uOttawa

L'Université canadienne
Canada's university

FACULTÉ DES ÉTUDES SUPÉRIEURES ET POSTOCTORALES

uOttawa

L'Université canadienne
Canada's university

FACULTY OF GRADUATE AND POSDOCTORAL STUDIES

Heriniaina Philibert Andrianirina

AUTEUR DE LA THÈSE / AUTHOR OF THESIS

M.A.Sc. (Electrical Engineering)

GRADE / DEGREE

School of Information Technology and Engineering

FACULTÉ, ÉCOLE, DÉPARTEMENT / FACULTY, SCHOOL, DEPARTMENT

PF-OBS: a Proportionally Fair Congestion Avoidance Algorithm for Optical Burst Switching Networks

TITRE DE LA THÈSE / TITLE OF THESIS

D. Makrakis

DIRECTEUR (DIRECTRICE) DE LA THÈSE / THESIS SUPERVISOR

CO-DIRECTEUR (CO-DIRECTRICE) DE LA THÈSE / THESIS CO-SUPERVISOR

EXAMINATEURS (EXAMINATRICES) DE LA THÈSE / THESIS EXAMINERS

J. Talim

T. Hall

Gary W. Slater

Le Doyen de la Faculté des études supérieures et postdoctorales / Dean of the Faculty of Graduate and Postdoctoral Studies

PF-OBS: a Proportionally Fair Congestion Avoidance Algorithm for Optical Burst Switching Networks

Heriniaina Philibert Andrianirina

Thesis submitted to the
Faculty of Graduate and Postdoctoral Studies
In partial fulfillment of the requirements for the degree of
Master of Applied Science in Electrical Engineering

Ottawa-Carleton Institute for Electrical and Computer Engineering
School of Information Technology and Engineering
University of Ottawa
Ottawa, Ontario, Canada

Your file Votre référence
ISBN: 978-0-494-50851-0
Our file Notre référence
ISBN: 978-0-494-50851-0

To my family

ABSTRACT

Optical Burst Switching (OBS) is an optical switching technique which eliminates the electronic bottleneck by keeping transmitted data in the optical domain. Therefore, it can effectively utilize the large bandwidth available in optical fibres. For these reasons, OBS is considered the future of optical networks. However, burst loss caused by congestion is a significant problem in OBS because flexible, practical optical buffers currently do not exist.

This thesis proposes a rate-based congestion control method, called PF-OBS, which limits the blocking probability in OBS networks. PF-OBS uses a utility maximization approach to congestion control. The method assigns prices to congested links. Ingress nodes receive bandwidth proportionally to the price they are willing to pay. The resulting rate allocation is proportionally fair.

The method can keep the traffic load on all links under a predefined level. Simulations show that PF-OBS maintains the overall blocking probability at lower levels as compared with plain OBS. In addition, PF-OBS achieves higher total throughput than existing OBS flow control methods. The method remains stable in the presence of propagation delays.

ACKNOWLEDGMENTS

I would like to express my deepest gratitude and appreciation to my supervisor, Professor Dimitrios Makrakis, for his guidance, patience, and support in preparing this thesis. During the past years, I greatly benefited from his insight and knowledge. I am also indebted to my co-supervisor, Professor Nicolas Georganas, who is always available for advice and assistance. My special thanks to Professor Luis Orozco-Barbosa who encouraged me to pursue graduate studies in computer networking.

I would like to thank Ognian Kabranov for helping me kick-start my research on OBS. I am thankful to Yixin for his assistance with OPNET, and to Miguel for the discussions regarding self-similar traffic.

My fondest gratitude goes to my family, whose prayers and encouragements are what keep me going. This work is dedicated to them. I would also like to give my heartfelt thanks to André, Suzanne, and Léandre for their unfailing support, *merci.*

Finally, I would like to thank the University of Ottawa, School of Information Technology and Engineering, for providing me with financial support.

Sincerely,

Heriniaina Andrianirina

Table of Contents

List of Figures

List of Tables

List of Abbreviations

$\boldsymbol{A}$	Routing matrix
A_{jr}	Elements of routing matrix. $A_{jr} = 1$ if resource $j \in R$. Otherwise, $A_{jr} = 0$
ABR	Available Bit Rate
AIMD	Additive Increase Multiplicative Decrease
AQM	Active Queue Management
ATM	Asynchronous Transfer Mode
BER	Bit error rate
BES	Bigger eats the smaller
BHP	Burst Header Packet
BORA	Burst Overlap Reduction Algorithm
BORA-FS	Burst Overlap Reduction Algorithm with Fixed-ordered Search
$\boldsymbol{C}$	Vector of elements C_j
CI	Congestion Indication bit
Cj	Capacity of resource j
C_l	Capacity of link l
$\hat{C}_l$	Maximum load allowed in link l
CV	Coefficient of Variation
CWDM	Coarse Wavelength Division Multiplexing
D_r	Round trip delay on route r
DWDM	Dense Wavelength Division Multiplexing
ECN	Explicit Congestion Notification
EDFA	Erbium Doped Fibre Amplifier
ERICA	Explicit Rate Indication for Congestion Avoidance
FDL	Fibre Delay Line
J	The set of resources in the network
j	One resource in the network. $j \in J$
JET	Just Enough Time
JIT	Just In Time

KKT	Karush-Kuhn-Tucker
L	The set of links in a network
l	A link in the network. $l \in L$
LAUC	Latest Available Unused Channel
LAUC-VF	Latest Available Unused Channel with Void Filling
LRD	Long Range Dependent
MAN	Metropolitan Area Network
Min-SV	Minimum Starting Void
MPLS	Multiprotocol Label Switching
n	Time in discrete domain
NACK	Negative Acknowledgment
OBS	Optical Burst Switching
OCS	Optical Circuit Switching
OEO	Optical-Electronic-Optical
OPS	Optical Packet Switching
$\hat{P}$	Maximum tolerated blocking probability at OBS switches
PF-OBS	Proportionally Fair OBS
$p_j(y)$	Per bandwidth price of resource j, function of load y
$Q_j(y)$	Total cost of resource j per time unit, for a load y
QoS	Quality of Service
R	The set of routes (users) in a network
r	One route (user) in the network. $r \in R$
RED	Random Early Detection
RTT	Round Trip Time
s	Designates a source-destination pair
SACK	Selective Acknowledgment
SCU	Switch Control Unit
SDH	Synchronous Digital Hierarchy
SONET	Synchronous Optical Network
SRD	Short Range Dependent
t	Time in continuous domain

T	Timeslot duration for congestion control
TAG	Tell and Go
TCP	Transmission Control Protocol
TWC	Tuneable Wavelength Converters
UDP	User Datagram Protocol
U_r	Utility function for user r
VoIP	Voice over IP
VPN	Virtual Private Network
WDM	Wavelength Division Multiplexing
$\boldsymbol{x}$	Vector of elements x_r
$x_r(t)$	Rate allocated to user r
$\boldsymbol{y}$	Vector of resource loads y_j
y_j	Total load on resource j
α	Maximum delay for BORA scheduler
κ_r	Rate Adjustment gain for route r
λ_r	Per bandwidth charge in the maximization problem for user r
$\mu_j(t)$	Per bandwidth price of resource j, function of time t
ω_r	Price user r is willing to pay

Chapter 1

Introduction

1.1 Background

There is a constant need for more data bandwidth in communication networks today. Global demand for bandwidth increased 42 percent in 2004 [1-3], following a remarkable 62 percent rise in 2003 [2, 3]. This demand is driven by the rapid growth and popularity of the Internet. The number and variety of applications running on this network expands continuously, with growing emphasis on bandwidth-intensive applications. In response to this explosive growth in global bandwidth demand, there is growing motivation to move to optical networking. It is generally accepted that future wired networks will move towards an all-optical solution because of the unmatched amount of bandwidth available in a single fibre. The use of multiple light frequencies in each fibre further increases the total bandwidth available in each strand. This technique is called Wavelength Division Multiplexing (WDM). Each frequency provides a separate transmission channel in this case. From this point forward, we use the terms *channel* and *wavelength* interchangeably. The multiplexing level achieved in current Dense WDM (DWDM) is over 100 wavelengths per fibre. The total data bandwidth achievable on a single fibre, with *currently available* technology, exceeds 1000 x 40 Gbps [4].

While optical networks offer a substantial amount of bandwidth, any data requiring transmission must be converted from electrical to optical at the source. The destination then converts the optical signal back to an electrical signal for further processing. The maximum channel data rate is presently restricted to around 40 Gbps because the electronic circuits required for the conversions cannot run any faster [4].

Consequently, it is essential to avoid any electronic processing inside an optical network. Optical Packet Switching (OPS) is an excellent solution for an all-optical network. OPS offers the flexibility of current electronic switching technologies such as ATM. It provides statistical multiplexing of individual packets while keeping transiting data in the optical domain. Nevertheless, required technology such as optical logic circuits and optical buffers are not mature enough for an implementation in the near future. At the moment, only a limited form of optical buffering is available using Fibre Delay Lines (FDL). An FDL is simply a long piece of optical fibre in which a signal circulates. The delay results from the propagation time inside the fibre section. FDLs are bulky and offer only a limited buffering time.

1.2 Optical Burst Switching

A new paradigm for optical networking, called Optical Burst Switching (OBS), was proposed to balance the present limitations of optical logic circuits and the need for a flexible switching technique [5, 6]. In this new paradigm, data bursts remain in the optical domain, avoiding optical-electronic-optical (OEO) conversion bottlenecks. Therefore, OBS provides high data rates and true statistical multiplexing. This is achieved through a balance of OPS and optical circuit switching (OCS) characteristics as shown on Figure 1.1.

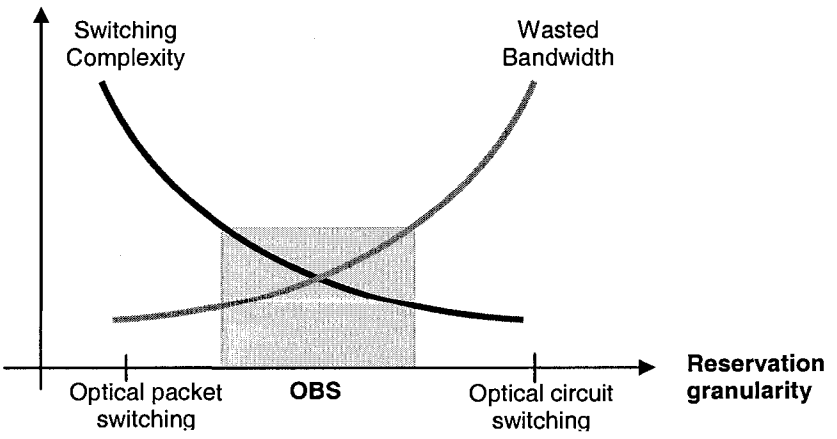

Figure 1.1: Comparison of optical switching techniques

The switched data carrying entity inside an OBS network is a burst. A burst is an aggregate of several end user (e.g. **IP**) packets processed as a single entity inside the OBS network. At an OBS network edge, ingress nodes receive end user packets and assemble them into bursts. Before sending a burst, a source node transmits the corresponding reservation information along the intended path through a separate control channel. A small control packet containing burst information is used for this purpose. This control packet contains information such as size, time offset between the control packet and the data burst, priority, and the destination address. One-pass reservation is used. A small delay after the control packet is sent, the corresponding burst is sent though the same path without waiting for an acknowledgment. This is done in order to avoid long delays introduced by a long circuit setup time.

Upon reception of a reservation packet, a switch node allocates a time slot on the appropriate output port, configures the optical switch and steers the incoming burst to the output. Otherwise, if all resources on the planned output link are occupied, the incoming burst is said to be blocked and is dropped. In case of successful reservation, a burst is forwarded to its output port using cut-through switching[1]. At the destination, egress OBS nodes disassemble incoming data bursts to recover individual end user packets.

1.3 Thesis Motivation

Because OBS uses one-pass reservation, burst loss due to contention within core nodes is a significant issue. Since OBS core nodes do not have buffers to resolve contentions, losses caused by conflict are higher than those observed with electronic networks. Within an OBS core node, a fraction of incoming traffic is expected to be lost because of blocking. As expected, this proportion increases with the load offered to the link. Applications running on any network may tolerate only a limited amount of data loss before the performance becomes unacceptable.

---

[1] Cut-through switching: when the switching fabric is configured, data bits are sent to the appropriate output as soon as they are received. The switch does not wait for the whole packet to be received.

Existing transport protocols, such as TCP, retransmit lost frames. Therefore, it is expected that end user (IP) frames lost due to blocking will eventually be retransmitted into the OBS network. However, given a network with high blocking probability, it is wasteful to retransmit data that will be lost later. Retransmitted frames increase the load of the network. This in turn increases the proportion of losses due to contention. This eventually leads to network congestion that produces collapse. In this situation, the real throughput[2] is almost zero and core traffic consists almost only of retransmitted frames. The average delay increases as well, eventually becoming unbounded[3]. For these reasons, regulation of the offered traffic load, in order to maintain the network under stable operation is necessary.

Most of the current OBS research focuses on reducing losses caused by blocking inside core nodes. This is achieved through various contention resolution techniques, applied at the core nodes. The solutions proposed in these works successfully reduce the probability of data loss, at the cost of added complexity in switching.

However, even with these improvements, congestion is still an issue. Their effect is only to increase the value of the load at which losses become too high. Furthermore, most of these methods can be considered *reactive*. They come into action once contention has occurred. Preferably, control actions should take place before the OBS network enters a congested state. This can be done by monitoring the state of the OBS network, and regulating the incoming traffic rate. Most of the research on OBS following this principle assumes TCP as the transport protocol [7-16]. This leaves the task of controlling traffic flow to the end users. However, end users cannot always be relied upon to regulate their traffic rates as pointed out in reference [17]. There is increasing focus on aggressive retransmission techniques in existing transport protocols or even completely removing

---

[2] Real throughput: the fraction of traffic successfully transmitted to the destination. It is equal to the traffic sent by the sources minus losses and retransmissions.

[3] The network becomes unstable.

congestion control. Also, there is no constraint on the types of traffic transiting through an OBS network. Some traffic classes are bandwidth-intensive and do not respond to congestion control signals (e.g.: Video using UDP).

For these reasons, it is important to regulate the traffic rate entering the network at the ingress nodes. This approach can prevent congestion from occurring in the network core, while the traffic regulation algorithms run at the ingress nodes, where memory and processing power are available. The flow control at the ingress nodes can be performed using feedback provided by the network. Similar approaches have been used in the past. A popular case being the congestion control developed and used for the ABR (Available Bit Rate) service of ATM [18]. However, existing methods cannot be applied directly to OBS due to the absence of buffers. Rate-based flow control applied to OBS is a subject that has not been widely studied. To the best of our knowledge, there are three previous works in this area [19-21]. More details about these proposals [19-21] are provided in Chapter 2.

1.4 Thesis Objectives

The goal of this thesis is to formulate and validate a congestion avoidance and control framework for OBS networks based on utility maximization and resource pricing. The basis for this method is introduced in [22-24]. The solution we propose includes rate control performed at the ingress nodes: a signalling mechanism that is used between OBS nodes and a rate adjustment rule. The congestion avoidance algorithm presented in this work has the following objectives:

- Ability to maintain a desired congestion level on every link;
- Proportional fairness;
- Distributed operation; and
- Stability under delay.

The performance of the proposed algorithm is studied through simulation models developed with *OPNET* software.

1.5 Thesis Outline

The remainder of this thesis is organised as follows: Chapter 2 presents OBS in more detail as well as the state of the art in OBS research. It also discusses in more detail previous work on rate-based congestion control for OBS. Chapter 3 introduces the theory behind pricing-based congestion control. Chapter 4 explains the relevance of pricing-based congestion control to OBS. A congestion control method tailored for OBS is then proposed, based on the theory outlined in Chapter 3. In Chapter 5, we present a simulation model of an OBS network enhanced with the congestion control algorithm. Simulation results obtained with *OPNET Modeler* are presented in order to assess the performance of our design. Finally, Chapter 6 concludes the thesis, and discusses future work.

Chapter 2

Optical Burst Switching Review

2.1 Introduction

This chapter discusses Optical Burst Switching in more detail. It has two main parts. The first part describes the operation of an OBS network and comprises section 2.2 to 2.7. It is also intended as a review of the research on this subject. Section 2.2 presents the building blocks of an OBS network. In section 2.3, we review the methods used to assemble end user packets into bursts. We also discuss the effects of burst assembly on traffic statistics. Section 2.4 discusses burst reservation mechanisms. Section 2.5 addresses support for QoS differentiation. Sections 2.6 and 2.7 deal with scheduling and contention resolution, respectively. The second part consists of section 2.8, which reviews previous works on congestion avoidance and control on OBS.

2.2 OBS Network Architecture

2.2.1 OBS Fundamentals

Optical burst switching [5] aims to take full advantage of the bandwidth available in optical fibres while retaining high resource utilization through statistical multiplexing. The basic principles that define an OBS network are the following [5]:

- *All-optical data plane:* Conceptually, an OBS network consists of two parts: the *Control Plane* and the *Data Plane*. Data is relayed optically through WDM channels within the network. It is kept entirely in the optical domain as long as it is within the OBS network boundary. Limitations caused by OEO conversion are avoided during switching. This allows high data transmission speed.
- *Out-of-band signalling:* The control plane is in charge of resource reservation, routing, optical switch configuration and other network management issues. One channel in each link is reserved for the control plane. This is called the *control channel*. Since the data plane needs to perform complex operations, the data transiting through this plane is processed electronically. This is possible because traffic on the control channel has lower bit rate.
- *Statistical multiplexing:* OBS reservation granularity is between those of circuit switching and packet or cell switching. A reservation is made for each burst. This allows high bandwidth utilization because all connections can share the available capacity, unlike circuit switching.

The architecture of an OBS network is shown on Figure 2.1. It is composed of *Edge Nodes* and *Core Nodes*. As their name implies, edge nodes are located at the boundary of an OBS network while core nodes are located within the network. When two edge nodes are communicating, the source node is called the *Ingress Node* while the destination node is called the *Egress Node*.

The basic switched entity in OBS is the burst. A burst is an aggregate of several end user (e.g. **IP**) packets processed as a single entity inside the OBS network. Note that burst switching imposes no constraints on the data format. At an ingress node, packets received from end users are assembled into bursts. Consequently, individual bursts contain data of different origin and format. Incoming data is buffered until a chosen threshold is reached. The data in the buffer is then framed into a single burst for transmission. Section 2.3 describes this process in more detail.

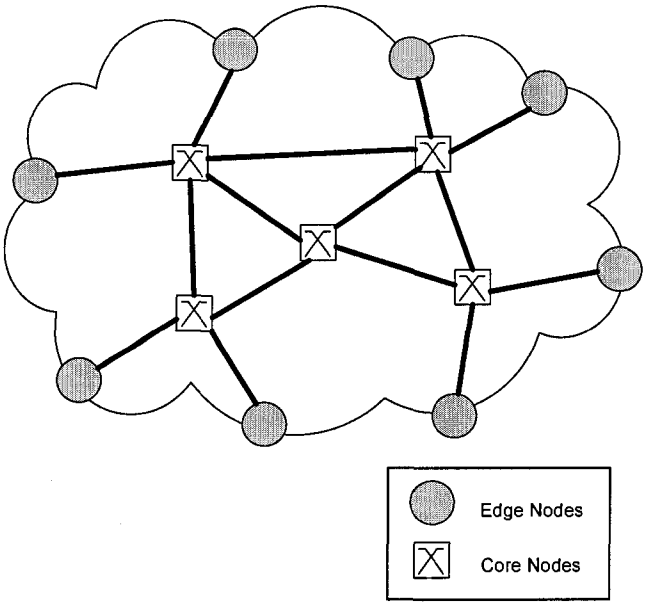

Figure 2.1: OBS network architecture

2.2.2 Edge Nodes

The main task of edge nodes is burst assembly and disassembly. Ingress nodes receive data packets from client networks and assemble them into bursts before transmission. At the destination, egress nodes recover the individual packets from the received bursts. Individual packets or cells then continue towards their final destination. The architecture and operation of an ingress node are presented in Figure 2.2.

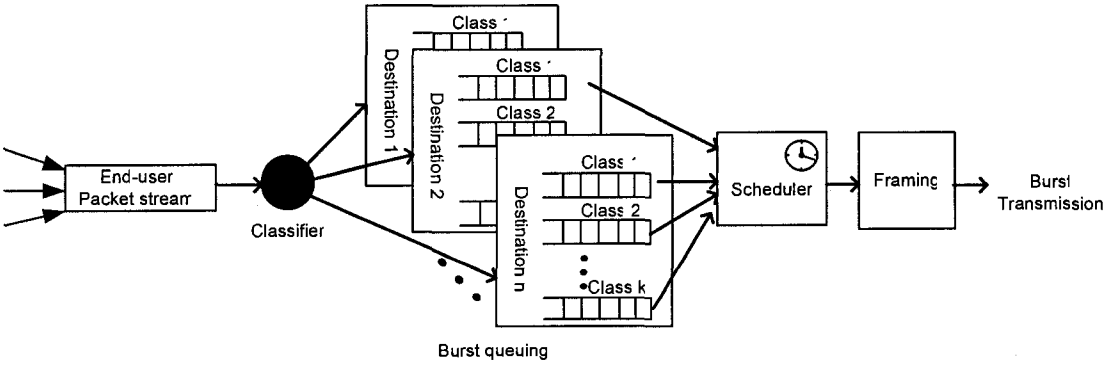

Figure 2.2: OBS Edge Node architecture (adapted from [25])

As shown on the Figure 2.2, end user packets received by an ingress node are first classified according to their destination node and service level. Packets having the same

destination and service class share a common logical queue. Therefore, the total number of logical buffers required is equal to the number of destinations multiplied by the number of service classes.

Once a buffer reaches a predefined threshold, the ingress node frames the packets into one burst. This threshold may be based on buffer size and/or a timer. A *burst header packet* (BHP) is sent along the intended burst path for bandwidth reservation. A BHP is a small header packet containing information about the burst associated with it. A BHP is also called a *control packet* or a *reservation packet.* These terms are used interchangeably in the remainder of the text. The ingress node sends the data burst a short time after the BHP, without waiting for an acknowledgment of the reservation. One-pass reservation is mainly used in OBS to avoid long circuit setup times associated with two-way reservation.

2.2.3 Core Nodes

The main task of a core node is the switching of incoming bursts, keeping them on the path leading them to their destination. The architecture of a core node is shown on Figure 2.3. It mainly consists of a switch control unit (SCU) and an optical switch fabric. The SCU receives burst header packets (BHP) from the control channel. The information contained in a BHP is retrieved and processed electronically. This information includes the burst length, its destination, and the time offset between header and burst. Based on this information, the control unit searches for available resources at the output port at the time the burst arrives. If a channel is found, the SCU configures the switching fabric and relays the BHP to the next node. In case of successful reservation, a burst is optically switched to the proper output link. Otherwise, the associated burst is discarded. In both cases, no acknowledgment is sent to the source.

Core nodes perform cut-through switching, as opposed to the store-and-forward method used by electronic routers. As a result, an optical burst does not suffer any delay within a core node. Delay within an OBS network is mainly due to propagation delay, which is a law of nature and cannot be reduced or avoided.

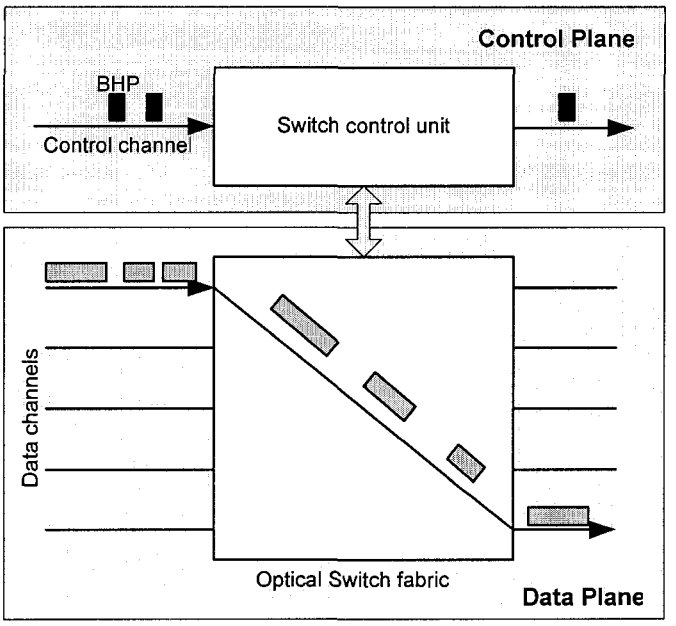

Figure 2.3: OBS Core node architecture (adapted from [26])

2.3 Burst Assembly

There are two main methods for burst assembly: size-based [27] and timer-based [28]. These two schemes are discussed more thoroughly in the next sections. In addition, combinations of both techniques have also been proposed [29, 30].

2.3.1 Size-based Assembly

This burst aggregation method uses a burst size threshold B as the primary decision factor. At an ingress node, packets are buffered until the cumulated size reaches the threshold B [27]. When the size of the buffer reaches B, its content is retrieved and is transmitted as a single burst.

As pointed out in [29] and [31], the choice of B determines the size of bursts produced using this scheme. Assume that packets contained in a burst have fixed size P and their arrival is Poisson with rate equal to $1/\mu$. If P is fixed, the burst size is also constant since the number n_p of packets contained in a burst is $n_p = \lfloor B/P \rfloor$ [31]. On the other hand,

burst interarrival (or interdeparture) is the sum of the interarrival times of packets in the burst. According to the central limit theorem, the resulting distribution should approach a Gaussian function with mean $\mu \cdot n_p$ and variance $\mu^2 \cdot n_p$ [31].

This is confirmed using a simple burst assembly simulator. The results for burst size distribution are shown on Figure 2.4, with the Gaussian approximation $N(\mu \cdot n_p, \mu^2 \cdot n_p)$. Results are shown for two cases: bursts containing $n_p = 10$ packets and bursts containing $n_p = 50$ packets. In both cases, the average packet interarrival time is 2 μs. As expected, the distributions obtained from simulation and from the drawing of the corresponding Gaussian PDF are very similar, with the similarity increasing as n_p becomes larger.

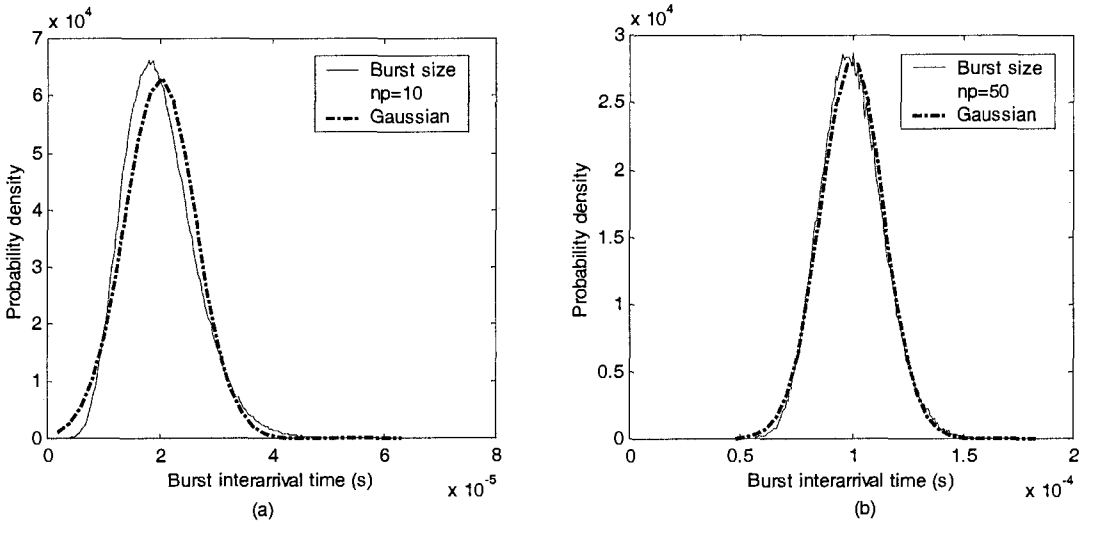

Figure 2.4: Burst interarrival time distribution, threshold-based assembly with (a) $n_p = 10$ (b) $n_p = 50$

Since burst assembly requires queuing until the threshold is reached, it introduces a delay for each incoming packet. This delay is directly related to the time interval separating consecutive bursts. If the data rate is high, the threshold is attained faster and the waiting time is small. On the other hand, if the data rate is slow, it takes longer to fill the buffer up to B and packets wait longer on average. Equivalently, for a given packet arrival rate at an assembly queue, the delay is longer for a larger value of the threshold B. This is

apparent by the curves displayed on Figure 2.4, where mean burst interarrival time is 20 μs for n_p = 10 and equal to 100 μs for n_p = 50. It is shown in [32] that the average packet delay due to threshold-based assembly decreases as the data rate increases.

2.3.2 Timer-based Assembly

The timer-based assembly [28] uses a different method for burst formation. Every τ seconds, a burst is formed from the content of the assembly buffer. If the resulting burst is smaller than the minimum size b, padding is added until it reaches b. As pointed out in [28], this algorithm is simple and offers the following advantages:

- It guarantees minimum burst size. As a result, it also reduces overhead.
- It provides an upper limit for the buffering delay experienced by user packets.

With timer-based assembly, the burst interarrival time is constant and equal to τ. Let's assume again that the size of packets contained in a burst is constant and equal to P. Let B be the burst size and n_p be the number of packets in a burst. If packet arrival at the burstification buffer is Poisson with rate $1/\mu$, then the number of packets contained in a burst equals the number of arrivals during time τ. For a Poisson process, the probability that k packets arrive during the interval τ is:

$$P\{n_p = k\} = \frac{(\tau/\mu)^k}{k!} e^{-\tau/\mu} \tag{2.1}$$

The average number of packets in a burst is E[n_p] = τ/μ. When τ/μ is large enough, the burst size distribution converges to a Gaussian distribution with mean $P \cdot \tau/\mu$ and variance $P^2 \cdot \tau/\mu$ [31].

2.3.3 Hybrid Burst Assembly

With size-based assembly, if the traffic rate at a buffer is low, it may take too long to gather enough data to reach the size B. This may considerably delay packets that are waiting in the queue for the threshold to be reached. Conversely, under heavy load, the timer-based algorithm may produce excessively large bursts, which occupy channels for too long.

For these reasons, combinations of both algorithms have been proposed [29, 30]. To avoid excessive delay in the size threshold-based technique, the basic algorithm can be improved by choosing a minimum burst length and a time-out value T for assembly [30]. A timer starts when the first packet of a burst arrives in its queue. A burst is formed when the minimum length is exceeded or the timer expires, whichever comes first. This algorithm is called *Min-Burst Length Max-Assembly Period* (MBMAP) algorithm [30]. The value of T should be large enough so time-out does not occur in normal operation. MBMAP guarantees a maximum delay and limits burst size.

Adaptive algorithms also exist, where the thresholds can be adjusted dynamically. Techniques such as the *Adaptive Assembly Period* (AAP) algorithm [30] adapt the timer while others [33] control the size threshold.

2.3.4 Impact of Burst Assembly on the Traffic's Self-Similar Behaviour

It is well-known that Internet traffic displays self-similar characteristics [34, 35]. Therefore, it is expected that traffic received by OBS edge nodes will exhibit self-similar behaviour. There is an incentive to study the effect of burstification on self-similar traffic. At first, it was suggested that assembly smoothes out traffic fluctuations and reduces long-range dependency [28]. However, it was later shown that burstification at ingress nodes does not change long-range dependency [29, 36]. This is further established in [37], where it is shown that after burstification, traffic is still self-similar and the degree of self-similarity is unchanged. Burst assembly only affects short-term traffic characteristics.

2.4 Resource Reservation

In this section, we discuss the signalling mechanisms used for bandwidth reservation and release along the path of a burst. We already mentioned that for each burst, a separate header is sent using a dedicated control channel. A burst header packet (BHP) carries information about the start time and duration of a reservation. All the methods described in the following sections use one-way reservation. Ingress nodes do not wait for an

acknowledgment before starting transmission. However, they differ on how the connection is established and released.

2.4.1 In-Band-Terminator (IBT)

In IBT, each burst contains a header as in packet switching [38]. A special delimiter, called an *In-Band Terminator* (IBT), is transmitted after each burst to release the connection. Header information may be carried on the same channel as the burst or through a separate control channel. The IBT is carried optically on the same channel as the burst. An intermediate node configures its optical switch as soon as a header is received.

As pointed out in [38], IBT and packet switching are very similar. However, IBT-based OBS uses cut-through switching. For this reason, the header is transmitted to the next node before the last bit of the burst is received. This contrasts with the store-and-forward method usually used for packet switching. A challenge in implementing this method is the optical recognition of the terminator.

2.4.2 Just-in-Time (JIT)

In *Just-in-Time* [39], an ingress node sends a reservation packet through the control channel. The first core node calculates the required delay between the header and the burst. The delay takes into account the processing and setup time required at each network element on the path. The information about this delay is sent to the source. At the end of this short wait-time, the ingress node sends the burst without waiting for an acknowledgment. If a connection request is accepted at a core node, it configures the optical switch immediately. The ingress node may send refresh messages along the path if the burst is long. When transmission is complete, the ingress node tears down the connection by sending a RELEASE message along the path. If the connection is successful, the destination node may send a notification message back to the source as soon as it receives the burst header [39]. Figure 2.5 shows the sequence of events for a successful end-to-end reservation, based on this technique.

JIT is simple and fast. However, since channel bandwidth is assigned before the arrival of the burst, bandwidth is wasted in the interval between the header packet and the data.

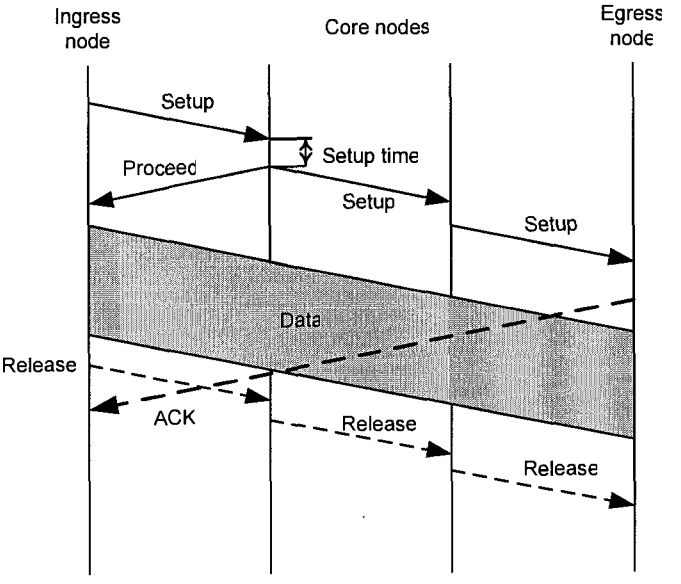

Figure 2.5: JIT reservation method [39]

2.4.3 Tell-and-Go

Tell-and-Go (TAG) is very similar to JIT [40]. Figure 2.6 shows its operation.

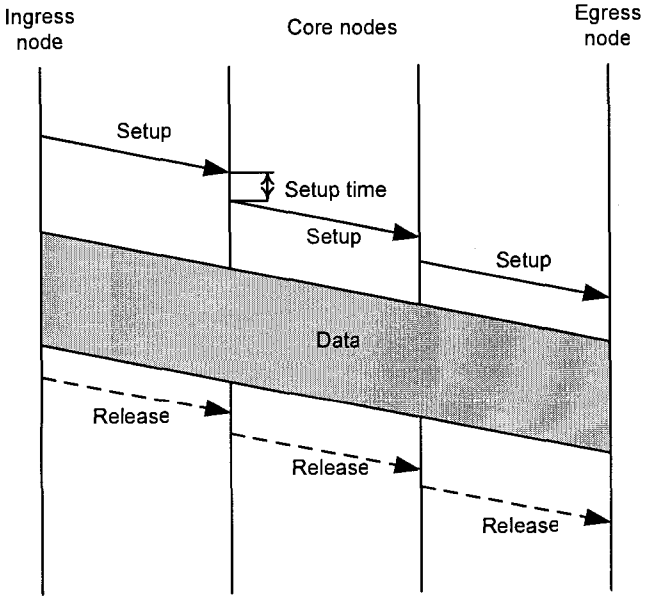

Figure 2.6: TAG reservation method

In TAG, the ingress node sends a header packet, followed by the burst. A guard time is placed between the header and the burst to allow switch configuration. A core node allocates a channel as soon as it receives a control packet. When the source completes burst transmission, it sends a release message to free the connection.

2.4.4 Just-Enough Time

Just-enough time (JET) [5, 41] is the method that has received the most attention from researchers. The main difference between JET and the other techniques described previously is that JET uses *delayed reservation* and *close-ended reservation*. These two ideas together are called *Reserve-a-Fixed-Duration* or RFD. When a burst is ready for transmission, the ingress node sends the BHP. After a small time offset t_{offset}, the burst is transmitted on an available wavelength. In JET, the BHP contains both the time offset and the burst duration, in contrast to the other reservation methods. These two values determine the exact time at which a burst begins and ends. When a core node receives a BHP at time t, it waits until time $t + t_{offset}$ to establish the connection. The connection is kept for the duration indicated in the BHP. There is no need for another packet to release the connection.

Figure 2.7 shows how JET works. As the BHP requires a processing time δ, the offset gets shorter after each intermediate node. Assume the configuration time at the egress node is also equal to δ. This can be observed on Figure 2.7. Thus, the time offset must be large enough to take into account the setup time incurred at each core node. If the path of a burst contains H intermediate nodes, the time offset must be at least equal to $t_{offset} = \delta.(H + 1)$. The time offset can also be used for other purposes, as explained in the next section.

Because of its design, JET involves less overhead than the other described techniques since no release message is needed. Also, JET uses the bandwidth more efficiently as it is assigned only for the duration of a burst. For a given load, this efficiency translates to a lower burst loss caused by blocking as compared to the burst loss experienced by the other techniques.

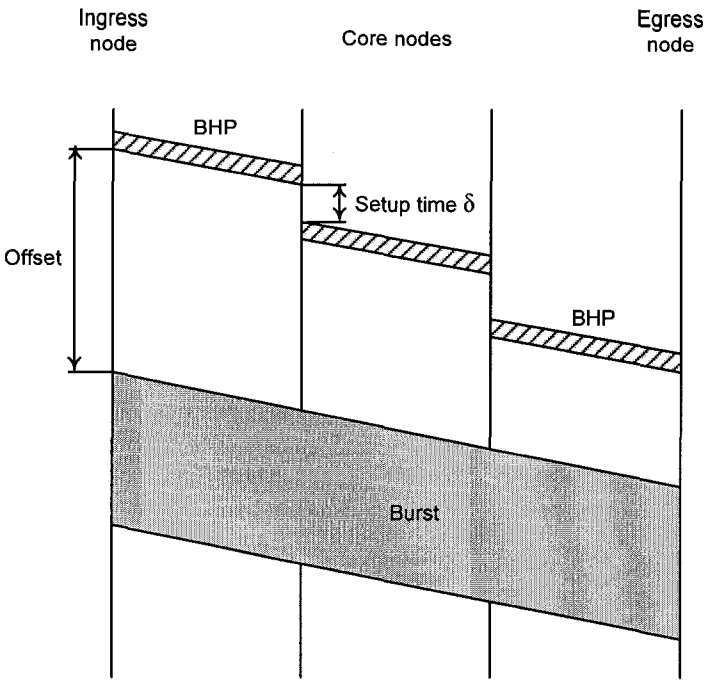

Figure 2.7: JET reservation

2.5 Service Differentiation

2.5.1 Offset-Based Service Differentiation

Service differentiation can be achieved in OBS by introducing an additional time offset in JET [42]. An extra offset t_{qos} is added to the base offset T between the BHP and a high priority burst as shown on Figure 2.8. This additional delay is called the *QoS Offset*. The base offset is equal to the delay needed for processing the BHP and configuring the optical switch at a core node.

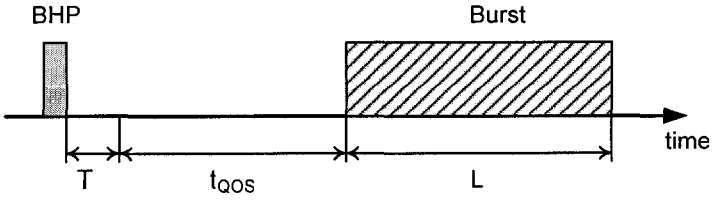

Figure 2.8: QoS-enabled JET

Using this method, traffic can be separated into several service classes with increasing time offsets. In this case, high priority bursts have a longer time offset. Lowest priority traffic, labelled *Class 0*, only has the base offset T. For *Class 1* traffic, a delay t_{QOS}^{1} is added to the base offset and so on. Service differentiation is achieved in this manner because bursts with longer time offset experience lower blocking. This is demonstrated in [43, 44] where an analysis of per-class blocking probability is presented. If the difference between QoS offsets of adjacent classes $t_{diff} = t_{QOS}^{n} - t_{QOS}^{n-1}$ is constant, then QoS separation only depends on the value of t_{diff}. It is shown that the degree of separation for adjacent classes increases with t_{diff}. If burst length is exponentially distributed with mean L, then $t_{diff} = 3 \cdot L$ results in 95% QoS separation [45]. This means that bursts from class n are *not* blocked by lower priority bursts from class $n - 1$ with probability 0.95.

This method achieves relative service differentiation in a simple and elegant way. No further change is required to the network to carry out this technique. In addition, the authors claim in [45] that the overall blocking probability when using this method is the same as that observed in classless OBS. In other words, the conservation law is satisfied.

However, this method introduces significant delay in burst transmission because of the increased offset. For an OBS network with several traffic classes, the end-to-end latency for the highest priority classes may become significant. This must be considered when high priority bursts carry traffic with strict delay requirements. In addition, this technique displays a burst-selecting effect [46]. This means that long bursts from lower priority classes are dropped more often than short bursts.

2.5.2 Other QoS separation methods

To address the issues with offset-based service differentiation, other methods have been proposed. This section discusses these techniques. Alternative service differentiation methods involve active dropping by intermediate nodes [46, 47].

