ALGORITHMS FOR QUALITY OF SERVICE PROVISIONING AND ENHANCEMENT IN OPTICAL BURST SWITCHED NETWORKS

PHUNG, MINH HOANG

(B. Eng. (Hons.))

A thesis submitted for the degree of

Doctor of Philosophy

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

NATIONAL UNIVERSITY OF SINGAPORE

2005

To my family

Acknowledgments

First of all, I would like to express my sincere thanks to my supervisors, A/Prof. Chua Kee Chaing, Dr. Mehul Motani, Dr. Wong Tung Chong and my unofficial supervisor, Dr. Mohan Gurusamy. Without their research guidance, this work would not have been possible. I am especially thankful to A/Prof. Chua, who cared about not just my research but also my personal well-being, and who guided me in every little nuances of academic life. He was a true mentor to me.

I would also like to thank the National University of Singapore and the Singapore Millennium Foundation for their generous financial support. It has helped me to lead a financially untroubled graduate life.

Finally and most importantly, I thank those closest to me, my parents who have endured incredible hardship to bring me up, my wife who understands me more than anyone else, and my daughter who brings me so much joy every day. Without them, I would not be who I am. This thesis is dedicated to them.

Contents

Summary

Quality of Service (QoS) support is becoming a crucial part of today's data networks. Due to the proliferation of real-time applications, network users have come to expect not only connectivity but also adequate network performance. At the same time, network operators would like to maximise network resource utilisation to increase profits. As Optical Burst-Switching (OBS) is generally regarded as the transport technology of choice in the Internet backbone in the medium term, it is important that QoS mechanisms be developed for OBS networks. In this thesis, we present several algorithms for provisioning and enhancing QoS in OBS networks at different operational levels, from link level to path and network levels.

We introduce an optimal channel scheduling algorithm called *Ordered Scheduling* to reduce burst loss probability at the link level. We propose two practical realisations for it, namely, Basic Ordered Scheduling and Enhanced Ordered Scheduling, that aim to minimise complexity and maximise burst loss performance, respectively. Several practical implementation issues such as timing, complexity and signalling overhead are also discussed.

At the path level, we develop an absolute QoS framework that can provide quantitative edge-to-edge loss probability guarantees for flows in OBS networks. The QoS framework comprises two parts. In the first part, a preemptive absolute QoS differentiation mechanism and a link-based admission control mechanism work together to provide per-hop loss probability threshold guarantees. Using these per-hop thresholds as building blocks, a signalling protocol in the second part coordinates the reservation along the edge-to-edge path to achieve quantitative edge-to-edge loss probability guarantees.

We investigate a phenomenon unique to OBS networks called the *streamline effect* and derive an analytical expression to accurately calculate the burst loss probability for a link. We incorporate this formula into the link cost function of a dynamic load balancing algorithm for absolute QoS traffic to improve network-wide loss performance.

The proposed solutions solve several QoS issues in Optical Burst-Switched networks, thereby making OBS more practical and deployable in the future.

Acronyms

ATM Asynchronous Transfer Mode

ABT-IT ATM Block Transfer with Immediate Transmission

DiffServ Differentiated Services

e2e edge-to-edge

FDL Fibre Delay Line

FEC Forward Equivalence Class

FIFO First In First Out

HDLC High Level Data Link Control Protocol

IP Internet Protocol

ITU-T International Telecommunication Union-Telecommunication Standardization Sector

JET Just-Enough-Time

JIT Just-In-Time

LAUC Latest Available Unscheduled Channel

LAUC-VF Latest Available Unused Channel with Void Filling

LDP Label Distribution Protocol

LIB Label Information Base

LSP Label Switched Path

MAN Metropolitan Access Network

Min-SV Minimum Starting Void

MPLS Multi-Protocol Label Switching

OADM Optical Add/Drop Multiplexer

OBS Optical Burst Switching

O-E-O opto-electronic-opto

OLT Optical Line Terminal

OPS Optical Packet Switching

OXC Optical Cross Connect

QoS Quality of Service

RAM Random Access Memory

WDM Wavelength Division Multiplexing

WFQ Weighted Fair Queueing

WR-OBS Wavelength-Routed Optical Burst Switching

List of Figures

Chapter 1

Introduction

1.1 Evolution of Optical Networks

Since the advent of the World Wide Web in 1990, the Internet has experienced tremendous growth. Everyday, more and more people turn to the Internet for their information, communication and entertainment needs. New types of applications and services such as web browsing, video conferencing, interactive online gaming continue to be created to satisfy those needs. They demand increasingly higher transmission capacity from the networks. This rapid expansion of the Internet is seriously testing the limits of the current computer and telecommunication networks. As a result, there is an immediate need for new high-capacity networks that are capable of supporting these growing bandwidth requirements.

Wavelength Division Multiplexing (WDM) [6] has emerged as a core transmission technology for next-generation Internet backbone networks. It provides enormous bandwidth at the physical layer with its ability to support hundreds of wavelength channels in a single fibre. Systems with transmission capacities of several Terabits per second have been reported [32]. In order to make efficient use of this raw bandwidth, efficient higher layer transport architectures and protocols are needed.

First-generation WDM systems, which are widely deployed in current backbone networks, comprise WDM point-to-point links. In these networks, routers are connected by high-bandwidth WDM links. At each router, all incoming Internet Protocol (IP) packets are converted from optics to electronics for processing. At

the output links, all outgoing packets are converted back from electronics to optics before being transmitted on outgoing fibres. Since the electronic processing speed is much lower than the optical transmission rate, the opto-electronic-opto (O-E-O) conversion of the entire traffic at every router creates significant overhead for the system, especially when most of the traffic is by-pass traffic.

