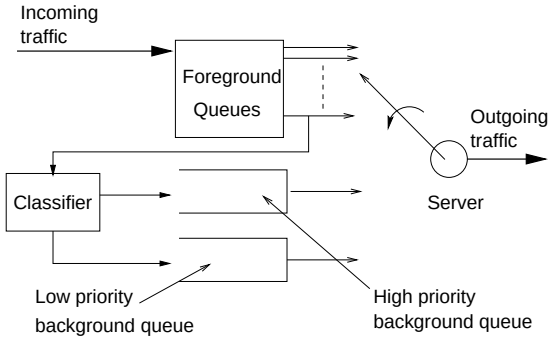



Figure 8.17: *Differential LAS-FCFS architecture.*

Pour chacune de ces politiques, nous avons calculé de manière analytique les temps des réponse moyens conditionnels des différentes classes de flots. Globalement les résultats obtenus montrent que la politique DF-LAS-FCFS offre de meilleures performances car les flots courts ordinaires ou prioritaires ne représentant qu'une faible fraction de la masse totale n'affectent pas notablement les performances des flots longs prioritaires qui sont traités eux en mode FCFS avec priorité sur les flots ordinaires.

## 8.7 Contributions et propositions de recherche future

Cette thèse contient plusieurs contributions. La première contribution est l'analyse de LAS pour une file M/G/1, où G est une distribution avec une forte variabilité. Nous avons comparé LAS avec un large éventail de politique de service : PS, SRPT, les politiques non-préemptives (NPP) telle FIFO, LIFO, etc. Nous avons également étudié LAS dans le cas de surcharge $\rho \geq 1$. De manière générale, nous avons montré que LAS réduit le temps de réponse des clients courts sans pénaliser notablement les clients si tant est que la distribution des tailles de clients est suffisamment variable.

La seconde contribution de cette thèse est l'analyse de performance de LAS dans un réseau paquet et l'étude de l'interaction de LAS avec le protocole TCP. Nous avons simulé LAS avec le simulateur ns-2 et comparé les performances de LAS avec FIFO dans le cas d'un unique goulot d'étranglement pour des tailles de flot fortement variable. Nos simulations ont montré que, comparé à FIFO, LAS diminue fortement le temps de réponse des flots ainsi que leur taux de pertes. Nous avons aussi considéré le cas de flots illimités. Pour ce cas, les simulations ont montré que LAS ne se comportait pas plus mal que FIFO tant que la charge restait modéré ; les performances ne se dégradant que pour des charge proche de la surcharge.

La troisième contribution de cette thèse est l'évaluation de LAS dans les réseau hétérogènes où la bande passante se trouve partagée inéquitablement entre les sources à cause d'une asymétrie en terme de latence des chemins des sources, à cause de protocoles de transport différents ou à cause de l'existence de multiples goulots d'étranglement. Les résultats de simulation montrent que LAS est plus efficace que FIFO dans ces cas car il rétablie toujours l'équité entre les sources.

La quatrième contribution de cette thèse consiste en la conception, la modélisation et l'analyse de politiques de services dérivées de LAS et offrant des services différenciés. L'objectif principal des ces politiques est de protéger les flots longs transportant des données prioritaires. Pour les différentes politiques proposées, nous avons développé des modèles analytiques au niveau flot permettant d'obtenir les temps de réponse conditionnels de ces politiques. Ces modèles ont été validés par simulation.

La dernière contribution de cette thèse est l'étude au niveau client de politiques de services hybrides servant les flots courts suivant LAS et longs suivant FCFS. L'idée est d'associer les bonnes qualités de LAS pour les flots courts et de FCFS pour les flots longs. Nous avons montré que les performances de LAS-FCFS étaient d'autant meilleures que la distribution des tailles des clients était plus variable. Nous avons également proposés et analysés des politiques de service différenciés basées sur LAS-FCFS.

Dans cette thèse, la modélisation de LAS et de ces variantes dans les réseaux paquets était limitée au cas où les pertes étaient faibles. Étendre ces modèles pour prendre en compte des pertes plus importantes fait l'objet d'un travail en cours. De manière plus générale, nous envisageons comme extension de cette thèse d'étudier plus précisément des aspects plus pratiques tels le choix des distributions ou l'implantation de LAS dans un noyau Linux. Un problème afférant à l'implantation de LAS est le lieu de son déploiement. Nous espérons que LAS puisse

être utile aux endroits où des goulots d'étranglement sont susceptibles d'apparaître tels les accès aux réseaux sans fil ou des " data centers ". Il faudra néanmoins veiller à ce que le niveau d'agrégation soit suffisant pour que l'hypothèse de forte variabilité des tailles des flots soit valide.