In [46], the authors make the case for proportional QoS on OBS. In proportional QoS, each traffic class is assigned a *proportional factor* s_i which determines the priority level. A higher value of s_i is associated with higher priority. At intermediate nodes, the arrival

rate and loss for each QoS class is monitored. Proportional QoS is achieved by keeping $lossrate_i$ / s_i constant for all QoS classes. If class i experiences lower blocking probability, i.e. if $lossrate_i / s_i < lossrate_N / s_N$, bursts from class i are dropped. This is complemented by a *Wait-Time-Priority* scheduler at the ingress node.

For *Assured Horizon* [47], per class bandwidth reservation is done at ingress nodes. Bursts are labeled as conforming on nonconforming depending on whether they satisfy the bandwidth limit for their class. At core nodes, there are two modes of operation. When the core node is in normal mode, no active dropping is performed. When the core node is in the congested state, bursts marked nonconforming are dropped ahead of the switch.

Several other publications propose a combination of early drop against low priority traffic and wavelength provisioning for high priority classes [48-50].

Other techniques use preemption against low priority bursts at core nodes [51-53]. Preemption means that low priority bursts already scheduled or in service at core nodes are discarded in favour of high priority ones.

Finally, in [54], it is proposed to assemble bursts from packets of different priorities. Packets are ordered within bursts according to their priority. Highest priority packets are placed at the head. In case of contention at core nodes, bursts are segmented[4] and the tail containing lower class packets is dropped. Consequently, higher priority packets experience lower blocking probability. This method achieves service differentiation at lower granularity level compared with the other techniques above.

2.6 Burst Scheduling

This section discusses channel scheduling algorithms on OBS. Burst scheduling controls how resource is allocated to incoming bursts based on the timing information carried by

[4] Burst segmentation as a contention resolution method is discussed in more detail in section 2.7.5.

control packets. Each channel on a link is divided in the time domain into *voids* separated by busy periods. A burst arriving at a switch node can be inserted into a void of adequate size. When a burst is successfully scheduled on a channel, the original void is replaced by two others as shown on Figure 2.9. One, in front of the burst is called the *starting void*, the other after the burst, called *ending void*. Various scheduling algorithms use different methods to find a suitable void where a received burst can be inserted.

There are two goals for scheduling algorithms: low blocking probability and fast processing speed [26]. These two objectives are in conflict with each other for the following reasons. To achieve low blocking probability, a switch needs more information on past and future bursts. This means slower processing speed.

Scheduling algorithms can be classified as *void filling* or *without void filling*, depending on whether they try to use voids between consecutive bursts or not.

2.6.1 Horizon Algorithm

Horizon is a scheduling algorithm without void filling proposed in [6]. It is the same as the LAUC algorithm (Latest Available Unused Channel) in practice [55]. The only information required in this method is the so-called *horizon*. The horizon is the time at which the last reservation on each channel ends. From this point on, the channel is free to accept new reservations. This results in fast processing as the scheduler only needs to determine if a new burst arrives after the horizon to accept the reservation. Second, when a burst is accepted, two new voids are created. If several suitable voids exist, the algorithm chooses the channel which results in the shortest starting void to minimise bandwidth waste. The motivation behind this is that the starting void is unusable afterwards. To satisfy this requirement, the channel with the latest horizon is chosen, hence the LAUC name. Consider the example shown on Figure 2.9. When the new burst arrives, it will be assigned to Channel 0 when Horizon scheduling is applied.

The Horizon algorithm is simple and fast. According to [26], the time complexity of the best Horizon implementation is O(log W) for a link with W channels. This is an important advantage when a switch must process a large number of reservations every second.

However, there is a trade-off. Because channel voids are ignored, bandwidth utilization is not efficient compared with more advanced techniques.

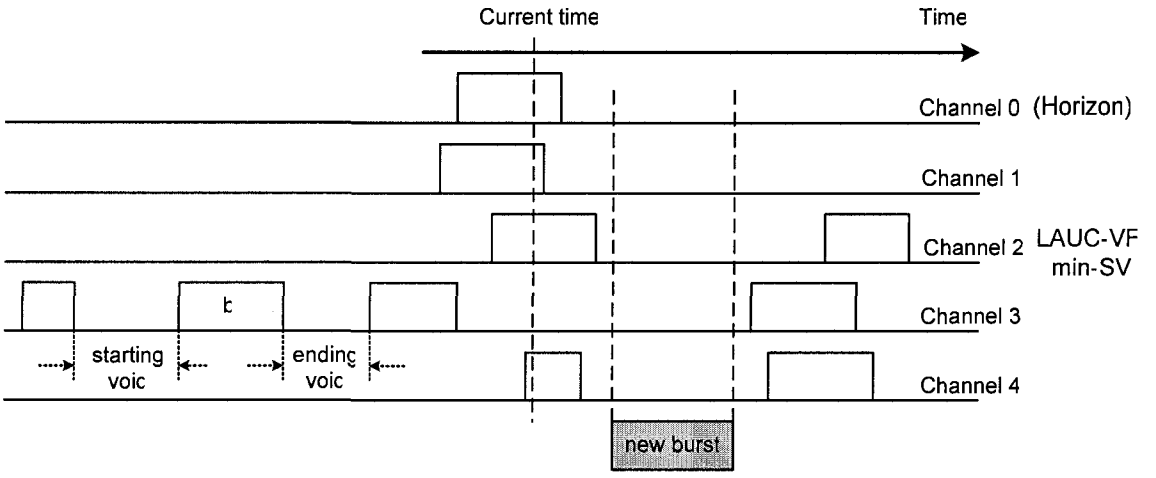

Figure 2.9: Comparison of burst scheduling algorithms (adapted from [26])

2.6.2 Scheduling with Void-Filling

2.6.2.1 Latest Available Unused Channel with Void Filling (LAUC-VF)

To improve bandwidth efficiency, LAUC-VF is proposed in [56]. LAUC-VF is similar to Horizon except that it maintains a list of all voids on a link. This list includes the void from the horizon to infinity. If a burst arrives at time *t*, channels in a link are sorted into two groups: channels which have scheduled bursts after *t* and channels with no scheduled reservation after *t*. The algorithm works as follows. Channels are searched in order of scheduled and unscheduled channels. Among the voids that can accommodate the new burst, the one that results in minimum starting void is chosen. Consider the example on Figure 2.9. If LAUC-VF algorithm is used, the new burst is assigned to Channel 2, which produces the smallest leading void.

LAUC-VF achieves high bandwidth utilization. However, it is more computationally expensive than Horizon. For a link with *m* voids, the time complexity of this algorithm is O(*m*) [26]. Variations of this algorithm exist, for which either minimum ending void or best-fit (minimises both starting and ending void) is used instead of minimum starting void [57].

2.6.2.2 Minimum Starting-Void (Min-SV)

The Min-SV algorithm [57] is functionally the same as the LAUC-VF algorithm described previously. Just as LAUC-VF, Min-SV maintains a list of channel voids and tries to minimise the starting void. However, the method used to store and search for available voids is different. Min-SV uses an augmented balanced tree to store channel voids. When a reservation arrives, a suitable interval is looked up in the tree. If a void is found, it is removed from the tree and replaced by two new ones corresponding to the starting and ending void. If we consider the previous example on Figure 2.9, the new burst is sent on Channel 2 when min-SV is used, just as in LAUC-VF.

A min-SV search tree is shown on Figure 2.10 to illustrate its operation [26]. The channel voids are first sorted in order of starting time. The voids make up the leaves of the tree, located at the bottom. The nonleaf nodes of the tree contain three information fields: the median starting time of all children leaves, a pointer P_{ymax} to the void with maximum ending time, and a pointer P_{ymin} to the void with minimum ending time. Scheduling a burst happens in three steps: searching for the min-SV void, deleting it, and inserting two new voids corresponding to the starting and ending voids. Let $b = (s, e)$ be a burst that requires scheduling, where s and e are the time at which it begins and ends respectively. The min-SV method performs a binary search for s in the tree. Let $s = 20$ and $e = 30$. Starting at the root, the search proceeds to the right if s is greater than the recorded median in the current node. Otherwise, it proceeds to the left. Following this rule, the search reaches the leaf $(35, \infty)$ in our example. All nodes to the left of the path (in thick lines) from the root down to the leaf node correspond to voids starting before $s = 20$. If the current void is feasible, then it is the min-SV void and the search is complete. If it is not feasible, as in the current example, a search is performed among the nodes to the left of the path (in thick lines). The algorithm only needs to compare the burst ending time with the value pointed to by P_{ymax} to establish if a node has an adequate void among its children. In our current example, the sibling leaf is feasible. Therefore, it is the min-SV void.

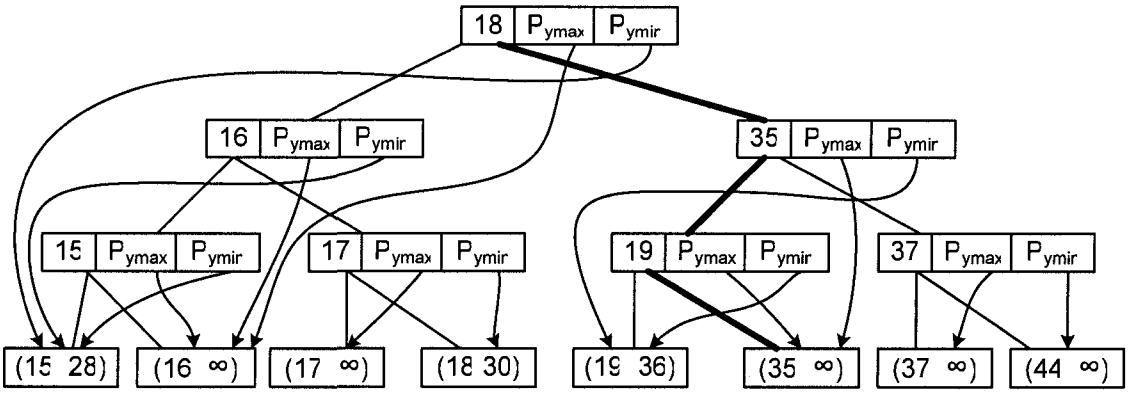

Figure 2.10: Augmented binary tree used by min-SV algorithm (copied from [26])

The use of a binary tree noticeably speeds up the search for an appropriate channel. If successful, it takes O (log m) time to complete the algorithm, where m is the number of voids on a link [57]. In addition, Min-SV achieves bandwidth utilization similar to LAUC-VF. However, this is achieved at the cost of a more complex data structure, which makes implementation less straightforward.

2.6.3 Proactive Algorithms

All methods previously discussed manage contentions locally, once they have occurred. For this reason, they are classified as *reactive* algorithms [26]. The methods presented in this section avoid contention before they occur. Accordingly, they are classified as *proactive* algorithms.

2.6.3.1 Priority-Based Wavelength Assignment

Priority-Based Wavelength Assignment (PWA) is proposed in [58] to reduce the blocking probability of OBS nodes which cannot perform wavelength conversion. The idea behind PWA is the following: at an ingress node, the channel that has the highest success rate so far has the most chance of delivering future bursts.

In PWA, each ingress node maintains a wavelength priority list for each destination. The priority of a wavelength increases when a reservation is successful. Otherwise, a blocked reservation decreases the priority of the wavelength on which it occurs. When an ingress node has a burst to send, it looks up the wavelength with the highest priority. If it is

available, the burst is sent on this channel. Otherwise, the list is searched sequentially starting with the highest priority until a free channel is found. When a burst is successfully delivered, the egress node sends an acknowledgment message to the source. Otherwise, a negative acknowledgment is generated. The ingress node then updates its wavelength priority table according to this information. Because of the priority scheme, some wavelengths are heavily used by one source. Therefore, wavelength priority tables are different for each ingress node.

According to [58], this method considerably decreases blocking probability on an OBS network, compared with random wavelength assignment. However, we note that PWA only applies to networks without wavelength conversion.

2.6.3.2 Burst Overlap Reduction Algorithm

Burst Overlap Reduction Algorithm (BORA) is a proactive blocking reduction technique [59]. The principle behind BORA is based on the observation that contentions result from simultaneous arrivals on a single link. If the number of bursts arriving simultaneously exceeds the number of channels on the output link, some bursts must be dropped.

Consider the example shown in Figure 2.11(a). Four overlapping bursts b_1 to b_4 are generated at an ingress node. If no proactive scheduling is used, these bursts are sent immediately, in the order they are formed. If b_1 to b_4 share a common path section, there is a high likelihood of contention at a switch downstream due to their simultaneous arrivals. If overlapping reservations formed at ingress nodes can be reduced, the overall blocking probability is expected to decrease. The idea is to use the memory available at ingress nodes or Fibre Delay Lines (FDLS) at core nodes to delay overlapping burst and transmit them sequentially instead. The aim is to use the fewest wavelengths to transmit a set of bursts.

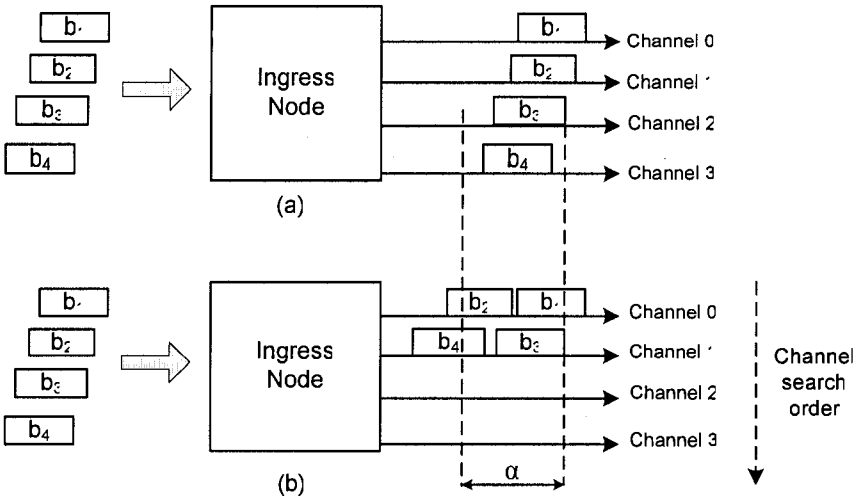

Figure 2.11: BORA scheduling algorithm (adapted from [26])

In BORA with Fixed-ordered Search (BORA-FS), data channels are sorted in a fixed order. For every burst generated, the ingress node queries the list of wavelengths in the order determined previously. If *channel 0* is available, the burst is sent out on this wavelength. This is the case of b_1 in Figure 2.11(b). If *channel 0* is not available, the scheduler verifies whether the unscheduled burst can wait until the transmission on *channel 0* is over. If so, the source postpones the new burst and assigns it to *channel 0* when it becomes available. This is the case of b_2 on Figure 2.11(b). To decide whether a burst can be delayed, a maximum delay α is defined. If the delay until the end of the current transmission exceeds α, the scheduler looks at the next channel in the list and the process is repeated. This is the case of b_3 and b_4 on Figure 2.11(b). At the end, one notes that transmission of the four bursts now occupies only two channels instead of four previously. Simulations show that BORA-FS with void filling significantly reduces overall blocking probability compared with LAUC-VF [59].

2.7 Contention resolution

In this section, we discuss methods used to avoid or resolve contentions occurring within core nodes. First, we discuss and quantify the effect of contention on data loss. Contention in OBS networks can be resolved using deflection of burst segmentation.

Deflection can take place either in time domain, wavelength domain of space domain. We discuss each of these solutions in the following sections. We note that a combination of two or more methods can be used to achieve even lower blocking.

2.7.1 Contention in OBS

Contention arises when more than one burst compete for the same channel on the same link. Originally, in case of contention, the channel is assigned to one burst and the others are simply discarded. Since core nodes cannot buffer packets indefinitely, contention is a major source of data loss in OBS. For this reason, methods to resolve contention and reduce data loss currently draw the attention of many researchers.

Assuming no buffers are available at switches and fixed time offset, an OBS link with N channels can be modeled as an M/G/n loss system [60]. Let A be the load on a link measured in Erlangs. If the burst arrival process is Poisson, then blocking probability P is given by the Erlang B formula [60]:

$$P = \frac{\frac{A^N}{N!}}{\sum_{x=0}^{N} \frac{A^x}{x!}} \tag{2.2}$$

The blocking probability computed using the Erlang B formula is insensitive to burst size distribution [61]. The only factors affecting blocking probability are the load A and the number of wavelength channels per link N. If no wavelength conversion (no contention resolution) is available in the switches, equation (2.2) simplifies to the following, which is the Erlang B formula with $N = 1$:

$$P = \frac{A}{1+A} \tag{2.3}$$

The blocking probability calculated with equation (2.3) is shown on Figure 2.12, with results obtained through simulations. Figure 2.12 shows that without contention resolution, the blocking probability in OBS is too high for any practical purpose. For instance, around 30% of bursts are lost with a 40% load.

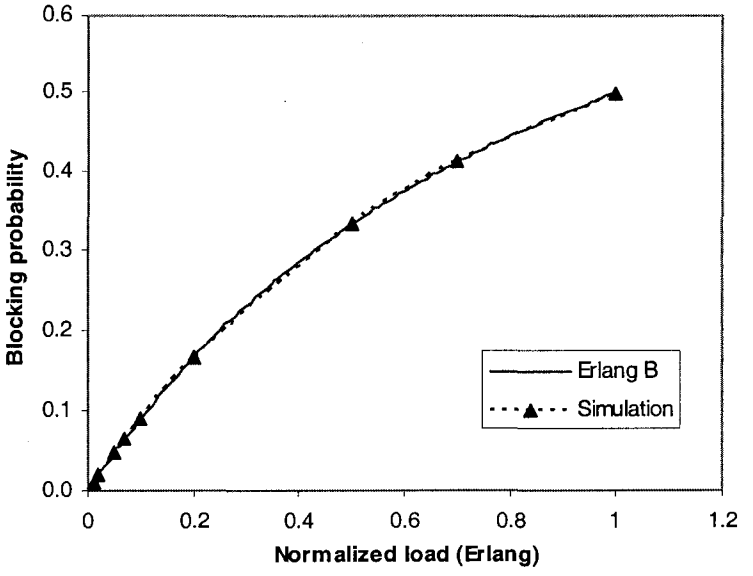

Figure 2.12: Blocking probability, no contention resolution

According to Zukerman *et al* [62], the Erlang B formula overestimates the blocking probability in an OBS core node when only a few wavelengths are available. They propose a more accurate analysis of the blocking probability based on the Engset loss system. For more details on this analysis, see [62].

2.7.2 Wavelength Conversion

The first contention resolution method we address is wavelength conversion. With this method, contending bursts are sent on different channels. This implies that some bursts arrive on wavelength i of the input link and are retransmitted on a different wavelength j on the output link. Tuneable wavelength converters (TWC) are used to change the carrier frequency of contending bursts. If a switch can convert any input wavelength to any of the N channels at the output, it is said to have *full wavelength conversion* capability. The Erlang B formula introduced in equation (2.2) gives the blocking probability of a switch with full wavelength conversion. In fact, most of the scheduling algorithms presented in the previous section assume that full wavelength conversion is available. Wavelength conversion reduces the blocking probability of an optical switch. For a given normalized load, blocking probability is lower when more wavelengths are available as seen on Figure 2.13. For instance, with $N = 64$ channels and 70% load, the calculated blocking

probability is around 10^{-3}. With similar blocking levels, practical deployment of OBS becomes feasible.

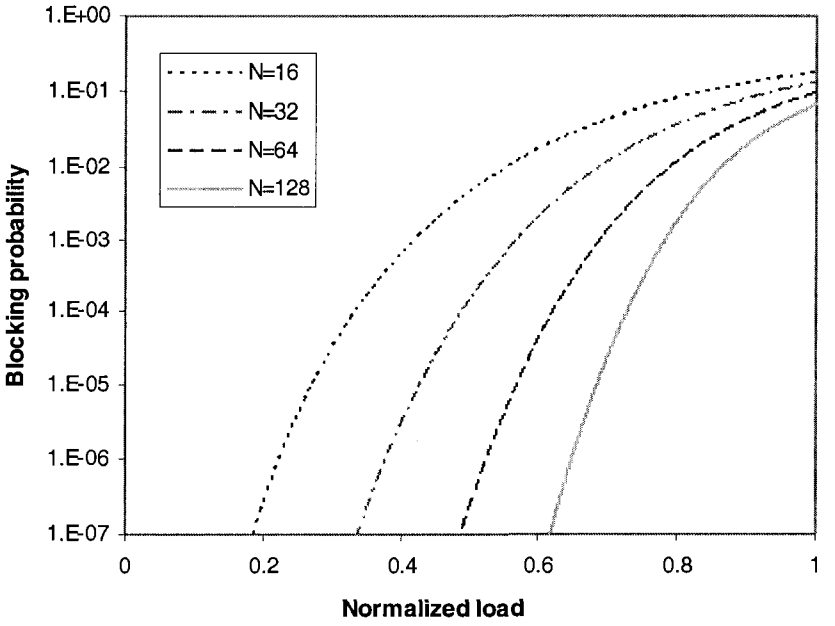

Figure 2.13: Blocking probability vs. load with full wavelength conversion

However, it should be noted that wavelength converters are expensive components. In practice, there is a limit on the number of available wavelength converters. The number of available TWCs may be a fraction of the channel count. Also, there is a limit to the tuning range of wavelength converters. A given input frequency may only be convertible to a reduced set of the output wavelengths. The effect of these constraints on the blocking probability is studied in [63] and [64].

2.7.3 Fibre Delay Lines

Fibre Delay Lines (FDL) are recommended in [5, 6] to resolve contention within a switch. FDLs offer a limited form of buffering where the delay results from propagation time inside the device. If all output wavelengths are busy, a burst can be stored in an FDL and be assigned to a channel when it becomes available later. If all channels and FDLs are occupied, the burst is discarded. FDLs of different lengths can be also used to get certain predefined values of delay. In addition, any channel can use an available FDL.

FDLs successfully reduce the blocking probability in an OBS switch. With 32 channels and 8 FDLs, blocking probability is 10^{-6} for a 50% load [6].

Because of its nature, this method increases the end-to-end latency in the network. Furthermore, it increases the complexity of channel scheduling algorithms. Also, the time complexity of scheduling algorithms is multiplied by the number B of delay lines. For example, the computational complexity of LAUC-VF becomes $O(B.m)$ for a link with m voids and B delay lines. Finally, FDLs are bulky devices since they typically contain tens of kilometres of optical fibre to achieve useful delays.

2.7.4 Deflection routing

With deflection routing, a contending burst is sent out on another output port and follows a suboptimal path to its destination [5]. The task of routing the burst to its destination is left to another switch downstream.

Hsu *et al* conduct a performance analysis of deflection routing in [65]. Simulation results show that deflection routing noticeably improves the blocking performance in an OBS network. However, this method also increases end-to-end transmission latency. We reported in section 2.4.4 that the burst time offset shrinks at each switch. Since deflection routing increases the hop count traversed by a burst, the original time offset may be insufficient for the deflected path. If this is the case, a burst may arrive at a switch before its header is processed. This is called the "insufficient offset time" problem in [65]. To avoid this problem, the authors propose FDLs to keep a suitable time offset between header and burst. Finally, deflection routing also results in out-of-order burst arrival at the destination node. To manage the drawbacks of this method, there should be a limit on the number of deflections allowed per burst [65].

2.7.5 Burst Segmentation

With the previous methods, if wavelength conversion, buffering or deflection routing fails, the entire burst is dropped. The objective of these methods is to reduce burst blocking probability. In contrast, the goal of burst segmentation is to reduce end user data loss (e.g.: IP packets) [66]. The idea behind burst segmentation is based on the

observation that contention often results from partial overlap. With burst segmentation, a data burst is divided into several "segments". One segment contains one or several end user data frames (e.g.: ATM cells). In case of contention, only the overlapping part is discarded instead of the whole burst.

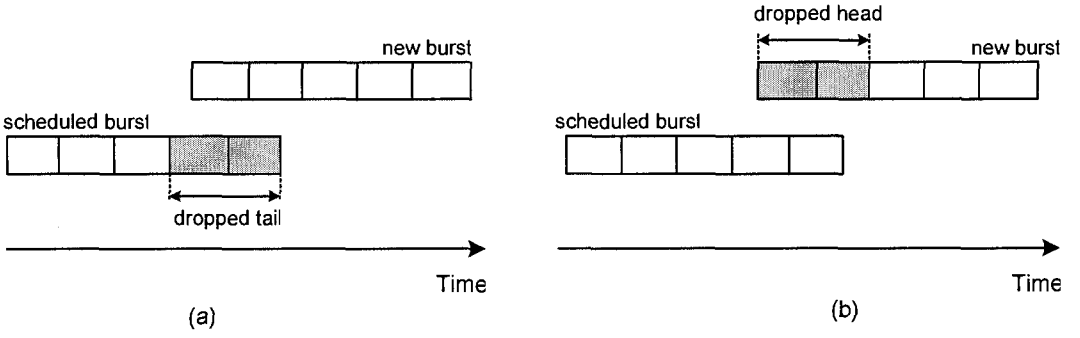

Figure 2.14: Burst segmentation (adapted from [26])

There are two possibilities when discarding part of a burst. Either the tail is dropped (see Figure 2.14(a)) or the head is lost (see Figure 2.14(b)). The head can only be discarded on a new burst whereas only a scheduled burst can have its tail dropped. Dropping the head of a new burst is more straightforward because all the required actions are carried out at the switch where contention occurs. However, if the tail of a burst is discarded, its reservation packet has already been relayed to nodes towards the destination. Intermediate nodes on the rest of the path will allocate bandwidth for the original size of the burst, which is no longer accurate. The unused bandwidth is wasted because it cannot be assigned to other bursts. To avoid this problem, Vokkarane *et al.* [66] suggest sending a second header to update the burst length along the same path. Several refinements of this base algorithm are discussed in [67]. The contending burst is always sent out on the channel that results in minimum segment loss.

Optical composite burst switching (OCBS) is an idea similar to OBS with burst segmentation [68]. With this method, each user data unit is encapsulated in a simple data link (SDL) frame. The format of an SDL frame is shown on Figure 2.15. An optical composite burst is simply a sequence of SDL frames [68].

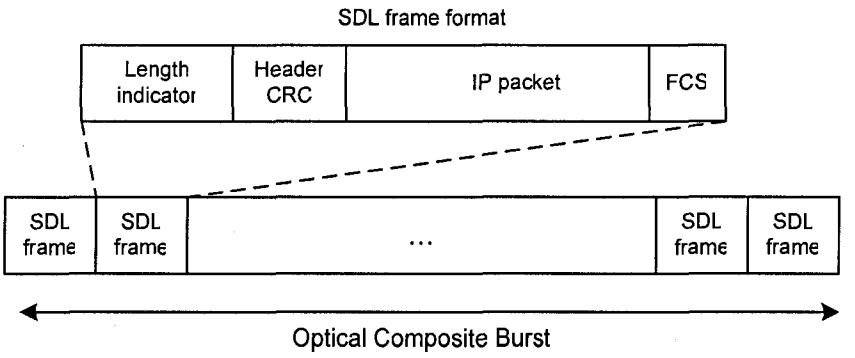

Figure 2.15: Optical Composite Burst format (adapted from [26])

In case of contention, only the head of the overlapping burst is dropped. At the egress node, payload from complete SDL frames is recovered while truncated SDL frames are discarded. To minimise user data loss, a contending burst is always sent out on the channel that results in minimum burst overlap.

Partial drop techniques such as burst segmentation and OCBS provide better performance in terms of end user data loss. This is achieved at the cost of increased complexity. Moreover, OCBS introduces overhead because it encapsulates every user data unit in a separate SDL frame.

2.8 Congestion Control on OBS

We have established earlier the main problem with OBS networks, which is burst loss caused by contention. Despite the performance improvements gained with contention resolution methods, blocking probability remains an issue. To complement these proposals, the traffic load offered to an OBS network must be monitored to keep it away from congestion. This section focuses on earlier works dealing with congestion control in OBS networks.

2.8.1 TCP over OBS

TCP is the dominant congestion control method used over the Internet today. It is therefore essential to study the performance of TCP on future OBS networks. Several previous works have been published on the performance of TCP on OBS. The main

problem in TCP over OBS results from the random losses in an OBS network and the burstification delay [69]. This issue stems from an apparent mismatch between the operation of OBS and the underlying assumptions of TCP. The bufferless nature of OBS makes it prone to random burst losses. On the other hand, TCP assumes that a packet loss implies network congestion. TCP responds to a lost frame by throttling back its sending rate and going through a recovery period. A similar behaviour is observed in wireless networks, where channel quality significantly affects the performance of TCP [70].

The steady state throughput of a TCP flow is [71]:

$$BW = k\frac{MSS}{RTT \cdot \sqrt{p}} \tag{2.4}$$

k is a constant ($k \approx \sqrt{3/2}$), MSS is the maximum segment size, RTT is the connection round-trip delay and p is the loss probability. As seen on equation (2.4), TCP throughput is adversely affected by the delay and packet losses. Also, each burst may contain multiple packets from the same TCP connection. In case of blocking, these packets are simultaneously lost. This type of loss is mistakenly interpreted by the various implementations of TCP as a time-out event. This is called a *false time-out* (FTO) in [12]. All current TCP implementations respond to a time-out by returning to slow-start. To address this issue, a version of TCP with FTO detection is proposed in [12]. It is shown that FTO detection improves the bandwidth utilization on OBS.

The first detailed study of the interaction between OBS and TCP is presented in [72]. The authors propose TCP throughput models for the most common TCP variants (SACK [73], Reno [74] and New-Reno [75]). The bandwidth penalty resulting from assembly delay, burst loss and multiple packet loss are quantified. In addition, a gain factor called *delayed first loss* (DFL) is identified. This performance gain results from the delay in the first packet loss observed by TCP resulting form burst assembly. This allows the congestion window to reach a large value before the first loss occurs.

Several studies suggest that, given an OBS network with certain burst loss probability and round-trip delay, there is an optimal burst assembly time or size threshold that results

in maximum throughput [31, 69, 72]. Adaptive assembly algorithms can take advantage of this [30].

Cameron *et al.* [11] analyzed the throughput of TCP flows over a single OBS link at equilibrium. This is accomplished by combining the TCP throughput model as shown in equation (2.4) with the OBS loss model and finding the convergence point. The authors also conclude that OBS networks must have many wavelengths with full wavelength conversion to support TCP connections efficiently.

Finally, *TCP decoupling* [14] applies TCP congestion control to regulate the traffic rate between ingress-egress node pairs. Decoupled TCP is the equivalent of having a *virtual* TCP connection between each ingress and egress node. TCP congestion avoidance techniques are applied to the aggregate traffic traversing the OBS network. This method uses the control channel to exchange necessary TCP control packets. Using this technique, OBS ingress nodes can regulate their sending rate according to the congestion level in the network core. Note, however, that this technique does not include retransmission of lost bursts. TCP decoupling is able to keep blocking probability under 1% and increase OBS throughput [14]. However, this improvement is partly caused by burst synchronization when all ingress nodes have the same RTT.

Most of the works cited above either assume a given burst loss probability to find the throughput, or use simple network settings (single link, single TCP session). Given a general meshed OBS network carrying many TCP flows, it is interesting to find the throughput and blocking probability if the network is left to itself. As far as we know, this problem is still open.

2.8.2 Robust OBS

The Robust OBS (ROBS) [76] proposal tries to change the OBS network core into a more reliable transmission medium. This technique adds Burst Drop (BD) and burst ACK (BACK) messages to OBS signalling. Burst sequence numbers are used to detect burst loss. In case of successful transmission, the egress node sends a BACK message to the source. Otherwise, the core node at which blocking occurs notifies the ingress node with

a BP message. Each burst is kept in memory until a BACK message is received or until the ROBS maximum hold time is exceeded. In case of BD, the burst is retransmitted and transmission between the affected ingress-egress nodes pair is *halted* until a BACK is received. ROBS improves the network throughput and reduces burst loss as is shown through simulations [76]. Note that optical burst retransmission, done for the purpose of supporting reliable burst transmission, has been discussed in earlier works [5].

This method retains the self-clocking advantage of TCP since it halts transmission in case of blocking until the retransmission is acknowledged. However, buffering bursts at OBS ingress nodes is arguable for the following reason: Memory size necessary for each destination is at least equal to the bandwidth-delay product of the path. This is expected to be very large in OBS. This means that this technique is limited to short span networks.

2.8.3 Load Balancing

Most studies in OBS assume that routing is based on shortest path. Path length may be measured either in physical distance or hop count. Shortest path routing is appealing because it minimises end-to-end delay as well as blocking probability. However, given a certain network topology, this method always makes the same routing decision regardless of congestion levels. Bursts invariably take the same route, even if a less congested but longer route exists. In contrast, dynamic load-balancing responds to changes in network condition. It distributes the load across the network links in an attempt to avoid congestion. Load balancing is a long-term congestion avoidance method. All the published studies [77-80] on this subject start from the same principle. Several link-disjoint paths are computed for each ingress-egress node pair. The paths are sorted according to their congestion levels. The load balancing algorithm dynamically chooses the least congested path. Congestion is measured periodically to adapt to traffic changes. However, oscillations may occur if several sources simultaneously switch their paths because of congestion to a certain link. After the switching is complete, the previously congested path may become underutilized. The sources may revert to the previous path at the next update, resulting in oscillation. Proper steps must be taken to ensure stable operation of load balancing algorithms. In [77], two algorithms are proposed:

a) Route selection using fixed alternate shortest path: The load balancing technique calculates a set of link-disjoint alternate paths between source-destination pairs. The transmission path is dynamically chosen from this set according to the congestion level.
b) Least congested dynamic routing: this is a route optimization method based on a set of performance metrics such as path length, number of hops, link utilization. These measurements are carried out periodically.

This method reduces the blocking probability in OBS network at the cost of increased delay [77].

Li *et al.* [79] take a similar approach. In their proposal, only two link-disjoint alternate paths are computed between every ingress-egress pair. The first path contains the fewest number of intermediate nodes to minimise burst loss probability. The second path is the next shortest path. Teng *et al.* [80] introduce another approach based on linear optimization.

2.8.4 Rate-based OBS congestion control

With high-speed data networks, rate-control is usually preferred to window-based congestion control. This is the technique used in ATM ABR flow control [18]. The techniques reviewed in this section propose rate-control as a congestion control method for OBS.

2.8.4.1 Robust Optical Burst Switching[5]

Maach *et al.* [19, 81] first studied the advantages of rate-based congestion avoidance on OBS. The goal of this method is to keep the loss rate in the network below a critical level represented as *CLR*. If the burst blocking probability exceeds *CLR*, the edge nodes reduce their sending rates. Every core node monitors the loss probability and reports this information to all edge nodes using a broadcast tree. In the current discussion, the term

[5] This technique has the same name as the one proposed by [76] and discussed in section 2.8.2 but its operation is different. We note that the work of Maach et al. [19, 81] appeared first.

source node means ingress node. These messages are produced either periodically or after a certain number of losses have occurred. A critical load *CL* is defined for the network. *CL* is the total load at which the overall loss rate is equal to critical loss rate *CLR*. Additionally, a critical rate CL_i is assumed to be defined for every ingress node i such that $CL < \sum CL_i$. CL_i can be thought of as a quota reserved for each node. For a source node, if its sending rate L_i is smaller than its critical load CL_i, then it is not involved in the rate adjustment process. However, if $L_i > CL_i$, the source rate is adjusted as follows:

- Ingress node i reduces its rate if network losses are greater than *CLR*. In simulations, this reduction is achieved by adding a fixed increment to the burst interarrival time or by setting $L_i = CL_i$.
- If source i has more traffic to send and losses are below *CLR*, it increases its rate. In [19], this is done by subtracting a fixed value from the burst interarrival time.
- If losses are equal to *CLR*, no rate adjustment is done.

In addition, a burst retransmission scheme is suggested to complement the congestion avoidance method. A sequence number is introduced to identify individual bursts. In case of blocking, the intermediate node sends a negative acknowledgment (NACK) back to the source with the burst sequence number. An ingress node keeps a copy of every transmitted burst. When it receives an NACK, the source retransmits the corresponding data. Detection of successful transmission is based on a timer at the source. Ingress nodes are assumed to know the round-trip time to each destination. If no NACK is received after the RTT has elapsed, the ingress node assumes the burst has been successfully transferred.

Simulations show that this technique is able to maintain the burst loss probability around a desired level. The retransmission scheme also improves the reliability of OBS although it requires additional memory to hold bursts until they are acknowledged. The method used to detect successful transmission also suggests that a separate timer is kept for each burst in transit. Both of these requirements limit the application of burst retransmission to LANs or MANs, as noted by the authors. As each source contributes to the network's

critical load *CLR*, it is reasonable to assume that the quota of each sender depends on the network topology. However, the authors assume that each edge node knows about its critical load L_i. No details are given, addressing the case occurring when more sources are added or leave the network. Also, every intermediate node sends congestion information to all edge nodes, even to sources that do not use the specific intermediate node.

2.8.4.2 Differentiated ABR

In Differentiated ABR (D-ABR) [21], an effective capacity is calculated for each link. The effective capacity (EC) is the rate at which the dropping probability P_{loss} reaches a predefined value. The goal of D-ABR is to keep the traffic load on each link under EC in order to prevent the dropping probability from crossing the set threshold. To do so, ingress nodes regulate the traffic flowing into the OBS network. An explicit rate control method derived from the ERICA [82] algorithm, used in ATM, is applied. Computation of source rates in D-ABR is essentially done as in ERICA, except that D-ABR supports service differentiation. Two traffic classes are supported. D-ABR achieves strict isolation between the high priority (HP) and low-priority (LP) classes in terms of bandwidth allocation precedence. In addition to burst header packets, ingress nodes send resource management (RM) cells periodically on a lightpath. Egress nodes send these RM cells back to the destination on the same path. HP and LP explicit rates are inserted in separate fields in backward RM cells. With this method, a lightpath denotes a traffic flow between a pair of ingress and egress nodes. Strict service class isolation is achieved as follows. The bandwidth allocated to a lightpath is entirely available to high priority traffic. Any capacity not used by HP flow is then assigned to LP traffic.

To enforce the traffic rate of a lightpath, an ingress node handles two queues for each egress node. Each traffic class uses one of these queues. Service rate of each buffer is then controlled by a token bucket.

Simulations indicate that this method is able to maintain the network load around the effective capacity and that bandwidth allocation is fair in the max-min sense.

2.8.4.3 Proportional Control Algorithm with Explicit Reduction Request

Farahmand *et al.* proposed another congestion avoidance method for OBS called *proportional control algorithm with explicit reduction request* (PCwER) [20]. PCwER also aims to maintain the blocking probability on every link around a targeted value. This is achieved by limiting the traffic load on every link. A link connecting node j to node k is denoted by (j, k). In PCwER, each core node measures the traffic load $\rho_{j,k}$ on each of its output ports. These measurements are carried out every Δ seconds. When the load on an output link exceeds a predefined threshold ρ_{TH}, the link is considered congested. It is assumed that intermediate nodes have the list of sources using a congested link. When ρ_{TH} is exceeded, the core node sends a *flow-rate reduction request* (FRR) directly to all ingress nodes using the congested link. The FRR contains the value $R_{j,k}$ by which an ingress node should reduce its rate. When $R_{j,k} > 0$, its value is calculated as follows:

$$R_{j,k} = (\rho_{j,k} - \rho_{TH}) / \rho_{j,k} \tag{2.5}$$

$R_{j,k} = 0$ means that no rate change is allowed whereas $R_{j,k} = -1$ means that an ingress node can increase its rate. To prevent oscillations due to delay, the value of Δ must be larger than the maximum RTT between an intermediate node n and any sources using n.

Upon reception of an FRR, an ingress node adjusts its rate as follows. If $R_{j,k}$ is negative, a constant increment is added to the rate. Otherwise, if $R_{j,k}$ is positive, the rate is multiplied by $(1 - R_{j,k})$. We note that this corresponds to an additive increase, multiplicative decrease (AIMD) algorithm. Sources regulate their sending rate by controlling the burst interdeparture time.

Chapter 3
Optimization Approach to Congestion Control

3.1 Introduction

This chapter introduces the theoretical foundation of an optimization approach to flow control. Existing end-to-end congestion control algorithms (for example TCP) are often the result of intuitive or heuristic methods. Once they are deployed in real networks, several improvements are then introduced, based on empirical observations of performance. In these cases, max-min fairness [83] is commonly chosen as a fairness criterion. A max-min fair allocation maximizes the smallest share[6]. In this chapter, we introduce a more systematic approach to congestion control, which is based on utility maximization. The underlying mathematical framework has been developed by Kelly *et al.* [22-24]. Based on this method, we present a distributed end-to-end congestion control algorithm. Section 3.2 presents the network model and assumptions. In Section 3.3 and 3.4, we look at the optimization goals and utility functions, respectively. Section 3.5 and 3.6 presents the rate control algorithms and, finally, section 3.7 discusses their stability.

[6] Fairness is defined formally in section 3.3.2 and discussed in more detail in section 3.4.

3.2 Background

3.2.1 Network Model

A network is modeled as a set of resources J. A single resource in this set is denoted by j, $j \in J$. A resource may either be a link or a queue (but not both). C_j is the capacity of resource j. Each direction of communication between neighbouring nodes is treated separately because they use different resources. Let L be the set of unidirectional links in the network and let l denote a link in this set, i.e. $l \in L$. A route r is represented by the set of resources $j \in J$ used by a connection. Thus, a route r is a nonempty subset of J. Let R be the set of all routes in the network ($r \in R$). A resource j may not appear twice in a route r. Let $A_{jr} = 1$ if j is a member of r and $A_{jr} = 0$ otherwise. This allows a routing matrix $\boldsymbol{A} = (A_{jr}, j \in J, r \in R)$ to be defined, whose elements are either 0 or 1. A *user* designates a source-destination pair or a flow. We assume that each user uses a fixed route for the duration of a transmission. Therefore, we can identify each route r with a unique user or flow. In the following discussion, we use the terms "user" and "route" interchangeably. Let x_r be the transmission rate of user r and $\boldsymbol{x}$ be the vector whose elements are x_r. The total offered load on resource j is $y_j = \sum_{r \in R} A_{jr} x_r$. In matrix notation, this is the same as $\boldsymbol{y} = \boldsymbol{A}.\boldsymbol{x}$, where $\boldsymbol{y}$ is the vector of elements y_j. In the following discussions, we ignore the discrete nature of the network traffic and use a deterministic fluid flow approximation instead.