Optical networking becomes possible with the arrival of three key optical network elements: Optical Line Terminal (OLT), Optical Add/Drop Multiplexer (OADM) and Optical Cross Connect (OXC) [67]. An OLT multiplexes multiple wavelengths into a single fibre and demultiplexes a composite optical signal that consists of multiple wavelengths from a single fibre into separate fibres. An OADM is a two-port device that takes in a composite optical signal that consists of multiple wavelengths and selectively drops (and subsequently adds) some of the wavelengths before letting the composite signal out of the output port. An OXC has multiple input and output ports. In addition to add/drop capability, it can also switch a wavelength from any input port to any output port. Both OADMs and OXCs may have wavelength conversion capability. These devices make it possible to switch data entirely in optical domain between a pair of source and destination.

Based on this optical routing capability, the next-generation optical Internet architecture is envisioned to have two main functional parts: an inner core network and multiple access networks [56]. The access networks, or Metropolitan Access Networks (MANs), are compatible with today's Internet transport architecture and are responsible for collecting IP traffic from end-users. They are built from electronic or lower-speed optical transport technologies such as Gigabit Ethernet or optical ring. The access networks are connected together by the inner core network through high-speed edge nodes. An ingress node aggregates traffic destined to the same egress node and forwards it through the core network. The core network consists of a mesh of reconfigurable optical switching network elements (e.g., OXCs and OADMs) interconnected by very high capacity long-haul optical links. To date, there are primarily three all-optical transport technologies proposed for

the optical core network, namely wavelength routing (circuit-switched), optical packet switching and optical burst switching. They are described below.

Wavelength routing [18], or optical circuit switching, is the first step towards an all-optical network. In this approach, dedicated WDM channels, or lightpaths, are established between a source and destination pair. The lightpath establishment may be static or dynamic. A lightpath is carried over a wavelength on each intermediate link and switched from one link to another at each intermediate node. If wavelength converters are present in the network, a lightpath may be converted from one wavelength to another along the route. Otherwise, it must utilise the same wavelength on all the links along the route. This property is known as the wavelength continuity constraint. A wavelength may be used by different lightpaths as long as they do not share any common link. This allows a wavelength to be reused spatially at different places in the network.

Although the wavelength routing approach is a significant improvement over the first generation point-to-point architectures, it has some limitations. Firstly, the number of connections in a network is usually much greater than the number of wavelengths and the transmission rate of a connection is much smaller than the capacity of a wavelength. Therefore, despite spatial reuse of wavelengths, it is neither possible nor efficient to allocate one wavelength to every connection. This problem can be alleviated by traffic grooming [54], which aggregates several connections into a lightpath. However, some connections must still take multiple lightpaths when there is no lightpath between a pair of source and destination. Such connections will have to undergo multiple O-E-O conversions and multiple crossings through the network, which increase network resource consumption and end-to-end delay. Furthermore, lightpaths are fairly static and fixed-bandwidth connections that may not be able to efficiently accommodate the highly variable and bursty Internet traffic.

Optical Packet Switching (OPS) [21,25,37,57] is an optical networking paradigm that performs packet switching in the optical domain. In this approach, optical

packets are sent along with their headers into the network without any prior reservation or setup. Upon reaching a core node, a packet will be optically buffered while its header is extracted and processed electronically. A connection between the input port and the output port is then set up for transmission of that optical packet and released immediately afterwards. As such, a link can be statistically shared among many connections at subwavelength level. OPS may have slotted/unslotted and synchronous/asynchronous variants.

The objective of OPS is to enable packet switching capabilities at rates comparable with those of optical links and thereby replacing wavelength routing in the next-generation optical networks. However, it faces several challenges involving optical technologies that are still immature and expensive. One such challenge is the lack of optical random access memory for buffering. Current optical buffers are realised by simple Fibre Delay Lines (FDLs) and not fully functional memories. Other required technologies that are still at a primitive stage include fast optical switching, optical synchronisation and the extraction of headers from optical packets.

Optical Burst Switching (OBS) [13, 65, 79, 85, 88] is a more recently proposed alternative to OPS. In OBS, the basic transport unit is a *burst*, which is assembled from several IP packets at ingress node. OBS also employs a one-pass reservation mechanism, whereby a burst header is sent first to reserve wavelengths and configure the switches along a path. The corresponding burst follows without waiting for an acknowledgment for the connection establishment. If a switch along the path cannot forward the burst due to congestion, the burst is simply dropped. This mechanism has its origin in an ITU-T standard for Asynchronous Transfer Mode (ATM) networks known as ATM Block Transfer with Immediate Transmission (ABT-IT) [42]. Other variants of ABT-IT include Ready-to-Go Virtual Circuit Protocol (RGVC) [80] and Tell-and-Go (TAG) [87]. The use of large bursts as the basic transport unit leads to lower switching frequency and overhead. Therefore, OBS nodes can use slower switching fabrics and processing electronics compared to OPS. The overhead reduction occurs in two places. Firstly, the

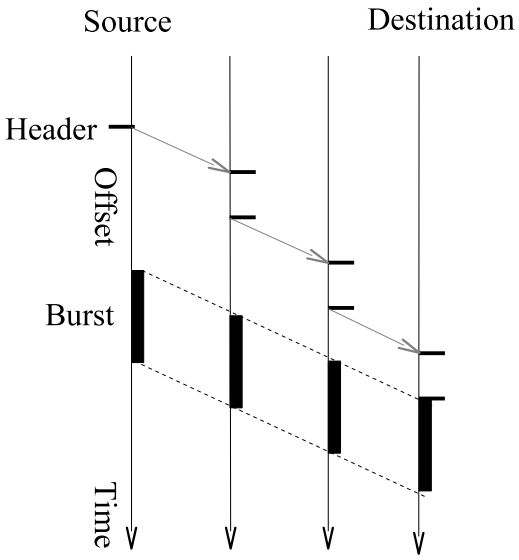


Figure 1.1: The use of offset time in OBS

header/payload ratio is reduced, leading to lower signalling overhead. Secondly, the ratio between guard intervals between bursts when a link is idle and the time it is transmitting is also reduced.