# Appendix A

# The Proof of Theorem 2.3.4

Proof: Given $f(x) = \mu e^{-\mu x}$ as the density function of exponential distribution we denote the cumulative distribution function as $F(x) = 1 - e^{\mu x}$. Given $x$ as job size, we define $\phi(x) = \rho F(x) = \rho(1 - e^{-\mu x})$ and derivative of $\phi(x)$ with respect to $x$ is denoted as $\phi'(x) = \rho e^{-\mu x} = \rho f(x)$. Hence, by substitution one can easily see that:

$$\int_0^x \frac{f(x)}{(1-\phi(x))^2} dx = \frac{\phi(x)}{(1-\phi(x))}$$

Similarly,

$$m_2(x) = \int_0^x t^2 f(t) dt = -x^2 e^{-\mu x} - \frac{2xe^{-\mu x}}{\mu} + \frac{2}{\mu^2} - \frac{2e^{-\mu x}}{\mu^2}$$

Hence,

$$\lambda[m_2(x) + x^2 F^c(x)] = \lambda[\frac{2}{\mu^2} - \frac{2xe^{-\mu x}}{\mu} - \frac{2e^{-\mu x}}{\mu^2}]$$

and

$$2x(1-\phi(x)) = 2x - \frac{2\lambda x}{\mu} + \frac{2\lambda x e^{-\mu x}}{\mu}$$

which implies that:

$$\begin{aligned} \lambda[m_2(x) + x^2 F^c(x)] + 2x(1-\phi(x)) &= \frac{2\lambda}{\mu^2}(1 - e^{-\mu x}) + 2x(1-\rho) && \text{(A.1)} \\ &= \frac{2}{\mu}\phi(x) + 2x(1-\rho) && \text{(A.2)} \end{aligned}$$

By definition:

$$\begin{aligned} T(x)_{LAS} &= \frac{\lambda[m_2 + x^2 F^c(x)] + 2x(1-\phi(x))}{2(1-\phi(x))^2} \\ &= \frac{\frac{2}{\mu}\phi(x) + 2x(1-\rho)}{2(1-\phi(x))^2} \\ &= \frac{\frac{\phi(x)}{\mu} + x(1-\rho)}{(1-\phi(x))^2} && \text{(A.3)} \end{aligned}$$

Also, we know that

$$\begin{aligned} T_{LAS} &= \int_0^\infty T(x)_{LAS}.f(x)dx \\ &= \int_0^\infty \frac{\frac{\phi(x)}{\mu} + x(1-\rho)}{(1-\phi(x))^2} f(x)dx \\ &= \frac{1}{\mu}\int_0^\infty \frac{\phi(x)}{(1-\phi(x))^2} f(x)dx + (1-\rho)\int_0^\infty \frac{x}{(1-\phi(x))^2} f(x)dx \end{aligned} \tag{A.4}$$

Integration by parts with $u(x) = \phi(x)$ and $dv = \frac{f(x)}{(1-\phi(x))^2}dx$, we get

$$\int_0^\infty \frac{\phi(x)f(x)}{(1-\phi(x))^2}dx = \frac{1}{(1-\rho)} + \frac{1}{\rho}ln(1-\rho) \tag{A.5}$$

Similarly, with $u(x) = x$ and $dv = \frac{f(x)}{(1-\phi(x))^2}$, integration by parts yields

$$\int_0^A \frac{f(x)}{(1-\phi(x))^2}dx = x\frac{\frac{1}{\rho}}{1-\phi(x)}|_0^A - \frac{1}{\rho}\int_0^A \frac{1}{(1-\phi(x))}dx. \tag{A.6}$$

In the above equation, we integrate between 0 and $A > 0$ and not between 0 and $\infty$ because $\lim_{A\to\infty} x\frac{\frac{1}{\rho}}{1-\phi(x)} = \infty$. We will see at the end of the computation that this term cancels out with its opposite. We integrate the second part of the above equation by substitution. We let $v = F(x)$, that is $\frac{dv}{dx} = \mu e^{-\mu x}$, i.e., $dx = \frac{dv}{\mu(1-v)}$. Hence

$$\int_0^A \frac{1}{(1-\phi(x))} = \int_0^c \frac{dv}{\mu(1-\rho v)(1-v)} \tag{A.7}$$