3.2.2 Rate Allocation Example

Consider the linear topology network shown in Figure 3.1. This network consists of L links. The network carries $L+1$ routes, denoted R_0 to R_L. Amongst these, L routes (R_1 to R_L) use exactly one link each. There is one route, R_0, which traverses all L links in the network. Let x_0 denote the traffic rate on route R_0 and x_s the rate of the remaining routes. All rates are normalized and the capacity of all links is equal to one.

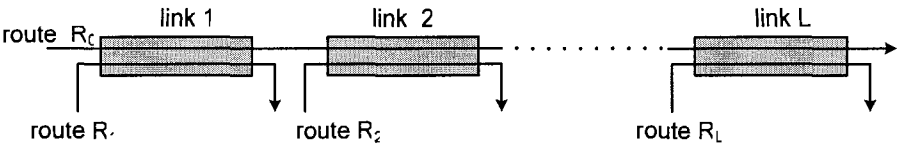

Figure 3.1: Linear topology network[7]

Suppose that our task is to share the available bandwidth among the $L+1$ routes. One possible solution is to give half of the bandwidth of each link to each route, i.e. $x_0 = 0.5$ and $x_s = 0.5$, for all s. Intuitively, we may call this distribution "fair" because it shares the bandwidth of each link equally among its users. It turns out that this distribution corresponds to what is called a "max-min" fair allocation. With this type of allocation, routes using more links do not suffer any rate penalty compared with the others.

Let us consider the efficiency of the previous rate allocation. To do so, we compare the total throughput of this allocation to the maximum possible for this topology. The total throughput of the max-min allocation is $\sum_{r \in R} x_r = (L+1)/2$. Compare this to the maximum throughput that can be achieved, which is equal to L (obtained by setting $x_0 = 0$ and $x_s = 1$ for all other routes). Therefore, it appears that although the max-min rate sharing is considered fair, it is not efficient because the total throughput is almost half the maximum achievable L. On the other hand, the maximum throughput L is not acceptable because it is highly "unfair".

3.2.3 Max-Min Fairness

We have seen an example of max-min allocation in the previous section. This section discusses max-min fairness in more detail. Max-min fairness [83] is arguably the most common definition of fairness in use for rate control methods. It is defined formally as follows [84, 85].

---

[7] Figure adapted from [94].

Definition 3.1 *A vector* $\boldsymbol{x}$ *of transmission rates is* max-min fair *if the following conditions are satisfied:*

- *The allocation is feasible, i.e. for all links, the load does not exceed the capacity.*
- *The rate of a flow cannot be increased without decreasing the rate of another already smaller flow.*

Max-min fairness can be described as the result of a water-filling procedure [84]:

1. Start with all flows set to zero;
2. Increase the rate of all flows at the same speed until some links reach capacity; freeze the bit rate of the flows using these links;
3. Repeat step 2 to nonfrozen flows until all traffic streams are bottlenecked.

Intuitively, a bottleneck is a link that limits the rate of a user from increasing further. However, we will mention often the concept of a bottleneck link in the text. Therefore, it is useful to define it formally as follows.

Definition 3.2 [85] *A link l is a bottleneck for the source r if it is filled to capacity and r has the largest flow among all sources using link l.*

Max-min fairness has the following notable properties [85, 86]:

- A rate allocation is max-min fair if and only if every source has a bottleneck link.
- If a max-min fair allocation exists for a network, it is unique.

3.3 Optimization-Based Rate Allocation

Max-min fairness can be considered as an extreme version of fairness because it always gives priority to the smallest rate in the network. As we saw in section 3.2.2, this comes at the cost of global efficiency. Suppose that in the network shown in Figure 3.1, the source R_1 needs at least $x_1 = 0.6$ units of bandwidth for its application while any bandwidth above 0.2 is of little value to source R_0 [85]. Therefore, a "fair" allocation in this case should take into account the requirements or preferences of users.

3.3.1 Utility Function

Let $U_r(x_r)$ be the *utility function* [24] of user r. It measures the benefit or "happiness" that user r derives from sending at rate x_r through the network. $U_r(x_r)$ is an increasing, strictly concave, and continuously differentiable function of x_r for $x_r \geq 0$. Due to its concavity, $U_r(x_r)$ follows the principle of diminishing returns[8]. We assume that $\lim_{x_r \to +\infty} U'_r(x_r) = 0$. To ensure that x_r is always positive, let: $\lim_{x_r \to 0} U_r(x_r) = -\infty$ [87]. We assume that utilities are additive so that the total utility of a given allocation is the sum of the utilities of individual users.

3.3.2 Alternate Definition of Fairness

Kelly *et al.* [22-24] framed the rate allocation problem as a global utility maximization problem, subject to resource capacity constraints. In this context, an allocation is said to be fair if it is feasible and it maximizes the aggregate utility of the network.

The initial work of Kelly *et al.* drew much attention, leading to an important volume of work on this subject becoming available at the present time [85, 87-94]. By interpreting congestion feedback as a resource price, classic economic optimization tools are used to maximize both the network's and the user's benefit. In addition, the concept of a resource price motivates users to use resources more efficiently, as we explain in section 3.4.

3.3.3 Optimization goals

Let the vector $\boldsymbol{x} = (x_r, r \in R)$ be the vector of user transmission rates, and $\boldsymbol{C} = (C_j, j \in J)$ be the vector of resource capacities. A social planner seeks to maximize the overall "happiness" of the network users. Therefore, the optimal point of operation of the network solves the following optimization problem [24], where the constraints are applied element by element:

---

[8] The principle of diminishing returns is well-known in the field of economics. It states that each unit of additional input yields less additional output than the previous one.

SYSTEM $(\boldsymbol{U}, \boldsymbol{A}, \boldsymbol{C})$:

$$\text{Maximize:} \quad \sum_{r \in R} U_r(x_r) \tag{3.1}$$

$$\text{subject to:} \quad \boldsymbol{Ax} \leq \boldsymbol{C} \tag{3.2}$$

$$\text{over:} \quad \boldsymbol{x} \geq \boldsymbol{0} \tag{3.3}$$

Since the objective function (3.1) is strictly concave and the feasible region (3.2), (3.3) is convex, a unique solution $\boldsymbol{x}^*$ exists for the previous problem, according to optimization theory [95]. Appendix A offers a brief introduction to convex optimization theory, which applies to the current problem. The solution can be found using the method of Lagrange multipliers. Consider the Lagrangian function corresponding to this problem:

$$L(x, \mu) = \sum_{r \in R} U_r(x_r) + \boldsymbol{\mu}^T (\boldsymbol{C} - \boldsymbol{Ax}) \tag{3.4}$$

where $\boldsymbol{\mu}$ is the vector of Lagrange multipliers μ_j, $j \in J$.

The Karush-Kuhn-Tucker (KKT) conditions [96] for optimality are the following:

$$\frac{\partial L}{\partial x_r} = U'_r(x_r) - \sum_{j \in r} \mu_j = 0 \tag{3.5}$$

$$\boldsymbol{\mu}^T (\boldsymbol{C} - \boldsymbol{Ax}) = 0 \tag{3.6}$$

$$\boldsymbol{Ax} \leq \boldsymbol{C} \tag{3.7}$$

$$\boldsymbol{\mu} \geq \boldsymbol{0} \tag{3.8}$$

In optimization theory, the Lagrange multipliers μ_j may be interpreted as the *cost* incurred by a unit of flow through bottlenecked resources or as *shadow prices* for additional resource capacity [24]. This introduces the concept of resource cost as a means for controlling network congestion. From (3.6), a resource incurs a nonzero cost μ_j only when its associated capacity constraint is active. In this case, we say that the resource is *congested*. Otherwise, a resource has zero cost when its load is less than its capacity. We will elaborate more on the definition of congestion later when we consider the case of various types of networks.

However, we note that solving the previous optimization problem requires knowledge of the utility functions of all users, their routes, and the capacity of each resource. This would be extremely difficult to do in a centralized manner because of the amount of information that needs to be exchanged and processed [23].

To solve this difficulty, the previous optimisation problem is reformulated and divided in two parts as follows. Suppose that user r is charged an amount λ_r per unit bandwidth. If each user is free to set its transmission rate to maximize its own benefit, a user will try to maximize its utility minus the cost. Thus, the optimal rate for a user r solves:

USER$_r$ (U_r, λ_r):

$$\text{Maximize:} \qquad U_r(x_r) - \lambda_r x_r \tag{3.9}$$

$$\text{over:} \qquad x_r \geq 0 \tag{3.10}$$

Since $U_r(.)$ is strictly concave, the problem *USER*$_r$ (U_r, λ_r) always has a unique solution, which satisfies [97]:

$$U'_r(x_r) = \lambda_r \tag{3.11}$$

Because of user charges, the network receives income from users at rate λ_r per unit flow. If the network is allowed to set user transmission rates x_r, it will try to maximize its revenue as follows:

NETWORK ($\boldsymbol{A}$, $\boldsymbol{C}$, $\boldsymbol{\lambda}$):

$$\text{Maximize:} \qquad \sum_r \lambda_r x_r \tag{3.12}$$

$$\text{subject to:} \qquad \boldsymbol{Ax} \leq \boldsymbol{C} \tag{3.13}$$

$$\text{over:} \qquad \boldsymbol{x} \geq 0 \tag{3.14}$$

We note that, after dividing the problem in equation (3.1) in two parts, the network problem does not require knowledge of the users' utility functions and the user problem does not require the routing matrix $\boldsymbol{A}$ or other users' utility functions. Therefore, the optimization problem can be solved in a distributed manner.

Theorem 3.1 [24] *There exists a price vector $\boldsymbol{\lambda} = (\lambda_r, r \in R)$ such that the vector $\boldsymbol{x}$, formed from the rates x_r that solve USER$_r$ (U_r, λ_r), solves NETWORK $(\boldsymbol{A}, \boldsymbol{C}, \boldsymbol{\lambda})$. In this case, the vector $\boldsymbol{x}$ also solves SYSTEM $(\boldsymbol{U}, \boldsymbol{A}, \boldsymbol{C})$.*

Put simply, this means the following. If the link prices are "right", i.e. if the prices are equal to the Lagrange multipliers of the *SYSTEM* problem, the users and the network will bring their rates to the global optimum by acting in a greedy manner.

Proof [24] We have already noted that the problem *USER*$_r$ (U_r, λ_r) has a unique solution x_r, which is determined by $U'_r(x_r) = \lambda_r$. Suppose that x_r also solves *NETWORK* $(\boldsymbol{A}, \boldsymbol{C}, \boldsymbol{\lambda})$. The Lagrangian function for the *NETWORK* $(\boldsymbol{A}, \boldsymbol{C}, \boldsymbol{\lambda})$ problem is:

$$L(x,q) = \sum_{r \in R} \lambda_r x_r + \boldsymbol{q}^T (\boldsymbol{C} - \boldsymbol{A}\boldsymbol{x})$$

If the pair (x_r, λ_r) that solves *USER*$_r$ (U_r, λ_r) also solves *NETWORK* $(\boldsymbol{A}, \boldsymbol{C}, \boldsymbol{\lambda})$, the following KKT conditions hold:

$$\frac{\partial L}{\partial x_r} = \lambda_r - \sum_{j \in r} q_j = 0 \tag{3.15}$$

$$\boldsymbol{q}^T (\boldsymbol{C} - \boldsymbol{A}\boldsymbol{x}) = 0 \tag{3.16}$$

$$\boldsymbol{A}\boldsymbol{x} \le \boldsymbol{C} \tag{3.17}$$

$$\boldsymbol{q} > 0 \tag{3.18}$$

Replacing q in (3.15)-(3.18) with μ and using the fact that $U'_r(x_r) = \lambda_r$, we obtain the same conditions as in (3.5)-(3.8), which means that x_r solves *SYSTEM* $(\boldsymbol{U}, \boldsymbol{A}, \boldsymbol{C})$ as well. We have established the last part of Theorem 3.1. As for the existence of the vector $\boldsymbol{\lambda}$, we

already know that *SYSTEM* ($\boldsymbol{U}, \boldsymbol{A}, \boldsymbol{C}$) always has a solution, with Lagrange multipliers μ_j. The elements of $\boldsymbol{\lambda}$ are $\lambda_r = \sum_{j \in r} \mu_j$.

3.3.4 Demand Function

Given a total price λ_r for route r, the user adjusts his transmission rate to maximize his benefit as seen in section 3.3.3. We can use a function $f_r(.)$ to describe this relationship between the route price and the transmission rate of user r as shown in equation (3.19). $f_r(.)$ is called the demand function [98].

$$x_r = f_r(\lambda_r) \tag{3.19}$$

Consider the function $g(z) = U'^{-1}_r(z)$, where $u^{-1}(.)$ denotes the inverse function of a function $u(.)$. Applying $g(z)$ to both sides of equation (3.11) and using the fact that $g^{-1}(g(z)) = z$, we obtain the following expression [98]:

$$x_r = f_r(\lambda_r) = U'^{-1}_r(\lambda_r) \tag{3.20}$$

Since U_r is concave, its derivative U'_r is a decreasing function. The inverse function of U'_r is also decreasing. Therefore, $f_r(.)$ is a decreasing function of the route price λ_r. This means that a user subscribes to less bandwidth as it gets more expensive.

3.4 Utility Functions and Fairness

The utility function determines the properties of the rate allocation as we explain next. Various fairness criteria can be achieved when users choose the appropriate utility function. To illustrate this, we discuss in this section two different fairness criteria: *proportional* fairness [24] and the so-called *minimum potential delay* fairness [94] and compare them with max-min fairness.

Let us revisit the linear topology network shown in Figure 3.1. A natural optimization objective is to use the available bandwidth as efficiently as possible. This means

allocating the bandwidth so that the total throughput $\sum_r x_r$ of the network is maximized [94]. However, this may result in a rate allocation where some flows must be zero, as we already mentioned in section 3.2.2. To reach the stated objective, each link should be filled to capacity. Referring back to Figure 3.1, if x_0 is the rate assigned to route R_0, then the rate of the other routes must be $x_r = 1 - x_0$. In this case, the sum of all rates is $x_0 + L.x_r = L - (L-1)x_0$. This is maximal for $x_0 = 0$, corresponding to a total throughput of L [94]. Although this method achieves the maximum feasible throughput for this network, this allocation is not acceptable because $x_0 = 0$. In the next sections, we discuss other rate allocations that do not display this problem.

3.4.1 Minimum Potential Delay Fairness

Minimum potential delay fairness was introduced by Massoulié and Roberts in [94]. It is achieved with a utility function of the following form:

$$U_r(x_r) = -\frac{1}{x_r} \tag{3.21}$$

Therefore, the network tries to maximize $\sum_r -\frac{1}{x_r}$, or conversely, to minimize $\sum_r \frac{1}{x_r}$. This is motivated as follows. Suppose each network user has a file of a given size to transfer. The transmission time for each file is proportional to $1/x_r$. Therefore, by minimizing $\sum_r \frac{1}{x_r}$, the network reduces the total transmission delay for the set of files. The "minimum potential delay" term follows from this property. The more general weighted version of this criterion minimizes $\sum_r \frac{\omega_r}{x_r}$.

Let x_0 be the rate of R_0 and x_s ($1 \le s \le L$) denote the rate of the other routes shown in Figure 3.1. It is obvious that all links must be fully utilized to reach the objective, i.e. $x_s = 1 - x_0$. The minimum potential delay fair allocation can be found as follows [94]. First, note that $1/x_r$ is strictly convex. Therefore, the objective function always has a minimum.

Objective function: $\min \sum \frac{1}{x_r} = \frac{1}{x_0} + \frac{L}{x_s} = \frac{1}{x_0} + \frac{L}{(1-x_0)}$

The minimum is reached when:

$$\frac{d}{dx_0}\left(\frac{1}{x_0} + \frac{L}{(1-x_0)}\right) = 0 \Rightarrow -\frac{1}{x_0^2} + \frac{L}{(1-x_0)^2} = 0$$
$$\Rightarrow \frac{1}{x_0^2} = \frac{L}{(1-x_0)^2}$$

The last expression admits two solutions. Only one is feasible, which determines the rate of R_0 as follows:

$$\frac{1}{x_0} = \frac{\sqrt{L}}{(1-x_0)} \Rightarrow x_0 = \frac{1}{(\sqrt{L}+1)}$$

For the other routes, the rate is:

$$x_s = 1 - x_0 = \frac{\sqrt{L}}{(\sqrt{L}+1)}$$

The total throughput is $L+1-\sqrt{L}$. This is better, compared with the case of max-min fairness since $L+1-\sqrt{L} > (L+1)/2$ for $L > 1$. However, as shown by the expression for x_0 above, the gain in total throughput comes at the cost of a penalty to long routes.

As an example, it is argued in [99] that the Additive Increase Multiplicative Decrease (AIMD) strategy used by TCP is minimum potential delay fair.

3.4.2 Proportional Fairness

Weighted proportional fairness with weights ω_r is obtained by choosing a utility function of the following form [24]:

$$U_r(x_r) = \omega_r \log x_r \tag{3.22}$$

From equation (3.1), it follows that a proportionally fair rate allocation maximizes $\sum_r \omega_r \log x_r$. Choosing a logarithmic utility function removes the possibility that x_r is zero for some flows.

A feasible rate allocation vector $\boldsymbol{x}$ is proportionally fair if, for any other feasible vector $\boldsymbol{x}^*$, the sum of proportional changes is zero or negative [23]. For the more general case of weighted proportional fairness, this can be expressed as:

$$\sum_{r \in R} \omega_r \frac{x_r^* - x_r}{x_r} \leq 0 \tag{3.23}$$

Loosely, in a proportionally fair rate sharing, the user rate x_r is proportional to its weight ω_r. This can be drawn from equation (3.20). Substituting U_r with $\omega_r \log x_r$, we get the following rate allocation at equilibrium: $x_r = \omega_r / \lambda_r$.

For the network depicted in Figure 3.1, the rate allocation is found as follows [94]. We use the same notation as in the previous section and assume that $\omega_r = 1$ for all r.

Objective function: $\max \sum \log x_r = \log x_0 + L \log x_s = \log x_0 + L \log(1 - x_0)$

The maximum is reached when:

$$\frac{d}{dx_0}\left(\log x_0 + L \log(1-x_0)\right) = 0 \Rightarrow \frac{1}{x_0} - \frac{L}{(1-x_0)} = 0$$

$$\Rightarrow \frac{1}{x_0} = \frac{L}{(1-x_0)} \Rightarrow x_0 = \frac{1}{(L+1)}$$

$$\Rightarrow x_s = \frac{L}{(L+1)}$$

The total throughput is $L-(L-1)/(L+1)$, which is the highest of the three cases we considered (including max-min). However, long routes are even more penalized compared with minimum potential delay fairness.

The previous discussion demonstrated the advantage of the optimization approach and the flexibility offered by the utility function.

3.5 Primal Rate Control Algorithm

We introduce in this section the rate adaptation strategy applied by users in order to reach the optimal operating point of the network.

3.5.1 Optimization Problem Relaxation

We saw in the proof of Theorem 3.1 that, for the users to converge by themselves to the global optimum, the vector of route prices λ_r must satisfy: $\lambda_r = \sum_{j \in r} \mu_j$, where μ_j ($j \in J$) are the Lagrange multipliers of *SYSTEM* (***U***, ***A***, ***C***). However, these Lagrange multipliers are no longer available when *SYSTEM* (***U***, ***A***, ***C***) is divided in two parts to allow a decentralized implementation (see section 3.3.3). For this reason, the primal rate control algorithm provided by Kelly *et al.* [23] solves a relaxation of the global optimization problem (3.1)-(3.3). In this relaxed version, the capacity constraints (3.2) are replaced by penalty functions.

The relaxed optimization problem is the following [23, 85]:

$$\text{Maximize:} \qquad \mathcal{U}(\boldsymbol{x}) = \sum_{r \in R} U_r(x_r) - \sum_{j \in J} \int_0^{\sum_{s: j \in s} x_s} p_j(z)dz \tag{3.24}$$

$$\text{over:} \qquad x_r \geq 0 \tag{3.25}$$

where $p_j(.)$ is the penalty function for resource j and $\sum_{s: j \in s} x_s$ denotes the total load on j.

Penalty functions are the subject of the following section.

3.5.2 Penalty Function

In order to regulate the network load, a resource j incurs a penalty $p_j(y_j)$ when it carries a load y_j. The function $p_j(.)$ is called a penalty function or barrier function [23, 85]. We can think of $p_j(y)$ as a cost of sending a *unit of bandwidth* through resource j. This price can be conveyed to users by marking the packets crossing a congested resource, as suggested in [23]. For example, this can be done using a scheme similar to Explicit Congestion Notification (ECN) [100]. In this case, $p_j(y)$ represents the proportion of traffic that is

marked at resource j. Individual users respond to this feedback according to their demand function (see equation (3.20)). Note that aggregate marking is applied at resources because the argument of the penalty function is the total load y. Therefore, the resource does not need to identify individual flows.

The function $p_j(.)$ satisfies the following conditions:

a. $p_j(.)$ is a non-negative, continuous and increasing function.

b. $p_j(y) < 1$ for $y > 0$, since $p_j(y)$ is the proportion of packets which receive a mark.

c. $\lim_{y \to +\infty} p_j(y) = 1$. This means that, as the offered load increases, the resource marks an increasing proportion of the offered load. A heavily loaded resource marks almost all packets that pass through it. Because y is the total load offered to resource j, it may exceed the resource capacity.

d. $\lim_{y_j \to \infty} \int_0^{y_j} p_j(z)dz = \infty$, i.e. as the load increases, the price of a resource does not stay "small."

Resource prices are assumed to be additive along a route such that the price of route r is equal to the sum $\sum_{j \in r} p_j(y_j)$ of the prices of the resources on r. This implies independent marking between resources. Since $p_j(y)$ represents the ratio of packets through j that receive a mark, the total number of marks received by a user r in each time unit is $x_r(t)\sum_{j \in r} p_j(y_j(t))$. Note that, although we use a fluid model assumption, marking can only be done at the packet level. Section 3.5.5 provides examples of how the price $p_j(y)$ can be computed.

3.5.3 Existence of a Solution

First, note that the objective function $\mathcal{U}(\boldsymbol{x})$ is strictly concave [23, 85]. We remind the reader that the utility functions $U_r(.)$ are concave. Let $C_j(y_j) = \int_0^{y_j} p_j(z)dz$. Since the

penalty function $p_j(.)$ is increasing, the function $C_j(.)$ is convex and $-Cj(.)$ is concave. Therefore, $\mathcal{U}(x)$ is concave because it is formed by the sum of concave functions. Next, we assumed in section 3.3.1 that $\lim_{x_r \to +\infty} U'_r(x_r) = 0$ and $\lim_{x_r \to 0} U_r(x_r) = -\infty$. Because of our assumptions regarding $p_j(.)$, we have: $\lim_{\|x\| \to 0} \mathcal{U}(\boldsymbol{x}) = -\infty$ and: $\lim_{\|x\| \to \infty} \mathcal{U}(\boldsymbol{x}) = -\infty$. Therefore, $\mathcal{U}(x)$ has a unique maximum in the feasible region $\boldsymbol{x} \geq 0$.

At the maximum, we have $\frac{d\mathcal{U}}{dx_r} = 0$ for all routes r. Therefore, the vector of rates that maximizes the objective function $\mathcal{U}(\boldsymbol{x})$ satisfies the following:

$$U'_r(x_r) - \sum_{j \in r} p_j \left(\sum_{s: j \in s} x_s \right) = 0 \tag{3.26}$$

3.5.4 Primal Rate Adaptation Rule

3.5.4.1 General Case

We now present the primal rate adaptation rule that individual users apply in order to converge towards this solution. The general form of the rate control algorithm is given by equation (3.27) and (3.28) [97]. We show in section 3.5.4.2 why the method described in equations (3.27) and (3.28) solves the optimization problem.

$$\frac{d}{dt} x_r(t) = \kappa_r \left(x_r(t).U'_r(x_r(t)) - x_r(t). \sum_{j \in r} \mu_j(t) \right) \tag{3.27}$$

where:

$$\mu_j(t) = p_j \left(\sum_{s: j \in s} x_s(t) \right) \tag{3.28}$$

κ_r is a positive constant acting as a control gain. As defined previously, $x_r(t)$ is the rate of user r at time t, $p_j(y)$ is the price of resource j when carrying a load y, and U_r is the utility function of user r.

3.5.4.2 Stability and Convergence

The convergence and stability of the rate update rules (3.27) and (3.28) can be established using a Lyapunov function argument as follows [23]. First, note that (3.27) has an equilibrium point defined by $\frac{d}{dt}x_r(t)=0 \Rightarrow U'_r(x_r(t))=\sum_{j\in r} p_j\left(\sum_{s:j\in s} x_s(t)\right)$, which corresponds to the maximum of $\mathcal{U}(\boldsymbol{x})$, as shown in equation(3.26). Let the constant K be the unique maximum of $\mathcal{U}(\boldsymbol{x})$, which always exists as we showed at the beginning of this section. Let $\boldsymbol{x}^*$ denote the vector of optimal rates x_r^*, i.e. $\mathcal{U}(\boldsymbol{x}^*)=K$.

Consider the following function:

$$Q(\boldsymbol{x})=K-\mathcal{U}(\boldsymbol{x}) \tag{3.29}$$

$Q(.)$ is strictly convex since $\mathcal{U}(\boldsymbol{x})$ is strictly concave. Moreover, $Q(.)$ admits a unique minimum $Q(\boldsymbol{x}^*)=0$. Since $Q(.)$ is strictly convex, $Q(\boldsymbol{x})>0$ for $\boldsymbol{x}\neq\boldsymbol{x}^*$.

Let us define the following axis translation: $\boldsymbol{a}=\boldsymbol{x}-\boldsymbol{x}^* \Rightarrow \boldsymbol{x}=\boldsymbol{a}+\boldsymbol{x}^*$. This implies that $dx_r=da_r$, where a_r is an element of the vector $\boldsymbol{a}$. Consider the following function: $\mathcal{V}(\boldsymbol{a})=Q(\boldsymbol{a}+\boldsymbol{x}^*)$. More explicitly:

$$\mathcal{V}(\boldsymbol{a})=K-\sum_{r\in R}U_r(a_r+x_r^*)+\sum_{j\in J}\int_0^{\sum_{s:j\in s}(a_s+x_s^*)} p_j(z)dz \tag{3.30}$$

$\mathcal{V}(\boldsymbol{a})$ is a Lyapunov function for (3.27), as we show next [23]. First, $\mathcal{V}(0)=Q(0+\boldsymbol{x}^*)=0$ and $\mathcal{V}(\boldsymbol{a})>0$ for $\boldsymbol{a}\neq\boldsymbol{0}$. In other words, $\mathcal{V}(.)$ is positive definite. Next, consider the following derivatives:

$$\frac{da_r}{dt}=\frac{dx_r}{dt}=\kappa_r\left(\left(a_r(t)+x_r^*\right).U'_r\left(a_r(t)+x_r^*\right)-\left(a_r(t)+x_r^*\right).\sum_{j\in r}p_j\left(\sum_{s:j\in s}\left(a_s(t)+x_r^*\right)\right)\right)$$

in addition:

$$\frac{\partial \mathscr{V}}{\partial a_r} = \sum_{j \in r} p_j \left(\sum_{s: j \in s} \left(a_s(t) + x_r^* \right) \right) - U'_r \left(a_r(t) + x_r^* \right)$$

Therefore,

$$\frac{d\mathscr{V}}{dt} = \sum_{r \in R} \frac{\partial \mathscr{V}}{\partial a_r} \cdot \frac{da_r}{dt} = -\sum_{r \in R} \kappa_r \left(a_s(t) + x_r^* \right) \left(U'_r \left(a_r(t) + x_r^* \right) - \sum_{j \in r} p_j \left(\sum_{s: j \in s} \left(a_s(t) + x_r^* \right) \right) \right)^2 \leq 0$$

Additionally, we note that $\frac{d\mathscr{V}}{dt} = 0$ when $\boldsymbol{a} = \mathbf{0}$ and that $\lim_{\|a\| \to \infty} \mathscr{V}(\boldsymbol{a}) = \infty$, which means that the equilibrium of the system is globally asymptotically stable [23]. Therefore, when users apply the rate adaptation algorithm given in equation (3.27), the rate allocation always converges to the optimal operation point determined by equation (3.26).

3.5.4.3 Proportionally Fair Rate Controller

For the particular case $U_r(x_r) = \omega_r \log x_r$, equation (3.27) simplifies to equation (3.31).

$$\frac{d}{dt} x_r(t) = \kappa_r (\omega_r - x_r(t) . \sum_{j \in r} \mu_j(t)) \tag{3.31}$$

The rate adjustment rule given in equation (3.31) is commonly known as the *Kelly control* and is first proposed in [23]. This method can be motivated as follows. The network charges each user r an amount equal to $\sum_{j \in r} \mu_j(t)$ per unit traffic flow. Therefore, cost per unit time to user r increases linearly with that user's sending rate and is equal to $x_r(t) . \sum_{j \in r} \mu_j(t)$. A user r picks a price ω_r. We can regard ω_r as the cost per time unit that user r is willing to "pay". It can also be interpreted as the number of marks per time unit that r is willing to sustain. Using the rate adjustment rule, user r receives bandwidth in proportion to ω_r. We note that the only network feedback needed by users to converge to optimality is the total price along its route.

This rate adjustment rule can also be interpreted as follows. When there is no congestion, user r additively increases its rate by an amount proportional to ω_r. When a congestion signal is received, the rate undergoes a multiplicative decrease [23]. Therefore, equation

(3.31) describes an AIMD rate control algorithm. The AIMD rate adjustment rule has several desirable properties as laid out by Chiu and Jain in [101]. These properties are summarized as follows:

a. Efficiency: ability to keep the network out of the congestion range.
b. Fairness.
c. Distributed operation: each user does not need knowledge of other users' state.
d. Convergence.

Most importantly, equation (3.31) results in a proportionally fair allocation because it corresponds to a logarithmic utility function. The equilibrium point x_r^* for the proportionally fair rate controller is given in equation (3.32).

$$x_r^* = \frac{\omega_r}{\sum_{j \in r} \mu_j} \tag{3.32}$$

From the previous equation, we note that the allocation of route r is proportional to ω_r. Therefore, we refer to ω_r as the weight of route r in the rest of the text.

3.5.5 Example Penalty Functions

In this section, we discuss two examples of suitable pricing functions p_j for use with the algorithm in equation (3.31). In a real network setting, the price may correspond to packet loss probability or queueing delay.

In the first example, a link j with capacity C_j is modeled as an M/M/1 queue. The total load offered to the link is y. Threshold marking may be performed at the queue to convey resource price. The price function is then equal to the probability that the queue length is greater than or equal to the threshold B and is given in equation (3.33) [102, 103]:

$$p_j(y) = \left(\frac{y}{C_j}\right)^B \tag{3.33}$$

The second marking function is proposed in [87, 91]. This method does not require a queue model and calculates the link price as follows:

$$p_j(y) = \frac{(y - C_j)^+}{y} \tag{3.34}$$

where $(\alpha)^+$ means $\sup\{0, \alpha\}$.

Using a deterministic fluid flow model, equation (3.34) may be interpreted as the fraction of traffic lost when offered load exceeds the link capacity C_j.

3.5.6 Marking vs. Charging

The term "price" traditionally implies monetary charges, and usually contains two parts: sunk costs and variable costs. However, in our discussion, price is merely used as a feedback signal for the purpose of flow control. In other words, a user interprets the number of marks (for example ECN marks) it receives as bandwidth cost.

To illustrate this, consider the case of TCP congestion control. We have already introduced the steady state throughput of TCP [71] in equation (2.4), which we repeat here for convenience.

$$BW = k\frac{MSS}{RTT \cdot \sqrt{p}}$$

where k is a positive constant ($k \approx \sqrt{3/2}$), MSS is the maximum segment size, RTT is the round-trip time, and p is the probability of packet loss. We can state from equation (2.4) that, if MSS and RTT are constant, the throughput of a TCP connection is adversely affected by packet loss. Since TCP halves the congestion window when a packet loss occurs, it is understandable that the average TCP throughput decreases when p increases.

However, we can also interpret equation (2.4) in terms of bandwidth price. If we regard p as a price per unit flow, we observe that a TCP user adjusts its transmission rate proportionally to $1/\sqrt{p}$. For this reason, we use the terms charging and marking as synonyms in the rest of the text.

3.6 Dual Algorithm

Because of the duality principle, there is a dual algorithm which also maximizes the objective functions presented in section 3.3.3 [23]. This section discusses the properties of the dual solution. In the dual algorithm, resources dynamically adjust the prices and the user rates are static functions of the price. In contrast, for the primal algorithm, users dynamically adjust their transmission rates and resource prices are function of these rates.

In the dual algorithm, resource j adjusts its price as follows, as described in equation (3.35) [23].

$$\frac{d}{dt}\mu_j(t) = \kappa\left(\sum_{r,j\in r} x_r(t) - q_j(\mu_j(t))\right) \tag{3.35}$$

where:

$$x_r(t) = \frac{\omega_r}{\sum_{k\in r}\mu_k(t)} \tag{3.36}$$

Again, $\kappa > 0$ and $q_j(\eta)$ may be interpreted as the load that generates a price η at resource j [23]. In other words, $q_j = p_j^{-1}$ and is strictly increasing [104]. The dual algorithm has a unique stable point $(\boldsymbol{\mu}, \boldsymbol{x})$ to which all trajectories converge. The stable point is the same as that of the primal algorithm, given in equation (3.32) [104].

The dual algorithm is included here for completeness. It is worth noting that there is a significant body of work studying another dual solution to the optimization problem introduced in section 3.3.3. Interested readers are referred to [88, 98, 105-113]. From this point forward, we will focus on the primal algorithm of equation (3.31).

3.7 Stability

The rate control described by equation (3.31) assumes instantaneous feedback and deterministic evolution of resource prices. In real networks, propagation delays cannot be

avoided and resource prices are based on random processes such as packet loss. This section looks into the perturbations caused by these factors on the rate allocation at equilibrium. We consider these two factors separately, although their effect is combined in a real network.

3.7.1 Stability under Delay

We observe that equation (3.31) describes a closed-loop congestion control. We know from control theory that network delays can destabilize its operation. Kelly *et al.* showed that, for a given round-trip time D_r, there is a maximum value of the gain κ_r above which the system regulated by equation (3.28) and (3.31) becomes unstable [23]. In this section, we introduce stability conditions for networks applying the algorithm (3.31), in the presence of delay. These stability conditions are taken from the work of Johari *et al.* [102]. We adopt the following assumptions about the delay [102]:

a. Queueing delays are negligible and network delays are entirely due to propagation delays. This means the delay of a given path is fixed and is only function of its length;
b. Each flow uses a fixed route. Combined with the previous point, this means the delay on every network path is fixed.

We switch to the discrete time domain for the following discussion. Let n denote the time in discrete domain. Let $d_{jr}^{\rightarrow}$ and $d_{jr}^{\leftarrow}$ denote the forward and backward delay between user r and resource j, respectively. The feedback received by user r at time n is based on resource prices $d_{jr}^{\leftarrow}$ time units ago, i.e. $\mu_j[n-d_{jr}^{\leftarrow}]$. Likewise, the load seen by resource j at time n is function of users' rates $d_{jr}^{\rightarrow}$ time units ago. Therefore, the feedback received at time n is based on the user's rate $d_{jr}^{\rightarrow}+d_{jr}^{\leftarrow}$ time units ago. Assume that $d_{jr}^{\rightarrow}+d_{jr}^{\leftarrow}=D_r$ for all resources j on the route r, which means D_r is the round-trip delay (RTT) of route r. This holds when feedback is transmitted to the sender by the destination, as in TCP [102]. Taking into account the previous delay terms, equation (3.28) and (3.31) yield the following delay-differential equations [102]:

$$x_r[n+1] = x_r[n] + \kappa_r\left(\omega_r - x_r[n-D_r]\sum_{j\in r}\mu_j[n-d_{jr}^{\leftarrow}]\right) \tag{3.37}$$

where:
$$\mu_j[n] = p_j\left(\sum_{s:j\in s} x_s[n-d_{js}^{\rightarrow}]\right) \tag{3.38}$$

In the previous discrete time form of the rate control method, $x_r[n]$ may represent the number of packets sent during timeslot n. Also, we note how the rate at time $n+1$ depends on the feedback at time n when $D_r = 0$. It is easy to see that the previous discrete time system has an equilibrium point, which is the same as in (3.32) [23].

3.7.1.1 Single Route, Single Resource

Let us consider first the case of a network with a single link and a single route. The single user adjusts its transmission rate as follows [102]:

$$x[n+1] = x[n] + \kappa\left(\omega - x[n-D]p\left(x[n-D]\right)\right) \tag{3.39}$$

where D is the RTT of the single route.

Let x^* be the transmission rate of the user at equilibrium, which satisfies $\omega = x^* p(x^*)$. Assume that $p(.)$ is differentiable at the equilibrium and that its derivative is $p'(.)$. To simplify the writing in this section, let p and p' denote $p(x^*)$ and $p'(x^*)$ respectively. Consider the following linearization of equation (3.39): $x[n] = x^* + y[n]$. The term $y[n]$ is updated as follows [102]:

$$y[n+1] = y[n] - \kappa\left(p - x^* p'\right)y[n-D] \tag{3.40}$$

The details of the previous linearization are given in Appendix B.1. The system described by equation (3.39) is locally stable if all the roots of the characteristic equation (3.41) have modulus less than one [102].

$$\lambda^{D+1} - \lambda^D + \kappa(p + x^* p') = 0 \tag{3.41}$$

Johari *et al.* [102] showed that the system described by equation (3.39) is locally stable if

$$\kappa(p + x^* p') < 2\sin\left(\frac{\pi}{2(2D+1)}\right) \tag{3.42}$$

Conversely, the system is unstable if

$$\kappa(p + x^* p') > 2\sin\left(\frac{\pi}{2(2D+1)}\right)$$

A proof is provided by Johari *et al.* [102] for this result and is reproduced in Appendix B.2.

3.7.1.2 Meshed Networks with Heterogeneous Delays

Johari *et al.* [102] proved that, for general mesh networks where all nodes have the same round-trip time D, the stability condition is the following:

$$\kappa_r \sum_{j \in r}\left(p_j + p'_j \sum_{s: j \in s} x_s\right) < 2\sin\left(\frac{\pi}{2(2D+1)}\right) \tag{3.43}$$

where p_j and p'_j denote the value of the penalty function and its derivative at equilibrium.

This condition has an appealing form because it involves only information available to users and does not require central control. Therefore, it is suitable for distributed operation. This condition is essentially the same as the one in (3.42) but takes into account the fact that the resource prices are function of the total load and that prices are additive over a route.

Obviously, the condition $D_r = D$ for all r is difficult, if not impossible, to achieve in a network of reasonable complexity. This seems to diminish the usefulness of this result. However, Johari *et al.* [102] conjectured that the criterion of equation (3.43) also applies to networks with heterogeneous delays. They simulated 10,000 random networks in an attempt to find a counterexample to their conjecture. In all cases, the systems were asymptotically stable when (3.43) is applied.

We note that Massoulié [93] and Vinnicombe [114] separately derived stability conditions for networks with diverse delays. However, the proofs only apply to the continuous time algorithm.

3.7.1.3 Bounds on Rate Oscillations

Even if the network is not stable according to the conditions above, it can be shown that the amplitude of rate oscillations is bounded [91]. Furthermore, the authors in [91] argue the peak rate of oscillations is close to the equilibrium rate.

Let x_r^* denote the equilibrium rate of a user r running the algorithm of equation (3.31). Assume a network with a single resource and in which all users have the same round-trip time D. A user r applies the rate adaptation rule shown in equation (3.37) and (3.38). From equation (3.37), we observe that the transmission rate of user r cannot increase faster than a certain amount determined by $x_r[n+1]-x_r[n]\leq \kappa_r\omega_r$. Note also that the feedback received by user r is based on its transmission rate D time units ago. Starting from the equilibrium x_r^*, the rate of user r can increase only for D time units before it receives feedback notifying it to reduce its transmission rate. Therefore, the upper bound M_r on x_r is given as follows [91]:

$$M_r = x_r^* + \kappa_r\omega_r D \tag{3.44}$$

To find a lower bound for x_r, we need to determine how fast it decreases during the adaptation process described by equation (3.37). The rate x_r decreases fastest when the price paid by user r reaches its maximum. Understandably, the maximum price is reached when the load is at its peak, which is determined by equation (3.44). Thus, the maximum speed of decrease of the rate x_r is determined by: $x_r[n+1]-x_r[n]\geq \kappa_r\left(\omega_r - M_r p\left(\sum_{s\in R} M_s\right)\right)$. Therefore, the lower bound on x_r is given by the following equation [91]:

$$l_r = x_r^* - D\kappa_r\left(M_r p\left(\sum_{s\in R} M_s\right) - \omega_r\right) \tag{3.45}$$

where $p\left(\sum_{s\in R} M_s\right)$ denotes the price of the resource when receiving the peak traffic from all users. The term M_r is computed using equation (3.44).

3.7.2 Stochastic Stability

This section discusses the effect of random price variations on the rate allocation at equilibrium. These random perturbations are mainly due to the fluid flow approximation of discrete packet streams and the stochastic nature of the pricing mechanism (for example packet loss, queue threshold marking).

Kelly *et al.* [23] modeled random perturbations to a linearized version of equation (3.28) and (3.31) using Brownian motions. They showed that, with perturbations coming from the price signals (i.e. from the network) or originating from users, the rate allocation is distributed according to a normal distribution, $x(t) \sim N(x^*, \Sigma)$ [23]. Furthermore, the variance Σ is proportional to the control gain κ_r. Increasing κ_r speeds up the convergence to the stable point. However, this also increases the spread of the rate around the equilibrium [23].

Chapter 4

Proportionally Fair Congestion Control for OBS

4.1 Introduction

The previous chapter introduced the mathematical foundation of optimized flow control. In this chapter, we examine the application of this theory to the development of a congestion control method for optical burst switching, which we named Proportionally Fair OBS (PF-OBS). Our goal is to keep the burst loss rate caused by contention under a chosen threshold. This is achieved by removing congestion from the optical core, alternatively managing it at the OBS edge nodes. Our technique runs in a distributed manner, without the need for coordination between traffic sources. Thanks to its theoretical foundation, PF-OBS inherits the advantages of the underlying principle. These include convergence to a unique stable point, predictability of the rate allocation at equilibrium, proportional fairness, and stability under delay.