Another distinguishing feature of OBS is the separation between a header and its data burst in both time and space. In OBS, a burst is not sent immediately after the header, but delayed by a predetermined *offset* time. The offset time is chosen to be at least equal to the sum of the header processing delays at all intermediate nodes. This is to ensure that there is enough headroom for each node to complete the processing of the header before the burst arrives. The use of the offset time is illustrated in Figure 1.1. Moreover, headers are transmitted in dedicated control channels, which are separate from data channels. This separation permits electronic implementation of the signalling control path while maintaining a completely transparent optical data path for high speed data transmission. It also removes the need for optical buffering, optical synchronisation and optical header extraction techniques.

Table 1.1 summarises the three all-optical transport paradigms. From the table, one can observe that OBS has the advantages of both optical circuit switching and OPS, while avoiding their shortcomings.

Optical Switching Paradigm	**Link Util.**	**Setup Latency**	**Switching Speed Req.**	**Complexity**	**Traffic Adaptivity**
Optical Circuit Switching	Low	High	Slow	Low	Low
Optical Packet Switching	High	Low	Fast	High	High
Optical Burst Switching	High	Low	Medium	Medium	High

Table 1.1: Comparison of of the different optical networking paradigms.

1.2 Overview of Optical Burst Switching Architecture

Figure 1.2 shows a diagram of an OBS network. It comprises a meshed network of core nodes linked by WDM links. In the present literature, the core nodes are usually assumed to have full wavelength conversion capability [65,79,88]. That is, they can convert a burst from any input wavelength to any output wavelength. In addition, depending on the switch architecture and design choice, the core nodes may or may not be equipped with optical buffering, which is in the form of FDLs. However, FDLs only offer deterministic delay and cannot be considered as fully functional memory. Some core nodes also act as edge nodes, which means that they are connected to some access networks and accept IP input traffic as well as all-optical transit traffic. Depending on whether an edge node acts as a source or a destination, it may be called an ingress node or egress node, respectively.

Optical bursts are assembled from input IP traffic by ingress nodes before being sent over the OBS core network. When a burst is ready for transmission, the ingress node sends a header packet towards the egress node on a dedicated control channel to reserve wavelengths and configure switches at core nodes along the path. The data burst is transmitted all-optically after a certain offset time without waiting for acknowledgment. Bursts are disassembled back into IP packets at egress nodes and forwarded onto adjacent access networks.

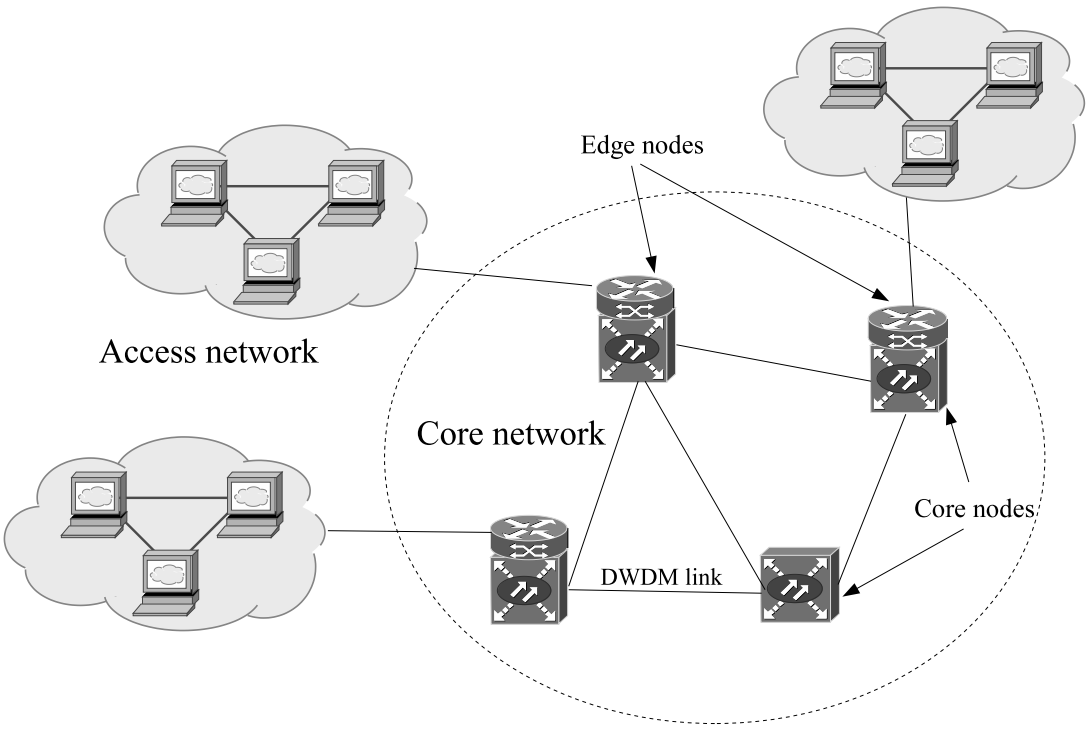


Figure 1.2: OBS network architecture

Due to the bufferless nature of OBS networks, burst loss is the main performance metric of interest. Any queueing and assembly delay is confined to edge nodes, making it easy to manage. The primary cause of burst loss is wavelength contention. This happens when the number of overlapping burst reservations at an output port of the core node exceeds the number of available data wavelengths. If the node has FDL buffers, it may delay the excess bursts and attempt to schedule them again. Otherwise, the excess burst will be dropped.

Some researchers propose a centralised version of OBS with two-way reservation for each burst, called Wavelength-Routed Optical Burst Switching (WR-OBS) [24]. In this proposal, before transmitting a burst, an ingress node must send a reservation message to a centralised server. For each reservation request, the server calculates the route from the ingress node to the egress node and reserves wavelengths at every link along the route for the burst. The burst is transmitted only after a successful acknowledgment message has been received. It is claimed that WR-OBS improves network throughput and includes explicit

Quality of Service (QoS) provisioning. However, the centralised nature of this scheme does not scale well and makes it unsuitable for large optical networks.