, where $c = F(A)$. Partial fraction integration then yields

$$\begin{aligned} \int_0^c \frac{1}{\mu(1-\rho v)(1-v)}dv &= \frac{-\rho}{\mu(1-\rho)}\int_0^c \frac{1}{(1-\rho v)}dv + \frac{1}{\mu(1-\rho)}\int_0^c \frac{dv}{1-v} \\ &= \frac{1}{\mu(1-\rho)}ln(1-\rho v)|_0^c - \frac{1}{\mu(1-\rho)}ln(1-v)|_0^c \\ &= \frac{ln(1-c\rho)}{\mu(1-\rho)} - \frac{ln(1-c)}{\mu(1-\rho)} \qquad \text{(A.8)} \\ &= \frac{ln(1-c\rho)}{\mu(1-\rho)} + \frac{A}{1-\rho} \qquad \text{(A.9)} \end{aligned}$$

The last term follows from $ln(1-c) = ln(1-F(A)) = ln(e^{-\mu A}) = -\mu A$. Note that this last term will cancel with the first term of Equation A.6. Putting all parts together in Equation A.4, we get

$$\int_0^\infty T(x)_{LAS}f(x)dx = \frac{1/\mu}{(1-\rho)}$$

The mean of the exponential random variable $X$ is given as $\overline{X} = \frac{1}{\mu}$, hence $T_{LAS}$ is:

$$\int_0^\infty T(x)_{LAS}f(x)dx = \frac{\overline{X}}{1-\rho} = T_{PS} \tag{A.10}$$

□

# Appendix B

# The Proof of Theorem 2.3.5

Proof: Since $\rho(x) \leq \phi(x) \leq \rho$, we obtain directly from Equations (2.1) and (2.2) and Equations (2.3) and (2.4): $\tilde{R}(x)_{LAS} \geq R(x)_{SRPT}$ and $\tilde{W}(x)_{LAS} \geq W(x)_{SRPT}$. Hence the left-hand side inequality. We begin the proof of the right-hand side inequality by studying $\tilde{R}(x)_{LAS} - R(x)_{SRPT}$:

$$\begin{aligned}
\tilde{R}(x)_{LAS} - R(x)_{SRPT} &= \frac{x}{1-\phi(x)} - \int_0^x \frac{dt}{1-\rho(t)} \\
&= \int_0^x \frac{dt}{1-\phi(x)} - \int_0^x \frac{dt}{1-\rho(t)} \\
&= \int_0^x \frac{\phi(x)-\rho(t)}{1-\phi(x)} \frac{1}{1-\rho(t)} dt
\end{aligned}$$

Applying the Chebyshev integral inequality [71] to $f(t) = \frac{\phi(x)-\rho(t)}{1-\phi(x)}$, which is a decreasing function of $t$ and $g(t) = \frac{1}{1-\rho(t)}$, which is an increasing function of $t$, we get

$$\tilde{R}(x)_{LAS} - R(x)_{SRPT} \leq \frac{1}{x} \int_0^x \frac{\phi(x)-\rho(t)}{1-\phi(x)} dt . \underbrace{\int_0^x \frac{dt}{1-\rho(t)}}_{R(x)_{SRPT}} \tag{B.1}$$

Besides:

$$\begin{aligned}
\frac{1}{x}\int_0^x \frac{\phi(x)-\rho(t)}{1-\phi(x)}dt &= \Big([t(\phi(x)-\rho(t)]_0^x + \int_0^x \lambda t^2 f(t)dt\Big) \\
&= \frac{1}{x(1-\phi(x))}(x(\lambda x F^c(x)) \\
&\quad +\lambda m_2(x)) \\
&= \frac{\lambda}{x(1-\phi(x))}(x^2F^c(x)+m_2(x)) \\
&\leq \frac{\lambda}{x(1-\phi(x))}(x^2F^c(x) \\
&\quad +x\int_0^x tf(t)dt) \\
&\quad \text{since} \int_0^x t^2 f(t)dt \leq x\int_0^x tf(t)dt \\
&= \frac{x\phi(x)}{x(1-\phi(x))} \\
&= \frac{\phi(x)}{1-\phi(x)} \qquad \text{(B.2)}
\end{aligned}$$

Using Equation (B.2) in Equation (B.1), one obtains:

$$\begin{aligned}
\tilde{R}(x)_{LAS} &\leq \Big(1+\frac{\phi(x)}{(1-\phi(x))}\Big)R(x)_{SRPT} \\
&= \frac{1}{1-\phi(x)}R(x)_{SRPT} \qquad \text{(B.3)}
\end{aligned}$$