First, we explain our motivation for choosing this method in the design of technology that applies congestion control at OBS networks. In section 4.3, we state the assumptions we make throughout the chapter. Section 4.4 discusses the generation of feedback messages in detail. Section 4.5 presents the rate control method applied at the edge nodes. Finally, section 4.6 addresses the stability under delay.

4.2 Rationale

Although many rate-based congestion control algorithms exist for electronic networks, these cannot be applied to OBS easily. The reasons are as follows:

- Many algorithms use queue length or queueing delay as a congestion indication. As we already pointed out, the current state of optical technology restricts the use of this method. It is safer to assume that optical switches used in OBS networks do not have any buffering capability.

- An example of techniques that do not use queueing information is the ERICA [82] algorithm, used in ATM ABR flow control. However, such explicit rate indication methods apply high computational load on the switches [94], as they require intermediate switches to monitor and police each flow.

Therefore, we argue that the proportionally fair rate control algorithm described in the last chapter is a better solution for OBS congestion control. For convenience, we reproduce equation (3.31), which forms the basis of our proposed method:

$$\frac{d}{dt}x_r(t) = \kappa_r(\omega_r - x_r(t).\sum_{j\in r}\mu_j(t))$$

This flow control algorithm has several advantages and fits nicely with the limitations and requirements of OBS as explained below:

a. In systems developed using approaches based on equation (3.31), the congestion is managed at the source nodes, which take up most of the computational load. The added load on intermediate switches is small since they only apply marking in case of congestion. We believe this is in agreement with the design principles behind OBS, which is to minimize processing at intermediate switches. This is the reason we chose the primal algorithm, although a dual solution is also given in equations (3.35) and (3.36).

b. Simplicity: this method manages end-to-end connections. Consequently, a source does not need to know about the congestion level of each link it uses. Similarly, feedback generated at congested resources is function of the aggregate load. There is no need to identify or police each flow.

c. Distributed operation: each source converges to an optimal rate on its own, using only feedback reporting the cumulate price. There is no need for central control.

d. This technique has a well-researched theoretical foundation. This framework introduces *predictability* into the rate allocations at equilibrium. In addition, the optimization approach results in high utilization of network bandwidth.

e. As a particular case of the last point, the method is proven to converge to a unique stable point. In the presence of feedback delay, a stability criterion is also available.

f. It guarantees proportional fairness of rate allocations at equilibrium.

4.3 Background

4.3.1 Assumptions

Before going into the details of our congestion control method, we state in this section general assumptions about the operation of OBS. We also define the scope of our proposal. These assumptions will apply to the rest of the thesis.

A1. We only consider congestion within the OBS domain, i.e. from an ingress node to the matching egress node. Presumably, an OBS network makes up only a portion of an end user's (e.g. IP) packets route. Our proposal applies to transmission rates between ingress and egress nodes. A study of the effect of PF-OBS on other protocols (e.g. TCP) or on adjacent networks is outside the scope of this thesis.

A2. We assume that routes from ingress to egress nodes are unique and fixed. This can be guaranteed using a technique similar to MPLS (Multiprotocol Label

Switching) in the OBS control plane. This defines a virtual connection between a pair of communicating ingress and egress nodes. Using the terminology introduced in Chapter 3, we use the terms "route" and "user" to designate these virtual connections. In addition, "source" refers to an OBS ingress node while "destination" means egress node. Unless specified otherwise, this naming convention applies to the rest of the thesis. For more detailed treatments of label switching in OBS networks, interested readers are referred to [40, 115, 116].

A3. We consider OBS networks as traffic backbones, carrying aggregate flows from access networks and spanning a large geographical area (possibly thousands of kilometres). As a result, propagation delays in OBS links[9] may be significant.

A4. We assume that OBS core nodes have no buffering capability, or that buffering delays are negligible. This is a reasonable supposition, given the current state of optical technology. Therefore, signalling lags in OBS networks are entirely caused by propagation delay. Combined with point A2 above, this means route round-trip times (RTT) do not change. This is mostly relevant to network stability under delay, discussed in section 4.6.

A5. Since the congestion control actions are performed by source nodes, the speed with which our algorithm may respond to congestion at intermediate nodes is limited by propagation delays. Therefore, PF-OBS is not designed to manage congestion duration shorter than the RTT. We are only concerned with congestion that lasts longer than the RTT.

A6. We assume that packets on OBS control channels are never lost. This includes burst header packets (BHP) and other control packets used for congestion control.

---

[9] In our discussion, an OBS link designates a section of optical fibre cable connecting two OBS nodes. A link may carry several wavelengths (also called channels). Communication between two OBS nodes requires a separate optical fibre for each direction. We consider each fiber as a distinct link because the connection may be congested in one direction but not in the other.

4.3.2 PF-OBS Overview

The main objective of our congestion control technique is to maintain the blocking probability on every link in an OBS network under a maximum value $\hat{P}$. To achieve this goal, the load on all links is kept under a threshold $\hat{C}_l$, as in [19, 20]. $\hat{C}_l$ is smaller than the link capacity C_l. Therefore, we may regard $\hat{C}_l$ as a virtual capacity for link l. $\hat{C}_l$ is the load at which the burst blocking probability on link l reaches $\hat{P}$. For example, if full wavelength conversion and void filling scheduling are applied, we may use the Erlang B formula to calculate the load matching a desired loss probability. Suppose the number of wavelengths in each link is $N = 64$. The normalized load must then be kept under $\hat{C}_l = 0.69$ if we want to maintain the blocking probability under $\hat{P} = 10^{-3}$. Obviously, the traffic load corresponding to $\hat{P}$ varies according the contention resolution methods available. A better contention resolution technique means that a higher load is allowed before the congestion controller takes action. We assume the maximum blocking $\hat{P}$ is the same for all links and does not change in time. It is the load threshold that varies for each link to meet the blocking limit $\hat{P}$. Note that the maximum blocking probability on a virtual connection depends on the number of links it contains. For a route traversing h links, the overall loss is at most $h.\hat{P}$, assuming $\hat{P}$ is small and blocking is independent between switches.

There are several choices regarding the moment at which source rates should be updated. For example, it can be done every i bursts or every m marks. However, the previous methods present some problems if some users have a low traffic rate or if the network is uncongested and no marking occurs. Moreover, in equation (3.37), ω_r refers to the number of marks per time unit that a user is willing to pay and we assume ω_r is a constant. Therefore, feedback generation and source rate adjustment should take place periodically. Furthermore, this guarantees timeliness of control actions. We adopt this method in PF-OBS. The timeslot duration T is the same for all sources. Accordingly, we use the discrete-time formulas in equation (3.37) and (3.38) in our congestion control method. The transmission rates to be used in the next timeslot are updated using the

feedback received at the end of the current timeslot. However, there is no synchronisation between different routes because of network delays.

We have already noted that each user regulates its rate using only their preferred price ω_r and the feedback it receives regarding the aggregate route cost. This cost is measured in terms of the number of marks received during a timeslot. Special fields are added to burst headers in order to carry this information, in a manner similar to Explicit Congestion Notification (ECN) [117].

We note that we do not include retransmission of blocked bursts in our proposal. The main reason is the memory requirement for each route, which must be at least equal to the bandwidth-delay product of the path. This is expected to be unusually large in OBS, except for short distance connections. If the congestion control works as intended, the fraction of bursts lost because of contention is expected to be small. For example, if $\hat{P}=10^{-3}$, retransmission is needed for only one out of 1000 bursts. However, a retransmitting ingress node must buffer all bursts in transit until an acknowledgment is received. Therefore, we believe the performance gained with burst retransmission does not justify the resources it requires. We rely on higher layers such as TCP to retransmit the packets contained in lost bursts.

4.4 Marking Strategy

This section details the marking strategy used in our congestion control method. A proper burst marking policy is vital for the operation of PF-OBS as it affects the speed of convergence [102] and smoothness of rate allocations at equilibrium [23, 91]. The OBS core nodes signal congestion by marking burst headers at a bottlenecked link. Intermediate nodes monitor the load offered to each link and detect congestion if this load exceeds the critical value $\hat{C}_l$ for the link. Marking means turning on a congestion indication bit in the BHP, as is done in ECN [117] for example. No marking takes place in uncongested links. Transmission of rate x_r incurs to user r a price equal to the total

number of marks received in a timeslot. For this reason, we may use the number of marks as a measurement unit for link price.

We introduce the principles behind our marking policy first and then present the implementation details.

4.4.1 Requirements of a Marking Scheme

We identify here the constraints that must be satisfied by a proper marking policy. Some of them follow directly from the theory presented in the previous chapter while others are due to practical considerations. Let us repeat the discrete-time rate update rule described by equation (3.37), ignoring the delay:

$$x_r[n+1] = x_r[n] + \kappa_r\left(\omega_r - x_r[n]\sum_{j\in r}\mu_j[n]\right) \tag{4.1}$$

where:

$$\mu_j[n] = p_j\left(\sum_{s:j\in s} x_s[n]\right) \tag{4.2}$$

The simplest interpretation of the previous rule is the following: in each timeslot, an ingress node compares the number of marks ω_r it is ready to sustain with the number of marks received from the network. If the received marks exceed ω_r, the source reduces its rate. Otherwise, the source increases its rate. The term that corresponds to the marks received from the network is $x_r[n]\sum_{j\in r}\mu_j[n]$. From this, we identify the following requirements:

R1. From equation (4.2), the probability that a packet receives a mark at a link is a function of the total load of this link. Therefore, the switch does not need to monitor individual flows.

R2. A flow receives from the network a number of marks proportional to its sending rate x_r.

R3. The overall probability that a packet from a flow receives a mark is equal to the sum of marking probabilities on each link. This also means that marking must be independent from link to link.

Besides the previous theoretical requirements, a marking policy should also satisfy the following requirements:

R4. In order to keep the rate allocations smooth, the number of marks in consecutive feedback periods should not show large variations when the network is at equilibrium. It is not enough that the long-term average converges to the desired value. The number of marks from timeslot to timeslot must also remain close to the average. Otherwise, the rate allocations will show large random deviations around the equilibrium, as discussed in section 3.7.2 (stochastic stability). Ideally, the marks should be generated by a deterministic rule, instead of being the result of a random process. In addition, since it is impossible to get a fraction of a mark, the number of marks received during congestion should be large enough to allow fine-tuning.

R5. The marking scheme should be able to work using a single bit in burst headers for marking. However, it may also take advantage of multi-bit marking (e.g. with variable size bursts, as explained in section 4.4.4).

4.4.1.1 Issues with Blocking Probability in OBS

Since we assumed that optical buffers are not available at the OBS switches, we cannot use queue length information marking in our method. Using data loss as indicator for congestion appears to be a natural and attractive solution at first, and it is used in many congestion control algorithms (e.g. TCP [118], R-OBS [81], RED [119]). However, we will show that it is not suitable for use in OBS congestion control, as it does not satisfy several of the constraints listed above.

First, as burst loss is stochastic in nature, the condition we stated in point R4 above is not satisfied. With randomness being unavoidable, if the losses counted in consecutive timeslots show a large variance, they force the congestion control scheme to take

conflicting decisions within a short period. As we will show, this typically holds for the range of timeslot durations that is of interest for us. In addition, since our goal is to keep the blocking probability low, we expect only a few burst losses for each user in a timeslot. This does not allow the accumulation of an adequate number of events, which provides strong confidence before taking an action. To illustrate this, we simulate a star topology OBS network, as shown in Figure 4.1. The link between the switch and destination node supports 64 wavelengths. For this experiment, we use 8 source nodes. Maximum burst size is 250 kbits. We run two simulations using different burst arrival processes. In the first run, burst arrival at the switch is Poisson and therefore short range dependent (SRD). In the second run, the burstification buffers at each source node are fed by self-similar sources with a target Hurst parameter H = 0.9, and therefore long-range dependent (LRD) [34]. Each self-similar traffic source is modeled as a superposition of 100 Pareto ON-Pareto OFF smaller sources, as described in [120]. In simulations, the Hurst parameter of the aggregate traffic at the switch is estimated using the variance-time plot method [34]. We obtain H = 0.886 ± 0.014 with 95% confidence.

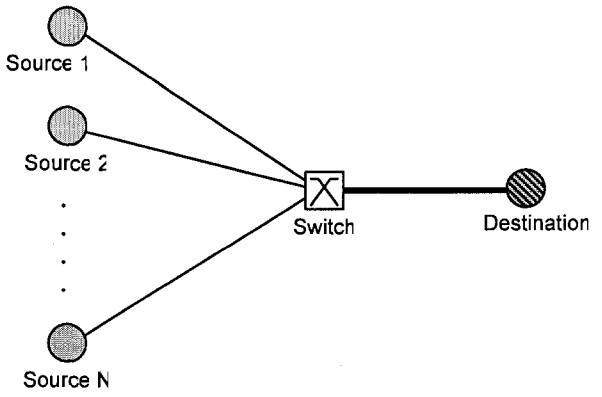

Figure 4.1: Simple star topology OBS network

We record the number of bursts blocked on the link between the switch and the destination node during each timeslot. This yields a time series $\boldsymbol{B} = (\beta_1, \beta_2, \ldots, \beta_N)$, where β_i is the number of blocked bursts in timeslot i. We compute the coefficient of variation (CV) of $\boldsymbol{B}$, defined as $CV_B = \frac{\sigma_B}{\mu_B}$, where σ_B is the standard deviation of $\boldsymbol{B}$ and μ_B is its mean. The coefficient of variation is an indicator of the statistical dispersion of a random

distribution around its mean. We run this experiment with four different values of timeslot duration T, ranging from 10^{-4} second to 1 second. For intervals smaller than 10^{-4} second, there are too few burst arrivals in each timeslot and the blocking probability estimate is inaccurate. Conversely, an observation period longer than a few seconds is considered too long and results in a sluggish response in case of congestion. For both source types, the load is such that the average blocking probabilities are the same and are equal to 0.66%.

The results are shown in Figure 4.2. Figure 4.2 (c) and (d) show the CV of the burst losses as function of the burst duration T. The number of burst arrivals at the switch in each timeslot is also recorded and its CV is plotted next to that of the burst loss for comparison. In Figure 4.2, the term "arrival" means the number of bursts received by the switch during a timeslot.

The dispersion of the number of blocked bursts is noticeable in Figure 4.2 (a) and (b), where the number of blocked bursts in consecutive timeslots is shown, for timeslot length $T = 0.1$. As expected, the CV decreases when the observation window duration increases. However, we note in Figure 4.2 (c) and (d) that the CV of burst loss is always higher than the CV of arrivals at the switch. Figure 4.2(e) and (f) show the CV of burst losses divided by the CV of burst arrivals. We observe of these figures that the CV of burst loss is roughly 30 times that of the burst arrivals with Poisson sources. With self-similar sources, the CV of burst loss is around 50 times the CV of the arrival process. Moreover the ratio stays roughly the same for all five timeslot durations. This suggests that, regardless of the length of the observation interval, the burst losses always exhibit high variability. Therefore, it is not recommended to use the burst losses as congestion indication.

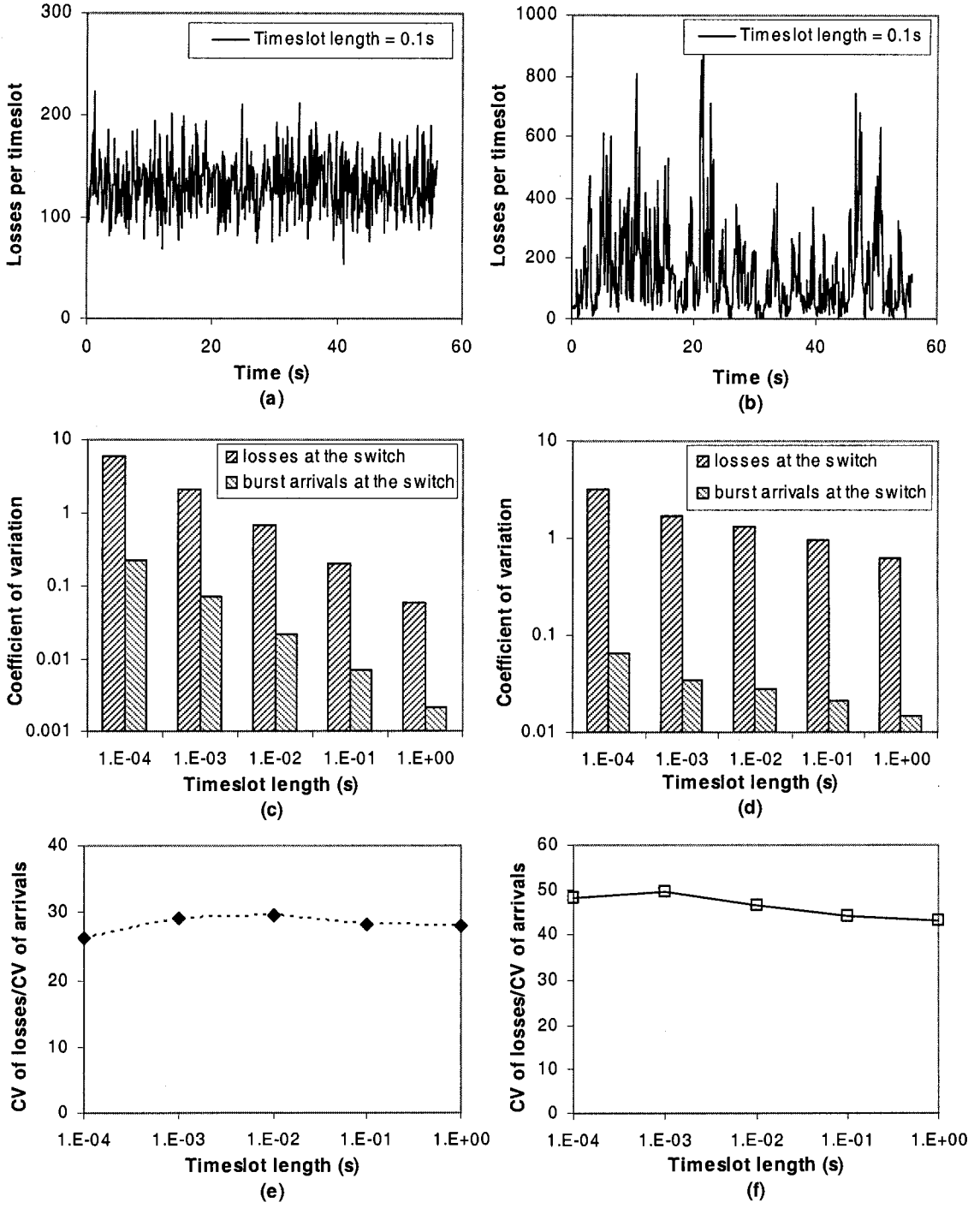

Figure 4.2: Number of blocked bursts counted in consecutive timeslots (a) Timeslot duration T = 0.1s, Poisson sources (b) Timeslot duration T = 0.1s, self-similar sources (c) CV of losses vs. timeslot duration for Poisson sources (d) CV of losses vs. timeslot duration for self-similar sources (e) Ratio of the CV of losses and arrivals for Poisson sources (f) Ratio of the CV of losses and arrivals for self-similar sources

The previous paragraph explained one of the reasons why burst loss is not a suitable congestion indicator for rate-based congestion control in OBS networks. The second problem lies in the fact that blocking probability in OBS favours the larger flows. This is incompatible with the requirements in points R1 and R2 (see page 71). Simply put, if two sources are using the same link, the source with higher sending rate will experience lower blocking probability. This issue has not received much attention from researchers. However, we found that it has a harmful effect on OBS flow control if burst loss is used as a congestion signal. Because the larger flow experiences a smaller blocking probability, it increases its rate. On the other hand, the smaller flow experiences a high blocking probability and reduces its rate further, as shown in Figure 4.4.

To the best of our knowledge, there is only one published work [121] that addresses this phenomenon and its effect on congestion control algorithms (TCP in the case of [121]). The authors in [121] call this effect "the bigger eats the smaller" or BES.

To demonstrate this problem, we simulate a simple OBS network similar to the one in Figure 4.1, but with only two source nodes. The total load on the link between the switch and destination is kept constant at 100% but the ratio of the traffic generated by the ingress nodes is varied from 1:9 to 5:5 (i.e. 10% and 90%, 20% and 80%, etc.). Packet arrival at the burst assembly buffer is Poisson and size based assembly is used. As previously, there are $N = 64$ wavelengths on the link. A plot of the blocking probabilities seen by both sources versus the rate of source 1 is shown in Figure 4.3. Let A_i be the normalized transmission rate of source i. We keep the total load on the switch at 100% ($A_1 + A_2 = 1$). Therefore, an increase in A_1 is obtained only by reducing A_2. We observe in Figure 4.3 that, when there is a large difference in the rates from the ingress nodes, the larger flow experiences lower burst loss.

We may explain this problem as follows. Suppose two ingress nodes are sending traffic through the same link. Ingress node 1 generates 1000 bursts/second while ingress node 2 generates 100 bursts/second. Assume that the traffic from both sources belong to the same QoS class. Therefore, when two bursts collide at the OBS switch, both have the same probability of getting through. Suppose the losses at the switch occur at a rate of 20

bursts/second. From the previous assumptions, this means each ingress node experiences 10 losses/second. For ingress node 1, this represents a blocking probability equal to $B_1 = 10/1000 = 0.01$ while ingress node 2 experiences $B_2 = 10/100 = 0.1$. Therefore, the blocking probability seen by ingress 1, which generates a higher traffic volume, is lower than the losses of ingress 2. Although the previous discussion does not constitute a formal proof, we believe it provides a valid explanation of the results we obtained in simulations.

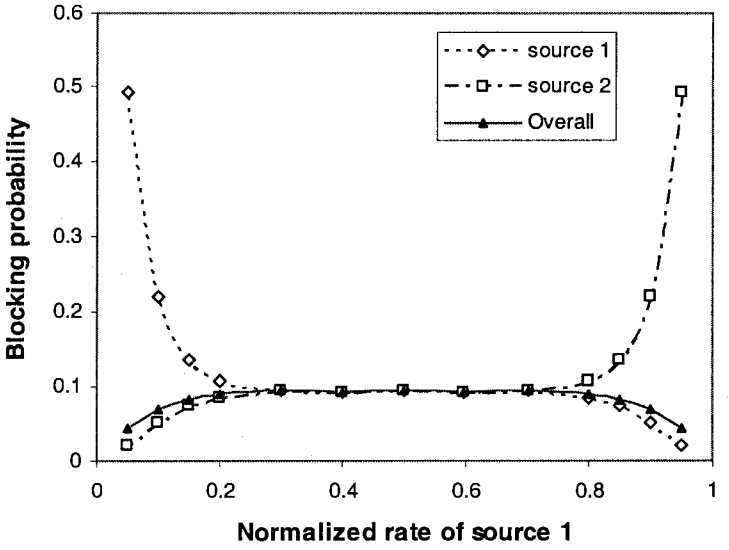

Figure 4.3: Blocking unfairness in OBS

Figure 4.4 shows the problem that arises when burst loss is used as congestion indication. To obtain the results in Figure 4.4, we simulate again an OBS network with two ingress nodes sharing a single link. The target burst loss probability is equal to 10^{-3} for both source nodes. The first ingress node (source 1) starts transmitting at 1.7 Gbps while the second ingress node (source 2) starts transmitting at 0.64 Gbps. During approximately 4 seconds, the total load on the link is low enough so there is no burst loss. Therefore, both nodes initially increase their transmission rates. After this period, both sources experience burst losses and adjust their transmission rates accordingly. However, source 2 experiences higher burst loss because it has a lower traffic volume compared with source 1. Thus, source 2 reduces its transmission rate more aggressively than source 1. On the other hand, source 1 keeps increasing its traffic volume. This cycle continues until

source 2 reaches the minimum transmission rate, which we set at 1 Mbps. Once it reaches this point, source 2 cannot increase its bandwidth share again.

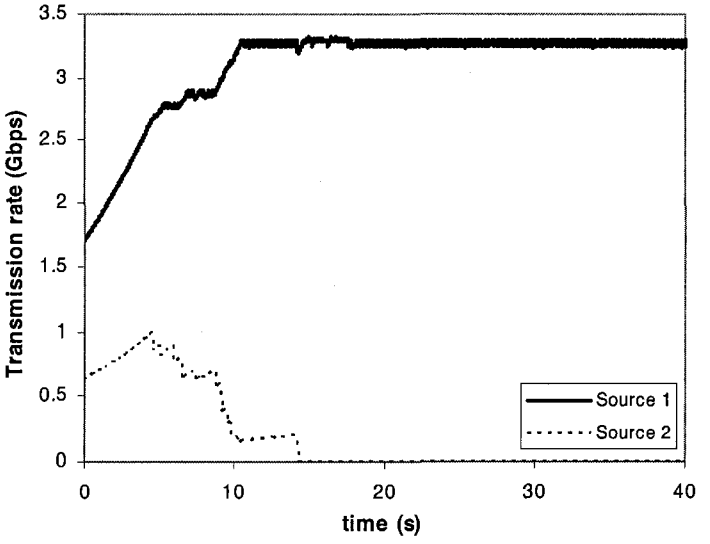

Figure 4.4: Rate allocation unfairness

4.4.2 Marking Function

We now describe the marking policy we apply at the OBS switches. In our method, we implement a marking rule, which is based on equation (3.34). Define:

$$p_l(y) = \frac{(y-\hat{C}_l)^+}{y} = \begin{cases} 0, & y \le \hat{C}_l \\ \dfrac{(y-\hat{C}_l)}{y}, & y > \hat{C}_l \end{cases} \tag{4.3}$$

where $p_l(.)$ is the fraction of bursts passing through link l that receive a mark when the load is equal to y. $\hat{C}_l$ is the desired load threshold or virtual capacity introduced in section 4.3.2. As pointed out in section 3.5.2, we may interpret this marking rule as the fraction of traffic lost when the offered load exceeds the capacity $\hat{C}_l$. However, in our case, $\hat{C}_l$ is only a virtual capacity that is used to signal congestion.

We chose this method for the following reasons:

a. It does not require a queue and is simple to implement. By marking the traffic that is over the threshold $\hat{C}_l$, the term $x_r[n]\sum_{j\in r}\mu_j[n]$ in equation (4.1) becomes equal to the number of marks received by the ingress node r, without the need for any further calculations. We explain this further in the next section.

b. Using (4.3), the term $p_j + p'_j \sum_{s:j\in s} x_s$ in the delay stability criterion of equation (3.43) becomes equal to one. Therefore, the term $\sum_{j\in r}\left(p_j + p'_j \sum_{s:j\in s} x_s \right)$ in (3.43) is equal to b, where b is the number of congested links on the route. This greatly simplifies the stability condition and allows a practical implementation.

4.4.3 Marking Policy Implementation at OBS Switches

In this section, we present the implementation of the previous marking policy at the OBS switches. This includes the signalling required between switches and edge nodes. A matching edge node policy must complement the implementation at the OBS switches. We discuss implementation of the marking scheme at edge nodes in the next section.

In order to mark bursts during congestion, an OBS switch monitors the traffic load on all of its ports. To estimate the traffic rate on one port, a switch calculates the average offered load within a timeslot. A timeslot is a period of length τ, during which the switch measures the load on all ports. Consecutive timeslots do not overlap and a new timeslot begins as soon as the previous one ends. The same value of threshold $\hat{C}_l$ (in bits/s) is used for all links. An accumulator counts the number of bits s_l offered to a link l within a timeslot. If s_l exceeds $\tau \cdot \hat{C}_l$ (i.e. the estimated load exceeds $\hat{C}_l$), any subsequent bursts will be marked and we declare that the switch is in "mark mode" until the period τ is over. Marking is done by turning on a Congestion Indication (CI) bit located in the header of incoming bursts. The accumulator is reset at the beginning of a new monitoring period and the switch stops marking.

τ should be long enough to allow accurate measurement of traffic load but short enough to maintain responsiveness of the control. Please note that τ is only for monitoring the load at the OBS switches and is independent from the timeslot duration T that is used to update the rates at the edge nodes. We also note that bursts are marked while on their way to the egress node. The egress node is responsible for transmitting the number of marks to the ingress nodes, as we will explain in more detail in section 4.4.4.

We introduce a new field in the OBS header format, called the Payload Identifier, which specifies the purpose of the header. The payload identifier consists of several single bit flags. The CI bit mentioned previously is one of these flags. Another flag is the RESV bit, which indicates that the header contains information for resource reservation at the OBS switch. Table 4.1 summarizes all signalling flags used by PF-OBS. Until now, we have explained the RESV and CI flags. We will explain the other flags later, when we will be discussing their purpose. Please keep in mind that a single control header may have more than one signalling flag turned on.

Flag name	Purpose	Generated at	Action triggered at core nodes	Action Triggered at edge nodes
RESV	Bandwidth reservation at Core nodes. Carries burst timing information	Ingress	Burst scheduling	Disassemble burst.
CI	Signal link congestion	Core	Pass to the next hop	Egress nodes increment route mark counter.
PING	Request a reply to measure route RTT	Ingress	Pass to the next hop	Egress nodes reply with PONG
PONG	Reply to PING	Egress	Pass to the next hop	Compute route RTT
PRICE_ADVERT	Compensation for load threshold increase	Ingress	- Adjust maximum load threshold. - Pass to the next hop	Ignored by egress nodes.
ROUTE_PRICE	Congestion feedback to ingress nodes	Egress	- Increment a counter in the header if the source link is congested - Pass to the next hop	Ingress nodes update route transmission rate.

Table 4.1: Summary of signalling flags

If $x_r[n]$ is the bit rate of flow r and burst size B is constant, as in size threshold assembly, the number of marks received by any user is proportional to its rate, as we show next. Using the marking method presented previously, the total number of marks $m_l[n]$ generated at link l is given by the following equation:

$$m_l[n] = \frac{\tau.\left(\sum_{r,l\in r} x_r[n] - \hat{C}_l\right)^+}{B} \tag{4.4}$$

We can rewrite $m_l[n]$ as: $m_l[n] = a.\left(\sum_{r,l\in r} x_r[n] - \hat{C}_l\right)^+$, where $a = \tau / B$ is of constant value. Using (4.3) and the fact that $y_l[n] = \sum_{r,l\in r} x_r[n]$, we get:

$$m_l[n] = a.p_l(y_l[n]).\sum_{r,l\in r} x_r[n] = a.\sum_{r,l\in r}(p_l(y_l[n]).x_r[n]) \tag{4.5}$$

Equation (4.5) gives the total number of marks at link l as the sum of the marks received by each ingress node that is using l. The marks received by ingress node r from link l is $a.x_r[n].p_l(y_l[n])$, which is proportional to the ingress node's transmission rate $x_r[n]$. We note that the term $a.x_r[n].p_l(y_l[n])$ appears in the rate adjustment rule of equations (4.1) and (4.2). Therefore, in our implementation, the term $x_r[n]\sum_{j\in r}\mu_j[n]$ in (4.1) does not require that the ingress node perform any calculations. If the burst size is constant, the value of the term is simply equal to the number of marks received during a timeslot.

4.4.3.1 Marking Independence

We already pointed out that marking must be independent from switch to switch. Since only one CI bit is available, a burst may not be marked more than once. If a switch is in the mark mode but receives a previously marked burst, the mark must not be simply discarded. Otherwise, the price calculated by the sources from the feedback will be lower than the actual value. To prevent this, when a switch is in the mark mode but receives a previously marked burst, the mark is assigned to the next unmarked header from any

source. In this manner, the total price of a route is equal to the sum of the prices of its component links.

Because marks are carried in OBS reservation packets and blocked burst headers are normally discarded, burst loss may affect the marking process. This occurs when a switch must drop a burst while it is in marking mode or the burst was previously marked at another link. To prevent such loss of marks, if a burst has the CI bit set, its header is not discarded even if the burst in the data channel is dropped. Instead, the OBS switch turns off the RESV flag in the header before sending it to the next switch. In this manner, the mark is not lost and the header does not trigger bandwidth reservation in the subsequent switches on its route.

4.4.3.2 Load Threshold Compensation

According to equation (4.3), if a link has nonzero price, the total load it carries must be slightly higher than the threshold $\hat{C}_l$. To demonstrate this, consider the OBS network configuration depicted in Figure 4.1. There is only one bottleneck link, which carries a load equal to $\sum_{r=1}^{N} x_r^*$, where x_r^* is the steady-state rate of user r. Assume that the users have enough traffic to fill the link capacity. The steady-state rate is calculated through equation (3.32) as follows:

$$x_r^* = \frac{\omega_r}{\mu}$$

where μ is the price of the congested link.

The price μ is the same as $p_l(y)$, shown in equation (4.3). ω_r is the price that an ingress node is willing to pay on route r.

At steady state, the link price $\mu = p_l(y)$ is constant. Then, using (3.32), the total traffic on the congested link is given by equation (4.6).

$$\sum_{r=1}^{N} x_r^* = \frac{\sum_{r=1}^{N} \omega_r}{p_l(y)} \tag{4.6}$$

Expressing $p_l(y)$ as in equation (4.3) and using the fact that $y = \sum_{r=1}^{N} x_r^*$, we easily get:

$$\sum_{r=1}^{N} x_r^* = \hat{C} + \sum_{r=1}^{N} \omega_r \tag{4.7}$$

where $\hat{C}$ is the desired maximum load on OBS links.

According to equation (4.7), the real load of a congested link at steady state is not $\hat{C}$ but $\hat{C}$ plus an excess term. The excess load contributed by a route r is equal to ω_r bursts per timeslot T when equilibrium is reached. If a link carries several flows, the total surplus is equal to $\sum_{r=1}^{N} \omega_r$ bursts in each timeslot T. If this value is not negligible compared to $\hat{C}_l$, the real link load at equilibrium might be considerably higher than our target.

To compensate for this, OBS switches must reduce the load threshold by the equivalent of $\sum_{r=1}^{N} \omega_r$ bursts per timeslot. Therefore, intermediate nodes need to know about the price ω_r of each route using their ports. This requires the ingress nodes to advertise their price preference ω_r for route r when it is established, using a special control packet. For this purpose, the ingress node turns on the "PRICE_ADVERT" flag in the control packet and places ω_r in another field of the same header. Only the intermediate OBS nodes need this information when a new route starts transmitting.

The previous compensation method works perfectly for routes containing only one intermediate node. If a route contains more than one intermediate node, all intermediate nodes should also reduce the link load threshold $\hat{C}_l$ by ω_r. This guarantees that, even in the worst case, the load on all links never exceeds the preset limit $\hat{C}_l$.

4.4.4 Marking Policy Implementation at Edge Nodes

An egress node keeps track of the number of marked headers it receives during a timeslot for each route it terminates. At the end of a period T, the egress node sends this information back to each source and resets the counters to zero. A header with the "ROUTE_PRICE" flag and containing the mark count is used for this purpose. When an OBS switch receives a header with the "ROUTE_PRICE" flag, it simply passes the control packet to the next hop. We use end-to-end feedback because it is significantly less computationally complex compared with direct feedback from switch to sources. Furthermore, an ingress node must know about the congestion level of all links on a route before adjusting its sending rate. End-to-end feedback provides this information in one pass. Since the whole network operation is also organized in timeslots, there is no advantage in using immediate feedback.

We now discuss the case of variable burst size. We showed in section 4.4.3 that a mark per excess burst is enough to convey link prices according to the requirements of the rate control algorithm. However, this works only when the burst size is constant. We remind the reader that time-based burst assembly results in bursts of variable size. In time-based burst assembly, the number of bursts per time unit in each flow is constant. Therefore, we need a different technique to make the marking proportional to the *bit rate* of each flow. To do this, the number of marks received by a burst must be proportional to its size. For example, we might have to use more than one CI bit in the BHP to count the marks.

We chose the following method to deal with variable size bursts: when the destination receives a marked burst, it increments the mark counter by the burst size (in bits). This way, proportionality between feedback and bit rate is preserved. The feedback received by ingress nodes now represents bits per time unit. To compensate for this change, the control coefficient ω_r must be scaled and its unit is bits/second. Therefore, instead of the number of marked bursts in a period, we may instead provide the number of *marked bits* in the same period. The number of marked bits is equal to the sum of the sizes of marked bursts. Since constant burst size is a special case of variable burst size, this marking method works for any burst size distribution.

4.5 Source Rate Control

4.5.1 Regulation of Transmission Rate

Each ingress node adjusts the rate of a route only when it receives a "ROUTE_PRICE" packet from the destination node. Since an egress node sends feedback every T time units, the sending rate of each route is also updated every T time units. The update rule is given in equation (4.1). Assuming variable burst size, we may regulate the rate of a flow by controlling the number of bits sent during a timeslot. Therefore, we may regard $x_r[n]$ in equation (4.1) as the number of bits sent on route r in timeslot n.

We control the sending rates at ingress nodes by adjusting the burst interdeparture time. This method is also used in previous OBS congestion control methods [20]. Suppose $x_r[n]$ is the number of bits that can be sent on route r in a period of length T, as calculated by (4.1). After sending a burst of size B_i bits, the ingress node waits $B_i \cdot T / x_r[n]$ time units before sending another burst on the same route.

4.5.2 Burst Randomization

Although the scheme introduced in the previous section effectively controls the sending rate at ingress nodes, it may cause burst synchronization at OBS switches [122]. Burst synchronization occurs when the interarrival times of bursts generated by several ingress nodes are roughly the same. This happens, for example, when two sources sharing a link have the same sending rate and burst size. Bursts from these sources may always overlap at the switch. This results in a long string of burst loss even if the total load on the switch is not high.

To solve this problem, we add a small random value Δ in the burst interdeparture time as is suggested by the authors of [122]. Δ has zero mean, so the average burst interdeparture time remains unchanged. Using a trial and error approach, we found that a uniform distribution $\Delta \sim U(-t_b/4, t_b/4)$ is sufficient to prevent synchronization, where t_b is the burst duration. To show the advantage of randomization, we simulate a simple star topology OBS network, similar to the one shown in Figure 4.1 but having only two ingress nodes, sending traffic to one destination node. Each OBS link supports one wavelength. The

burst size is constant and identical for both ingress nodes. The total load on the shared link is equal to 50% and shared equally by both sources. We record the loss cumulated at the OBS switch with and without randomization at the rate controller. We run ten independent simulation replications in order to compute confidence intervals. A sample of the result obtained is shown in Figure 4.5.

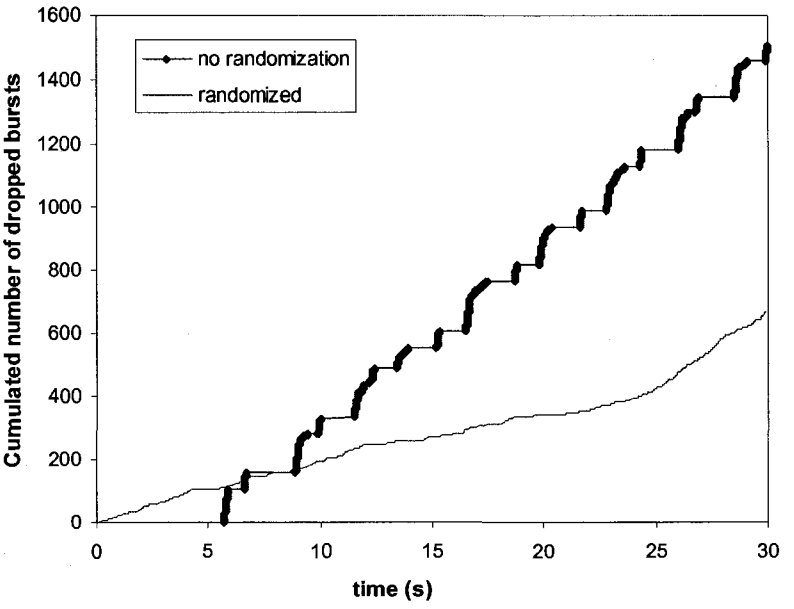

Figure 4.5: Effect of burst interdeparture randomization on burst blocking

We note that, when there is no randomization, there are intervals lasting a few seconds during which there are no losses at the switch. However, during other periods, the loss increases suddenly because of synchronized burst arrival at the switch. On the other hand, the number of losses with randomization increases smoothly. At the end of the simulation, the blocking probability is equal to $0.245 \pm 7.5\times10^{-4}$ with 99% degree of confidence when randomization is done. On the other hand, the blocking probability is always equal to 0.497 in all 10 replications where randomization was not used.

Figure 4.6 shows the architecture of an ingress node running our congestion control method. The new components are pointed out using a shaded background.

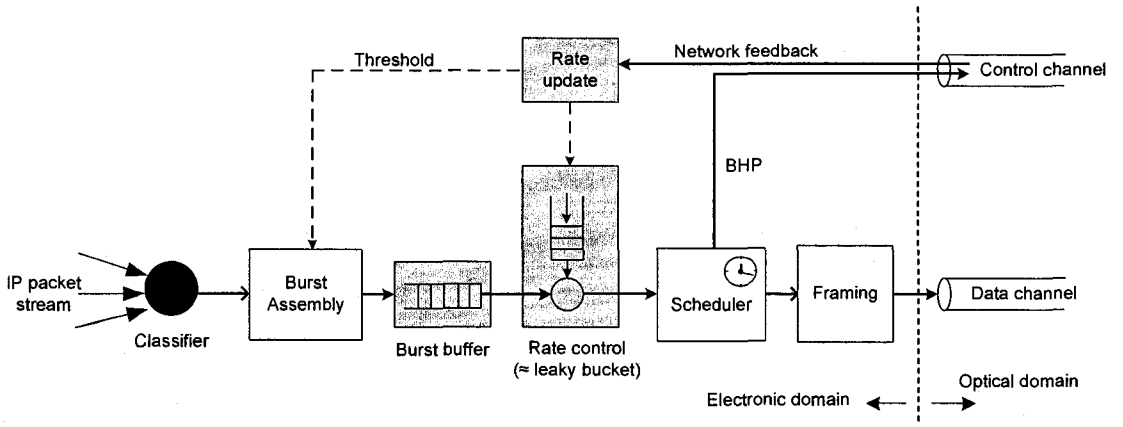

Figure 4.6: PF-OBS ingress node architecture

4.6 Stability under Delay

We stated earlier that we consider OBS networks that may span a large geographical area. Therefore, the propagation delay may destabilize the control loop. To guarantee stability, we must make sure the control gain κ_r in equation (4.1) has a proper value. We use the stability criterion given in equation (3.43) for this purpose.