Most existing OBS proposals assume that a label switching framework such as Multi-Protocol Label Switching (MPLS) [69] is included. Reference [64] discusses methods and issues for integrating MPLS into OBS. Generally, this is done by running IP/MPLS software on every OBS core node. Each header is sent as an IP packet, carrying a label to identify the Forward Equivalence Class (FEC) it belongs to. Based on this assigned label, the core nodes route the header from source to destination, establishing the all-optical path or Label Switched Path (LSP) for the data burst that follows later. Label switching is the preferred routing method in OBS instead of hop-by-hop routing since its short label processing time per hop is particularly suitable for the high burst rate in OBS networks. Besides, label switching offers the possibility of explicit path selection, which enables traffic engineering.

To date, there have been three main OBS signalling proposals that differ mostly in the way wavelengths are reserved. In Just-In-Time (JIT) [85, 95], an output wavelength is reserved as soon as a header packet arrives at a node and released only after a release message is received. Terabit Burst Switching [79] works the same way except that burst length information is carried in header packets to enable automatic release of wavelengths. These two techniques are simple to implement. However, they do not utilise the channels during the period between the arrivals of a header and the corresponding burst, which may be inefficient. In Just-Enough-Time (JET) [65, 88], the time offset information is included in a header packet in addition to burst length information. This allows a node to reserve a wavelength for a burst just before its actual arrival. Therefore, in the period between the header and burst arrival epochs, the channel can be used to transmit other bursts. This can lead to significant improvement in burst loss performance if the offset times in the network are large [88]. Thus, JET is probably the most popular OBS signalling scheme. In the rest of the thesis, we will focus primarily on JET OBS.

1.3 Need for Quality of Service Support in Optical Burst Switched Networks and Challenges

Due to the extreme popularity and success of the Internet, there is great diversity in current Internet applications with very different requirements of network performance or Quality of Service (QoS). Non-interactive and semi-interactive applications such as email, web browsing can cope with a wide range of QoS. On the other hand, highly interactive application such as video conferencing, online gaming have very stringent operating requirements. In addition, not all users need the same level of QoS and wish to pay the same price for it. Some companies that rely on the Internet for critical transactions would be willing to pay high premiums to ensure network reliability. In contrast, casual home users only need cheap Internet access and can tolerate a lower service level. The central point of this discussion is that *some degree of controllability on the QoS provided to users is desirable*. This is to ensure that applications and users get the service level they need and at the same time, network service providers maximise their returns from the networks. We refer to such controllability of the QoS provided to users as QoS support.

In general, offering QoS support to end users, or end-to-end QoS provisioning, requires the participation of all network entities along end-to-end paths. This is because the network performance perceived by an end user is the cumulative result of the service received by the user's packets at network entities along the end-to-end path. For example, consider a particular application that requires an end-to-end packet loss probability of no more than 1%. If the packet loss probability at just one single router on the path becomes larger than 1%, the required end-to-end QoS cannot be achieved. This requirement implies that OBS networks, which are to form the backbone of the next-generation Internet, must have QoS support across ingress/egress pairs in order to realise end-to-end QoS support.

Closely related to QoS provisioning is the issue of network performance enhancement in general. To maximise profits, network operators would like to provide the required QoS levels with the least amount of network resources. Alternatively, they would like to provide QoS support for as many users as possible with a fixed amount of network resources. This applies for communication networks in general and OBS networks in particular. Therefore, *if QoS provisioning algorithms are important to network users, QoS enhancement algorithms are equally important to network operators.*

A solution used in wavelength-routed networks is to treat the optical connection between an ingress/egress pair as a virtual link. The ingress and egress nodes then become adjacent nodes and QoS mechanisms developed for IP networks can be directly applied. This approach works well for wavelength-routed networks because wavelengths are reserved exclusively. Therefore, there is no data loss on the transmission path between an ingress node and an egress node, which makes the connection's characteristics resemble those of a real optical link. On the other hand, wavelengths in OBS networks are statistically shared among many connections. Hence, there is a finite burst loss probability on the transmission path between an ingress node and an egress node, which renders this approach unusable for OBS networks.

Since OBS is also a datagram transport protocol as IP and there has been extensive QoS literature for IP networks, it is attractive to adapt IP-QoS solutions for use in OBS. However, there are unique features of OBS that must be considered in this process. In the following paragraphs, these differences will be discussed in detail.

A primary difference between OBS and IP networks is that there is no or minimal buffering inside OBS networks. Therefore, an OBS node must schedule bursts as they come. This poses a great challenge in adapting IP-QoS solutions for OBS because most of the QoS differentiation algorithms in IP networks rely on the ability of routers to select specific buffers or even specific packets to transmit next. It also makes it more difficult to accommodate high priority traffic classes

with very low loss probability thresholds. For example, if two overlapping high priority bursts attempt to reserve a single output wavelength, one of them will be dropped. This is unlike the situation in IP networks where one of them can be delayed in a buffer while the other is being transmitted. Furthermore, without buffering at core nodes, burst loss performance of a traffic class depends strongly on its burst characteristics. Bursts with long durations or short offset times are more likely to be dropped than others. Hence, it is difficult to have a consistent performance within one traffic class.

There are two other unique features of OBS networks. The first feature is that there is a time interval between a header and its data burst. This offset time gives a scheduler information about an arriving burst some time in advance, which can be utilised. Two of our proposals presented later in this thesis, Ordered Scheduling and Preemptive Differentiation, take advantage of this unique feature of OBS networks. The second feature is that a burst only occupies one wavelength when it is being transmitted instead of engaging the entire link. Thus, several bursts of different traffic classes can be transmitted simultaneously.