Consider now $\tilde{W}(x)_{LAS}$ and $W(x)_{SRPT}$, we have:

$$\tilde{W}(x)_{LAS} = \frac{(1-\rho(x))^2}{(1-\phi(x))^2}W(x)_{SRPT} \qquad \text{(B.4)}$$

Adding Equations (B.3) and (B.4), we obtain:

$$\begin{aligned}
T(x)_{LAS} &\leq \max\Big(\frac{(1-\rho(x))^2}{(1-\phi(x))^2}, \frac{1}{1-\phi(x)}\Big)T(x)_{SRPT} \\
&= \frac{1}{1-\phi(x)}\max\Big(\frac{(1-\rho(x))^2}{1-\phi(x)}, 1\Big)T(x)_{SRPT}
\end{aligned}$$

Since $(1-\rho(x))^2 = 1-2\rho(x)+\rho(x)^2 = 1-\rho(x)+\underbrace{\rho(x)^2-\rho(x)}_{=\rho(x)(1-\rho(x))\geq 0}$ and $1-\phi(x) = 1-\rho(x)-\underbrace{xF^c(x)}_{\geq 0}$, then $\frac{(1-\rho(x))^2}{1-\phi(x)} \geq 1$ and the right-hand side of the theorem follows directly from Equation (B.5). □

# Bibliography

[1] "The Network Simulator ns2", http://www.isi.edu/nsnam/ns/.

[2] A. Aggarwal, S. Savage, and T. Anderson, "Understanding the Performance of TCP Pacing", In *Proceedings of IEEE INFOCOM*, Tel Aviv, Israel, March 2000.

[3] J. Aikat et al., "Variability in TCP Round-trip Times", In *Internet Measurement Conference 2003*, October 2003.

[4] "Akamai", http://www.akamai.com.

[5] M. Allman et al., "An Evaluation of TCP with Larger Initial Windows", *ACM Computer Communication Review*, 28(3):41–52, July 1998.

[6] M. Allman et al., "Increasing TCP's Initial Window", Request for Comments RFC 2414, Internet Engineering Task Force, September 1998.

[7] M. Allman, V. Paxson, and W. Stevens, "TCP Congestion Control", RFC 2581, Internet Engineering Task Force, April 1999.

[8] V. Almeida, A. Bestavros, M. Crovella, and A. de Oliveira, "Characterizing Reference Locality in the WWW", In *Proceedings of the Fourth International Conference on Parallel and Distributed Information Systems (PDIS'96)*, pp. 92–103, 1996.

[9] K. Avrachenkov, U. Ayesta, and P. Brown, "Batch Arrival Processor Sharing with Application to Multilevel Processor Sharing Scheduling", INRIA Technical Report RR-5043, 2003.

[10] K. Avrachenkov, U. Ayesta, P. Brown, and N. Nyberg, "Differentiation between Short and Long TCP flows: Predictability of the response time", In *Proc. IEEE INFOCOM*, March 2004.

[11] H. F. Badran, "Service Provider Networking Infrastructures with MPLS", In *Proceedings of the Sixth Symposium on Computers and Communications*, pp. 312–318, Hammamet, Tunisia, 2001.

[12] H. Balakrishnan, V. Padmanabhan, S. Seshan, S. M., and R. Katz, "TCP Behavior of a Busy Internet Server: Analysis and Improvements", In *Proc. Infocom 98*.

[13] N. Bansal and M. Harchol-Balter, "Analysis of SRPT Scheduling: Investigating Unfairness", In *Sigmetrics 2001 / Performance 2001*, pp. 279–290, June 2001.

[14] P. Barford and M. E. Crovella, "Generating Representative Web Workloads for Network and Server Performance Evaluation", In *Proceedings of Performance'98/SIGMETRICS'98*, pp. 151–169, July 1998.

[15] Y. Bernet, F. Baker, P. Ford, R. Yavatkar, and L. Zhang, "A Framework for End-to-End QoS Combining RSVP/Intserv and Differentiated Services", Internet-Draft <draft-bernet-intdiff-00.txt>, 2000.

[16] R. Bhagwan and B. Lin, "Fast and Scalable Priority Queue Architecture for High-Speed Network Switches", In *Proc. IEEE INFOCOM*, pp. 538–547, 2000.

[17] R. Braden, L. Zhang, S. Berson, S. Herzog, and S. Jamin, "Resource ReSerVation Protocol (RSVP)- Version 1 Functional Specification", Request for Comments RFC 2205, Internet Engineering Task Force, September 1997.