In order to implement (3.43), ingress nodes need to know the RTT on each route. We assumed previously that routes are fixed. This is mainly to ensure the RTT does not change. In our method, ingress nodes measure this round-trip time by sending a probe along each route when it is established. The probe is a special control packet sent through the OBS control channel. This control packet carries a "PING" flag in conjunction with its creation timestamp. Ingress nodes also advertise the route weight ω_r at the same time, as discussed in section 4.4.3.2. A "PING" message does not reserve bandwidth because intermediate OBS nodes just pass it to the next hop. Whenever an OBS edge node receives a header with the "PING" code, it changes the code to "PONG" and sends the header back to the source. An ingress node receives the reply to a probe D_r timeslots later. The source node may repeat this measurement several times and average the results to get a more reliable estimate of the RTT. Also, D_r may be refreshed periodically or updated in response to routing changes (e.g. because of link failures). Because of the fixed route assumption, we can use the measured D_r until there is a routing change.

As shown in section 4.4.2, ingress nodes also need to know the number of bottlenecked links on each route. To convey this information, OBS switches increment a counter in the "ROUTE_PRICE" messages received from a congested link. We keep in mind that "ROUTE_PRICE" messages travel from the destination to the ingress nodes. In order to guarantee stability under delays, the ingress nodes choose the control gain κ_r as follows:

$$\kappa_r < \frac{2}{b}\sin\left(\frac{\pi}{2(2D_r+1)}\right) \tag{4.8}$$

where b is the number of congested links on route r.

Chapter 5

Performance Evaluation

5.1 Introduction

The previous chapter discussed in detail the architecture and operation of our proposed OBS congestion control method. In this chapter, we show that PF-OBS can effectively manage the congestion in OBS networks and assess its performance in various scenarios using simulation. We use the OPNET [123] software package to model and simulate an OBS network running our congestion control method. OPNET is a network simulation software based on a discrete-event system. It allows detailed modeling and simulation of a network, down to the process level.

First, we discuss the methodology and modeling assumptions. In section 5.3, we validate the marking strategy introduced in the previous chapter. We begin the performance evaluation with a simple OBS network containing a single switch and a single bottleneck link. We then consider a more realistic configuration, involving a meshed network with heterogeneous propagation delays.

5.2 Background

5.2.1 Methodology

To assess the performance of PF-OBS, we built detailed models of OBS nodes and links using OPNET. These include the basic building blocks such as burst assembly,

burst scheduling, cut-through switching and signalling. The congestion control modules are added to the previous building blocks to give a complete model of a PF-BOS network. As we stated in Chapter 4, we are only concerned with congestion occurring in the OBS network. Therefore, our simulations only cover events occurring between packet arrival at the assembly buffers and burst disassembly at egress nodes. We do not consider events outside this setting.

We generate the OBS network load using different traffic arrival patterns at the ingress nodes as follows [18]:

1. Persistent sources: also called "greedy" or "infinite" sources because they always have traffic to send. Therefore, all flows in the network cross at least one bottleneck. Using these sources, the network can be kept under heavy load, allowing us to study the fairness and convergence of the rate control algorithm.

2. Staggered source: the sources start at different times, and in our case, they stop transmitting at different times. This reveals the performance of PF-OBS during transient periods. This includes convergence speed and any oscillations during transient periods. During their "ON" period, staggered sources behave like persistent sources.

5.2.2 Assumptions and Simulation Parameters

For our simulations, we make the following assumptions and choice of parameters:

A1. All information regarding the operation of an OBS network is available through the control channel. For this reason, we focus on the OBS control plane and do not simulate the data plane.

A2. All nodes use the same interval T to update their transmission rates. We use $T = 0.01s$ throughout this chapter. We consider this value appropriate because it permits frequent updates while minimizing the overhead introduced by the feedback and update process.

A3. All links in the network carry the same number of wavelengths and each wavelength has the same bit rate. In our simulations, we will use $N = 65$ wavelengths per link: one for control and 64 for data. The capacity of each wavelength is 1 Gbps.

A4. Operation of an OBS network without contention resolution is not considered practical, as the losses are too high (see Figure 2.12). In our simulations, the only contention resolution method applied at OBS switches is wavelength conversion. We assume that full wavelength conversion is available. We chose wavelength conversion for our simulations because it is commonly used in similar works [16, 20, 21, 124] and maintains a constant RTT on every route. We do not consider use of delay lines, burst segmentation, or burst deflection. Therefore, RTT is constant on every route.

A5. The same load threshold $\hat{C}_l$ is used for all links.

A6. We implement the LAUC-VF channel scheduling algorithm reviewed in section 2.6.2.1.

A7. We use classless JET bandwidth reservation on OBS. JET signalling is previously reviewed in section 2.4.4.

A8. The overall time allocated for header processing and optical switch configuration at an OBS core node is set at 10 μs. This value is chosen in light of the current state of technology. For example, a FPGA-based burst reservation module using JET reservation, void-filling and 64 wavelengths is able to process a burst in 60 ns [125]. Once the reservation module finds a suitable wavelength for an incoming burst, the optical switch must be configured. A switching time of 200 ps to 2 ns is reported with Semiconductor Optical Amplifiers (SOA) in [126]. With additional processing required for feedback generation at OBS core nodes, we assume a total processing delay of 10 μs is sufficiently realistic and is used in our simulations.

A9. The burst time offset is fixed for the whole network, which means that no offset-based service differentiation is modeled. We use t_{offset} = 100 μs. This takes into account the processing time discussed above. With this value, an OBS path may contain up to 10 hops.

A10. For each simulation scenario, we run 10 replications with different seeds and compute confidence interval for all each point presented. By default, all intervals provided in the text correspond to a 99% confidence level.

A11. All ingress nodes use the size-based burst assembly described in section 2.3.3. The size distribution of IP packets arriving at the OBS assembly buffers is taken from [127], which is based on measurement of packet sizes on the Internet as follows:

- 55% of packets are 40 bytes in length.
- 15% of packets are 576 bytes in length.
- 12% of packets are 1500 bytes in length.
- 18% of packets are uniformly distributed between 40 and 1500 bytes in length.

This distribution is 98.5% accurate [127]. We set the burst size threshold at 250 kbits. Assuming no assembly timeout, the resulting burst size distribution is shown in Figure 5.1. We observe in Figure 5.1 that the sizes of all bursts are within the interval 238 kbits to 250 kbits and most burst sizes are very close to the threshold. The corresponding mean burst size is equal to 245,625 bits. Since the burst size is variable, we use the appropriate marking method as discussed in section 4.4.4. We remind the reader that in this case, an accumulator at the egress node is incremented with the size of every marked burst. Therefore, the feedback packet sent to the source node contains the total size of marked bursts in bits. For this reason, the unit we will use to count the number of marks is a bit. In addition, the number of marks received during an update period is counted in bits/s.

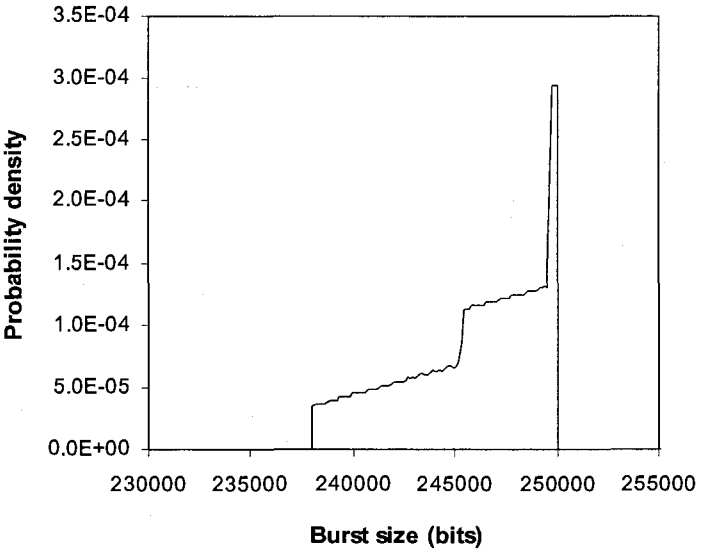

Figure 5.1: Burst size distribution for size-based assembly with a 250 kbit threshold and accurate IP packet size distribution

5.3 Evaluation of Feedback Generation

A simplified view of the operation of PF-OBS identifies two major components: feedback generation and source rate adjustment. As discussed in the previous chapter, proper feedback generation is an essential component of our congestion control method. Before we begin with the actual performance evaluation for a complete PF-OBS network, this section focuses on the marking scheme alone.

The goal of this section is to show the following:

a. As implemented, the feedback generation satisfies all the requirements for a proper marking scheme, discussed in section 4.4.1: The number of marks received by an OBS traffic flow must be proportional to its sending rate and link prices must add up over a route.

b. The marking scheme implements equation (4.3), i.e. $p_l(y) = \frac{(y - \hat{C}_l)^+}{y}$, where y is the total link load, $\hat{C}_l$ is the chosen maximum load for link l and $x^+ = \max(0, x)$.

5.3.1 Methodology

To study the feedback generation method, we simulate a single bottleneck link in the simple star OBS network depicted in Figure 5.2. This network has 10 ingress nodes sending traffic to one egress node. Since all links have the same capacity, the link between the switch and the egress node is the bottleneck.

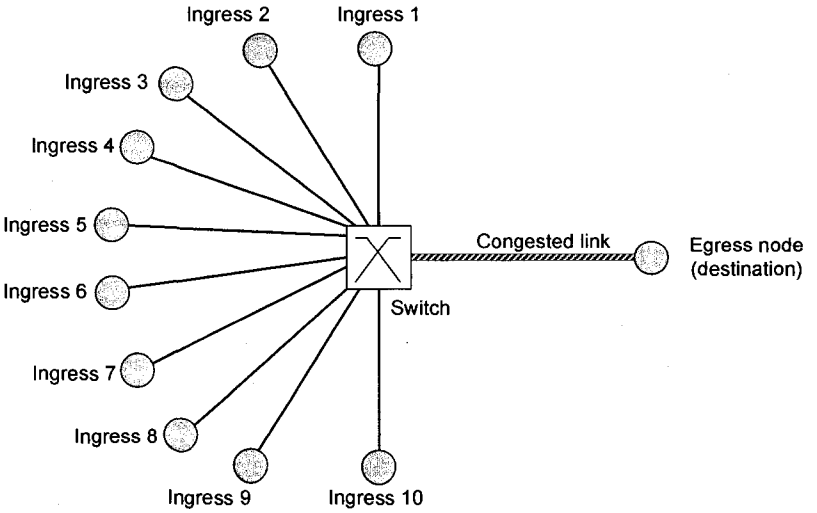

Figure 5.2: Single bottleneck star network

The total load y on the bottleneck link is manually controlled through the arrival rates at the ingress nodes. This means that no congestion control is taking place at the ingress nodes as we only wish to investigate the marking strategy in this section. The switch marks any traffic above a predefined threshold in each timeslot, according to equation (4.3). For this experiment, we set the load threshold $\hat{C}_l$ at 70% of the link capacity, i.e. $\hat{C}_l = 44.8$ Gbps. Each ingress node contributes a different amount of the total switch load y. If the rate of Ingress 1 is x_1, then Ingress 2's rate is $x_2 = 2 \cdot x_1$, Ingress 3's rate is $x_3 = 3 \cdot x_1$ etc. This relation between the rates of individual sources is maintained while the total load on the bottleneck link is varied. This means that the total switch load is always equal to $y = \sum x_r = 55 \cdot x_1$. The reason we chose this combination of individual source rates is the following. According to the theory presented in Chapter 3, the number of marks received by a source is function of its traffic rate. In the current experiment, we

assign a different sending rate to each ingress node to observe the effect on the feedback received. The combination of source rates introduced previously is a simple way to achieve this and covers a satisfactory range. We will clarify this further in section 5.3.3. All rates x_r are kept constant during each simulation run. The burst randomization method used was described in section 4.5.2. We record the fraction of total traffic marked as seen at the switch and as seen by each source.

5.3.2 Evaluation of Marking Scheme Implementation at Switches

In this section, we show that the implementation of the marking scheme in the core nodes follows equation (4.3). The price p_l of the bottleneck link as function of the load is shown in Figure 5.3. Since a PF-OBS switch does not identify individual traffic flows, the price shown in Figure 5.3 is function of the total load $y = \sum x_r$ carried by the bottleneck link.

For each data point displayed in Figure 5.3, we computed 99% confidence intervals (C.I.). However, the widths of all C.I. obtained are smaller than 0.1% of the average value. Therefore, we do not show them on the figure to improve readability.

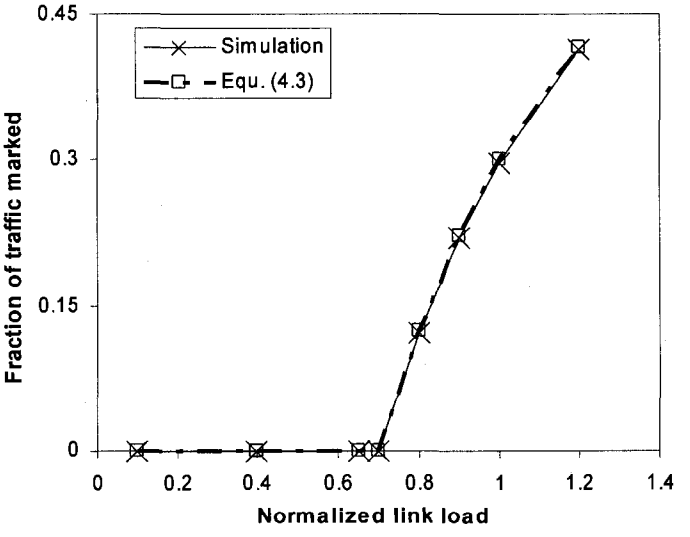

Figure 5.3: Price $p_l(\sum x_r)$ of the bottleneck link vs. the total load

Figure 5.3 shows the proportion of bursts through the bottleneck link that receive a mark. This is interpreted as the price per unit flow $p_l(\sum x_r)$ of this link, as computed by the switch. We observe that p_l is zero when the load is under the chosen threshold (70%). When the load exceeds this value, the link incurs a price. As the link becomes more congested, the link price increases gradually. We note that, in the simulations, the variation of p_l as function of the load follows exactly the curve obtained using equation (4.3).

5.3.3 Feedback Delivery

The previous section focused on the overall number of marks generated at the congested link. Since the switch does not identify individual flows, we are interested in how the marks generated by the switch are distributed among all users of the congested link. If marks are fairly distributed among all flows, we expect each source to get the same link price from the feedback it receives. This means that the proportion of bursts marked should be the same for all flows. In Figure 5.4, we compare the prices seen by each source as well as the price seen by the switch. We observe that the price curves of all flows and at the switch are identical and are superimposed in Figure 5.4. This is achieved regardless of the fact that the switch does not identify the source of each burst.

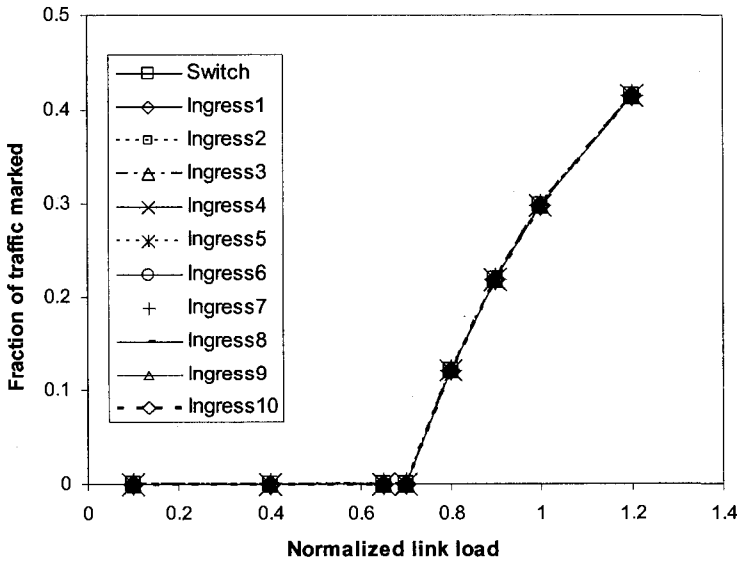

Figure 5.4: Link price seen by each source, compared to price seen by the switch

Figure 5.3 and Figure 5.4 show the price *per unit flow* of a congested link. Since all flows are charged the same price according to Figure 5.4, the total number of marks received by a particular flow must be proportional to its bit rate. As indicated in section 5.3.1, the transmission rates for the ingress nodes in Figure 5.2 are arranged according to an arithmetic progression. Figure 5.5 shows the average number of marks received by each OBS ingress node in reference to its transmission rate. While Figure 5.3 and Figure 5.4 show the fraction of traffic marked by the OBS switch, Figure 5.5 shows the average number of marks received during a measurement period at the switch. Therefore, the unit for the y-axis is in bits per second. The curve shown in Figure 5.5 corresponds to a total load of 100% at the switch. The results obtained by the simulations show that the number of marks received by each ingress node is proportional to its sending rate.

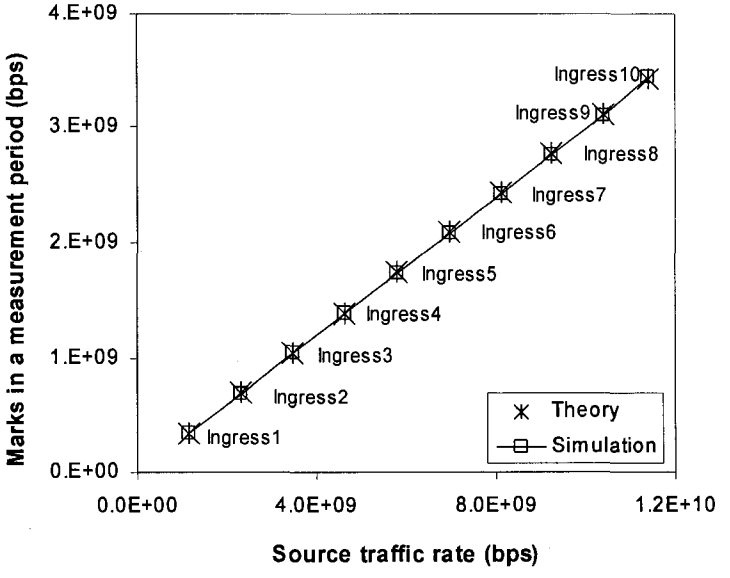

Figure 5.5: Total number of marks received by each source vs. transmission rate

The widths of the 99% confidence intervals for the results in Figure 5.4 and Figure 5.5 are all under 0.2% of the mean values. Again, we do not show the intervals on the previous figures to maintain clarity.

5.3.4 Multiple Congested Links

The final part of our evaluation of the feedback generation method addresses the case of multiple congested links in a route. The goal of this section is to establish that the total price for a route is equal to the sum of individual link prices along the path. For this purpose, we simulate the OBS network depicted in Figure 5.6. From now on, we will be referring to it as the "linear OBS network" because of its layout. We focus our attention on the set $L^* = \{L_1, L_2, L_3\}$ of links connecting two OBS switches because congestion takes place only in these links. This network carries four traffic flows or routes. Route 0 crosses all three congested links between Ingress 0 and Egress 0. Each of the remaining three routes (Route i, $1 \le i \le 3$) crosses only one congested link (L_i) between Ingress i and Egress i. Therefore, each congested link carries two traffic flows. The traffic rates are the same for all routes and are varied together. The load threshold for all links is set at $\hat{C}_l = 70\%$. We are interested in the feedback received by Ingress 0 compared to the feedback received by the other three ingress nodes.

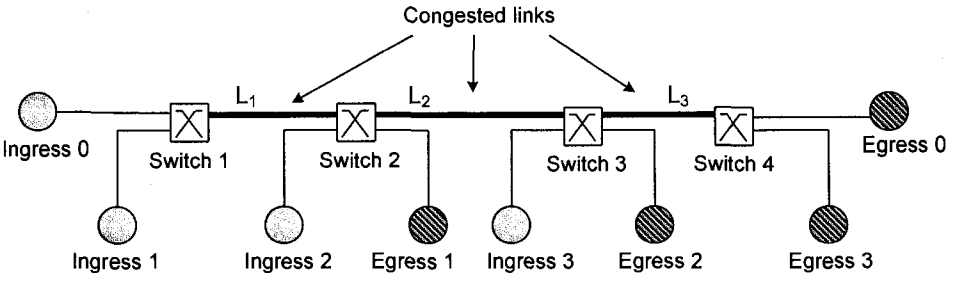

Figure 5.6: Linear topology OBS network

Since Route 0 crosses three congested links, the price of Route 0 should be equal to the sum of the prices of Route 1, 2, and 3. Figure 5.7(a) shows the price of Route 0 in relation to the sending rate of Ingress 0. The sum of the prices of Route 1, 2, and 3 is also shown on the same figure for comparison purposes. We note in Figure 5.7(a) that the two curves are very closely matched, except when the transmission rates exceed 45%. This corresponds to a load greater than 90% on the congested links because each carries traffic from two sources. The discrepancy between the curves at high load is caused by blocking at the OBS switches. At each switch on its route, a fraction of the traffic from Ingress 0 is

lost because of contention. Therefore, the amount of traffic from Ingress 0 that can be marked is progressively reduced. This explains why the price seen by Route 0 becomes smaller than the sum of the prices of each link and why this effect is only noticeable when the load is high. However, operation of the network at 90% load and above is an extreme scenario that should be avoided, and by no means should be considered a norm in operating a network. We intend to maintain the load on each link around the threshold $\hat{C}_l$. It is evident that the two curves are closely matched all up to the threshold.

Individual prices for Route 1, 2, and 3 are plotted in Figure 5.7(b). We note that, at high load, link prices decrease from L_1 to L_3 because the traffic is reduced by blocking. For transmission rates under 45%, all tree curves are closely matched. For all data points in Figure 5.7, the width of the 99% confidence interval is smaller than 0.2% of the average value.

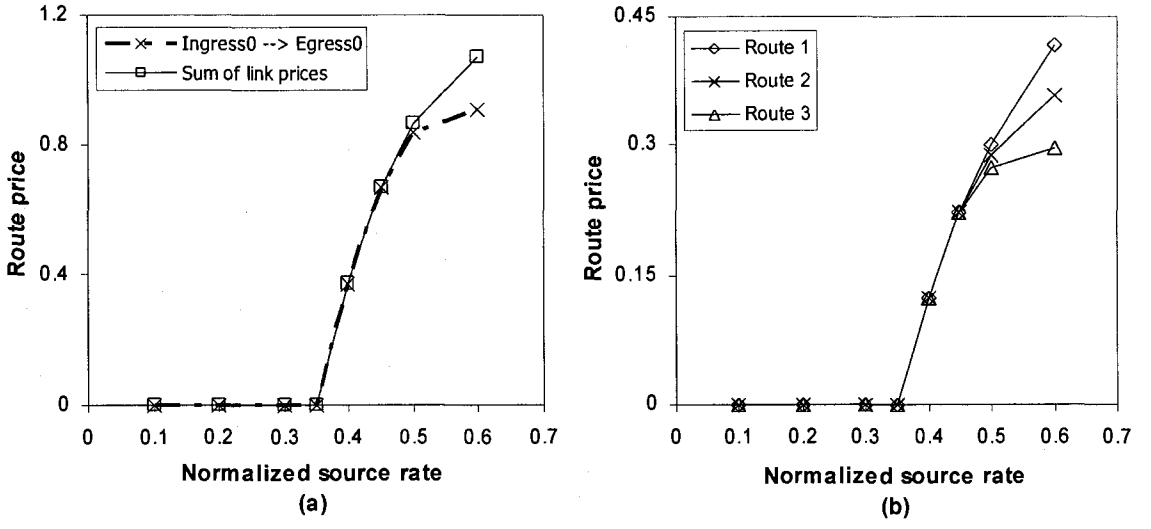

Figure 5.7: Route prices for linear network: (a) Price of Route 1 is compared to the sum of the prices of individual links (b) Individual prices for Route 1, Route 2, and Route 3

5.4 OBS Blocking Probability

In Chapter 4, we introduced a technique similar to the leaky bucket algorithm with burst randomization to control the transmission rate at OBS ingress nodes. This method

changes the burst arrival process at the OBS switches and therefore has an impact on the burst blocking probability. We examine in this section the blocking probability at OBS switches when our traffic policing method is applied at the ingress nodes.

We use the star-topology network shown in Figure 5.2. However, we vary the number of active ingress nodes. We run simulations for three scenarios: with 2, 8, and 16 active sources respectively. Each simulation run generates 10^7 bursts in total.

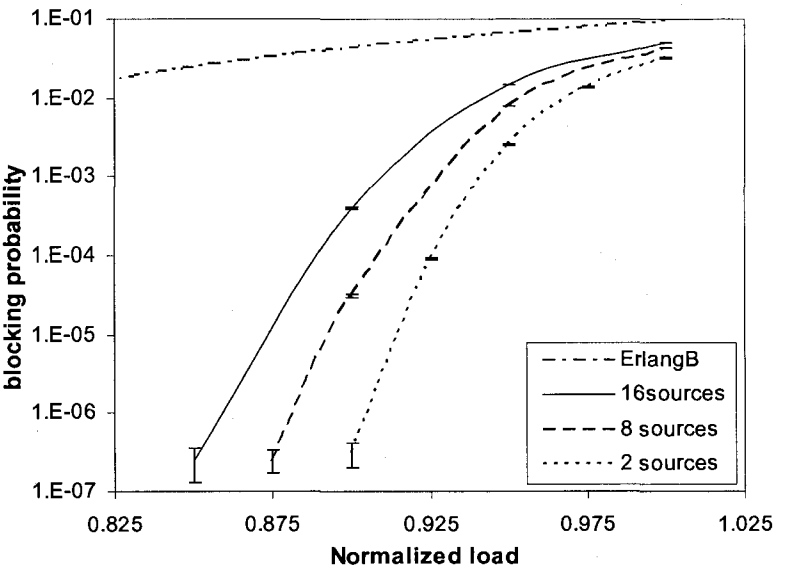

Figure 5.8: OBS blocking probability versus the aggregate traffic load

Figure 5.8 shows the blocking probability as a function of the aggregate load on the congested link. The 99% confidence intervals are also shown. The confidence intervals are too narrow to be noticeable on the figure (less than 12%), except when the load is in the 10^{-7} range.

The blocking probability measured in simulations is much lower compared with the values computed using the Erlang B formula. In fact, with eight ingress nodes transmitting and 10^7 bursts generated during the simulation, the switch load must exceed 87.5% before we observe any loss. This is presumably because the leaky bucket scheme used for rate control smoothes the traffic and reduces the number of simultaneous arrivals at the OBS switch. Moreover, the blocking probability increases with the number of

ingress nodes generating the traffic. This is easily explained by the fact that an increase in the number of uncoordinated users leads to higher burstiness, which unavoidably increases the number of collisions in a random access based network. We remind the reader that only one burst is delivered in case of contention, the others are discarded. Therefore, the percentage of loss is higher when more than two bursts are in contention.

5.5 Single Congested Link, No Delay

In the previous sections, we studied each component of our congestion control algorithm in isolation. In this section, these components are put together and we start evaluation of the PF-OBS congestion control algorithm as a whole. We start with a simple network architecture, where only one link is congested. We also ignore the effect of network propagation delays. This ideal scenario will allow us to understand the basic properties of PF-OBS before considering more realistic and complex cases. For this part, we will use again the star-topology network shown in Figure 5.2. We assume that there are 10 OBS ingress nodes and one egress node. As it was discussed in section 5.2.1, we evaluate the network using two types of traffic sources: persistent and staggered sources. As pointed out previously, persistent sources allow the study of steady state rate allocations while staggered sources highlight the transient response of our algorithm.

5.5.1 Persistent sources

In this scenario, each ingress node generates a single traffic flow to the egress node. All OBS ingress routers receive traffic from persistent traffic sources. This guarantees that any available bandwidth is fully used during the simulation. Therefore, the link between the switch and the destination node will be the bottleneck. We want the congestion control algorithm to keep the load on all links under 70% of capacity or 44.8 Gbps. This corresponds to a very low blocking probability in the OBS network as seen in Figure 5.8.

We assign to each route r a different weight or price preference ω_r, such that $\omega_r = r \cdot \omega_1$, where $r = 1, 2, \ldots, 10$, i.e. the values of ω_r form an arithmetic progression. We consider two scenarios (Scenario A and B), which are the same, except they use different

sequences for ω_r. To examine the effect of the gain κ_r on the congestion control, each of the previous scenarios is simulated with ten different values of the control gain, ranging from $\kappa_r = 0.1$ to $\kappa_r = 1.0$. In one simulation run, all ingress nodes use the same value of κ_r.

5.5.1.1 Scenario A

In scenario A, ω_1 = 250,000 bits/timeslot. Since the average burst size is close to 250 kbits, this means Ingress 1 is willing to sustain approximately one marked burst per interval of 10ms. The Ingress node with the highest weight is source 10 with $\omega_{10} = 2.5$ Mbits/timeslot.

To investigate the convergence and stability of PF-OBS, we initialize the bandwidth allocated to sources with random values at the beginning of a simulation. If the control algorithm is stable, the vector of source rates must always converge to the same point after the initial transient period. To calculate the long-term average rate of each ingress node at equilibrium, the measurements obtained during the transient period are first discarded from the simulation results. We consider that equilibrium is reached when all source rates change by less than 10% afterwards.

Figure 5.9 shows the evolution of the network during the first 5 seconds of a simulation. The curves have been formed by data points, each point providing the measured parameter within a specific timeslot. A timeslot corresponds to a 10 ms time duration. Figure 5.9(a) shows the number of marks received by each ingress node. The data shown in Figure 5.9(a) was smoothed using a moving window of 10 samples to improve clarity. Figure 5.9(b) shows the fraction of traffic that receives a mark at the switch. Figure 5.9(c) shows the bandwidth allocated to each ingress node. Figure 5.9(d) shows the total load on the bottleneck link.

We observe in Figure 5.9(a) that the variations in the number of marks received by individual sources follow closely the changes in their transmission rates (Figure 5.9(c)). This matches the observations we made earlier in section 5.3.3. Moreover, we notice that, when the number of marks received is greater than the target value ω_r defined earlier, a source decreases its transmission rate, and *vice versa*. In Figure 5.9(d), the load on the

bottleneck link initially increases and then settles close to the desired maximum, which we set at $\hat{C}_l = 44.8\,\text{Gbps}$. As soon as the switch starts generating marks, the load on the congested link stabilizes, as seen in Figure 5.9(b) and Figure 5.9(d). This point is reached at approximately 0.15 s after the start of the simulation, i.e. after 15 periods. However, we observe in Figure 5.9(c) that the individual source rate allocations continue to change after this point. This is accomplished while maintaining the total load on the switch under the predefined threshold. All the source rate allocations reach equilibrium after 2s approximately.

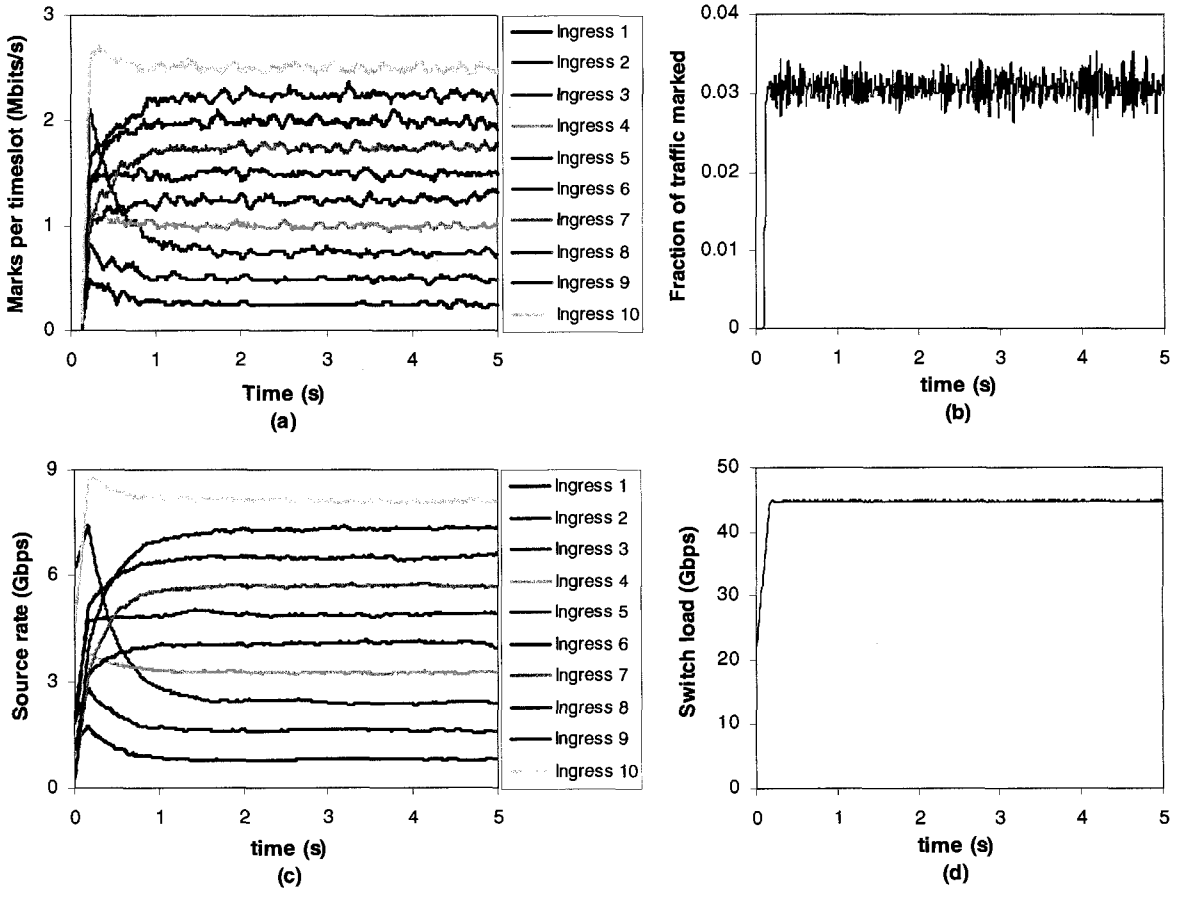

Figure 5.9: Sample trace showing the start of a simulation, gain $\kappa_r = 1.0$: (a) Number of marks received by each source (b) Ratio of aggregate traffic receiving a mark at the switch (c) Bandwidth received by each source (d) Aggregate load on the bottleneck link

It is clear by the curves shown in Figure 5.9(c) that, at equilibrium, the rate allocations are sorted according to the price ω_r chosen by the sources. This is also obvious by the results shown in Figure 5.10, where the average bandwidth received by an ingress node is plotted with respect to the price ω_r it is willing to pay. Figure 5.10 indicates that the bandwidth received by each source at equilibrium is proportional to the price it is willing to pay. This is in perfect agreement with the theory presented in Chapter 3.

According to equation (3.32), the bandwidth assigned to route r at equilibrium is:

$$x_r^* = \frac{\omega_r}{p_l\left(\sum_{s=1}^{10} x_s^*\right)}$$

where $p_l(.)$ is the price of the single bottleneck link l as function of the load. Since we want the available bandwidth to be fully utilized, i.e. $\left(\sum_{s=1}^{10} x_s^* = \hat{C}_l\right)$, it is desired that the total load on the link is: $\sum_{s=1}^{10} x_s^* = \frac{\sum_{s=1}^{10} \omega_s}{p_l\left(\hat{C}_l\right)} = \hat{C}_l$. Therefore: $p_l\left(\hat{C}_l\right) = \frac{\sum_{s=1}^{10} \omega_s}{\hat{C}_l}$. Combining with equation (3.32), this yields:

$$x_r^* = \hat{C}_l \frac{\omega_r}{\sum_{s=1}^{10} \omega_s} \tag{5.1}$$

Therefore, each ingress node should receive an amount of bandwidth proportional to its contribution compared to the total price paid by all users of the congested link.

Each data point displayed in Figure 5.10 is obtained from 100 independent replications of the simulation (10 replications for each value of κ_r). The widths of the 99% confidence intervals are all smaller than 0.5% of the average values. This confirms that the traffic rates in the simulated OBS network always converge to the same point at equilibrium, although the initial rates are set randomly.

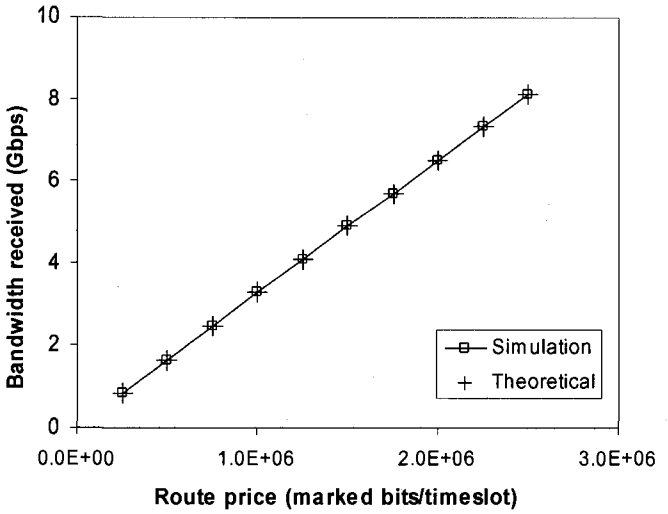

Figure 5.10: Proportionally fair bandwidth allocation

Figure 5.11 shows that the congestion control algorithm is capable of effectively maintaining the total load on the bottleneck link close to the desired limit.

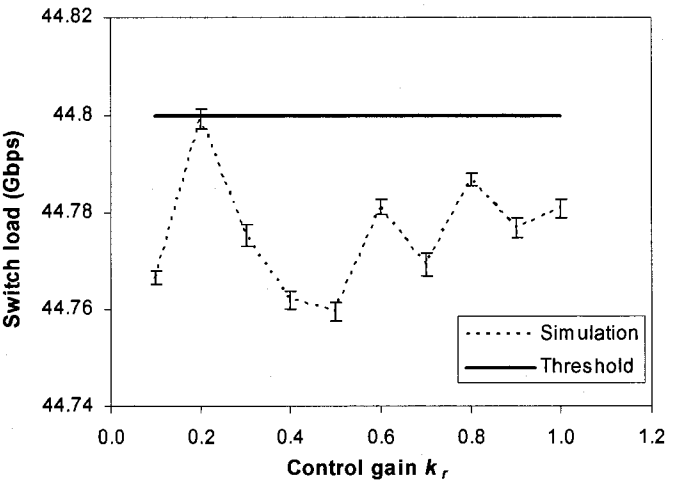

Figure 5.11: Aggregate load at the switch for scenario A

Although the equilibrium of the system is stable, the traffic rates shown in Figure 5.9(a) and Figure 5.9(c) exhibit small random variations around their average values in time. This is mainly caused by random variations in the feedback generated by the switch as seen in Figure 5.9(b). We explained this effect in section 3.7.2 when we discussed the

stochastic stability of the proportional rate controller. The amplitude of these variations depends on the control gain κ_r as pointed out in section 3.7.2. To verify this, we plot the standard deviation of the rate allocations in function of the control gain in Figure 5.12(a). The standard deviation of the rate allocations appears to increase linearly with the gain. We also observe that, for the same value of κ_r, the flows with a bigger weight ω_r exhibit larger variance. Figure 5.12(b) shows the coefficient of variation (CV) of the sources transmission rates, defined as the ratio of the standard deviation to the mean. The CV increases with the control gain. It is smaller, however, for larger traffic flows (larger ω_r). According to Figure 5.12(b), the transmission rates deviate from the long-term average by a few percentage points. The widths of the confidence intervals are less than 0.1%.

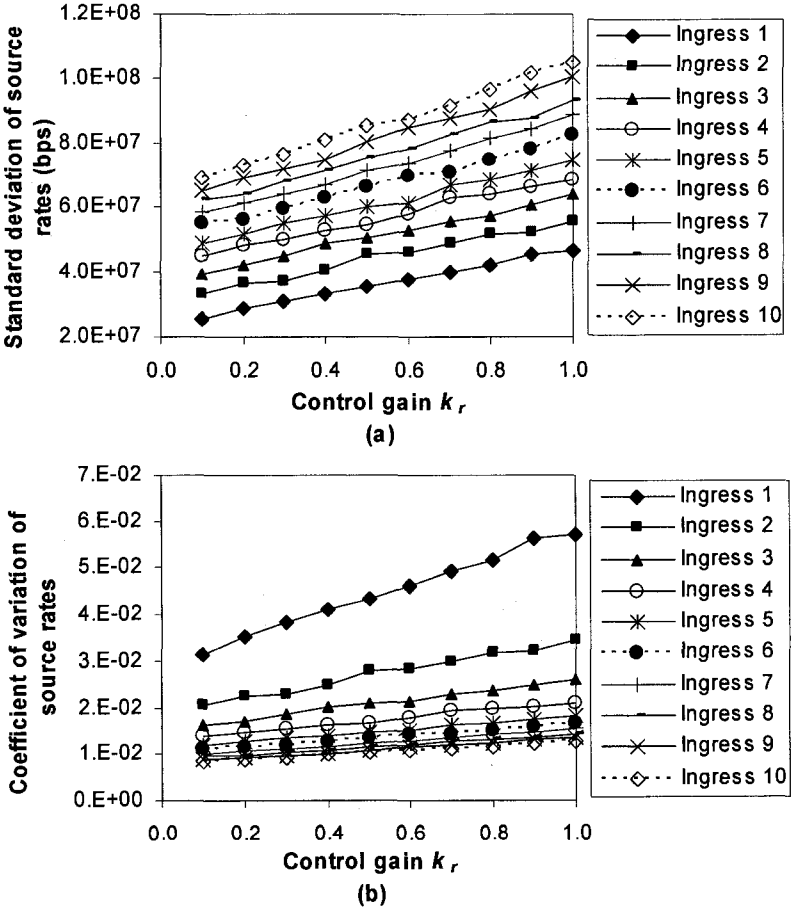

Figure 5.12: Effect of the control gain κ_r: (a) Standard deviation of bandwidth allocations (b) Coefficient of variation of bandwidth allocations

On the other hand, the control gain does not affect the point to which the system converges. As seen in Figure 5.13, the average bandwidth allocated to each ingress node remains constant when κ_r changes. Only the price paid by an ingress node determines the bandwidth it receives. The width of the confidence interval for each point on Figure 5.13 is less than 3% of the mean.