In summary, the diversity of Internet applications and users makes it desirable to have QoS support built into the Internet. In addition, from the network operators' point of view, QoS enhancement algorithms that maximise network performance are also important for economic reasons. As OBS is envisioned to be the optical transport architecture in the core of the Internet, it is imperative to develop QoS provisioning and enhancement algorithms for OBS networks. An attractive approach is to modify QoS solutions designed for IP networks for use in OBS networks. However, there are unique features of OBS networks that must be taken into consideration. These features present both challenges and opportunities for OBS-QoS algorithms.

1.4 Objectives and Contributions of the Thesis

In this thesis, we develop algorithms for QoS provisioning and performance enhancement in OBS networks at different levels of operation. Specifically, we de-

velop a burst scheduling algorithm to improve burst loss performance at the node level. At the connection level, we develop an edge-to-edge QoS framework to cater for specific QoS requirements of LSPs. Finally, on the network level, a load balancing algorithm is developed to distribute traffic more evenly across the network in order to reduce congestion at bottlenecks. These contributions are summarised below.

Since there are many wavelength channels per link in a WDM system, a key component in an OBS node is an efficient channel scheduling algorithm. The scheduling algorithm needs to pack as many arriving bursts onto the wavelengths as possible so as to reduce the number of bursts that are lost. We introduce an optimal scheduling approach, called *Ordered Scheduling*, for use with the JET OBS variant. There are two ways of implementing Ordered Scheduling, namely *Basic Ordered Scheduling* and *Enhanced Ordered Scheduling*. Enhanced Ordered Scheduling fully implements the Ordered Scheduling approach but is more complex. On the other hand, Basic Ordered Scheduling is simpler but has poorer performance. Through extensive simulations, it is shown that both implementations outperform a popular burst scheduling algorithm called Latest Available Unused Channel with Void Filling (LAUC-VF) [88]. The superiority is especially significant if offset times are large. We also analyze their complexity and discuss various implementation options.

Next, we propose an *absolute edge-to-edge QoS framework* for OBS networks. The framework employs a differentiation mechanism and a per-hop admission control mechanism to realise a set of quantitative per-hop loss guarantees. When an LSP with a certain edge-to-edge loss requirement needs to be established, the network decomposes the edge-to-edge loss requirement into a series of per-hop loss guarantees and assigns them to the intermediate nodes along the path. This approach avoids the problem of outdated traffic information in traditional QoS schemes with edge-based admission control. Using a small set of per-hop classes as building blocks for loss guarantees, it can offer a large number of possible edge-to-edge loss guarantees, and thereby achieve good scalability. The proposed

framework is evaluated through analysis and simulation and is shown to achieve reliable loss guarantees under various network scenarios.

Finally, we analyze a unique phenomenon in OBS networks called the *Streamline effect*, which is caused by the lack of buffering at core nodes. In certain scenarios, this effect causes the burst loss probability at a link to be significantly lower than that estimated by the traditional Erlang B formula. We then utilise this Streamline effect in designing a load balancing algorithm to better distribute traffic among different alternative paths between ingress/egress node pairs. The new load balancing algorithm is evaluated through simulation. It is shown to outperform shortest path routing and the load balancing versions that do not include the Streamline effect.

In summary, the proposed solutions solve several QoS issues in OBS networks, thereby making OBS more practical and deployable in the near future.

1.5 Thesis Organisation

This thesis consists of six chapters. This chapter has introduced OBS and provided the motivation for developing QoS provisioning and enhancement algorithms in OBS. It has also defined the research objectives. Chapter 2 provides a survey of the current literature on QoS issues in OBS, focusing on burst scheduling, QoS differentiation and load balancing. Ordered Scheduling and its two implementation versions are presented in Chapter 3. Chapter 4 introduces the absolute edge-to-edge QoS framework and its components, namely, differentiation, per-hop admission control and edge-to-edge signalling and reservation. Chapter 5 has two parts. In the first part, we study the Streamline effect and validate the analysis through simulation. In the second part, we propose a load balancing algorithm that utilises the Streamline effect for better performance. The thesis is summarised and suggestions for future works are given in Chapter 6.

Chapter 2

Quality of Service in Optical Burst Switching - A Survey

Since OBS is based on the same datagram transport model as IP, many of the general QoS approaches used in IP networks can also be used for OBS networks. However, due to certain unique features of OBS as discussed in §1.3, with the lack of buffering being the most notable, they require significantly modified or entirely novel algorithms to implement.

The objective of this chapter is to provide a detailed survey of the current QoS literature in OBS. Since our contributions in this thesis are on different operational levels in OBS networks, the literature survey is organised in the same way. We first start by examining proposals at the node level, i.e., contention resolution approaches and channel scheduling algorithms, in §2.1 and §2.2. We then move onto QoS provisioning proposals at the path level. QoS differentiation algorithms, which play a central role in any QoS framework, are reviewed in §2.3. These algorithms can be used to implement both relative and absolute QoS models. Since the absolute QoS model requires other mechanisms such as admission control and signalling in addition to QoS differentiation, it is discussed separately in more detail in §2.4. Finally, load balancing algorithms will be discussed in §2.5.

2.1 Contention Resolution Approaches

Since a wavelength channel may be shared by many connections in OBS networks, there exists the possibility that bursts may contend with one another at interme-

diate nodes. Contention occurs when multiple bursts from different input ports are destined for the same output port simultaneously. The general solution to burst contention is to move all but one burst "out of the way". An OBS node has three possible dimensions to move contending bursts, namely, time, space and wavelength dimensions. The corresponding contention resolution approaches are optical buffering, deflection routing and wavelength conversion, respectively. In addition, there is another approach unique to OBS called burst segmentation. Below we will examine each of these approaches.

2.1.1 Optical Buffering

Typically, contention resolution in traditional electronic packet switching networks is implemented by storing excess packets in Random Access Memory (RAM) buffers. However, RAM-like optical buffers are not yet available. Currently, optical buffers are constructed from Fibre Delay Lines (FDLs) [16,17,38,70]. An FDL is simply a length of fibre and hence offers a fixed delay. Once a packet/burst has entered it, it must emerge after a fixed length of time later. It is impossible to either remove the packet/burst from the FDL earlier or hold it in the FDL longer. The fundamental difficulty facing the designer of an optical packet/burst switch is to implement variable-length buffers from these fixed-length FDLs.