[18] L. S. Brakmo, S. W. O'Malley, and L. L. Peterson, "TCP Vegas: New techniques for congestion detection and avoidance", In *Proceedings of the ACM SIGCOMM Conference*, London, England, 1994.

[19] L. Breslau et al., "Comments on the Performance Measurement-Based Control Algorithms", In *IEEE INFOCOM 2000*, pp. 1233–1242, 2000.

[20] "Castify Networks", http://www.castify.net.

[21] X. Chen and J. Heidemann, "Preferential Treatment for Short Flows to Reduce Web Latency", *Computer Networks: International Journal of Computer and Telecommunication Networking*, 41(6):779–794, 2003.

[22] K. Claffy, G. Miller, and K. Thompson, "The Nature of the Beast: Recent Traffic Measurements from an Internet Backbone", In *Proceedings of INET*, July 1998.

[23] E. G. Coffman and P. J. Denning, *Operating Systems Theory*, Prentice-Hall Inc., 1973.

[24] E. G. Coffman and L. Kleinrock, "Feedback Queueing Models for Time-Shared Systems", *Journal of ACM (JACM)*, 15(4):549–576, October 1968.

[25] R. W. Conway, W. L. Maxwell, and L. W. Miller, *Theory of Scheduling*, Addison-Wesley Publishing Company, 1967.

[26] M. Crovella and A. Bestavros, "Self-Similarity in World Wide Web Traffic: Evidence and Possible Causes", *IEEE/ACM Transactions on Networking*, pp. 835–846, December 1997.

[27] M. Crovella, R. Frangioso, and M. Harchol-Balter, "Connection Scheduling in Web Servers", In *USENIX Symposium on Internet Technologies and Systems (USITS '99)*, pp. 243–254, Boulder, Colorodo, October 1999.

[28] M. E. Crovella, "Performance Evaluation with Heavy Tailed Distributions", In *Job Scheduling Strategies for Parallel Processing (JSSPP)*, pp. 1–10, 2001.

[29] M. E. Crovella, M. Harchol-Balter, and C. D. Murta, "On Choosing a Task Assignment Policy for a Distributed Server System", In *ACM SIGMETRICS (poster paper)*, July 1998.

[30] M. E. Crovella, M. Harchol-Balter, and C. D. Murta, "Task Assignment in a Distributed System: Improving Performance by Unbalancing Load", In *ACM SIGMETRICS (poster paper)*, July 1998.

[31] M. E. Crovella and L. Lipsky, "Long-lasting Transient Conditions in Simulations with Heavy-tailed Workloads", In *Proc. of Winter Simulation Conference*, pp. 1005–1012, 1997.

[32] A. Demers, S. Keshav, and S. Shenker, "Analysis and Simulation of a Fair Queueing Algorithm", In *Proc. of ACM SIGCOMM'89*, pp. 1–12, Austin, Texas, September 1989.

[33] "Differentiated Services (diffserv) Charter", http://www.ietf.org/html.charters/diffserv-charter.html.

[34] A. B. Downey, "The structural cause of file size distributions", In *Proc. MASCOTS 20001*, August 2001.

[35] V. Elek, G. Karlsson, and R. Ronngren, "Admission Control Based on End-to-End Measurement", In *IEEE INFOCOM 2000*, pp. 623–630, 2000.

[36] K. Fall and S. Floyd, "Simulation-based Comparisons of Tahoe, Reno, and SACK TCP", *Computer Communication Review*, 26(3):5–21, June 1996.

[37] F. L. Faucheur, "Requirements for Support of Differentiated Services-aware MPLS Traffic Engineering", Request for Comments RFC 3564, Internet Engineering Task Force, July 2003.

[38] A. Feldmann, A. Gilbert, P. Huang, and W. Willinger, "Dynamics of IP Traffic: A Study of the Role of Variability and the Impact of Control", In *ACM SIGCOMM*, August 1999.

[39] A. Feldmann, J. Rexford, and R. Caceres, "Efficient Policies for Carrying Web Traffic Over Flow-Switched Networks", 6(6):673–685, December 1998.

[40] H. Feng and M. Misra, "Mixed Scheduling Disciplines for Network Flows", In *The Fifth Workshop of Mathematical Performance Modeling and Analysis (MAMA 2003), San Diego, California, USA*, 2003.

[41] M. J. Fischer, D. Gross, D. M. B. Masi, and J. F. Shortle, "Analyzing the Waiting Time Process in Internet Queueing Systems With the Transform Approximation Method", *The Telecommunications Review*, pp. 21–32, 2001.