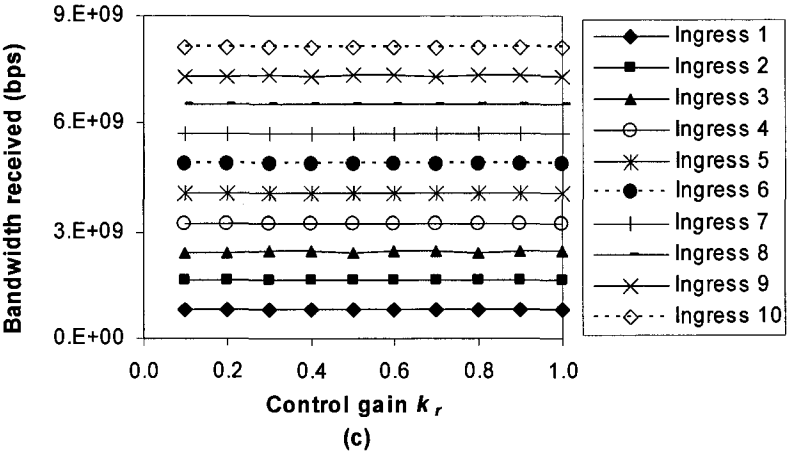

Figure 5.13: Average rate of ingress nodes at equilibrium

5.5.1.2 Scenario B

This scenario is the same as the previous one, except for the vector of price preference ω. In this scenario, each source reduces the price it is willing to pay by a factor of ten, i.e. ω_0 = 25,000 bits/timeslot, ... , ω_{10} = 250,000 bits/timeslot. While the values of price preferences ω_r are entirely different from scenario A, they still form an arithmetic progression.

In Figure 5.14, we compare the resulting rate allocation to the one obtained in scenario A. The widths of the 99% confidence intervals for the data shown in Figure 5.14 are all under 3% of the mean. We note on Figure 5.14 that the average bandwidth received by each ingress node is the same as in Scenario A. Therefore, more than one combination of prices ω_r results in the same rate allocation. Only the value of price choices relative to each other determines the amount of bandwidth received by users. This is also a consequence of equation (5.1).

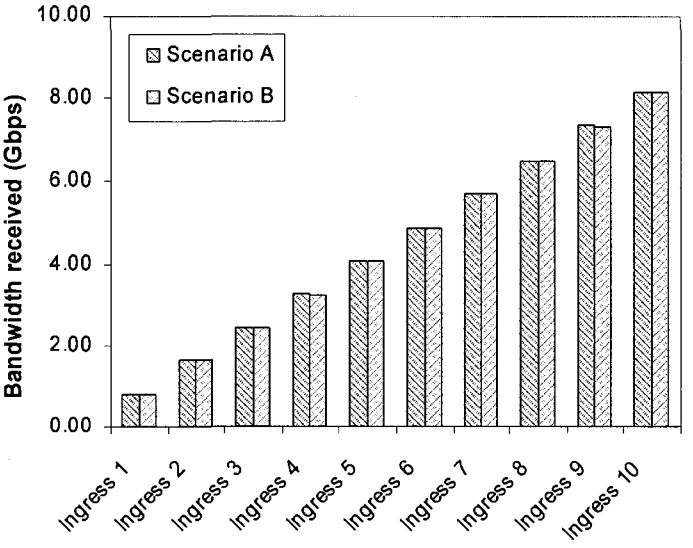

Figure 5.14: Rate allocation scenario A vs. scenario B

Figure 5.15 shows the evolution of the network during the first 40 seconds of a simulation of Scenario B. Figure 5.15(a) shows the number of marks received by each ingress node. The data shown in Figure 5.15(a) was smoothed using a moving window of 100 samples to improve clarity. Figure 5.15(b) shows the fraction of traffic that receives a mark at the switch. The data shown in Figure 5.15(b) was smoothed using a moving window of 10 samples. Figure 5.15(c) shows the bandwidth allocated to each ingress node. Figure 5.15(d) shows the total load on the bottleneck link.

We note in Figure 5.15 that, when users collectively reduce the price ω_r they are willing to pay, the speed of convergence is much slower compared with scenario A. The load on the congested link converges after roughly 1.5s, compared with 0.15 in Scenario A. It also takes approximately 20s for all the source rates to settle down, compared with 2s in Scenario A. We observe that the values of these parameters have increased by a factor of ten ($0.15 \rightarrow 1.5$; $2 \rightarrow 20$). This is the rate of decrease we used to scale down the values of price preferences ω_r of case A, in order to obtain those for case B. This behaviour is understandable, given that ω_r is the additive component of the rate adjustment rule applied by the ingress nodes. Therefore, when the gain κ_r is constant, reducing ω_r means reducing the width of the rate adjustment steps and the response speed of the algorithm.

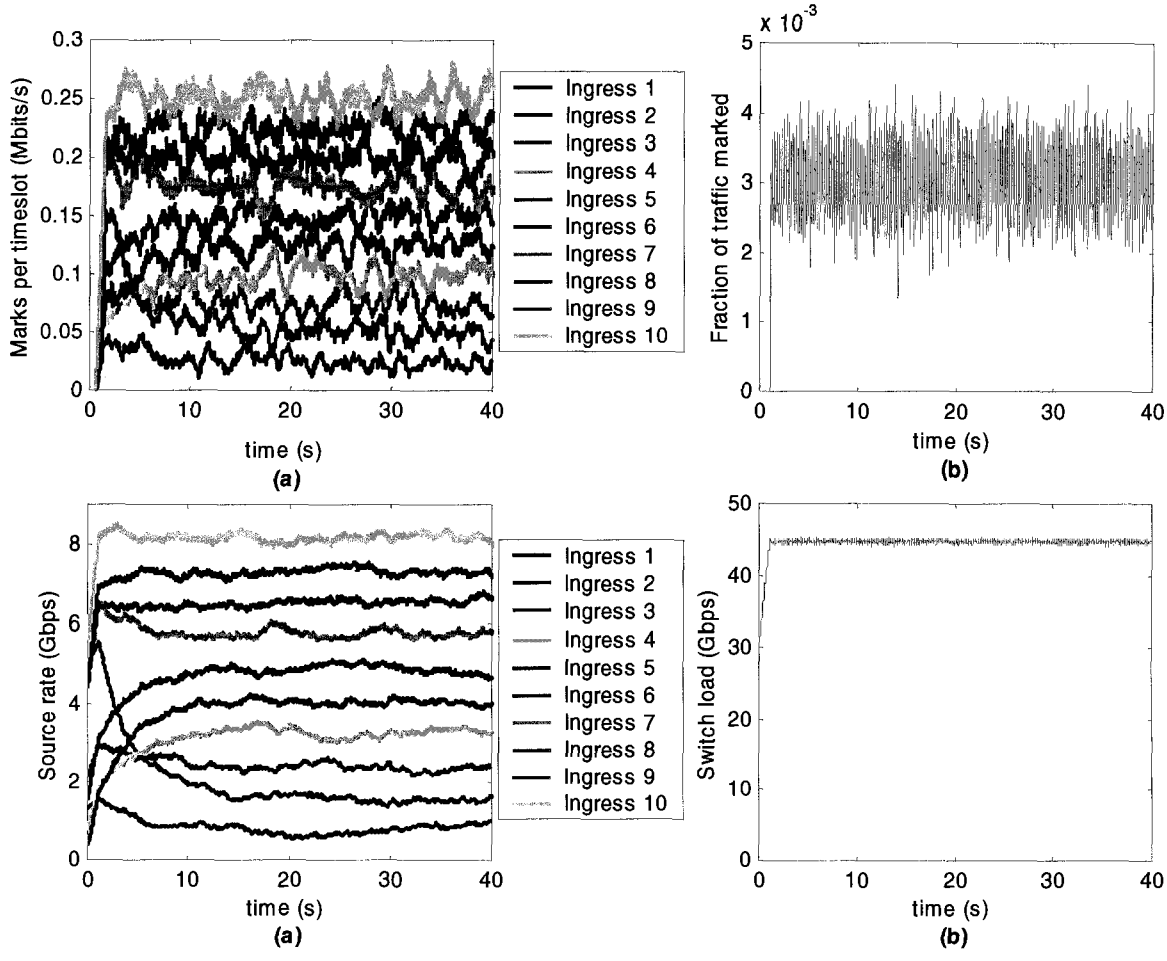

Figure 5.15: Sample traces showing the start of a simulation for Scenario B, gain $\kappa_r = 1.0$: (a) Number of marks received by each source (b) Ratio of aggregate traffic receiving a mark at the switch (c) bandwidth received by each source (d) Load on the bottleneck link

From equation (4.1), the maximum rate of increase of the bandwidth allocated to route r is $\kappa_r \cdot \omega_r / T$ in each timeslot of duration T. The transmission rates are updated $1/T$ times in one second. Therefore, the maximum increase of the transmission rate in one second is given by equation (5.2).

$$\Delta x_{r,\max} = \kappa_r \cdot \omega_r / T^2 \tag{5.2}$$

where the unit of $\Delta x_{r,\max}$ is bits/s^2. Equation (5.2) summarizes the factors affecting the response speed of our congestion control algorithm. It identifies three factors affecting the convergence speed, namely the control gain κ_r, the choice of price ω_r and the timeslot

duration T. Once the link on the bottleneck link stabilises, the bandwidth used by one ingress node can only decrease when other sources increase their share. The effect of the control gain κ_r and the price preference ω_r on the convergence speed is illustrated in Figure 5.16. For the purpose this discussion, we define the response speed as the inverse of the time required for the allocation to converge when all rates are initialized to zero. In simulations, the convergence speed increases linearly with the control gain, as expected. In addition, the convergence speed of Scenario A is 10 times higher than that of Scenario B. This is also expected because the set of price preferences ω_r in Scenario A is 10 times higher than in Scenario B. For Figure 5.16, the confidence intervals are smaller than 0.1% of the mean.

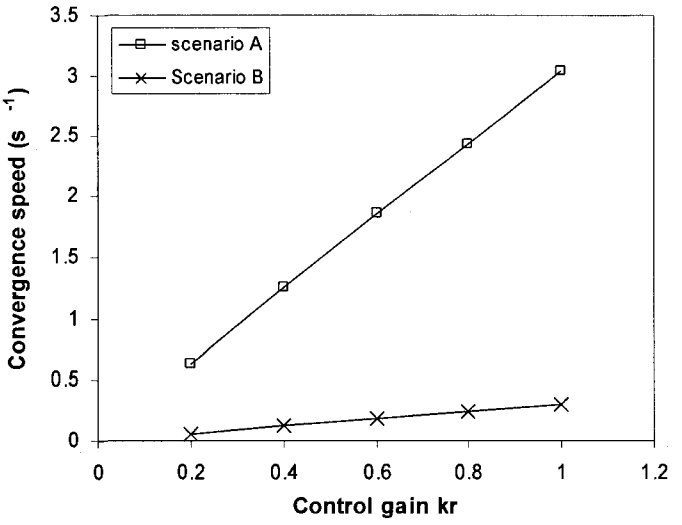

Figure 5.16: Effect of the control gain κ_r and price preference ω_r on convergence speed

While the average-bandwidth received by each source is the same in scenario A and B, we notice when comparing Figure 5.9(c) and Figure 5.15(c) that, in scenario B, the rate allocations display larger deviations from the mean. This is made more obvious in Figure 5.17, where the standard deviation of transmission rates is plotted in reference to the control gain κ_r.

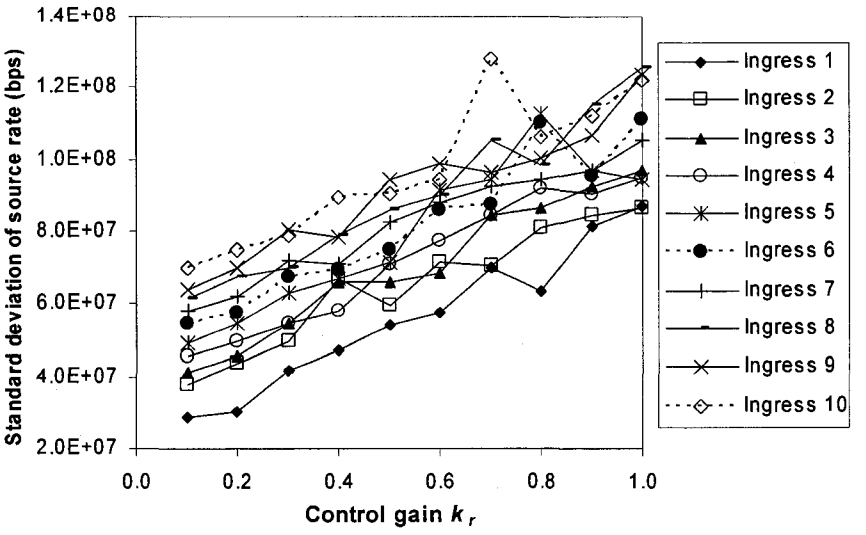

Figure 5.17: Effect of the control gain κ_r on the smoothness of rate allocation, scenario B

We note that, for the same gain κ_r, the values shown in Figure 5.17 are higher compared to those observed for Scenario A (see Figure 5.12). This may be explained as follows. We recall that in the current scenario, ingress nodes have ω_r less than or equal to 250,000 bits/timeslot. For example, the Ingress node 1 has ω_r = 25,000 bits/timeslot. Since a burst with the CI flag set generates a number of marks equal to 250 kbits, this means that Ingress node 1 accepts on average one marked bursts in 10 timeslots. Therefore, the problem for Ingress node 1 consists in adjusting its traffic rate in order to keep the average number of marks equal to ω_r = 25,000 bits/timeslot, given that the number of marks it receives in one timeslot is a multiple of 250 kbits. This results in larger deviations around the mean in the transmission rate. This is not a problem in scenario A, where all the route weights ω_r are multiples of 250,000 bits/timeslot. For this reason, it is preferable to keep the minimum value of ω_r roughly equal to the average burst size.

Despite this, the total load at the bottleneck link remains close to the desired threshold (44.8 Gbps) in both scenarios, as shown in Figure 5.18. Additionally, the blocking probability corresponding to this load is low enough that no blocking occurred in any of the simulation replications. We have run 200 replications for scenarios A and B combined, each generating 5.10^6 bursts once the load stabilises. The 99% confidence intervals are shown for each data point in Figure 5.18.

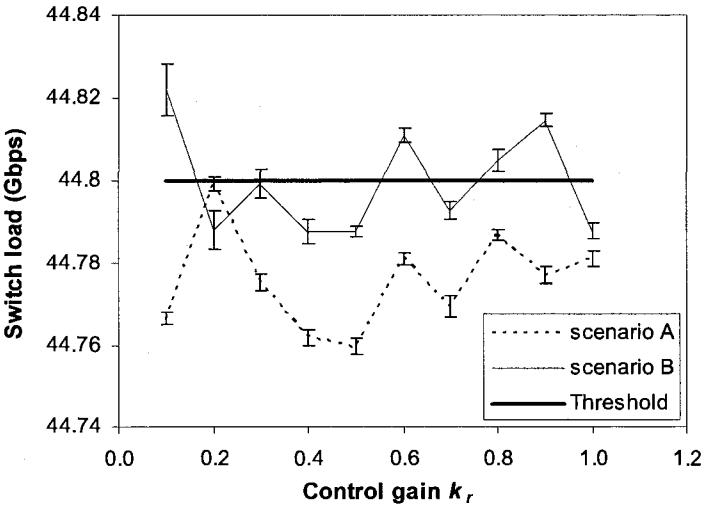

Figure 5.18: Average load occurring at the congested link

5.5.2 Staggered Sources

5.5.2.1 Transient Response

The cases we examined in the previous section represent simplified and static scenarios. Once the bandwidth allocation converges, no further changes occur in the network because we used greedy traffic sources. In this section, we use staggered traffic sources to represent fluctuating demand in the network and study the transient behaviour of PF-OBS. We use the network layout used previously and depicted in Figure 5.2. We run the simulation for 90s. The Ingress nodes generate traffic as follows:

- Ingress 1 and 2 start transmission at 0s until the end of the simulation.
- Ingress 3 and 4 start transmission at 10s and stop at 80s.
- Ingress 5 and 6 start transmission at 20s and stop at 70s.
- Ingress 7 and 8 start transmission at 30s and stop at 60s.
- Ingress 9 and 10 start transmission at 40s and stop at 50s.

When active, traffic generators behave as greedy sources. We set the load threshold at 90% of capacity. The control gain is $\kappa_r = 0.9$. All Ingress nodes pay the same amount per

timeslot: $\omega_r = 10^6$ bits/timeslot, which means that when equilibrium is reached, all users should receive the same amount of bandwidth.

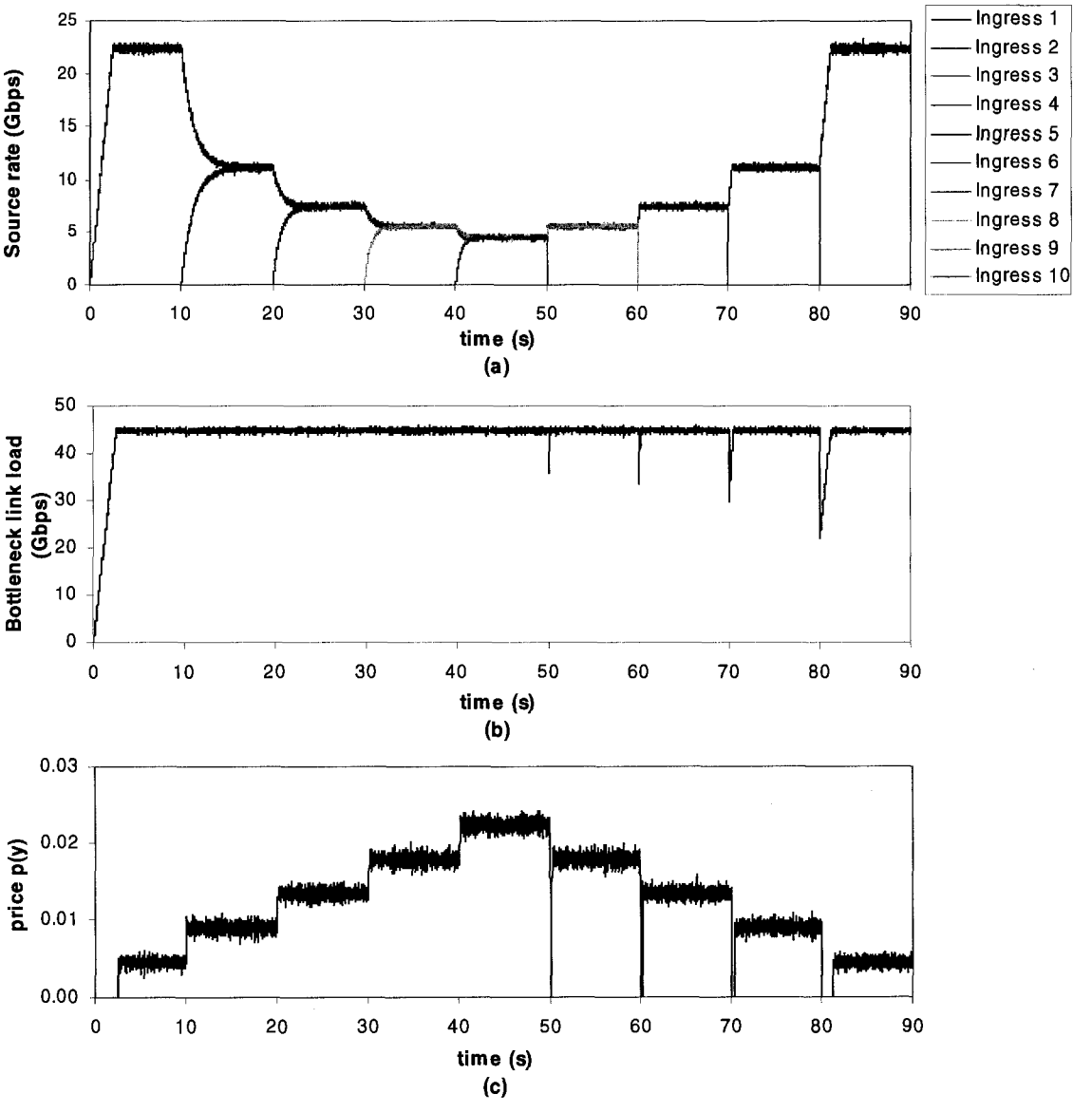

Figure 5.19: Staggered sources: (a) bandwidth received by each source (b) Total load at the bottleneck link (c) Price of the congested link

Figure 5.19(a) shows the progression of the bandwidth allocated to each ingress node in time. It shows that each source adjusts its rate when other users start or stop transmitting. We note that all sources converge to the same rate after each transient period. This is

expected since all sources pay the same price ω_r. With every arrival/departure of other ingress nodes, the available bandwidth is repartitioned equally among all active sources.

We observe in Figure 5.19(b) that while individual user rates are constantly changing, the load at the switch is almost constant and under the predefined threshold. In fact, the only time when there is any noticeable difference between the switch load and the load threshold is when a pair of sources abruptly stops transmitting. In these cases, the load seen by the switch drops suddenly but quickly returns to its previous value. When Ingress 7 and Ingress 8 stop transmitting at t = 80s, the link load is suddenly halved but returns to its previous value after approximately 2s. Sudden reduction in load has no adverse effects in terms of data loss, as the load threshold is never crossed.

Figure 5.19(c) shows the evolution of the price of the congested link as sources start and stop transmitting. This figure indicates the fraction of bits through the bottleneck link that receive a mark. We note that, as more ingress nodes compete for the bandwidth, its price increases in accordance with the law of supply and demand. This adjustment of the link price is what allows the switch to maintain the total load under the desired threshold.

5.5.2.2 Threshold Compensation

We mentioned in section 4.4.3.2 that the link load threshold $\hat{C}_l$ must be reduced when a new source starts using a link. We stated that this is because the load on a link must slightly exceed the predefined threshold $\hat{C}_l$ before it incurs a price. As the offered load is regulated using this price, it follows that the total load on the link must be slightly above the threshold when equilibrium is reached. We showed in section 4.4.3.2 that the excess load is equal to $\sum_r \omega_r$ bits per timeslot when only one link is congested. To compensate for this, $\hat{C}_l$ must be reduced by an amount equal to $\sum_r \omega_r$ bits per timeslot. To illustrate this, we repeat the previous simulation and compare the load on the switch with and without the adjustment of $\hat{C}_l$. The differences are clear in Figure 5.20. On both plots, the predefined threshold (90% or 57.6 Gbps) is shown using dashed lines. In Figure 5.20(a), when compensation is used, the load remains very close to the threshold during the

simulation, except when some ingress nodes stop transmitting. On the other hand, when no compensation occurs, the load increases slightly with the number of active ingress nodes. We notice that, with all 10 ingress nodes transmitting, the threshold is exceeded by 1 Gbps. This results in higher losses in the congested link.

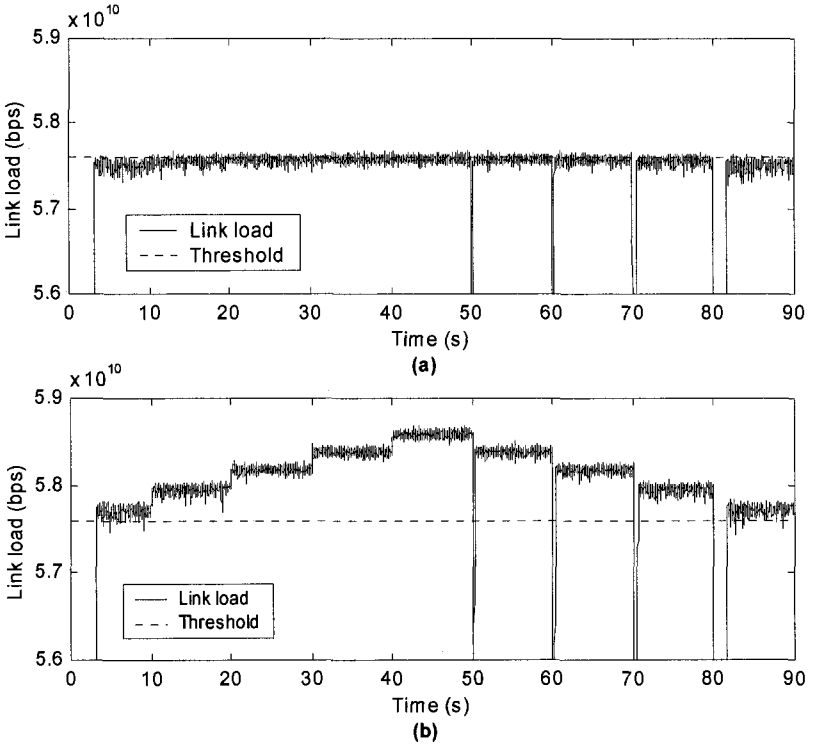

Figure 5.20: Total load on the bottleneck link, staggered sources (a) With threshold compensation (b) Without threshold compensation

5.6 Multiple Congested Links, No Delay

In this section, we revisit the linear OBS network depicted in Figure 5.6. This network carries four routes or traffic flows. We remind the reader that one of these routes spans three congested links while the remaining three contain only one congested link. Again, greedy sources generate traffic for the ingress nodes. The load threshold on the links is set at 90% or 57.6 Gbps. This setup highlights the proportional fairness property of our congestion control algorithm.

5.6.1 Uniform Route Weights

First, we consider the case where users of all routes pay the same price ω_r in each timeslot. We studied theoretically a similar setup in section 3.4.2 and found that the bandwidth is shared as follows. Route 0, from Ingress 0 to Egress 0, receives $x_0 = \hat{C}_l/(L+1)$, where $L = 3$ is the number of congested links. The other routes receive $x_r = L\hat{C}_l/(L+1)$

Table 5.1 and Table 5.2 summarize the simulation results for this scenario, as well as theoretical results for comparison purposes. Simulation results are provided indicating the average and the 99% confidence intervals. Table 5.1 presents the bandwidth received by each route. We note that the simulation data is in line with theoretical results. The routes crossing one congested link receive the same amount of bandwidth (43.6 Gbps) while the route crossing three congested links receives three times less bandwidth.

Route	Number of congested links	Theoretical rate (Gbps)	Average rate, simulation (Gbps)
Ingress 0 → Egress 0	3	14.4	$14.5 \pm 8.7\times10^{-3}$
Ingress 2 → Egress 2	1	43.2	$43.6 \pm 9.3\times10^{-3}$
Ingress 3 → Egress 3	1	43.2	$43.6 \pm 8.5\times10^{-3}$
Ingress 4 → Egress 4	1	43.2	$43.6 \pm 1.0\times10^{-2}$

Table 5.1: Average rate of each route, constant route weight

Table 5.2 presents the total load carried by the congested links. We see that the load is close to the desired threshold. The difference comes from imperfect adjustment of the load threshold as depicted in Figure 5.20, as there are more than one bottleneck.

Link	Threshold (Gbps)	Average link load, simulation (Gbps)
Switch1 → Switch2	57.60	$58.17 \pm 2.6\times10^{-4}$
Switch2 → Switch3	57.60	$58.17 \pm 3.6\times10^{-4}$
Switch3 → Switch4	57.60	$58.17 \pm 3.3\times10^{-4}$

Table 5.2: Average load of each link, constant route weight

5.6.2 Uniform Bandwidth Shares

In this section, Route 0 pays three times the amount paid by users of the other three routes (i.e. $\omega_0 = 3\times\omega_r$, $r \neq 0$). As a result, Route 0 receives the same amount of bandwidth as the other flows as seen in Table 5.3. Table 5.4 shows that the load on each link remains close to the desired limit, as in the previous setup.

We observe that, to receive the same bandwidth, the user of Route 0 must pay three times more than the other routes because it uses three congested links. The current results, together with those shown in Figure 5.10, establish that our congestion control algorithm is proportionally fair. As in the previous section, the degree of confidence is 99%.

Route	Number of links	Theoretical rate (Gbps)	Average rate, simulation (Gbps)
Ingress 1 → Egress 1	3	28.8	$28.90 \pm 1.2\times10^{-2}$
Ingress 2 → Egress 2	1	28.8	$28.94 \pm 1.4\times10^{-2}$
Ingress 3 → Egress 3	1	28.8	$28.94 \pm 1.6\times10^{-2}$
Ingress 4 → Egress 4	1	28.8	$28.94 \pm 1.4\times10^{-2}$

Table 5.3: Average rate of each route on the linear network, all routes have the same rate

Link	Threshold (Gbps)	Average link load, simulation (Gbps)
Switch1 → Switch2	57.60	$57.84 \pm 8.2\times10^{-4}$
Switch2 → Switch3	57.60	$57.84 \pm 5.9\times10^{-4}$
Switch3 → Switch4	57.60	$57.84 \pm 5.5\times10^{-4}$

Table 5.4: Average load of each link on the linear network, all routes have the same rate

5.7 Single Congested Link, Homogeneous Delay

As discussed in section 3.7.1, network delay may destabilize the rate control algorithm of PF-OBS. To illustrate this, we run simulations using the same setup used in Scenario A (see section 5.5.1.1), with the only exception that the link between the switch and the destination node now incurs a nonzero round-trip time (RTT). Therefore, the RTT is the same on all routes. The link load threshold is also set at 90% instead of 70%. We increased the load threshold to highlight the negative effect of instability on the blocking

probability. As the current target load (90%) is close to capacity, any overshoot will result in a noticeable increase in the blocking probability. We consider two cases: a RTT of 4 timeslots (equivalent to a link length between 3,000 and 4,000 km) and 10 timeslots (equivalent to a link length between 9,000 and 10,000 km). According to equation (3.43), the maximum values of κ_r that guarantee a stable control for these scenarios are $\kappa_r = 0.347$ and $\kappa_r = 0.150$, respectively. We would like to show that the network is stable when the gain κ_r is smaller than these values and investigate the behaviour of the system when this condition is not satisfied.

5.7.1 Unstable Congestion Control

We use greedy sources. Figure 5.21 below shows the rate allocations as well as the load on the bottleneck link when κ_r = 1.0. This value clearly exceeds the bounds given previously. The two plots on the left of Figure 5.21 correspond to a delay $D = 4$ timeslots and the two plots on the right correspond to D = 10 timeslots.

Figure 5.21 shows that all traffic streams experience large, synchronized oscillations. The oscillations appear to be periodic and the period increases with the delay. The magnitudes of the oscillations appear roughly constant in time and increase with the delay. Moreover, traffic flows with higher average rates display larger oscillations. Since the oscillations of the individual traffic streams are synchronized, they combine positively, forming a strongly oscillating traffic pattern at the switch.

Such oscillations have detrimental effect on the performance. They cause the traffic at the switch to exceed the load threshold (90%) periodically, significantly increasing the blocking probability on the congested link. Table 5.5 presents the overall blocking probability at the switch for each delay. As expected, a larger delay results in higher blocking probability because of the wider oscillations. Results in Table 5.5 are provided with 99% confidence intervals.

Delay (timeslots)	Blocking probability
4	$5.13\times10^{-3} \pm 8.2\times10^{-5}$
10	$2.74\times10^{-2} \pm 2\times10^{-4}$

Table 5.5: Blocking probability under unstable operation

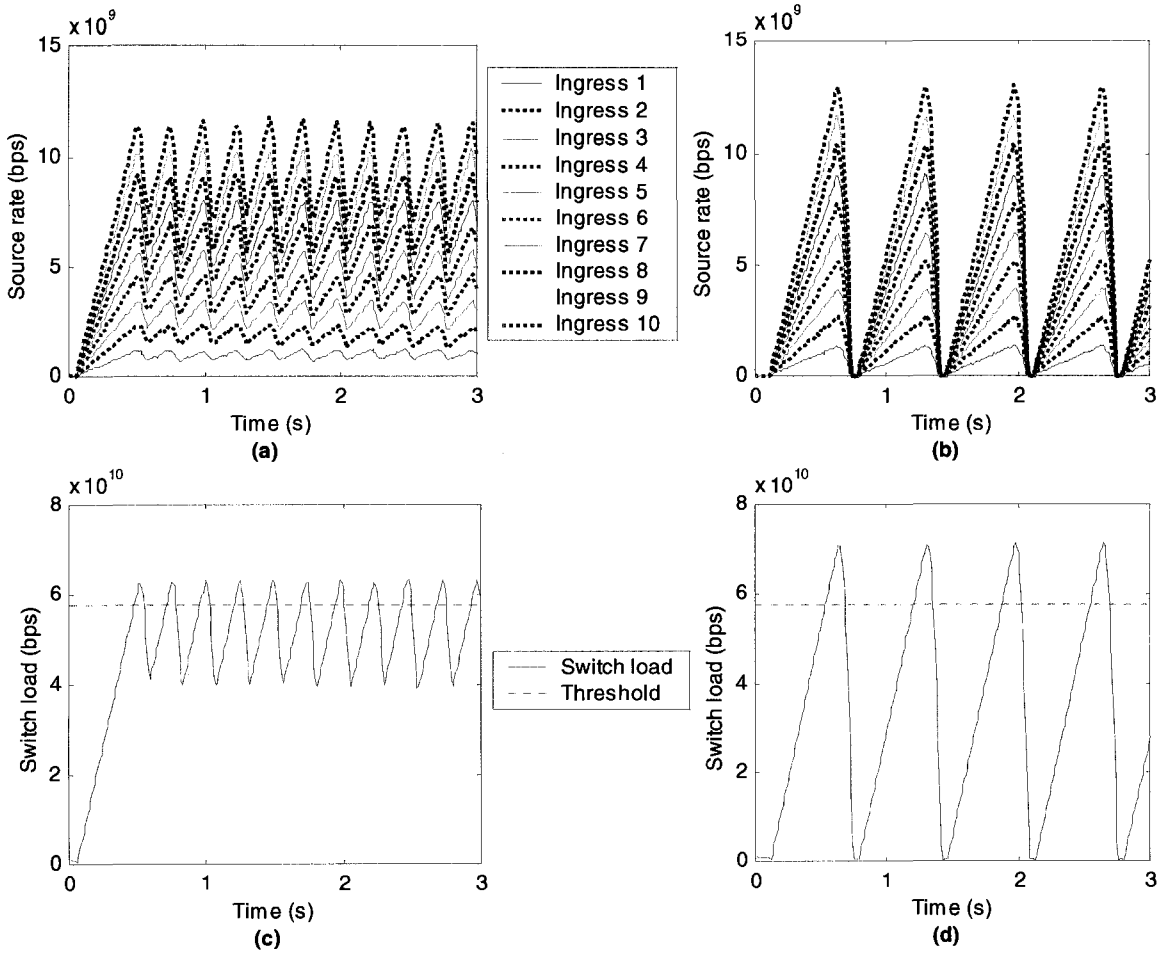

Figure 5.21: Instability caused by propagation delay, when $\kappa_r = 1.0$: (a) Bandwidth allocated to each ingress node for $D = 4$ (b) Bandwidth allocated to each ingress node for $D = 10$ (c) Total load on the congested link for $D = 4$ (d) Total load on the congested link for $D = 10$

Although the network is unstable, we explained in section 3.7.1.3 that the oscillations are bounded. We presented theoretical bounds for the magnitude of oscillations. The amplitudes of oscillations observed in simulations closely match the bounds computed with equations (3.44) and (3.45). Figure 5.22 provides a visual example of this, where only the rate of Ingress node 10 is displayed. Detailed simulation results for each ingress node are provided in Table 5.6 with 99% confidence intervals included.

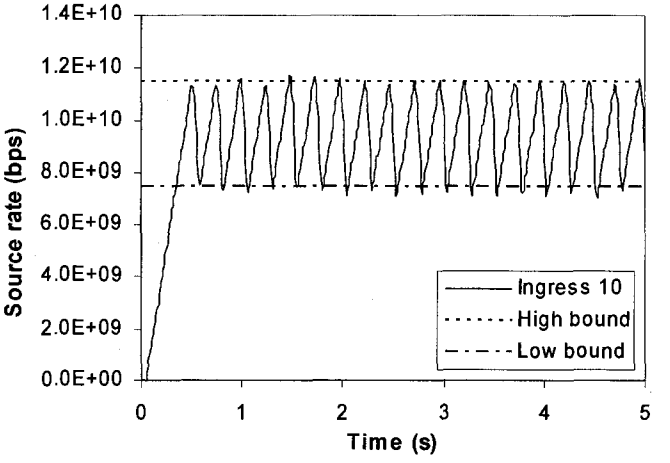

Figure 5.22: Simulation results and theoretical bounds for the rate of Ingress 10, $D = 4$

	Theoretical bounds		Simulation results	
Node	Lower bound (Gbps)	Upper bound (Gbps)	Lower bound (Gbps)	Upper bound (Gbps)
Ingress 1	0.75	1.15	0.65 ± 1.6×10-2	1.21 ± 1.8×10-2
Ingress 2	1.49	2.29	1.38 ± 2.5×10-2	2.38 ± 2.7×10-2
Ingress 3	2.24	3.44	2.07 ± 2.5×10-2	3.52 ± 3.3×10-2
Ingress 4	2.99	4.59	2.78 ± 5.3×10-2	4.68 ± 2.5×10-2
Ingress 5	3.74	5.74	3.49 ± 3.3×10-2	5.81 ± 3.6×10-2
Ingress 6	4.48	6.88	4.19 ± 4.1×10-2	6.98 ± 2.7×10-2
Ingress 7	5.23	8.03	4.90 ± 6.6×10-2	8.12 ± 3.4×10-2
Ingress 8	5.98	9.18	5.59 ± 5.2×10-2	9.28 ± 4.4×10-2
Ingress 9	6.73	10.33	6.31 ± 3.2×10-2	10.42 ± 5.8×10-2
Ingress 10	7.47	11.47	6.99 ± 7.2×10-2	11.60 ± 4.6×10-2

Table 5.6: Comparison of theoretical bounds and simulation results

5.7.2 Stable Congestion Control

Figure 5.23 shows the rate allocations when the gain κ_r satisfies the stability condition described by equation (3.43). The value of κ_r used for Figure 5.23 is set to 0.95 times the limit provided by equation (3.43). The limits calculated using equation (3.43) are 0.347 and 0.149 for $D = 4$ and $D = 10$, respectively. Therefore, we use $\kappa_r = 0.330$ and $\kappa_r = 0.142$ in our simulations. We observe small oscillations just after the switch load reaches the threshold value. However, these oscillations eventually die down and the system reaches a steady state after 3 to 4 seconds when $D = 4$.

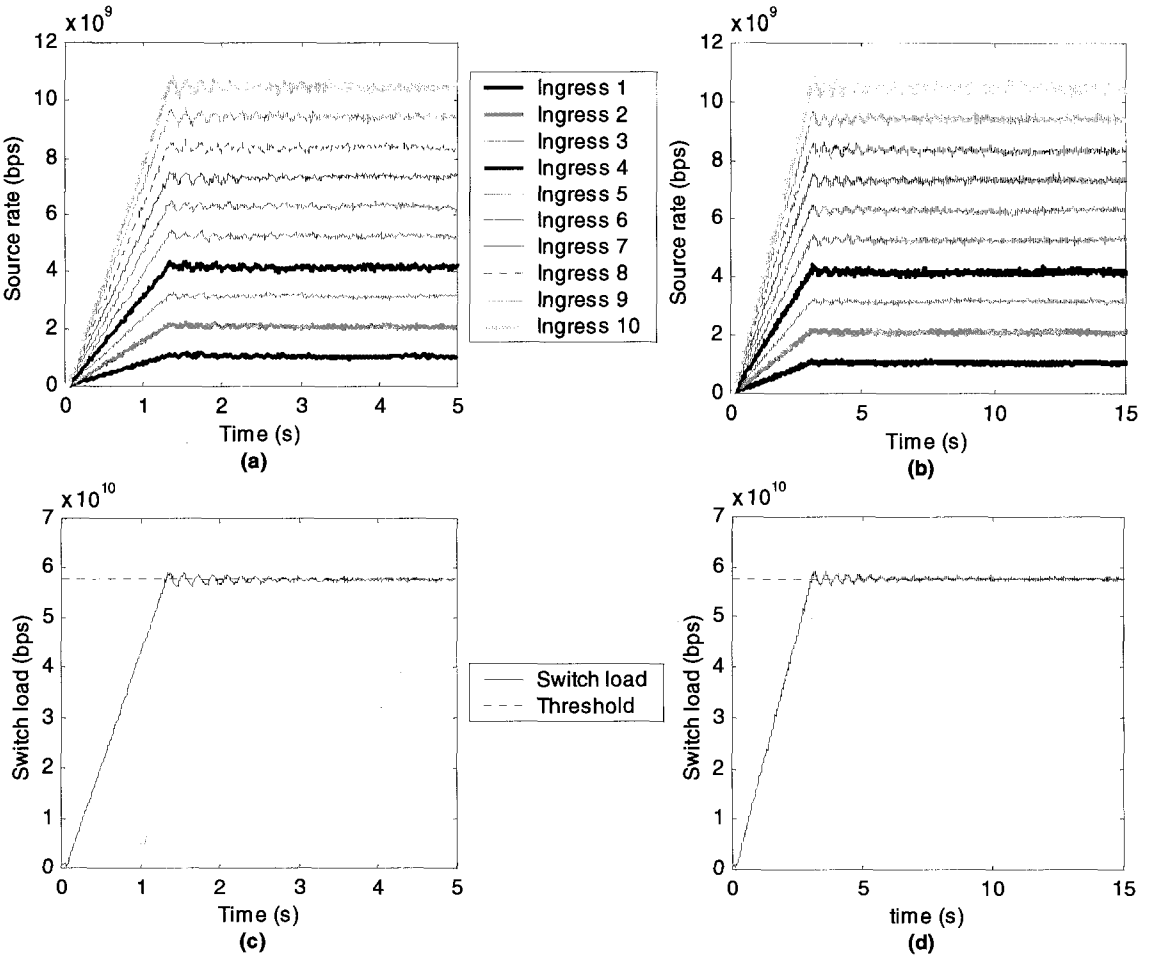

Figure 5.23: Traffic rates for stable OBS network with delay: (a) Bandwidth allocated to each ingress node for $D = 4$ (b) Bandwidth allocated to each ingress node for $D = 10$ (c) Total load on the congested link for $D = 4$ (d) Total load on the congested link for $D = 10$

Since there is no oscillation when equilibrium is reached in the current scenario, the load threshold is not exceeded. Therefore, the blocking probability is lower, compared with the unstable system. The measured blocking probability of the stabilized system is summarized in Table 5.7.