According to [38], optical buffers may be categorised in two fundamental ways. They can be classified as either single-stage, i.e., having only one block of parallel delay lines, or multi-stage, which have several blocks of delay lines cascaded together. Single-stage optical buffers are easier to control, but multi-stage implementation may lead to more savings on the amount of hardware used. A multi-stage architecture proposed in [39] achieves buffer depths of several thousands. Optical buffers can also be classified as having feed-forward or feedback configurations. In a feed-forward configuration, delay lines connect the output of a switching stage to the input of the next switching stage. In a feedback configuration, delay lines connect the output of a switching stage back to the input of the

same stage. Long holding time and certain degrees of variable delays can be easily implemented with a feedback configuration by varying the number of loops a packet/burst undergoes [45]. On the other hand, in a feed-forward configuration, delay lines with different lengths must be used to achieve variable delays [35, 39]. However, the disadvantage of a feedback configuration is that it degrades optical signal quality due to the many switching passes. Hybrid combinations of the above are also possible [101].

Based on the position of buffers, packet switches fall into one of three major categories: input buffering, output buffering and shared buffering. In input-buffered switches, a set of buffers is assigned for each input port. This configuration has poor performance due to the head-of-line blocking problem. Consequently, it is never proposed for purely optical implementation. In output-buffered switches, a set of buffers is assigned to each output port. Most optical switches emulate output buffering since the delay in each output optical buffer can be determined before the packet/burst enters it. Shared buffering is similar to output buffering except that all output ports share a common pool of buffers. In optical switches, this can significantly reduce the number of FDLs required.

Despite the considerable research efforts on FDL-based optical buffers, there remain certain problems that limit their effectiveness. Firstly, by their nature, they can only offer discrete delays. The use of recirculating delay lines can give finer delay granularity but it also degrades optical signal quality. Secondly, the size of FDL buffers is severely limited not only by signal quality concerns but also by physical space limitations. A delay of 1 ms requires over 200 km of fibre. Due to the size limitations of buffers, a switch may be unable to effectively handle high load or bursty traffic conditions.

2.1.2 Deflection Routing

Deflection routing is a contention resolution approach ideally suited for photonic networks that have little buffering capacity at each node. If no buffer is provided at all, it is also referred to as hot-potato routing. In this approach, if the intended

output port is busy, a burst/packet is routed (or deflected) to another output port instead of being dropped. The next node that receives the deflected burst/packet will try to route it towards the destination. The performance of deflection routing has been extensively evaluated for regular topologies [1, 3, 31, 33]. It is found that deflection routing generally performs poorly compared to store-and-forward routing unless the topology in use is very well connected. Nevertheless, its performance can be significantly improved with a small amount of buffers. In [15], an unslotted deflection routing algorithm is proposed. The analysis indicates that unslotted deflection routing can achieve comparable performance as the slotted approach. This removes the need for synchronisation techniques, which are difficult to implement. Deflection routing for arbitrary topologies is studied in [9]. The proposal assigns a priority to each output port of a node. When a burst/packet needs to be routed, the ports are chosen in the prioritised order.

In addition to the earlier literature on deflection routing, a number of papers have studied the application of deflection routing in JET-based OBS networks. In [36], the problem of insufficient offset time is examined. This problem is caused by a burst traversing more hops than originally intended as a result of being deflected. Since the offset time between the burst and its header decreases after each hop, the burst may overtake the header. The paper studies various solutions such as setting extra offset time or delaying bursts at some nodes on the path. It finds that delaying a burst at the next hop after it is deflected is the most promising option. Various performance analyses of deflection routing in OBS networks are presented in [14, 36, 96]. In [76], a congestion control scheme is presented that deflects bursts at upstream links rather than at the congested link itself. To cater for loss-sensitive traffic, reference [47] proposes to reserve some wavelengths on the alternate paths in advance for such traffic classes in case their bursts need to be deflected.

Deflection routing may be regarded as "emergency" or *unplanned* multipath routing. It causes deflected bursts to follow a longer path than other bursts in the same flow. This leads to various problems such as increased delay, degradation

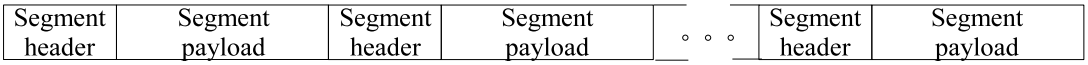
of signal quality, increased network resource consumption, out-of-order burst arrivals. A better method to reduce congestion and burst loss is probably *planned* multipath routing, or load-balancing. We will survey the literature on load balancing in §2.5.

2.1.3 Burst Segmentation

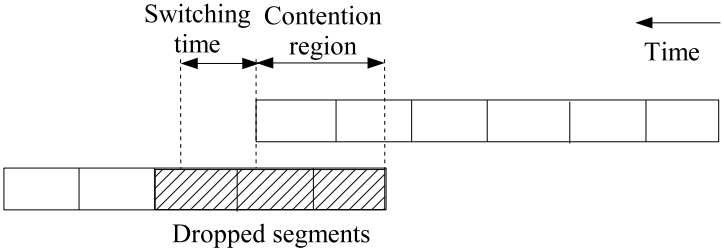
Burst segmentation [20, 82] is a contention resolution approach unique to OBS networks. It takes advantage of the fact that a burst is composed of multiple IP packets, or segments. Therefore, in a contention between two overlapping bursts, only the overlapping segments of a burst need to be dropped instead of the entire burst. Network throughput is improved as a result. Two currently proposed variants of burst segmentation are shown in Figure 2.1. In the head-dropping variant [20], the overlapping segments of the later arriving burst, or the head segments, are dropped. On the other hand, the tail-dropping variant in [82] drops the overlapping segments of the preceding burst, or the tail segments. A number of strategies to combine burst segmentation with deflection routing are also discussed in the same paper. The authors of [82] claimed that the tail-dropping approach results in a better chance of in-sequence delivery of packets at the destination. Burst segmentation is later integrated with void-filling scheduling algorithms in [75, 81]. A performance analysis of burst segmentation is presented in [68].