[42] S. Floyd, "Connections with Multiple Congested Gateways in Packet-Switched Networks", *ACM Computer Communication Review*, 21(5):30–47, October 1991.

[43] S. Floyd and V. Jacobson, "Random Early Detection Gateways for Congestion Avoidance", *IEEE/ACM Transactions on Networking*, 1(4):397–413, August 1993.

[44] C. Fraleigh, S. Moon, B. Lyles, C. Cotton, M. Khan, D. Moll, R. Rockell, T. Seely, and C. Diot, "Packet-level Traffic Measurement from the Sprint IP Backbone", *IEEE Network Magazine*, November 2003.

[45] S. B. Fredj, T. Bonald, A. Proutiére, G. Réginié, and J. W. Roberts, "Statistical Bandwidth sharing: a Study of Congestion at Flow Level", In *Proc. ACM SIGCOMM*, August 2001.

[46] E. J. Friedman and S. G. Henderson, "Fairness and Efficiency in Web Servers", In *Proc. ACM SIGMETRICS*, pp. 229–237, June 2003.

[47] W. Gong, Y. Liu, V. Misra, and D. Towsley, "On the Tails of Web File Size Distributions", In *Proc. of 39-th Allerton Conference on Communication, Control, and Computing*, October 2001.

[48] L. Guo and I. Matta, "The War between Mice and Elephants", In *Proc. 9th IEEE International Conference on Network Protocols (ICNP)*, Riverside, CA, 2001.

[49] L. Guo and I. Matta, "Differentiated Control of Web Traffic: A Numerical Analysis", In *SPIE ITCOM'2002: Scalability and Traffic Control in IP Networks*, August 2002.

[50] L. Guo and I. Matta, "Scheduling Flows with Unknown Sizes: Approximate Analysis", In *Proc. ACM SIGMETRICS*, pp. 276–277, June 2002.

[51] M. Harchol-Balter, B. Bansal, and M. Agrawal, "Implementation of SRPT Scheduling in Web Servers", In *IPDS 2001*, pp. 1–24, 2001.

[52] M. Harchol-Balter and A. Downey, "Exploiting Process Lifetime Distributions for Dynamic Load Balancing", *ACM Transactions on Computer Systems*, 15(3):253–285, 1997.

[53] 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, May 2003.

[54] M. Harchol-Balter, K. Sigman, and A. Wierman, "Asymptotic Convergence of Scheduling Policies with respect to Slowdown", *Performance Evaluation*, 49:241–256, September 2002.

[55] M. Harchol-Balter, "The Effect of Heavy-Tailed Job Size Distributions on Computer System Design", In *Proc. of ASA-IMS Conf. on Applications of Heavy Tailed Distributions in Economics*, June 1999.

[56] G. Hasegawa and M. Murata, "Survey of Fairness Issues in TCP Control Mechanisms", *IEICE Trans. on Communications*, E84-B(6):1461–1472, June 2001.

[57] J. Heinanen, F. Baker, W. Weiss, and J. Wroclawski, "Assured Forwarding PHB Group", Request for Comments RFC 2297, Internet Engineering Task Force, June 1999.

[58] J. Hoe, "Improving Start-up Behavior of a Congestion Scheme for TCP", In *ACM SIGCOMM*, pp. 270–280, 1996.

[59] "Integrated Services (intserv) Charter", http://www.ietf.org/html.charters/intserv-charter.html.

[60] S. Iyer, S. Bhattacharyya, N. Taft, N. McKeown, and C. Diot, "An Approach to Alleviate Link Overload as Observed on an IP backborne", In *INFOCOM*, April 2003.

[61] V. Jacobson, K. Nichols, and K. Poduri, "An Expedited Forwarding PHB", Request for Comments RFC 2298, Internet Engineering Task Force, June 1999.

[62] J. Ke, "Towards a Rate-based TCP Protocol for the Web", In *International Workshop on Modeling, Analysis and Simulation of Computer and Telecommunications Systems - MASCOTS*, pp. 36–45, October 2000.

[63] S. Keshav, *An Engineering Approach to Computer Networking*, Prentice Hall, 1st edition, 1997.

[64] L. Kleinrock, *Queuing Systems, Volume II: Computer Applications*, Wiley, New York, 1976.

[65] H. Krayl, E. J. Neuhold, and C. Unger, *Grundlagen der Betriebssysteme*, Walter de Gruyter, Berlin, New York, 1975.