Delay (timeslots)	Blocking probability
4	$1.19\times10^{-4} \pm 1.2\times10^{-5}$
10	$1.32\times10^{-4} \pm 1.9\times10^{-5}$

Table 5.7: Blocking probability for stable OBS network

5.7.3 Staggered and Unresponsive Sources

Until now, we have assumed that all edge nodes in an OBS network apply the PF-OBS algorithm. In this section, we examine the behaviour of the congestion control when some ingress nodes do not respond to congestion signals. Unresponsive sources are modeled using CBR sources. Again, we use the star-topology network depicted in Figure 5.2. Ingress 9 and Ingress 10 fill the role of the unresponsive sources. The other ingress nodes behave as staggered sources and apply the PF-OBS algorithm. When active, they behave as greedy sources. Ingress nodes start and stop transmitting at various times during the simulation. When active, the unresponsive sources consume half of the available bandwidth. This means that, together, the two CBR sources generate 28.8 Gbps when the load threshold on the bottleneck link is set at 90% (57.6 Gbps). The round-trip delay is $D = 4$ timeslots for all routes. This setup will also highlight the transient state performance of PF-OBS in the presence of delay.

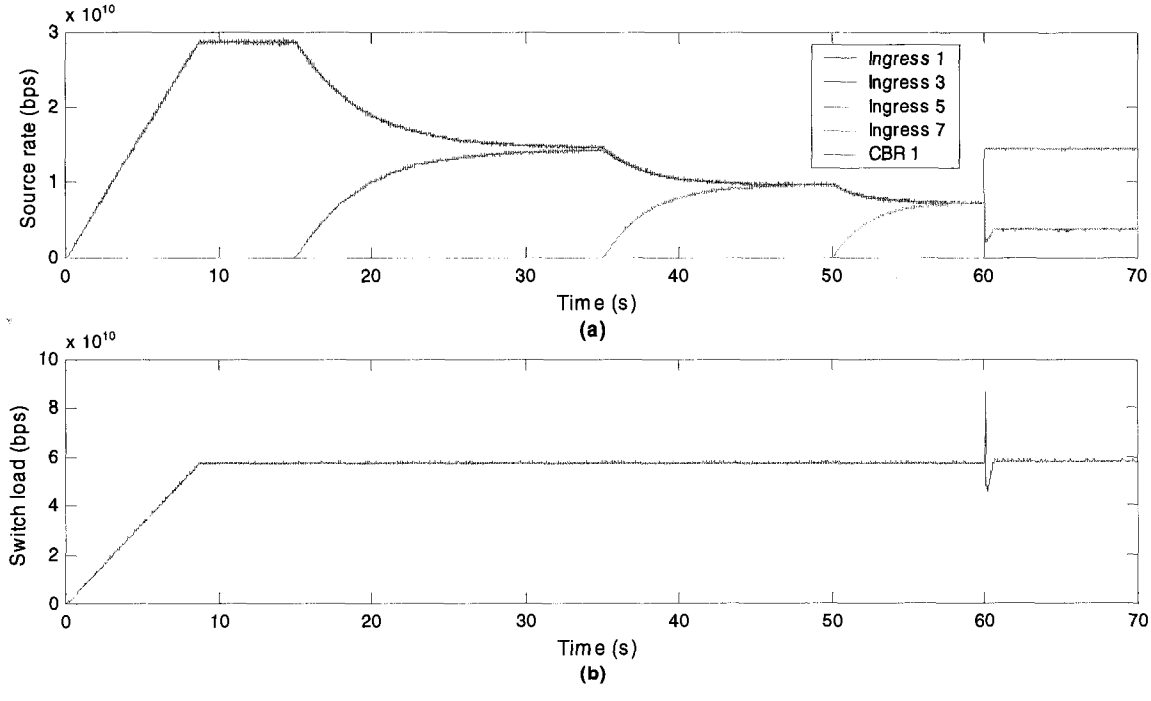

Figure 5.24: Staggered and unresponsive sources, stable with $D = 4$ (a) Transmission rates of ingress nodes (b) Overall load on bottleneck link

Figure 5.24 shows the behaviour when the network is stable. Ingress 1 and Ingress 2 start transmitting at the same time and their responses are the same. Therefore, we only plot the response of Ingress 1 in Figure 5.24 to keep the figure tidy. The same is true for Ingress 3 and Ingress 4, etc.

We note the slower response compared with Figure 5.19. This is due to the reduction of the control gain required for stability ($\kappa_r = 0.330$). Aside from this, the Ingress nodes that apply congestion control behave as in Figure 5.19. The interval starting at $t = 60$s is of particular interest for us. This is when the unresponsive sources start transmitting. These sources begin their transmission at full speed. This results in a sudden spike in the load received by the switch as seen in Figure 5.24(b). This spike remains for the duration of an RTT. After a delay equal to the RTT, the feedback regarding the change arrives at the sources. At this point, the ingress nodes that apply the PF-OBS congestion control reduce their sending rates. The minimum is reached 0.14s after the start of the spike. There is some overcompensation as the minimum switch load reached during this adjustment is roughly 46 Gbps. The traffic rates return to steady state roughly 0.6s after the start of the spike. To sum it up, in case some ingress nodes do not respond to congestion signals, the ingress routers that apply congestion control will adjust their rates in order to keep the load on the bottleneck close to the threshold. However, the load on the congested link does not return exactly to its previous value after 60s. When steady state is reached after 60s, the average load on the congested link is roughly 0.75 Gbps above the threshold. The reason for this is the following. The OBS switch marks all traffic streams and cannot distinguish the traffic from unresponsive ingress nodes. Therefore, the marks assigned to unresponsive sources are lost and cannot be used to control the congestion. This results in an apparent reduction of the link price seen by other ingress nodes, causing a small increase in the load received by the switch.

In contrast, Figure 5.25 shows the response of the same network when it is unstable ($\kappa_r = 1.0$).

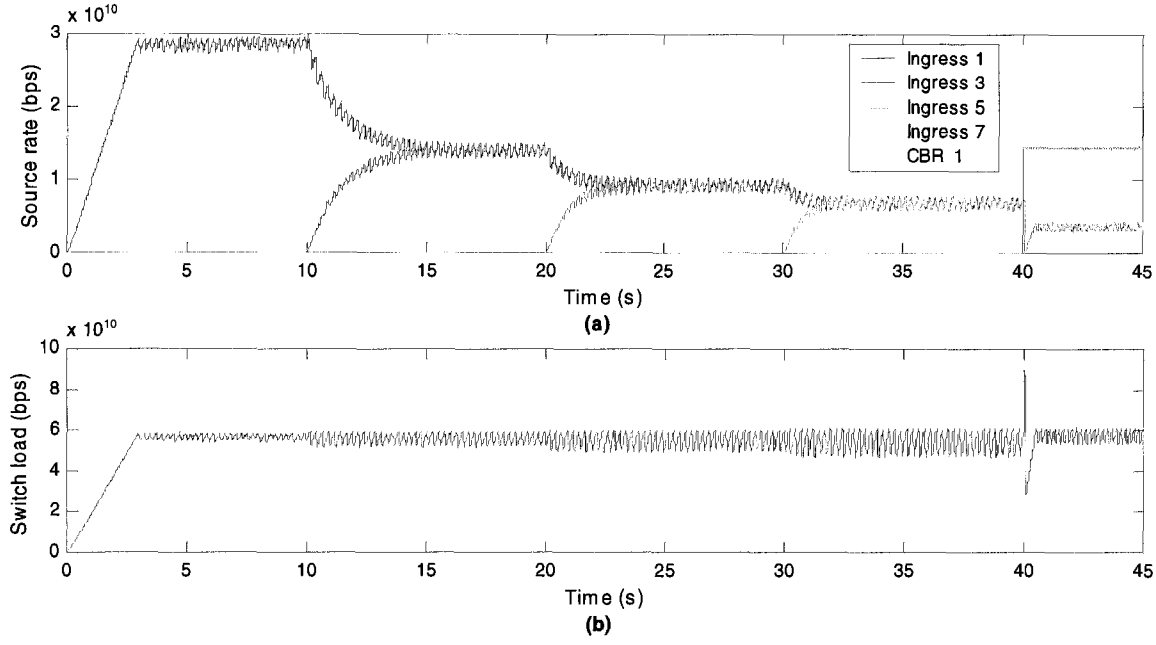

Figure 5.25: Staggered and unresponsive sources, unstable with D = 4 (a) Transmission rates of ingress nodes (b) Overall load on bottleneck link

The transmission rates of all responsive sources exhibit oscillations, as seen in Figure 5.25(a). The oscillations are also synchronized, although ingress nodes begin transmission at different times in the current case. The magnitude of the oscillations is the same for all sources running the PF-OBS algorithm. Consequently, the amplitude of the oscillations at the switch increases when more ingress nodes generate traffic.

5.8 Meshed Network with Delay

In previous sections, we have simulated only simple network scenarios, involving a single congested link or a linear topology network. This allowed us to demonstrate the basic properties of PF-OBS, analyze and explain its behaviour. However, real networks are understandably more complex. To assess the performance of PF-OBS in a realistic setting, we simulate our congestion control algorithm using the 14-node NSF network depicted in Figure 5.26. The numbers next to the links indicate their lengths in kilometres. Each vertex in Figure 5.26 represents one OBS edge node connected to a

local OBS core node. The OBS core node is connected to neighbouring core nodes. OBS edge nodes act as traffic sources and sinks.

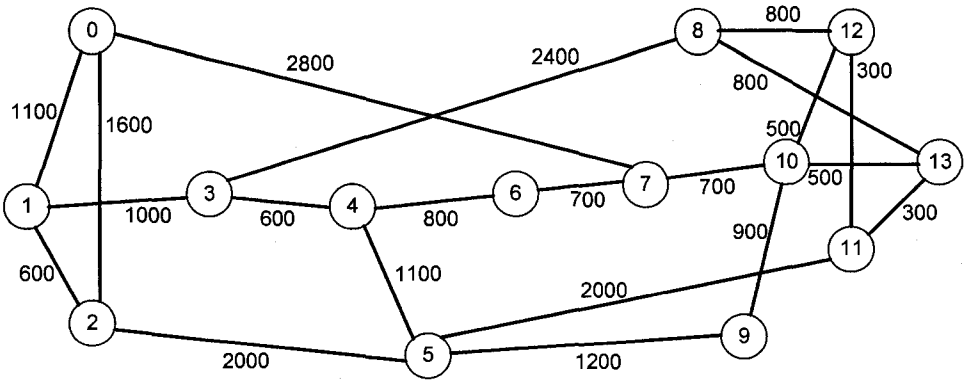

Figure 5.26: 14-node NSF network

Our goal is to maintain the load on all links under 90% of their capacity, i.e. 56.7 Gbps. This corresponds to a blocking probability under 10^{-3} on each link, according to Figure 5.8. We consider this level of loss acceptable for most applications. We apply minimum hop routing. This is to minimize the end-to-end blocking probability since each link in a path introduces a possibility of burst loss. When several minimum hop routes exist between two edge nodes, we choose one randomly and maintain it for the entire duration of the simulation.

Each edge node sends traffic to the 13 other edge nodes in the network. This means there are 182 routes or traffic flows in this OBS network, which are individually regulated by the PF-OBS algorithm. Ingress routers are alimented by greedy sources in order to use all available bandwidth. The network depicted in Figure 5.26 features a mix of diverse simulation parameters, as opposed to the simple cases seen previously. This includes:

- Various path lengths in terms of the number of hops.
- Heterogeneous delays: Each route in the NSF network has a different RTT, depending on its length. The RTT values in the network range from 1 to 6 times the update period, as shown in Figure 5.27. For this reason, the control gain κ_r is chosen according to equation (3.43) to guarantee stability under delay.

- Each ingress node pays a random price ω_r in the interval 250,000 to 2,500,000 bits/timeslot for each of the 13 traffic flows it generates. Each OBS link carries traffic from different routes.

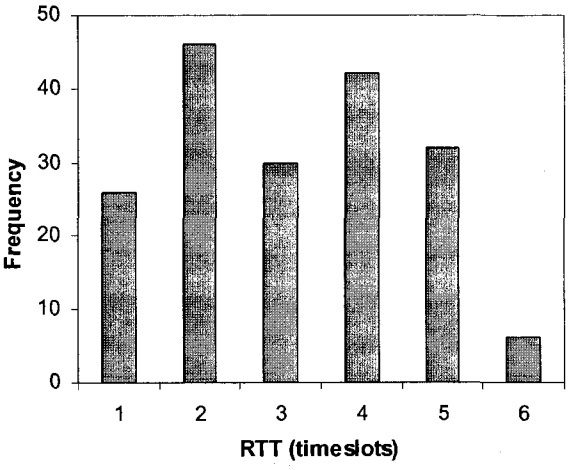

Figure 5.27: Distribution of RTT in the NSF network

We consider this setup a reasonably accurate model of a real network. We ran 10 independent simulation replications with the previous parameters, each lasting 90 simulated seconds and generating approximately 2.5×10^8 bursts. All transmission rates are initialized to 6.4 Mbps (i.e. 0.01% of the link capacity).

Figure 5.28 and Figure 5.29 provide a representative sample of the traffic flowing through the network in a simulation run. Figure 5.28 shows the traffic rates of all 13 flows generated by Edge node 1. Figure 5.29 shows the load on all output ports of switch4. The overshoot seen in Figure 5.28 is caused by differences in the values of the control gain κ_r. Note that the routes that overshoot are those destined to the closest neighbors (Edge 0 and Edge 2), which means lower delay and higher gain. These routes increase their rates faster. However, as the other routes increase their rates, the flow to Edge 0 and Edge 2 must be reduced.

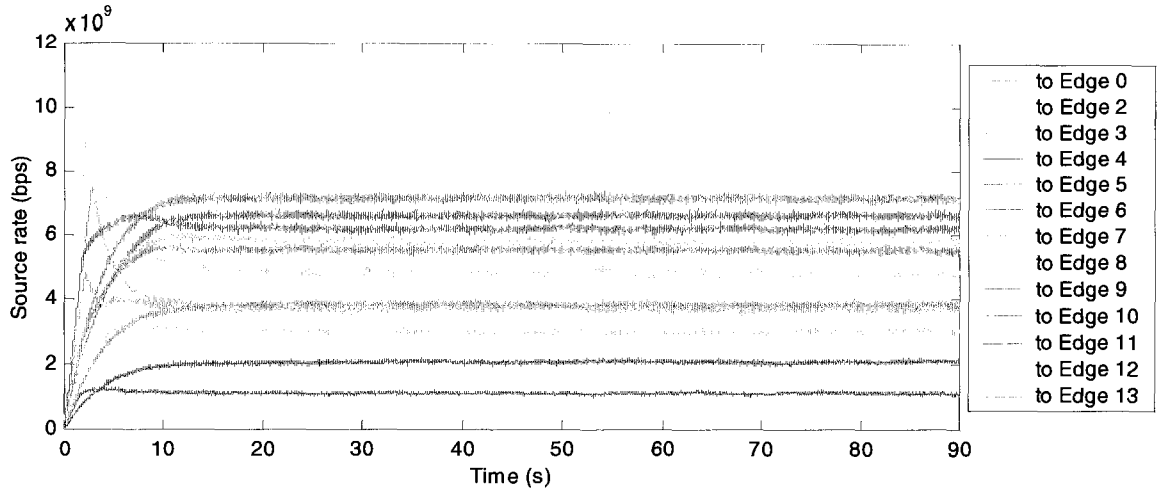

Figure 5.28: Traffic flows generated by Edge node 1

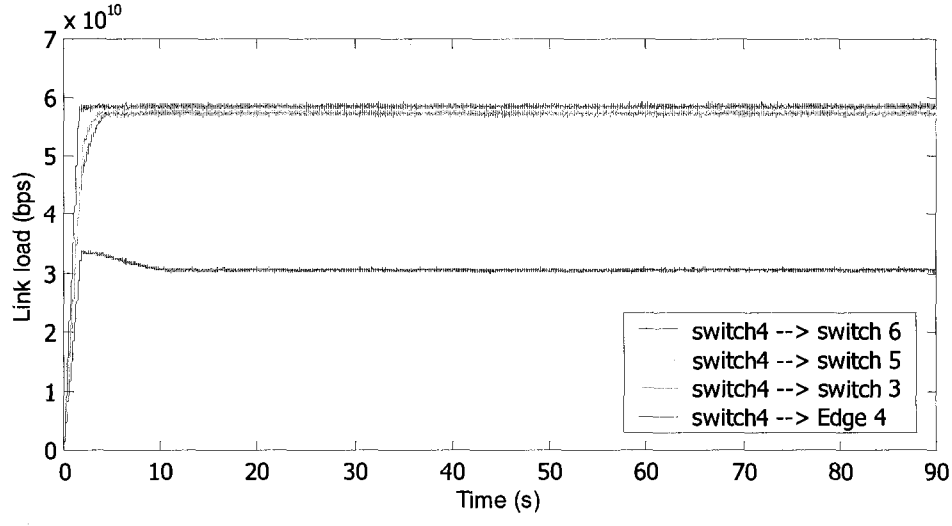

Figure 5.29: Traffic load at the output links of switch 4

In all replications, the network stays stable and the load on all links remains close to the predefined threshold. The link that carries the most routes (16 routes) is the link from switch 10 to switch 7 while the link from switch 11 to switch 13 has the fewest (3 routes). The bottlenecks for this setup are mostly the ports connecting OBS switches to their associated edge nodes. As every edge node sends to and receives traffic from all other nodes, these ports tend to be congested. Since route prices ω_r are randomized, the repartition of load in the network is not the same in all replications but depends on the simulation seed. Not all links connecting two switches together are congested. Based on

10 replications, the maximum blocking probability for a switch port is $1.9\times10^{-3} \pm 2.6\times10^{-4}$ while the overall blocking probability in the network is $4.8\times10^{-4} \pm 3.5\times10^{-5}$, with 99% confidence. Detailed results regarding the congestion at each link are provided in Appendix C.

5.9 Discussion

The PF-OBS algorithm described in Chapter 4 appears effective in managing the congestion in the OBS network. In all scenarios simulated in this chapter, the load on all links remains close to the desired limit. Consequently, the blocking probability stays bounded, which is our first objective.

There is good agreement between the results obtained in simulations and the theory presented in Chapter 3. We observe this concordance in the marking policy assessment in section 5.3 and in the traffic rates obtained in section 5.5. Another example is the calculation of the appropriate control gain to ensure stability in the presence of delay. The critical value of the gain given in Chapter 3 is the point at which oscillations appear in simulations. Furthermore, the amplitudes of observed oscillations closely match the theoretical bounds. Therefore, we are confident that the congestion control method we proposed in Chapter 4 accurately implements the theoretical foundation seen in Chapter 3.

Fast convergence and stability are generally needed for any congestion control technique. We know from control theory and from our simulations that these two objectives conflict with each other. In our case, they depend on three parameters: the control gain κ_r, the route price or weight ω_r, and the timeslot duration T. Increasing either κ_r or ω_r speeds up convergence but increases the spread of the traffic rates around the equilibrium. With κ_r unacceptably large, the network may even become unstable. Reducing T results in more frequent updates but it also increases number of update periods during a round trip. Therefore, choosing appropriate values requires a tradeoff between response speed and smoothness of rate allocations.

We also notice that the rate update rule (see equation (4.1)) is similar to the well-known proportional controller from control theory. We remember that a proportional controller can steer a system towards its target input but theoretically never reaches it. In practice, this is not an issue due to the bufferless nature of OBS networks. However, in electronic networks, a small discrepancy between the target rate and the actual rate leads to a queue build-up or starvation. A queue drops all incoming frames when it is full, whereas an OBS switch does not show such sudden successive losses. The curve for the loss probability is always smooth as we saw in Figure 5.8. Therefore, small discrepancies between the optimal and current rate are tolerable in OBS networks.

5.9.1 Comparison with OBS without Congestion Control

To assess the benefit of the PF-OBS algorithm, we compare the performance of an OBS network without any traffic policing with another one running our algorithm. For this purpose, we revisit the star-topology network seen in Figure 5.2. We remind the reader that this network has 10 ingress nodes sending traffic to one egress node. We assume that the incoming traffic is Poisson before it is assembled into bursts. The simulation parameters are the same as previously, except for the number of wavelength channels per fibre. In this section, we use $N = 4$ wavelengths for data transmission. The load threshold is set at 50% of link capacity. We use these simulation parameters because they are fairly similar to those used in the PCwER scheme[20]. We remind the reader that PCwER is one of the previous works on rate-based congestion control for OBS, which we reviewed in section 2.8.4. In this manner, we can also compare our results with [20]. In addition, choosing $N = 4$ wavelengths per fibre highlights the advantage of congestion control because the losses are expected to be high. The authors in [20] used the following parameters: 2 ingress nodes sharing a bottleneck link, Poisson burst arrival at the switch, $N = 4$ wavelengths per fibre.

The results are shown in Figure 5.30. We observe that the performance of both networks is the same when the offered load is below the threshold. This is understandable because PF-OBS does not take any action when it does not receive any congestion signal. When the load increases above 50% on Figure 5.30(a), PF-OBS maintains the blocking probability around 7%, which is lower than the losses experienced by the plain OBS

network. However, this is achieved at the cost of lower throughput through the network as shown in Figure 5.30(b). The results shown in Figure 5.30 are similar to those obtained with PCwER in [20].

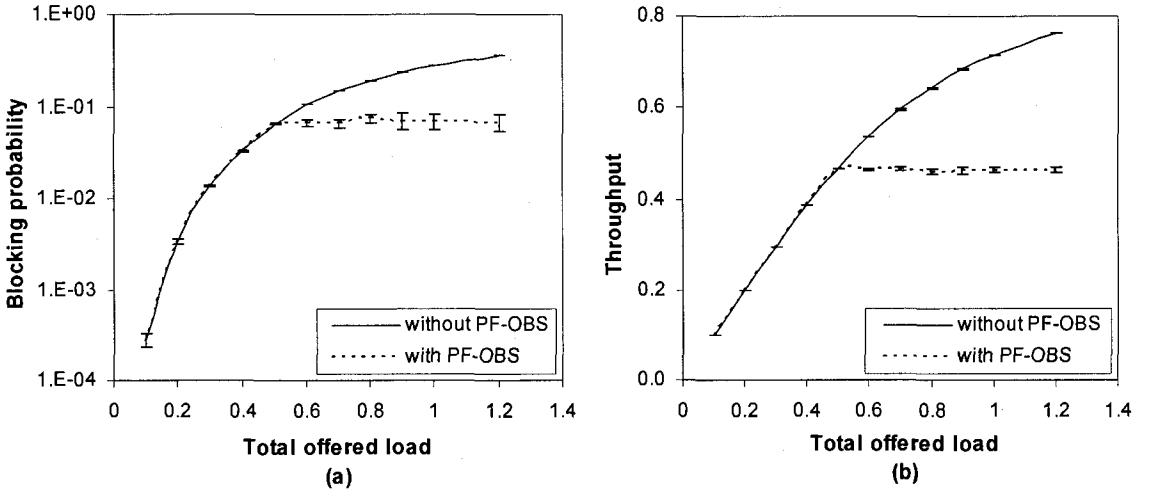

Figure 5.30: Comparison between OBS without congestion control and with PF-OBS algorithm (a) Blocking probability (b) Throughput

5.9.2 Comparison with Existing Methods

In section 2.8.4, we reviewed three previous proposals for rate-based congestion control on OBS:

- Robust OBS (ROBS) [81].
- Differentiated ABR (D-ABR) [21].
- Proportional Control with Explicit Reduction Request (PCwER) [20].

Although the names used are different, D-ABR, PCwER and PF-OBS all define a load threshold on all links. The value of the threshold is slightly lower than the link capacity in order to keep the burst losses low. When the load exceeds this threshold, a congestion signal is generated, prompting the ingress nodes to reduce their transmission rates. Unlike these three methods, ROBS uses burst loss probability as a congestion signal.

Because D-ABR is based on the ERICA algorithm, it converges faster than our method. In contrast, ROBS, PCwER, and PF-OBS change the rate allocations in fixed increments.

It is well known that the explicit rate approach used by D-ABR imposes a heavy processing load on the network nodes [94]. This is also the case with PCwER because it requires edge nodes to organise a large amount of information for each feedback received, unlike PF-OBS.

When compared to the previous methods, the main advantage of PF-OBS comes from its utility maximisation approach. The ERICA scheme, on which D-ABR is based, achieves exact max-min fairness while PCwER adopts a relaxed form of max-min fairness, based on a maximisation of the following fairness index:

$$FI(x) = \frac{\left(\sum x_r\right)^2}{n\sum x_r^2} \tag{5.3}$$

where n is the number of flows sharing a congested link, x_r is the rate of flow r, and x is the set of flows sharing the link.

If a max-min fair allocation exists for a network, it is unique [85]. This makes the allocation too rigid because it does not respond to the bandwidth requirements or preferences of users. In contrast, the weighted proportional fairness adopted by PF-OBS allows users to indicate the priority of their traffic using the target price ω_r. In addition, PF-OBS is more efficient in terms of total throughput of the network. We define the total throughput as the sum of the rates of all flows entering the network.

To demonstrate this, let us consider again the linear topology network shown in Figure 5.6. Let x_r denote the rate of the flow from Ingress r to Egress r. As in section 5.9.1, assume that each link supports $N = 4$ wavelength channels of 1 Gbps each. Assume the load threshold is $\hat{C}_l = 3.2$ Gbps (80%). We use the same value of ω_r for all ingress nodes. In a max-min allocation such as the D-ABR scheme, it is obvious that each ingress node receives half the available bandwidth, i.e. $x_r = 1.6$ Gbps . We ran a simplified, high-level simulation of the PCwER scheme to assess its performance in the same scenario. The resulting rate allocations are shown in Table 5.8 and compared with PF-OBS.

We note that the load remains close to the desired threshold (3.2 Gbps) for all methods. However, the total throughput with PF-OBS is 24% higher (7.94 ± 0.02 Gbps) compared with the other schemes (6.4 Gbps).

	D-ABR (max-min)	PF-OBS	PCwER
Rate Ingress 0 (Gbps)	1.6	0.79 ± 0.007	1.60 ± 0.05
Rate Ingress 1 (Gbps)	1.6	2.38 ± 0.007	1.60 ± 0.04
Rate Ingress 2 (Gbps)	1.6	2.38 ± 0.007	1.64 ± 0.06
Rate Ingress 3 (Gbps)	1.6	2.38 ± 0.007	1.65 ± 0.06
Link L_1 load (Gbps)	3.2	$3.18 \pm 2\times10^{-4}$	3.20 ± 0.05
Link L_2 load (Gbps)	3.2	$3.18 \pm 1\times10^{-4}$	3.22 ± 0.03
Link L_3 load (Gbps)	3.2	$3.18 \pm 1\times10^{-4}$	3.22 ± 0.02

Table 5.8: Throughput of PF-OBS, D-ABR and PCwER

5.9.3 Disadvantages

PF-OBS works by gradually probing the network for available bandwidth. This introduces ramp up times and increases the time required to converge to optimality. The number of iterations required for convergence of PF-OBS is not fixed. Depending on initial conditions, it may take more than one hundred iterations for the load at the links to converge to the desired level, and longer for the allocations to reach proportional fairness (see Figure 5.15). However, the rate of change is faster when the load must be reduced, preventing a prolonged increase in the loss probability. In contrast, explicit rate schemes converge in a fixed number of iterations. For example, the ERICA scheme often converges to max-min fairness in one RTT [18]. The convergence speed of PF-OBS is made worse in the presence of propagation delays because of the required reduction in the control gain κ_r. PF-OBS trades off response speed in exchange of ease of implementation.

We note that the load threshold compensation discussed in section 4.4.3.2 and illustrated in Figure 5.20 is not perfect, especially with meshed networks. As shown in Appendix C, the average load on some links of the NSF network (e.g. switch8 → Edge8) slightly exceeds the desired limit, even after compensation. The discrepancy depends on the combination of ω_r used by users. Links carrying many flows are more affected by this

problem (e.g. switch8 → Edge8 carries 13 flows). Despite this, the maximum burst loss probability in all links observed during simulations remains tolerable.

Finally, we pointed out previously (see section 5.5.1.2) that there are more than one combination of price choice that result in the same bandwidth allocation. This poses a problem if one ingress node chooses to pay more to increase its share of bandwidth while, unaware of this, other sources are doing the same. Consequently, the rate allocations remain unchanged. For example, we saw in section 5.5.1.2 that the rate allocations in Scenarios A and B are the same although the prices paid by individual users are different (see Figure 5.14). The bandwidth prices might randomly drift when changes in the price choices are correlated. This can also happen due to deliberate action from users. Users may form "coalitions" to influence link prices as discussed in [128]. However, this is unlikely in the case of PF-OBS where prices do not imply monetary charges but are used to control network congestion.

Chapter 6
Conclusion

6.1 Conclusion

The goal of this thesis was to develop a rate-based congestion control method for OBS networks. An approach based on optimization theory was used. Kelly *et al.* developed the mathematical foundations of the optimization-based rate-control we used in this thesis. In this approach, each user has a utility function which measures the benefit it gains from using the network. On the other hand, network links incur a price when congested. The goal of rate control is then to maximize the overall utility minus the cost of congestion. When the utility functions are logarithmic, the resulting rate sharing is proportionally fair. Loosely speaking, proportional fairness means each user receives a bandwidth share proportional to the price it is willing to pay for a congested resource.

Based on this theoretical work, we developed a congestion control framework (PF-OBS) tailored for OBS as follows. The proposed method is an extension to the OBS control plane. PF-OBS works in a distributed manner. OBS ingress nodes regulate the transmission rate on each route independently. To control the congestion, a load threshold is defined for every link in the network. The load threshold is the load level at which the highest acceptable blocking probability is reached. It is usually slightly lower than the link capacity (e.g. 90%). OBS switches monitor the traffic load on all of their ports. In case of congestion, the switch marks the header of any burst that exceeds the threshold. The price of a link is defined as the fraction of traffic it marks. These marks are reported to each ingress node using special burst headers. In turn, ingress nodes periodically adjust

their traffic rates in function of the current price received from feedback and the price they are willing to pay.

We evaluated the performance of this method through simulations using OPNET Modeler. We found that the results obtained in simulations are in good agreement with those predicted by the underlying theory. The bandwidth allocation is proportionally fair as it is distributed according to the price paid by the user of each route (Figure 5.10). In simulations, our algorithm can keep the load on all links under the desired limit (Figure 5.9, Figure 5.29). As a result, blocking probability remains low in the network. We used greedy traffic sources in most simulations to keep the network under heavy load and to use all available bandwidth. When propagation delays are nonnegligible, the rate allocations can be kept stable thanks to the stability condition obtained from theory (Figure 5.23). The main problem with the method resides in the threshold compensation technique when applied to multihop paths. As a result, the actual load on congested links slightly exceeds the desired limit. However, we believe the small divergence is tolerable because it does not result in excessive added losses. In a 14-node meshed network, the overall blocking probability is equal to $4.8\times10^{-4} \pm 3.5\times10^{-5}$ when the load threshold is set at 90%. For the same setting, the maximum blocking probability of any link in the network is equal to $1.9\times10^{-3} \pm 2.6\times10^{-4}$. Both results are given with 99% degree of confidence.

The advantage of PF-OBS is clear when we consider the burst loss in the network. Without PF-OBS, the blocking probability always increases with the offered load. In a network with PF-OBS, the burst loss probability is bounded. In networks with multiple congested links, we showed that the optimization approach adopted in PF-OBS yields a higher total throughput when compared with existing flow control methods for OBS.

6.2 Contributions

Although optimization-based rate control has received much attention, most of the existing works are theoretical in nature. To the best of our knowledge, this thesis contributed the first application of this theory to develop a congestion control framework

for OBS. We argued that the principles behind proportionally fair rate control conform to the constraints imposed by OBS. We showed that this method can effectively control the congestion in OBS networks and therefore limit the blocking probability.

According to the theory, a mark may also correspond to a lost frame. However, we showed in this work that burst loss alone cannot be used as a congestion signal for OBS. There are two main reasons for this. First, burst losses occur randomly. It requires a long observation interval to get an accurate measurement of the blocking probability. This reduces the response speed of the control method. Second, when several flows are in contention, the OBS blocking probability is lower for larger streams. This results in an unfair bandwidth sharing. Although this is has been pointed out in one previous research, we provided more details on the problem and offered an explanation.

6.3 Future Work

Although we focused on size-based burstification in our simulations, PF-OBS does not exclude the use of time-based burst assembly. The rate adjustment rule fixes the amount of data that may be sent during a timeslot. For size-based burst assembly, burst sizes are nearly constant and the time between bursts is varied to meet the limit. If time-based burst assembly is used, the interval between bursts is constant and their sizes must be regulated. In this case, the rate controller must communicate the maximum burst size to the burst assembly module. Future OBS networks will presumably include a mix of various burst assembly methods. Further study of this solution is left as future work.

In the present work, we studied the performance of PF-OBS only within the boundaries of an OBS network and ignored its effects outside this limit (for example on TCP sessions). Given that TCP is the prevailing congestion control method on the Internet, the interactions between TCP and PF-OBS is of much interest to us. However, because of limitations in computing power, we cannot at this time simulate enough TCP flows to fill the available bandwidth.

In case of congestion on the OBS side, TCP connections crossing this network must reduce their congestion window quickly enough. Otherwise, the buffers at the ingress nodes will overflow. To prevent this, the mechanism we proposed in this thesis may require additional signalling to end users. A simple solution would be to transfer the marks from an OBS header packet to the appropriate field of disassembled IP packets at the egress node. A similar solution was proposed in [97] for ECN on Virtual Private Networks (VPN). A careful study of this solution, including simulation of TCP sessions crossing a PF-OBS network needs to be conducted.

Bibliography

[1] Bandwidth Demand Outpaces Price Declines," Telegeography, [Online document], (2005), available at: http://www.telegeography.com/press/releases/2005-04-07.php

[2] J. Glasner,"P2P Fuels Global Bandwidth Binge," in *Wired News*, [Online article], (2005), available at: http://www.wired.com/news/business/0,1367,67202,00.html

[3] Voice and Video Fuels Record Bandwidth Growth," in *MCI 360°*, [Online Newsletter], (2005), available at:

http://global.mci.com/ie/news/newsletters/360degrees/360deg_enl.xml?snlid=1761&scountry=ie&slang=en-ie

[4] A. S. Tanenbaum, *Computer networks*, 4th ed. Upper Saddle River, NJ: Prentice Hall, 2003.

[5] C. Qiao and M. Yoo, "Optical burst switching (OBS) - A new paradigm for an Optical Internet," *Journal of High Speed Networks,* vol. 8, pp. 69-84, 1999.

[6] J. S. Turner, "Terabit burst switching," *Journal of High Speed Networks,* vol. 8, pp. 3-16, 1999.

[7] L. Zhu, N. Ansari, and J. Liu, "Throughput of high-speed TCP in optical burst switching networks," *IEE Proceedings-Communications,* vol. 152, pp. 349-52, 2005.

[8] Z. Shao, T. Tong, J. J. Liao, Z. Li, Z. Wang, and A. Xu, "Performance analysis of TCP variants over time-space-labeled optical burst switched networks," in *4th International Conference on Networking Lecture Notes in Computer Science. Proceedings*, Reunion Island, France, 2005, pp. 149-155.

[9] R. Pleich, M. de Vega Rodrigo, and J. Gotz, "Performance of TCP over optical burst switching networks," in *31st European Conference on Optical Communication, 2005. ECOC 2005 Proceedings*, Glasgow, Scotland, 2005, pp. 883-884 vol.4.

[10] M. Casoni and M. L. Merani, "On the performance of TCP over optical burst switched networks with different QoS classes," *Lecture Notes in Computer Science. Proceedings*, Catania, Italy, 2005, pp. 574-585.

[11] C. Cameron, V. Hai Le, C. Jung Yul, S. Bilgrami, M. Zukerman, and K. Minho, "TCP over OBS - fixed-point load and loss," *Optics Express,* vol. 13, 2005.

[12] X. Yu, C. Qiao, and Y. Liu, "TCP implementations and false Time Out detection in OBS networks," in *IEEE INFOCOM Proceedings - IEEE INFOCOM. Proceedings*, Hongkong, China, 2004, pp. 774-784.

[13] D. U. Ping and A. B. E. Shunji, "TCP performance analysis of optical burst switching networks with a burst acknowledgment mechanism," *2004 Joint Conference of the 10th Asia-Pacific Conference on Communications and the 5th International Symposium on Multi-Dimensional Mobile Communications Proceedings, APCC/MDMC'04. Proceedings*, Beijing, China, 2004, pp. 621-625.

[14] S. Y. Wang, "Using TCP congestion control to improve the performances of optical burst switched networks," in *IEEE International Conference on Communications IEEE International Conference on Communications. Proceedings*, Anchorage, AK, United States, 2003, pp. 1438-1442.

[15] S. Gowda, R. K. Shenai, K. M. Sivalingam, and H. C. Cankaya, "Performance evaluation of TCP over optical burst-switched (OBS) WDM networks," in *Communications, 2003. ICC '03. IEEE International Conference on Proceedings*, 2003, pp. 1433-1437 vol.2.

[16] J. Luo, H. Zhang, Z. Zhang, S. Qiu, and J. Wang, "Dynamic TCP performance over optical burst-switched meshed networks," in *Network Architectures, Management, and Applications III Proceedings*, Shanghai, China, 2005, pp. 602245-10.

[17] S. Floyd and K. Fall, "Promoting the use of end-to-end congestion control in the Internet," *IEEE/ACM Transactions on Networking,* vol. 7, pp. 458-472, 1999.

[18] R. Jain, "Congestion control and traffic management in ATM networks: Recent advances and a survey," *Computer Networks and ISDN Systems,* vol. 28, pp. 1723-38, 1996.

[19] A. Maach, G. V. Bochman, and H. Mouftah, "Congestion Control and Contention Elimination in Optical Burst Switching," *Telecommunication Systems,* vol. 27, pp. 115-131, 2004.

[20] F. Farahmand, Z. Qiong, and J. P. Jue, "A closed-loop rate-based contention control for optical burst switched networks," in *GLOBECOM '05. IEEE Proceedings*, 2005, pp. 1989-1993.

[21] N. Akar and H. Boyraz, "Differentiated ABR: a new architecture for flow control and service differentiation in optical burst switched networks," *2005 Conference on Optical Network Design and Modelling. Proceedings*, Milan, Italy, 2005, pp. 11-18.

[22] R. J. Gibbens and F. P. Kelly, "Resource pricing and the evolution of congestion control," *Automatica,* vol. 35, pp. 1969-1985, 1999.

[23] F. P. Kelly, A. K. Maulloo, and D. K. H. Tan, "Rate control for communication networks: shadow prices, proportional fairness and stability," *Journal of the Operational Research Society,* vol. 49, pp. 237-52, 1998.

[24] F. Kelly, "Charging and Rate Control for Elastic Traffic," *European Transactions on Telecommunications,* vol. 8, pp. 33-37, 1997.

[25] O. Se-yoon, H. H. Ha, and K. Minho, "A Data Burst Assembly Algorithm in Optical Burst Switching Networks," *ETRI Journal,* vol. 24, pp. 311-322, 2002.

[26] J. Li, C. Qiao, and Y. Chen, "Recent progress in the scheduling algorithms in optical-burst-switched networks [Invited]," *Journal of Optical Networking,* vol. 3, pp. 229-241, 2004.

[27] V. M. Vokkarane, K. Haridoss, and J. P. Jue, "Threshold-based burst assembly policies for QoS support in optical burst-switched networks," *Proceedings of the SPIE - The International Society for Optical Engineering,* vol. 4874, pp. 125-36, 2002.

[28] A. Ge, F. Callegati, and L. S. Tamil, "On optical burst switching and self-similar traffic," *IEEE Communications Letters,* vol. 4, pp. 98-100, 2000.

[29] X. Yu, Y. Chen, and C. Qiao, "A study of traffic statistics of assembled burst traffic in optical burst switched networks," *Proceedings of SPIE - The International Society for Optical Engineering. Proceedings*, Boston, MA, United States, 2002, pp. 149-159.

[30] X. Cao, J. Li, Y. Chen, and C. Qiao, "Assembling TCP/IP packets in optical burst switched networks," *Conference Record / IEEE Global Telecommunications Conference. Proceedings*, Taipei, Taiwan, 2002, pp. 2808-2812.

[31] X. Yu, J. Li, X. Cao, Y. Chen, and C. Qiao, "Traffic statistics and performance evaluation in optical burst switched networks," *Journal of Lightwave Technology,* vol. 22, pp. 2722-2738, 2004.

[32] J. Choi, H. Le Vu, C. W. Cameron, M. Zukerman, and M. Kang, "The Effect of Burst Assembly on Performance of Optical Burst Switched Networks," in *Lecture Notes in Computer Science Proceedings*, 2004, pp. 729-739.

[33] O. Se-yoon and K. Mincho, "A burst assembly algorithm in optical burst switching networks," *Optical Fiber Communications Conference. (OFC). Postconference Technical Digest (IEEE Cat. No.02CH37339). Proceedings*, Anaheim, CA, USA, 2002, pp. 771-3.

[34] W. E. Leland, M. S. Taqqu, W. Willinger, and D. V. Wilson, "On the self-similar nature of Ethernet traffic (extended version)," *IEEE/ACM Transactions on Networking,* vol. 2, pp. 1-15, 1994.

[35] W. Willinger, M. S. Taqqu, W. E. Leland, and D. V. Wilson, "Self-Similarity in High-Speed Packet Traffic: Analysis and Modeling of Ethernet Traffic Measurements," *Statistical Science,* vol. 10, p. 67, 1995.

[36] M. Izal and J. Aracil, "On the influence of self-similarity on optical burst switching traffic," *Conference Record / IEEE Global Telecommunications Conference. Proceedings*, Taipei, Taiwan, 2002, pp. 2308-2312.

[37] J. Luo, Q. Zeng, Z. Zhang, and H. Zhao, "On the changes of self-similar traffic through optical burst-switched networks," *Journal of Optical Communications,* vol. 25, pp. 230-234, 2004.

[38] C. Qiao and M. Yoo, "Choices, features and issues in optical burst switching," *Optical Networks Magazine,* vol. 1, pp. 36-44, 2000.

[39] J. Y. Wei and R. I. McFarland Jr, "Just-in-time signaling for WDM optical burst switching networks," *Lightwave Technology, Journal of,* vol. 18, pp. 2019-2037, 2000.