2.1.4 Wavelength Conversion

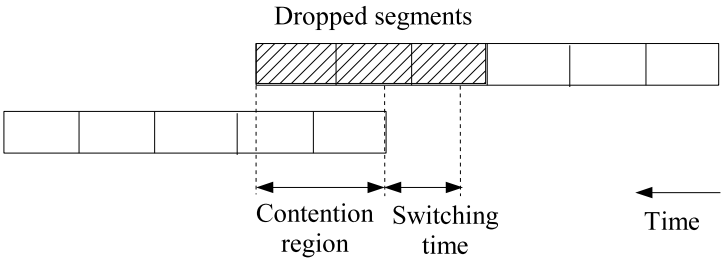
Wavelength conversion is the process of converting the wavelength of an incoming signal to another wavelength for transmission on an outgoing channel. In WDM, each fibre has several wavelengths, each of which functions as a separate transmission channel. When contention for the same output wavelength happens between some bursts, the node equipped with wavelength converters can convert all except one burst to other free wavelengths. Wavelength conversion enables an output wavelength to be used by bursts from several input wavelengths, thereby increasing the degree of statistical multiplexing and the burst loss performance.



(a) Segment structure of a burst



(b) Head dropping



(c) Tail dropping

Figure 2.1: Burst segmentation approaches

As the number of wavelengths that can be coupled into a fibre continues to grow, this approach becomes increasingly attractive. For example, with 32 wavelengths per link, the burst loss probability at loading of 0.8 is about 4×10^{-2}. With 256 wavelengths per link, the burst loss probability drops to less than 10^{-4}.

Although optical wavelength conversion has been demonstrated in the laboratory environment, the technology remains expensive and immature. Therefore, to be cost-effective, an optical network may be designed with some limitations on its wavelength conversion capability. Following are the different categories of wavelength conversion [66]:

- *Full conversion:* Any incoming wavelength can be converted to any outgoing wavelength at every core node in the network. This is assumed by most current OPS and OBS proposals. It is the best performing and also the most expensive type of wavelength conversion.
- *Sharing of converters at a node:* Converter sharing is proposed and studied in [26] for synchronous OPS and in [27] for asynchronous OPS. It allows

great saving on the number of converters needed. However, the drawbacks are the enlargement of the switching matrix and additional attenuation of the optical signal.

- *Sparse location of converters in the network:* Only some nodes in the network are equipped with wavelength converters. Although this category is well-studied for wavelength-routed networks, it has not been considered for OPS and OBS networks due to the poor loss performance at nodes without wavelength conversion capability.
- *Limited-range conversion:* An incoming wavelength can only be converted to some of the outgoing wavelengths. In [28,71,99], various types of limited-range converters for OPS networks are examined. It shows that nodes with limited-range wavelength converters can achieve loss performance close to those with full conversion capability.

2.2 Channel Scheduling

Since the large number of wavelengths per link in WDM offers excellent statistical multiplexing performance, wavelength conversion is the primary contention resolution approach in OBS. In this approach, every OBS core node is assumed to have full wavelength conversion capability. When a burst header arrives at a node, the node invokes a channel scheduling algorithm to determine an appropriate outgoing channel to assign to the burst. Channel scheduling plays a crucial role in improving the burst loss performance of an OBS switch. A good scheduling algorithm can achieve several orders of magnitude performance improvement over a first-fit algorithm. Because of its importance, channel scheduling in OBS has been the subject of intense research in the last few years.

In the JET OBS architecture, each burst occupies a fixed time interval, which is characterised by the start time and the end time carried in the burst header. Therefore, channel scheduling can be regarded as a packing problem wherein the primary objective is to pack as many incoming bursts onto the outgoing channels as possible. This problem is complicated by the fact that the order of the burst

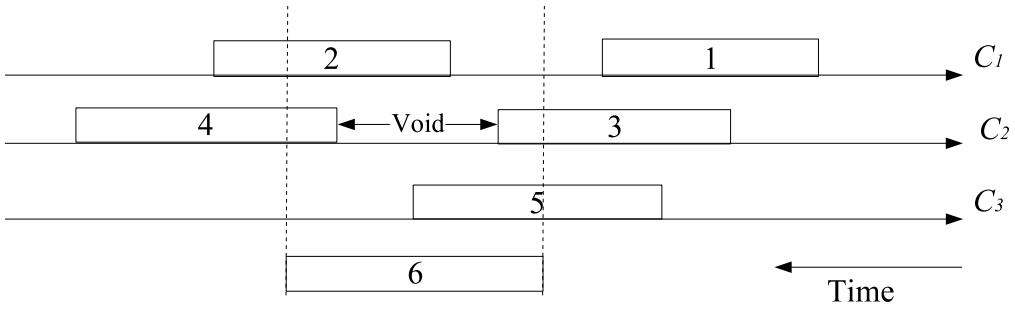


Figure 2.2: Illustration of the channel fragmentation problem

header arrivals is not the same as the arrival order of the bursts themselves. Thus, bursts with long offset times are able to reserve a channel before those with shorter offset times. Their reservations fragment a channel's free time and produce gaps or *voids* among them that degrade the schedulability of bursts with shorter offset times. This is illustrated in Figure 2.2 where the numbers inside the bursts indicate their header arrival order. Although all six bursts can theoretically be accommodated, burst 6 cannot be scheduled because of the channel fragmentation caused by the other bursts. Many channel scheduling algorithms have been proposed to deal with this problem. In this section, we will give a detailed survey of the existing channel scheduling algorithms.