[66] J. F. Kurose and K. W. Ross, *Computer Networks: A Top-Down Approach Featuring the Internet*, Addison Wesley, 2nd edition edition, 2003.

[67] D. Lin and R. Morris, "Dynamics of Random early Detection", In *Proc. of the ACM SIGCOMM*, pp. 127–137, October 1997.

[68] B. A. Mah, "An Emperical Model for HTTP Network Traffic", In *IEEE INFOCOM 1997*, pp. 592–600, April 1997.

[69] I. Matta and L. Guo, "Differentiated Predictive Fair Service for TCP Flows", In *Proc. 8th IEEE International Conference on Network Protocols (ICNP'00)*, pp. 49–57, Osaka, Japan, 2000.

[70] D. E. McDysan and D. L. Spohn, *ATM: Theory and Application*, McGraw Hill, New York, 1995.

[71] D. S. Mitrinovic, *Analytic Inequalities*, Springer-Verlag, 1970.

[72] S. Moon, J. Rexford, and K. G. Shin, "Scalable Hardware Priority Queue Architectures for High-Speed Packet Switches", *IEEE Transactions on Computers*, 49(11):1215–1227, November 2000.

[73] "Multiprotocol Label Switching (mpls) Charter", http://www.ietf.org/html.charters/mpls-charter.html.

[74] K. Muthukrishnan et al., "A Core MPLS IP VPN Architecture", Request for Comments RFC 2917, Internet Engineering Task Force, September 2000.

[75] S. Muthukrishnan, R. Rajaraman, A. Shaheen, and J. E. Gehrke, "Online Scheduling to Minimize Average Stretch", In *40th Annual Symposium on Foundation of Computer Science*, pp. 433–442, 1999.

[76] J. Nagle, "On packet switches with infinite storage", *IEEE Transactions on Communications*, COM-35(4):435–438, April 1987.

[77] K. Nichols, V. Jacobson, and L. Zhang, "Two-bit Differentiated Services Architecture for the Internet", Request for Comments RFC 2638, Internet Engineering Task Force, June 1999.

[78] M. Nuijens, "The Foreground-Background Queue", PhD Dessertation, University of Amsterdam, May 2004.

[79] J. Padhye, V. Firoiu, D. Towsley, and J. Kurose, "Modeling TCP Throughput: A Simple Model and its Empirical Validation", In *Proc. of ACM SIGCOMM'98*, pp. 303–314, Vancouver, Canada, August 1998.

[80] V. N. Padmanabhan, L. Qiu, and H. J. Wang, "Server-based Inference of Internet Link Lossiness", In *Proc. IEEE INFOCOM*, 2003.

[81] Padmanabhan, V. and R. Katz, "TCP Fast Start: a Technique for Speeding Up Web Transfers", In *Globecom Internet Mini-Conference*, November 1998.

[82] A. K. Parekh and R. G. Gallager, "A Generalized Processor Sharing Approach to Flow Control in Integrated Services Networks", In *Proc. IEEE INFOCOM'92*, pp. 915–924, 1992.

[83] V. Paxson and M. Allman, "Computing TCP's Retransmission Timer", Request for Comments RFC 2988, Internet Engineering Task Force, November 2000.

[84] V. Paxson and S. Floyd, "Wide-Area Traffic: The Failure of Poisson Modelling", *IEEE/ACM Transactions on Networking*, 3:226–244, June 1995.

[85] E. Rosen and Y. Rekhter, "BGP/MPLS VPN", Request for Comments RFC 2547, Internet Engineering Task Force, March 1999.

[86] "Agere Systems", http://www.agere.com.

[87] R. Schassberger, "A Detailed Steady State Analysis of the M/G/1 Queue under various Time Sharing Disciplines", In P. J. C. In O.J. Boxma G. Iazeolla, editor, *Computer Performance and Reliability*, pp. 431–442, North Holland, 1987.

[88] R. Schassberger, "The Steady State Distribution of Spent Service Times Present in the M/G/1 Foreground-Background Processor-sharing Queue", *Journal of Applied Probability*, 25(7):194–203, 1988.

[89] L. E. Schrage, "The Queue M/G/1 with Feedback to Lower Priority Queues", *Management Science*, 13(7):466–474, 1967.

[90] L. E. Schrage, "A Proof of the Optimality of the Shortest Remaining Service Time Discipline", *Operations Research*, 16:670–690, 1968.

[91] L. E. Schrage and L. W. Miller, "The Queue M/G/1 with the Shortest Processing Remaining Time Discipline", *Operations Research*, 14:670–684, 1966.