[40] C. Qiao, "Labeled optical burst switching for IP-over-WDM integration," *IEEE Communications Magazine,* vol. 38, pp. 104-114, 2000.

[41] M. Yoo and C. Qiao, "Just-enough-time (JET): A high speed protocol for bursty traffic in optical networks," *LEOS Summer Topical Meeting. Proceedings*, Montreal, Can, 1997, pp. 26-27.

[42] M. Yoo and C. Qiao, "New optical burst switching protocol for supporting quality of service," *Proceedings of SPIE - The International Society for Optical Engineering. Proceedings*, Boston, MA, USA, 1998, pp. 396-405.

[43] C. Gauger, K. Dolzer, J. Späth, and S. Bodamer, "Service differentiation in optical burst switching networks," *ITG-Fachtagung Photonische Netze, March,* 2001.

[44] M. Yoo and Q. Chunming, "Supporting multiple classes of services in IP over WDM networks," *Seamless Interconnection for Universal Services. Global Telecommunications Conference. GLOBECOM'99. (Cat. No.99CH37042). Proceedings*, Rio de Janeireo, Brazil, 1999, pp. 1023-7.

[45] M. Yoo, C. Qiao, and S. Dixit, "Optical burst switching for service differentiation in the next-generation optical internet," *IEEE Communications Magazine,* vol. 39, pp. 98-104, 2001.

[46] C. Yang, M. Hamdi, and D. H. K. Tsang, "Proportional QoS over OBS networks," *Proceedings*, 2001, pp. 1510-1514 vol.3.

[47] K. Dolzer, "Assured Horizon-A New Combined Framework for Burst Assembly and Reservation in Optical Burst Switched Networks," *Proceedings of 7th*

European Conference on Networks and Optical Communications (NOC 2002),(Darmstadt, Germany), June, 2002.

[48] Q. Zhang, V. M. Vokkarane, J. P. Jue, and B. Chen, "Absolute QoS differentiation in optical burst-switched networks," *IEEE Journal on Selected Areas in Communications,* vol. 22, pp. 1781-1795, 2004.

[49] Z. Qiong, V. M. Vokkarane, C. Biao, and J. P. Jue, "Early drop scheme for providing absolute QoS differentiation in optical burst-switched networks," *HPSR 2003. Workshop on High Performance Switching and Routing (Cat. No.03TH8660). Proceedings*, Torino, Italy, 2003, pp. 153-7.

[50] G. Xin, I. L. J. Thng, J. Yuming, and H. Li, "Providing absolute QoS through virtual channel reservation in optical burst switching networks," *Computer Communications,* vol. 28, pp. 967-86, 2005.

[51] H. Overby and N. Stol, "Providing absolute QoS in asynchronous bufferless optical packet/burst switched networks with the adaptive preemptive drop policy," *Computer Communications,* vol. 28, pp. 1038-1049, 2005.

[52] H. C. Cankaya, S. Charcranoon, and T. S. El-Bawab, "A preemptive scheduling technique for OBS networks with service differentiation," *Global Telecommunications Conference, 2003. GLOBECOM'03. IEEE,* vol. 5, 2003.

[53] C. H. Loi, W. Liao, and D. N. Yang, "Service differentiation in optical burst switched networks," *Global Telecommunications Conference, 2002. GLOBECOM'02. IEEE,* vol. 3, pp. 2313-2317, 2002.

[54] V. M. Vokkarane and J. P. Jue, "Prioritized Burst Segmentation and Composite Burst-Assembly Techniques for QoS Support in Optical Burst-Switched Networks," *IEEE Journal on Selected Areas in Communications,* vol. 21, pp. 1198-1209, 2003.

[55] Y. Xiong, M. Vandenhoute, and H. C. Cankaya, "Design and analysis of optical burst-switched networks," in *All-Optical Networking 1999: Architecture, Control, and Management Issues Proceedings*, Boston, MA, USA, 1999, pp. 112-119.

[56] Y. Xiong, M. Vandenhoute, and H. C. Cankaya, "Control architecture in optical burst-switched WDM networks," *Selected Areas in Communications, IEEE Journal on,* vol. 18, pp. 1838-1851, 2000.

[57] J. Xu, C. Qiao, J. Li, and G. Xu, "Efficient channel scheduling algorithms in optical burst switched networks," *Proceedings - IEEE INFOCOM. Proceedings*, San Francisco, CA, United States, 2003, pp. 2268-2278.

[58] W. Xi, H. Morikawa, and T. Aoyama, "Priority-based wavelength assignment algorithm for burst switched photonic networks," *Proceedings*, 2002, pp. 765-767.

[59] J. Li and C. Qiao, "Schedule burst proactively for optical burst switched networks," *Computer Networks,* vol. 44, pp. 617-629, 2004.

[60] K. Dolzer, C. Gauger, J. Spath, and S. Bodamer, "Evaluation of reservation mechanisms for optical burst switching," *AEU - International Journal of Electronics and Communications,* vol. 55, pp. 18-26, 2001.

[61] V. Hai Le and M. Zukerman, "Blocking probability for priority classes in optical burst switching networks," *Communications Letters, IEEE,* vol. 6, pp. 214-216, 2002.

[62] M. Zukerman, E. W. M. Wong, Z. Rosberg, G. M. Lee, and H. Le Vu, "On Teletraffic Applications to OBS," *IEEE COMMUNICATIONS LETTERS,* vol. 8, 2004.

[63] H. Luo, G. Hu, and L. Li, "Performance evaluation of optical burst switching networks with limited number wavelength conversion," *Proceedings of SPIE - The International Society for Optical Engineering. Proceedings*, Wuhan, China, 2003, pp. 96-101.

[64] N. Akar and E. Karasan, "Exact calculation of blocking probabilities for bufferless optical burst switched links with partial wavelength conversion," *Broadband Networks, 2004. BroadNets 2004. Proceedings. First International Conference on,* pp. 110-117, 2004.

[65] C. F. Hsu, T. L. Liu, and N. F. Huang, "Performance analysis of deflection routing in optical burst-switched networks," *INFOCOM 2002. Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE,* vol. 1, pp. 66-73, 2002.

[66] V. M. Vokkarane, J. P. Jue, and S. Sitaraman, "Burst segmentation: An approach for reducing packet loss in optical burst switched networks," *IEEE International Conference on Communications. Proceedings*, New York, NY, 2002, pp. 2673-2677.

[67] V. M. Vokkarane, G. P. V. Thodime, V. U. B. Challagulla, and J. P. Jue, "Channel scheduling algorithms using burst segmentation and FDLs for optical burst-switched networks," *IEEE International Conference on Communications. Proceedings*, Anchorage, AK, United States, 2003, pp. 1443-1447.

[68] A. Detti, V. Eramo, and M. Listanti, "Performance evaluation of a new technique for IP support in a WDM optical network: optical composite burst switching(OCBS)," *Journal of Lightwave Technology,* vol. 20, pp. 154-165, 2002.

[69] A. Detti and M. Listanti, "Impact of segments aggregation on TCP reno flows in optical burst switching networks," *Proceedings - IEEE INFOCOM. Proceedings*, New York, NY, United States, 2002, pp. 1803-1812.

[70] Y. Tian, K. Xu, and N. Ansari, "TCP in Wireless Environments: Problems and Solutions," *IEEE Communication Magazine,* vol. 43, pp. 27-32, 2005.

[71] M. Mathis, J. Semke, J. Mahdavi, and T. Ott, "Macroscopic behavior of the TCP congestion avoidance algorithm," *Computer Communication Review,* vol. 27, pp. 67-82, 1997.

[72] X. Yu, C. Qiao, Y. Liu, and D. Towsley, "Performance evaluation of TCP implementations in OBS network," CSE Dept., SUNY, Buffalo, NY, Technical Report 2003-13, 2003.

[73] M. Mathis, J. Mahdavi, S. Floyd, and A. Romanow, "TCP Selective Acknowledgment Options," RFC 2018, October 1996.

[74] W. Stevens, "RFC2001: TCP Slow Start, Congestion Avoidance, Fast Retransmit, and Fast Recovery Algorithms," *Internet RFCs,* 1997.

[75] J. C. Hoe, "Start-up dynamics of TCP' s congestion control and avoidance schemes," Master's Thesis, Massachusetts Institute of Technology, Cambridge, MA, USA, 1995.

[76] J. Luo, Z. Zhang, S. Qiu, and J. Wang, "ROBS: A novel architecture of Reliable Optical Burst Switching with congestion control," *Proceedings of SPIE - The International Society for Optical Engineering. Proceedings*, Beijing, China, 2005, pp. 440-447.

[77] G. P. V. Thodime, V. M. Vokkarane, and J. P. Jue, "Dynamic Congestion-Based Load Balanced Routing in Optical Burst-Switched Networks," *Conference Record / IEEE Global Telecommunications Conference. Proceedings*, San Francisco, CA, United States, 2003, pp. 2628-2632.

[78] L. Yang and G. N. Rouskas, "Path switching in OBS networks," *Lecture Notes in Computer Science. Proceedings*, Waterloo, Ont., Canada, 2005, pp. 406-418.

[79] J. Li, G. Mohan, and K. C. Chua, "Load Balancing using Adaptive Alternate Routing in IP-over-WDM Optical Burst Switching Networks," *Proceedings of SPIE - The International Society for Optical Engineering. Proceedings*, Dallas, TX, United States, 2003, pp. 336-345.

[80] J. Teng and G. N. Rouskas, "Traffic engineering approach to path selection in optical burst switching networks," *Journal of Optical Networking,* vol. 4, pp. 759-776, 2005.

[81] A. Maach, G. V. Bochmann, and H. Mouftah, "Robust optical burst switching," *Networks 2004 - 11th International Telecommunications Network Strategy and Planning Symposium. Proceedings*, Vienna, Austria, 2004, pp. 447-452.

[82] R. Jain, S. Kalyanaraman, R. Goyal, S. Fahmy, and R. Viswanathan, "ERICA Switch Algorithm: A Complete Description," *ATM Forum Contribution,* pp. 96-1172, 1996.

[83] J. M. Jaffe, "Bottleneck Flow Control," *IEEE Transactions on Communications,* vol. COM-29, pp. 954-962, 1981.

[84] D. P. Bertsekas and R. G. Gallager, *Data networks*. Englewood Cliffs, N.J.: Prentice-Hall, 1987.

[85] R. Srikant, *The Mathematics of Internet Congestion Control*: Birkhauser, 2004.

[86] J. Y. Le Boudec,"Rate Adaptation, Congestion Control and Fairness: A Tutorial," [Online], (2005), available at:
http://ica1www.epfl.ch/PS_files/LEB3132.pdf

[87] S. Kunniyur and R. Srikant, "End-to-end congestion control schemes: utility functions, random losses and ECN marks," *Networking, IEEE/ACM Transactions on,* vol. 11, pp. 689-702, 2003.

[88] S. H. Low and D. E. Lapsley, "Optimization flow control - I: Basic algorithm and convergence," *IEEE/ACM Transactions on Networking,* vol. 7, pp. 861-874, 1999.

[89] J. Mo and J. Walrand, "Fair end-to-end window-based congestion control," *Networking, IEEE/ACM Transactions on,* vol. 8, pp. 556-567, 2000.

[90] S. Deb and R. Srikant, "Global Stability of Congestion Controllers for the Internet," *IEEE Transactions on Automatic Control,* vol. 48, p. 1055, 2003.

[91] S. Shakkottai, R. Srikant, and S. P. Meyn, "Bounds on the throughput of congestion controllers in the presence of feedback delay," *Networking, IEEE/ACM Transactions on,* vol. 11, pp. 972-981, 2003.

[92] L. Ying, G. E. Dullerud, and R. Srikant, "Global stability of internet congestion controllers with heterogeneous delays," *IEEE/ACM Transactions on Networking (TON),* vol. 14, pp. 579-591, 2006.

[93] L. Massoulie, "Stability of distributed congestion control with heterogeneous feedback delays," *Automatic Control, IEEE Transactions on,* vol. 47, pp. 895-902, 2002.

[94] L. Massoulie and J. Roberts, "Bandwidth sharing: objectives and algorithms," *INFOCOM'99. Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE,* vol. 3, 1999.

[95] D. P. Bertsekas, *Nonlinear programming*, 2nd ed. Belmont, Mass.: Athena Scientific, 1999.

[96] H. W. Kuhn and A. W. Tucker, "Nonlinear Programming," in *Proceedings of 2nd Berkeley Symposium Proceedings*, Berkeley, 1951, pp. 481-492.

[97] P. Key, D. McAuley, P. Barham, and K. Laevens, "Congestion Pricing for Congestion Avoidance," Microsoft Research, Technical Report MSR-TR-99-15. [Online]. Available at: http://research.microsoft.com/network/publications/msrtr99_15.pdf, , February 1999.

[98] F. Paganini, Z. Wang, J. C. Doyle, and S. H. Low, "Congestion control for high performance, stability, and fairness in general networks," *IEEE/ACM Transactions on Networking,* vol. 13, pp. 43-56, 2005.

[99] L. Kang-Won, K. Tae-eun, and V. Bharghavan, "A comparison of two popular end-to-end congestion control algorithms: the case of AIMD and AIPD," *Proceedings*, 2001, pp. 1580-1584 vol.3.

[100] S. Floyd and K. Ramakrishnan, "A proposal to add Explicit Congestion Notification to IP," *RFC2481, Jan,* 1999.

[101] D.-M. Chiu and R. Jain, "Analysis of the increase and decrease algorithms for congestion avoidance in computer networks," *Computer Networks and ISDN Systems,* vol. 17, pp. 1-14, 1989.

[102] R. Johari and D. K. H. Tan, "End-to-end congestion control for the internet: delays and stability," *IEEE/ACM Transactions on Networking (TON),* vol. 9, pp. 818-832, 2001.

[103] G. Vinnicombe, "Robust congestion control for the Internet," *submitted to SIGCOMM,* 2002.

[104] F. Kelly, "Fairness and stability of end-to-end congestion control," *European Journal of Control,* vol. 9, pp. 159–176, 2003.

[105] F. Paganini, J. Doyle, and S. Low, "Scalable laws for stable network congestion control," *Decision and Control, 2001. Proceedings of the 40th IEEE Conference on,* vol. 1, pp. 185-190, 2001.

[106] W. Zhikui and F. Paganini, "Global stability with time-delay in network congestion control," *Proceedings*, 2002, pp. 3632-3637 vol.4.

[107] F. Paganini, "A global stability result in network flow control," *Systems and Control Letters,* vol. 46, pp. 165-172, 2002.

[108] F. Paganini, Z. Wang, S. H. Low, and J. C. Doyle, "A new TCP/AQM for stable operation in fast networks," *Proceedings*, 2003, pp. 96-105 vol.1.

[109] Z. Wang and F. Paganini, "Global stability with time-delay of a primal-dual congestion control," *Proceedings*, 2003, pp. 3671-3676 vol.4.

[110] F. Paganini, J. Doyle, and S. H. Low, "A Control Theoretical Look at Internet Congestion Control," *The Mohammed Dahleh Symposium,* 2002.

[111] S. H. Low, F. Paganini, J. Wang, S. Adlakha, and J. C. Doyle, "Dynamics of TCP/RED and a scalable control," *Proceedings of IEEE INFOCOM,* pp. 239–248, 2002.

[112] S. H. Low, F. Paganini, J. Wang, and J. C. Doyle, "Linear stability of TCP/RED and a scalable control," *Computer Networks,* vol. 43, pp. 633-647, 2003.

[113] D. Lapsley and S. Low, "An IP implementation of optimization flow control," *Proceedings*, 1998, pp. 3023-3028 vol.5.

[114] G. Vinnicombe, "On the Stability of End-to-end Congestion Control for the Internet," University of Cambridge Tech Report CUED/F-INFENG/TR. 398, December 2000.

[115] Z. Jing, W. Shuang, Z. Keyao, D. Datta, K. Young-Chon, and B. Mukherjee, "Pre-planned global rerouting for fault management in labeled optical burst-switched WDM networks," *Proceedings*, 2004, pp. 2004-2008.

[116] D. M. Shan, G. Mohan, and K. C. Chua, "Offline wavelength assignment in labelled optical burst switching networks," *Proceedings*, 2005, pp. 467-471.

[117] K. Ramakrishnan, S. Floyd, and D. Black, "The Addition of Explicit Congestion Notification (ECN) to IP," RFC 3168, September 2001.

[118] V. Jacobson, "Congestion avoidance and control," *Computer Communication Review,* vol. 18, pp. 314-29, 1988.

[119] S. Floyd, "Random early detection gateways for congestion avoidance," *IEEE/ACM Transactions on Networking (TON),* vol. 1, pp. 397-413, 1993.

[120] M. S. Taqqu, W. Willinger, and R. Sherman, "Proof of a Fundamental Result in Self-Similar Traffic Modeling," *Computer Communication Review,* vol. 27, pp. 5-23, 1997.

[121] H. Zheng, Y. Zhao, and C. Chen, "AOS: Adaptive offset time scheduling for TCP fairness in optical burst-switched network," *Proceedings of SPIE - The International Society for Optical Engineering. Proceedings*, Shanghai, China, 2005.

[122] S. Verma, H. Chaskar, and R. Ravikanth, "Optical burst switching: a viable solution for terabit IP backbone," *IEEE Network,* vol. 14, pp. 48-53, 2000.

[123] Opnet Technologies Inc., OPNET Modeler, http://www.opnet.com.

[124] B. Zhou, M. Bassiouni, and G. Li, "Improving fairness in optical-burst-switching networks," *Journal of Optical Networking,* vol. 3, pp. 214-228, 2004.

[125] S. Junghans and C. M. Gauger, "Resource reservation in optical burst switching: architectures and realizations for reservation modules," *Proceedings of SPIE OptiComm,* pp. 409-413.

[126] M. Xiaohua and K. Geng-Sheng, "Optical switching technology comparison: optical MEMS vs. other technologies," *Communications Magazine, IEEE,* vol. 41, pp. S16-S23, 2003.

[127] Agilent_Technologies,"How can I generate traffic representative of 'Realistic' internet traffic?," in *Insight*, [Online article], (Aug. 2001), available at: http://advanced.comms.agilent.com/insight/2001-08/Questions/traffic_gen.htm

[128] J. E. Carroll and P. A. Kirkby, "Proportionally fair pricing: dynamics, stability and pathology," *Communications, IEE Proceedings-,* vol. 147, pp. 23-31, 2000.

[129] R. A. Horn and C. R. Johnson, *Matrix analysis*. Cambridge [Cambridgeshire] ; New York: Cambridge University Press, 1985.

APPENDIX A: CONVEX OPTIMIZATION

This appendix is intended as a brief introduction into convex optimization theory. Our goal is to present the reader with the essentials in order to follow the discussion in Chapter 3. The material presented here is taken from section 2.3 of [85].

Definition A.1 *A set* $C \subset \mathbf{R}^n$ *is said to be convex if the following property holds:*

$$\alpha\mathbf{x} + (1-\alpha)\mathbf{y} \in C, \qquad \forall \mathbf{x}, \mathbf{y} \in C, \alpha \in [0,1] \tag{A.1}$$

This means that for any pair of points in a convex set, all the points on the segment joining the two points also lie within the set. This is illustrated in Figure A.1.

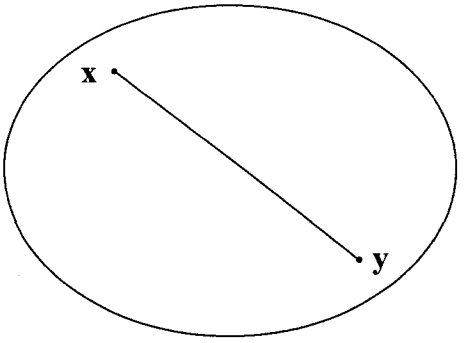

Figure A.1: A convex set

Definition A.2 *Consider the function f:* $C \to \mathbf{R}$, *where* $C \subset \mathbf{R}^n$ *is a convex set. The function f is said to be convex if:*

$$f(\alpha\mathbf{x} + (1-\alpha)\mathbf{y}) \le \alpha f(\mathbf{x}) + (1-\alpha) f(\mathbf{y}), \qquad \forall \mathbf{x}, \mathbf{y} \in C, \qquad \forall \alpha \in [0,1] \tag{A.2}$$

The function is said to be strictly convex if the above inequality is strict when $\mathbf{x} \neq \mathbf{y}$ *and* $\alpha \in (0,1)$.

Definition A.3 *Consider the function f:* $C \to \mathbf{R}$, *where* $C \subset \mathbf{R}^n$ *is a convex set. The function f is said to be concave if –f is convex, or equivalently:*

$$f(\alpha \mathbf{x} + (1-\alpha)\mathbf{y}) \geq \alpha f(\mathbf{x}) + (1-\alpha) f(\mathbf{y}), \qquad \forall \mathbf{x}, \mathbf{y} \in C, \qquad \forall \alpha \in [0,1] \qquad \text{(A.3)}$$

The function is said to be strictly concave if -f is strictly convex, or equivalently, if the above inequality is strict when $\mathbf{x} \neq \mathbf{y}$ *and* $\alpha \in (0,1)$.

Examples of convex and concave functions are shown in Figure A.2.

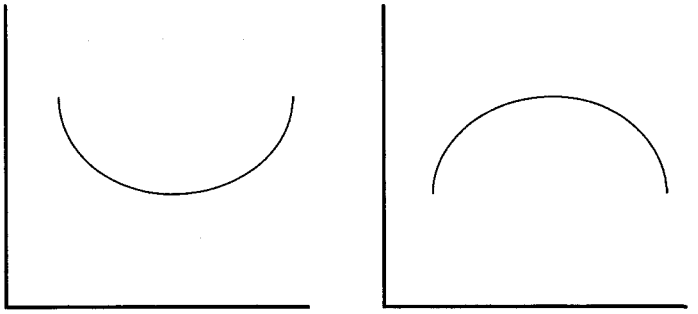

Figure A.2: Example of convex and concave functions (a) Convex function (b) Concave function

For a convex function, if a line joins any two points on the curve of the function, then all points on the line are located above the curve of the function. For a concave function, all points on the line are located below the curve of the function.

We now consider the maximization of concave functions over a convex set.

Lemma A.1 *Consider a concave function f(x) defined over a convex set* $C \subset \mathbf{R}^n$. *The following statements are true:*

- *If* $\hat{\mathbf{x}}$ *is a local maximum of f(x) over C, then it is also a global maximum over C. If f(x) is strictly concave, then* $\hat{\mathbf{x}}$ *is also the unique global maximum over C.*
- *If C is compact (closed and bounded), then a global maximum exists over C.*

The type of problem that we consider in this thesis is one where the objective function is strictly concave and the constraints are linear. Consider the following problem:

$$\max_{\mathbf{x}} f(\mathbf{x}) \tag{A.4}$$

subject to:

$$R\mathbf{x} \leq \mathbf{c},$$
$$H\mathbf{x} = 0$$

where $f(.)$ is a differentiable concave function and R and H are matrices of appropriate dimensions. It is easy to verify that the feasible set, i.e., the values of $\mathbf{x}$ that satisfy the constraints, is convex. The first step in solving such a problem is to form the Lagrangian function given by:

$$L(\mathbf{x}, \lambda, \mu) = f(\mathbf{x}) - \lambda^T (R\mathbf{x} - \mathbf{c}) - \mu^T H\mathbf{x} \tag{A.5}$$

where $\lambda \leq 0$ is called the Lagrange multiplier. The Karush-Kuhn-Tucker theorem states that, if $\mathbf{x}$ is a solution to (A.4), then it has to satisfy the following conditions:

$$\begin{aligned} \frac{\partial L}{\partial \mathbf{x}} = 0 \Rightarrow \nabla f(\mathbf{x}) - R^T\lambda - H^T\mu = 0, \\ \lambda^T (R\mathbf{x} - \mathbf{c}) = 0, \\ R\mathbf{x} \leq \mathbf{c}, \\ H\mathbf{x} = 0, \\ \lambda \geq 0 \end{aligned} \tag{A.6}$$

Since the objective function is concave and the feasible set is convex, the previous conditions are also sufficient for optimality. In other words, if $\mathbf{x}$ satisfies (A.6), then it solves (A.4).

The final topic introduced in this appendix is duality. Consider the function

$$D(\lambda, \mu) = \max_{\mathbf{x} \in C} L(\mathbf{x}, \lambda, \mu) \tag{A.7}$$

where C is the set of all $\mathbf{x}$ that satisfy the constraints in (A.4). This is called the dual function. We note that:

$$\inf_{\lambda \geq 0, \mu} D(\lambda, \mu) \geq f(\mathbf{x}^*) \tag{A.8}$$

where $\mathbf{x}^*$ is the solution to (A.4). To see this, note that, if $\mathbf{x}$ satisfies the constraints in (A.4), then:

$$f(\mathbf{x}) \leq f(\mathbf{x}) - \lambda^T (R\mathbf{x} - \mathbf{c}) - \mu^T H\mathbf{x}$$

Thus,

$$\max_{\mathbf{x} \in C} f(\mathbf{x}) \leq \max_{\mathbf{x} \in C} \left[f(\mathbf{x}) - \lambda^T (R\mathbf{x} - \mathbf{c}) - \mu^T H\mathbf{x} \right]$$

and further

$$\max_{\mathbf{x} \in C} f(\mathbf{x}) \leq \inf_{\lambda \geq 0, \mu} \max_{\mathbf{x} \in C} \left[f(\mathbf{x}) - \lambda^T (R\mathbf{x} - \mathbf{c}) - \mu^T H\mathbf{x} \right] = \inf_{\lambda \geq 0, \mu} D(\lambda, \mu)$$

this result is called the *weak duality* theorem. Under certain conditions called the *Slater constraint qualification* conditions, we further have:

$$\inf_{\lambda \geq 0, \mu} D(\lambda, \mu) = f(\mathbf{x}^*)$$

i.e., the optimal primal and dual objectives are equal. The Slater constraint qualification conditions are always satisfied for a problem with a concave objective function and linear constraints. Thus, the optimal primal and dual objectives are equal for the problem in (A.4).

APPENDIX B: PROOF OF EQUATION (3.42)

B.1. Linearization of equation (3.39)

Before presenting the actual proof of equation 3.42, we explain how the derivation for the linearization of equation 3.39 is obtained. We reproduce equation 3.39 here for convenience:

$$f(u,v) = x[n+1] = x[n] + \kappa\left(\omega - x[n-D]p\left(x[n-D]\right)\right)$$

where $u = x[n]$ and $v = x[n-D]$

Therefore:

$$f(u,v) = u + \kappa\left(\omega - vp\left(v\right)\right) \tag{A.1}$$

The equilibrium is $x^* = f(u_0, v_0)$, which means that $x^* = u_0 = v_0$

Let $x[n] = x^* + y[n]$, therefore:

$$\begin{aligned} y[n] &= x[n] - x^* = u - u_0 \\ y[n-D] &= x[n-D] - x^* = v - v_0 \end{aligned} \tag{A.2}$$

$f(u,v)$ can be linearized as follows:

$$f(u,v) = f(u_0,v_0) + \frac{\partial f(u_0,v_0)}{\partial u}(u-u_0) + \frac{\partial f(u_0,v_0)}{\partial v}(v-v_0)$$

Replacing $f(u,v)$ with $u + \kappa\left(\omega - vp\left(v\right)\right)$, we obtain:

$$f(u,v) = f(u_0,v_0) + (u-u_0) - \kappa\left(p\left(v_0\right) + v_0 p'\left(v_0\right)\right)(v-v_0)$$

Therefore:

$$y[n+1]+x^* = x^* + y[n]-\kappa\left(p(v_0)+v_0p'(v_0)\right)y[n-D]$$
$$\Rightarrow \quad y[n+1]=y[n]-\kappa\left(p(v_0)+x^*p'(v_0)\right)y[n-D]$$

The last result is the same as the linearized equation (3.40).

B.2. Proof of equation (3.42)

The actual proof for equation (3.42) begins here. The material presented in this section is taken entirely from the work of Johari *et al.* [95] and is reproduced here only for convenience.

We remind the reader that the stability condition described by equation (3.42) only applies to a network with a single link and a single user. The system described by equation (3.39) is locally stable if all the roots of the following characteristic equation have modulus less than one.

$$\lambda^{D+1}-\lambda^D+k(p+x^*p')=0 \qquad \text{(A.3)}$$

Lemma A.2 *Let* $p(\lambda,\kappa)$ *be a polynomial in* λ, *with coefficients which are continuous functions of* κ *(where* κ *may be vector valued). Then the maximum modulus of the roots* λ *of* $p(\lambda,\kappa)=0$ *is continuous in* κ.

Proof: The roots of any polynomial are continuous functions of the coefficients [125]. If the coefficients are continuous functions of κ, this immediately implies the conclusion of the lemma.

Lemma A.3 *For sufficiently small* κ, *all roots of (A.3) have modulus less than unity.*

Proof: We define the polynomial $p(\lambda,\kappa)$ by

$$p(\lambda,\kappa)=\lambda^{D+1}-\lambda^D+ka=0$$

where $a = p + x^* p' > 0$. Then the characteristic equation (A.3) is $p(\lambda, \kappa) = 0$. When $\kappa = 0$, the roots are $\lambda = 0$ with multiplicity D and $\lambda = 1$ with multiplicity 1. By direct computation, we find that $\partial p / \partial \lambda$ is nonzero at $\lambda = 1$, $\kappa = 0$. Thus, we can apply the Implicit Function Theorem to find an open interval $(-\varepsilon, \varepsilon)$ and a differentiable complex-valued function $g(\kappa)$ such that $g(0) = 1$, and $\lambda = g(\kappa)$ satisfies $p(\lambda, \kappa) = 0$ for $-\varepsilon < \kappa < \varepsilon$. For such κ, differentiating $p(\lambda, \kappa) = 0$ with respect to κ yields

$$(D+1)g(\kappa)^D g'(\kappa) - Dg(\kappa)^{D-1} g'(\kappa) + a = 0$$

Evaluating this through at $\kappa = 0$ yields $g'(0) = -a$. We take a Taylor series expansion of $g(\kappa)$ around $\kappa = 0$:

$$g(\kappa) = 1 - \kappa a + o(\kappa)$$

Thus, as increases away from zero, the root $\lambda = 1$ initially moves approximately as $1 - \kappa a$, and hence for sufficiently small κ this root will have modulus less than unity. For the root $\lambda = 0$ of multiplicity D, we note that by taking even smaller κ if necessary, we can appeal to Lemma A.2 to ensure that these D roots remain of modulus less than unity as κ increases away from zero. Thus, for sufficiently small κ, (A.3) has all roots of modulus less than unity.

Proof of equation (3.42): We can easily check the result for $D = 0$ since the linearized system becomes $y[n+1] = \left(1 - \kappa\left(p + x^* p'\right)\right) y[n]$; so assume $D > 0$ for the rest of the proof.

By Lemma A.2, the maximum modulus of the roots of (A.3) varies continuously with κ. Also, we know that the system is locally stable for small κ from Lemma A.3. Hence, for the first part of the result, it suffices to look for the smallest κ such that (A.3) has a root of unit modulus. Let the root be $\lambda = e^{2i\theta}$; then (A.3) may be rewritten as

$$2 \sin\theta e^{i((2D+1)\theta - \pi/2)} = a$$

where $a = \kappa(p + x^* p')$. Hence we conclude that

$$2|\sin\theta| = a, \qquad \theta = \frac{\pi}{2(2D+1)} + \frac{2\pi n}{2D+1}$$

where n is an integer. We choose $n = 0$ since we are looking for the smallest positive a such that λ is of unit modulus. Hence, substituting for θ in $2|\sin\theta| = a$, we see that there are no solutions for θ if $a < 2\sin(\pi/(2(2D+1)))$. This gives the first part of the result.

For the second part of the result, it suffices to prove that there exists a root $\lambda = re^{2i\theta}$ of modulus $r > 1$ to (A.3) when $a > 2\sin(\theta^*/2)$, where $\theta^* = \pi/(2D+1)$. From (A.3), note that $\lambda = 1 - a\lambda^{-D}$, and that as vectors in the complex plane, the angle between λ and 1 is θ. Thus, applying the cosine rule, we have

$$\cos\theta = (1 + r^2 - a^2 r^{-2D})/(2r) \tag{A.4}$$

Further, by equating imaginary parts in (A.3), we have

$$r = \sin(D\theta)/\sin((D+1)\theta) \tag{A.5}$$

We know that for $a = 2\sin(\theta^*/2)$, $\cos\theta^* = 1 - (a^2/2)$ and $r = 1$. Suppose $a > 2\sin(\theta^*/2)$. Then the right-hand side of (A.4) is less than $\cos\theta^*$ for $r = 1$. Notice that the left-hand side of (A.4) decreases from $\cos\theta^*$ to $\cos(\pi/(D+1))$ for $\theta \in [\theta^*, \pi/(D+1)]$. To prove that a root of modulus greater than 1 exists, it suffices to show that r increases from 1 to ∞ for $\theta \in [\theta^*, \pi/(D+1)]$ and that the right-hand side of (A.4) is increasing in r for $r \geq 1$. From (A.5), we may conclude $dr/d\theta > 0$ for $\theta \in [\theta^*, \pi/(D+1)]$, since $\sin(D\theta) > \sin((D+1)\theta)$ and $\cos(D\theta) - \cos((D+1)\theta) > 0$. So r increases from 1 to ∞ for $\theta \in [\theta^*, \pi/(D+1)]$. Further, differentiating the right-hand side of (A.4) with respect to r shows that the right-hand side of (A.4) is increasing in r for $r \geq 1$, tending to $r/2$ when r is large. Thus, a root of modulus greater than unity exists for which $\theta \in [\theta^*, \pi/(D+1)]$ when $a > 2\sin(\theta^*/2)$. This proves second part of the result.

APPENDIX C: NSF NETWORK SIMULATION RESULTS

This appendix provides detailed results obtained from simulations of the NSF network described in section 5.8.

The following table provides simulations results for each link of the NSF network studied in Chapter 5. The results provided are from 10 independent replications. 99% confidence intervals are also given. Although the traffic pattern is not the same in all replications, we note on Table C.1 that some links are constantly congested while others tend to be underloaded.

Link	Average load (Gbps)	Maximum load (Gbps)	Blocking probability
switch0 → Edge0	57.5 ± 0.33	59.2 ± 0.16	$1.10\times10^{-3} \pm 2.97\times10^{-4}$
switch0 → switch1	26.6 ± 4.82	27.8 ± 4.97	0
switch0 → switch2	30.5 ± 5.8	32.0 ± 5.94	0
switch0 → switch7	56.1 ± 0.43	58.3 ± 0.22	$2.31\times10^{-4} \pm 1.21\times10^{-4}$
switch1 → Edge1	57.8 ± 0.15	59.3 ± 0.15	$1.23\times10^{-3} \pm 2.22\times10^{-4}$
switch1 → switch0	37.6 ± 3.77	39.1 ± 3.93	0
switch1 → switch2	25.2 ± 3.92	29.0 ± 3.76	0
switch1 → switch3	39.5 ± 4.28	41.1 ± 4.36	0
switch10 → Edge10	58 ± 0.28	59.3 ± 0.23	$1.25\times10^{-3} \pm 4.28\times10^{-4}$
switch10 → switch12	33.8 ± 4.65	39.9 ± 4.94	0
switch10 → switch13	41.8 ± 3.81	43.8 ± 4.11	0
switch10 → switch7	57.2 ± 0.32	58.5 ± 0.31	$6.77\times10^{-4} \pm 3.69\times10^{-4}$
switch10 → switch9	25.7 ± 4	33.3 ± 4.21	0
switch11 → Edge11	58.2 ± 0.22	59.6 ± 0.21	$1.65\times10^{-3} \pm 3.66\times10^{-4}$
switch11 → switch12	14.3 ± 4.12	15.8 ± 4.66	0
switch11 → switch13	30 ± 5.89	32.0 ± 5.89	0
switch11 → switch5	53.3 ± 4.2	55.5 ± 4.29	$2.07\times10^{-5} \pm 4.17\times10^{-5}$
switch12 → Edge12	58.1 ± 0.15	59.4 ± 0.15	$1.38\times10^{-3} \pm 1.85\times10^{-4}$
switch12 → switch10	19 ± 4.31	21.4 ± 4.84	0
switch12 → switch11	27.6 ± 4.99	33.4 ± 6.42	0
switch12 → switch8	22.3 ± 4.98	24.4 ± 5.32	0
switch13 → Edge13	58.1 ± 0.13	59.4 ± 0.11	$1.39\times10^{-3} \pm 2.15\times10^{-4}$

Link	Average load (Gbps)	Maximum load (Gbps)	Blocking probability
switch13 → switch10	49.7 ± 5.76	51.2 ± 5.83	$1.54\times10^{-5} \pm 3.35\times10^{-5}$
switch13 → switch11	14 ± 2.71	17.5 ± 4.31	0
switch13 → switch8	42.9 ± 4.2	44.9 ± 5.02	0
switch2 → Edge2	57.7 ± 0.23	59.3 ± 0.23	$1.16\times10^{-3} \pm 3.78\times10^{-4}$
switch2 → switch0	32.9 ± 5.66	34.4 ± 5.82	0
switch2 → switch1	27.8 ± 3.96	30.6 ± 4.76	0
switch2 → switch5	52.8 ± 4.15	55.2 ± 4.08	$1.73\times10^{-5} \pm 3.10\times10^{-5}$
switch3 → Edge3	57.8 ± 0.27	59.2 ± 0.18	$1.21\times10^{-3} \pm 4.18\times10^{-4}$
switch3 → switch1	48.1 ± 4.01	50.2 ± 4.13	$5.10\times10^{-9} \pm 1.66\times10^{-8}$
switch3 → switch4	49 ± 4.55	50.7 ± 4.64	$1.96\times10^{-6} \pm 6.37\times10^{-6}$
switch3 → switch8	55.4 ± 1.57	57.6 ± 1.21	$1.37\times10^{-4} \pm 1.05\times10^{-4}$
switch4 → Edge4	57.9 ± 0.27	59.3 ± 0.19	$1.31\times10^{-3} \pm 3.52\times10^{-4}$
switch4 → switch3	54.1 ± 4.79	55.9 ± 4.77	$1.05\times10^{-4} \pm 9.54\times10^{-5}$
switch4 → switch5	51.7 ± 5.37	53.5 ± 5.35	$2.73\times10^{-5} \pm 4.27\times10^{-5}$
switch4 → switch6	35.2 ± 5.18	37.0 ± 5.45	0
switch5 → Edge5	57.9 ± 0.24	59.4 ± 0.18	$1.43\times10^{-3} \pm 3.78\times10^{-4}$
switch5 → switch11	53.3 ± 4.31	55.5 ± 4.23	$2.85\times10^{-5} \pm 4.55\times10^{-5}$
switch5 → switch2	55.8 ± 1.25	57.9 ± 0.90	$1.49\times10^{-4} \pm 1.29\times10^{-4}$
switch5 → switch4	45.5 ± 5.69	47.2 ± 5.79	$7.70\times10^{-7} \pm 2.50\times10^{-6}$
switch5 → switch9	40 ± 3.82	41.5 ± 3.93	0
switch6 → Edge6	57.8 ± 0.21	59.2 ± 0.14	$1.19\times10^{-3} \pm 3.26\times10^{-4}$
switch6 → switch4	48.4 ± 7.3	50.1 ± 7.41	$3.58\times10^{-7} \pm 1.16\times10^{-6}$
switch6 → switch7	42.4 ± 8.29	44.1 ± 8.52	0
switch7 → Edge7	57.9 ± 0.26	59.1 ± 0.17	$9.89\times10^{-4} \pm 3.29\times10^{-4}$
switch7 → switch0	44.8 ± 5.87	47.0 ± 6.09	$5.13\times10^{-9} \pm 1.67\times10^{-8}$
switch7 → switch10	56.8 ± 0.27	58.2 ± 0.24	$2.87\times10^{-4} \pm 1.12\times10^{-4}$
switch7 → switch6	52.3 ± 3.82	54.0 ± 3.82	$3.78\times10^{-5} \pm 5.71\times10^{-5}$
switch8 → Edge8	58.1 ± 0.18	59.5 ± 0.13	$1.59\times10^{-3} \pm 3.78\times10^{-4}$
switch8 → switch12	23 ± 4.23	24.2 ± 4.49	0
switch8 → switch13	38.5 ± 4.43	39.9 ± 4.62	0
switch8 → switch3	56.4 ± 0.38	58.4 ± 0.23	$3.02\times10^{-4} \pm 1.13\times10^{-4}$
switch9 → Edge9	58 ± 0.27	59.5 ± 0.21	$1.65\times10^{-3} \pm 4.28\times10^{-4}$
switch9 → switch10	32.1 ± 3.26	33.4 ± 3.28	0
switch9 → switch5	37.7 ± 4.16	39.1 ± 4.33	0

Table C.1: Simulation results for each link of the 14-node OBS network

Table C.2 presents the results of 10 independent simulations of the NSF network. The second column contains the maximum load on a port. The port carrying the maximum load is not the same in all replications. However, it is always a port connecting an OBS switch to an edge node. The third column presents the overall blocking probability of the OBS network, computed from all the bursts injected into the network. The last column contains the maximum blocking probability observed in a link. This often corresponds to the link that has the highest traffic rate (second column).

	Load, most congested port (Gbps)	**Overall blocking probability**	**Maximum blocking probability for a link**
Replication 1	58.3	4.93×10^{-4}	2.29×10^{-3}
Replication 2	58.5	4.97×10^{-4}	2.16×10^{-3}
Replication 3	58.3	4.31×10^{-4}	1.80×10^{-3}
Replication 4	58.2	4.47×10^{-4}	1.79×10^{-3}
Replication 5	58.2	4.71×10^{-4}	1.70×10^{-3}
Replication 6	58.4	4.66×10^{-4}	1.96×10^{-3}
Replication 7	58.4	4.96×10^{-4}	2.10×10^{-3}
Replication 8	58.3	5.21×10^{-4}	1.96×10^{-3}
Replication 9	58.1	4.36×10^{-4}	1.59×10^{-3}
Replication 10	58.4	5.18×10^{-4}	2.41×10^{-3}
Mean and 99% confidence interval	58.3 ± 0.11	$4.76\times10^{-4} \pm 3.5\times10^{-5}$	$1.94\times10^{-3} \pm 2.6\times10^{-4}$

Table C.2: Traffic load and blocking results for 10 independent simulation replications of the NSF network