[92] M. Schwartz, *Broadband Integrated Networks*, Prentice Hall, 1996.

[93] A. Shaikh, J. Rexford, and K. G. Shin, "Load-sensitive Routing of Long-lived IP Flows", In *Proc. ACM SIGCOMM*, pp. 215–226, September 1999.

[94] M. Shreedhar and G. Varghese, "Efficient Fair Queueing using Deficit Round Robin", *ACM Computer Communication Review*, 25(4):231–242, October 1995.

[95] J. Shudong, G. Liang, I. Matta, and A. Bestavros, "A Spectrum of TCP-friendly Window-based Congestion Control Algorithms", *IEEE/ACM Transactions of Networking*, 11(3).

[96] F. D. Smith, F. H. Campos, K. Jeffay, and D. Ott, "What TCP/IP Protocol Headers Can Tell us about the Web", In *Proc. ACM SIGMETRICS*, pp. 245–256, June 2001.

[97] H. D. Tan, D. L. Eager, M. K. Vernon, and H. Guo, "Quality of Service Evaluations of Multicast Streaming Protocols", In *Proc. ACM SIGMETRICS*, pp. 183–194, June 2002.

[98] "Traffic Engineering (tewg) Charter", http://www.ietf.org/html.charters/tewg-charter.html.

[99] K. Tutschku, "A Measurement-based Traffic Profile of the eDonkey Filesharing Service", In *Proc. Passive and Active Measurements (PAM) Workshop*, April 2004.

[100] J. van der Merwe, S. Sen, and C. Kalmanek, "Streaming Video Traffic: Characterization and Network Impact", In *Proc. of the Seventh International Web Content Caching and Distribution Workshop*, August 2002.

[101] J. Walrand and P. Varaiya, *High Performance Communication Networks*, Morgan Kaufmann Publishers, 2000.

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

[103] A. Wierman and M. Harchol-Balter, "Classifying Scheduling Policies with Respect to Unfairness in an M/GI/1", In *Proc. ACM SIGMETRICS*, pp. 238–249, June 2003.

[104] C. Williamson and Q. Wu, "A Case for Context-Aware TCP/IP", *Performance Evaluation Review*, 29(4):11–23, March 2002.

[105] R. W. Wolff, "Time Sharing with Priorities", *SIAM Journal of Applied Mathematics*, 19(3):566–574, 1970.

[106] X. Xiao and L. M. Ni, "Internet QoS: A Big Picture", *IEEE Network*, pp. 8–18, March/April 1999.

[107] S. Yang and G. Veciana, "Size-based Adaptive bandwidth Allocation: Optimizing the Average QoS for Elastic Flows", In *INFOCOM*, 2002.

[108] S. Yilmaz and I. Matta, "On Class-Based Isolation of UDP, Short-Lived and Long-Lived TCP Flows", In *International Workshop on Modeling, Analysis and Simulation of Computer and Telecommunications Systems - MASCOTS*, August 2001.

[109] H. Zhang, "Service Disciplines For Guaranteed Performance Service in Packet-Switching Networks", *IEEE Proceedings*, October 1995.

[110] P. Zhang and R. Kantola, "Building MPLS VPNs with QoS Routing Capability", In *Fifth International Symposium on Interworking*, pp. 292–301, October 2000.

[111] Z. Zhang et al., "Decoupling QoS Control from Core Routers: A Novel Bandwidth Broker Architecture for Scalable Support of Guaranteed Services", In *ACM SIGCOMM*, 2000.

[112] Zhang, Y. et al., "Speeding up Short Data Transfers", In *NOSSDAV*, June 2000.

# Related Publications

- "Performance modeling of LAS based scheduling policies in packet switched networks", *Proceedings of ACM Sigmetrics, June, 2004, New York USA.*
- "LAS scheduling approach to avoid bandwidth hogging in heterogeneous TCP networks", *7th IEEE International Conference on High Speed Networks and Multimedia communications HSNMC'04, July 2004, Toulouse, France.*
- "Analysis of LAS scheduling for job size distributions with high variance", *Proceedings of ACM Sigmetrics, June 2003, Sandiego, USA.*
- "Size-based scheduling with differentiated services to improve response time of highly varying flows", *5th ITC Specialist Seminar, Internet Traffic Engineering and Traffic Management, July 2002, Wurzburg, Germany.*
- "LAS scheduling to improve the performance of short TCP flows, *To appear for a journal publication 2004.*