2.2.1 Algorithms without Void Filling

Non-void filling algorithms are the simplest type of channel scheduling algorithms. In order to maximise processing speed, they do not utilise voids caused by previously scheduled bursts to schedule new bursts. There are currently two non-void filling algorithms in the literature, Horizon [79] and Latest Available Unscheduled Channel (LAUC) [88], which are essentially the same. They only keep track of the *unscheduled* time, which is the end time of the last scheduled burst, for each channel. When a header arrives, they assign to the burst the channel with the unscheduled time being closest but not exceeding its start time. The idea is to minimise the void produced before it. This is illustrated in Figure 2.3(a) where t_1, t_2 and t_3 are the unscheduled times. Channel C_3 is selected to schedule the new burst because $t - t_3 < t - t_2$. By storing the unscheduled times in a binary

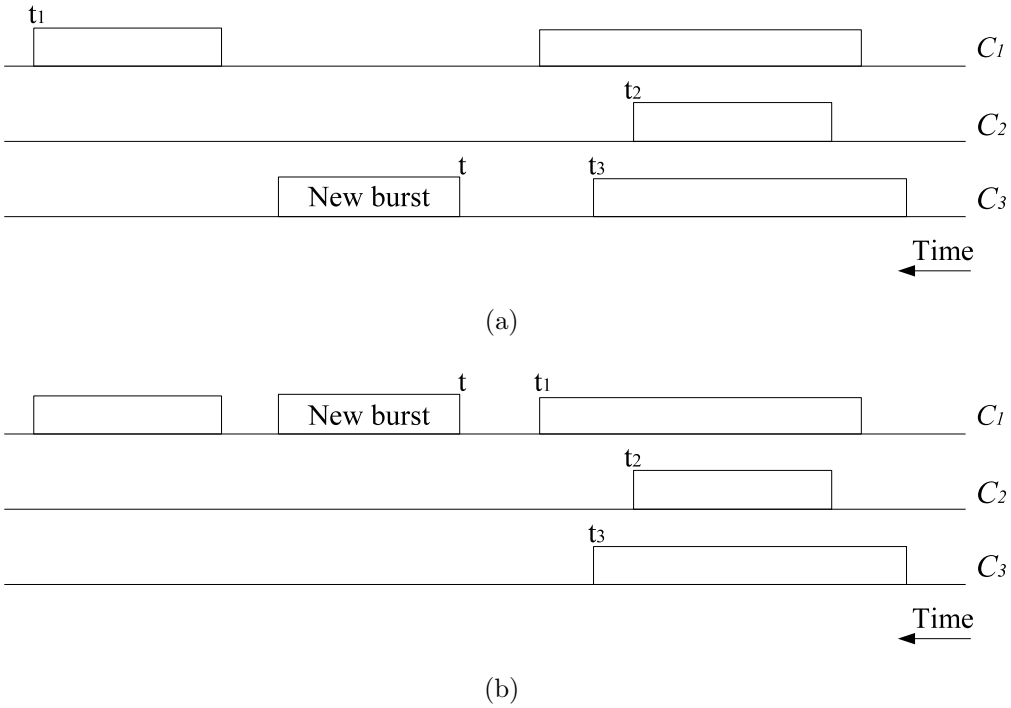


Figure 2.3: Examples of channel assignment using: (a) non-void filling algorithm (Horizon or LAUC), and (b) void filling algorithm (LAUC-VF)

search tree, the two algorithms can be executed in $O(logW)$ time, where W is the number of wavelengths per link.

2.2.2 Algorithms with Void Filling

Void-filling algorithms utilise voids to schedule new bursts to improve burst loss performance. They keep track of every void on the outgoing channels and check all of them as well as unscheduled channels when an incoming burst needs to be scheduled. A void-filling scheduling algorithm is proposed in [77] for OPS, but can be used for OBS as well. However, it does not specify the selection criteria if several wavelength channels are available. Another algorithm called Latest Available Unused Channel with Void Filling (LAUC-VF) is proposed in [88]. This is perhaps the most popular OBS channel scheduling algorithm to date. When an incoming burst needs to be scheduled, LAUC-VF calculates the *unused* time of each available channel, which is the end time of the burst preceding the incoming one. The channel with the unused time closest to the start of the incoming burst is selected. This is illustrated in Figure 2.3(b) where t_1, t_2 and t_3 are the unused

times. Channel C_1 is selected to schedule the new burst because t_1 is closest to t. Since the unused times for each burst are different, LAUC-VF has to recalculate all of them for each new burst. Using a binary search tree structure, each unused time calculation takes $O(logN_b)$, where N_b is the average number of scheduled bursts per channel. Thus, LAUC-VF takes $O(WlogN_b)$ to execute.

A variant of LAUC-VF is proposed in [40, 100]. The authors observe that LAUC-VF may select an unscheduled channel to schedule a new burst even though suitable voids are available because it bases its decision only on the unused times. This creates more voids and degrades performance. Therefore, they propose to minimise the number of voids generated by giving priority to channels with voids. The authors in [40] use simulation to show that the new algorithm performs better than LAUC-VF in terms of burst loss. The authors in [100] also propose to use parallel processing and associative memory to implement LAUC-VF in order to reduce its time complexity.

Another implementation of LAUC-VF is proposed in [89] under the name Minimum Starting Void (Min-SV). Min-SV has the same scheduling criteria as LAUC-VF. However, it uses an augmented balanced binary search tree as the data structure to store the scheduled bursts. This enables Min-SV to achieve processing time as low as that of Horizon without requiring special hardware or parallel processing.

2.2.3 Batch Scheduling

As the name implies, batch-scheduling schemes make scheduling decisions for multiple bursts at one go. They generally attempt to collect header packets for all bursts within a time window before carrying out the scheduling operation. This is done by delaying some header packets until the batch scheduling times. It is believed that such a collective view of multiple bursts results in more efficient scheduling decisions. However, delaying the headers may make their offset times too small for processing at downstream nodes. A batch-scheduling proposal must

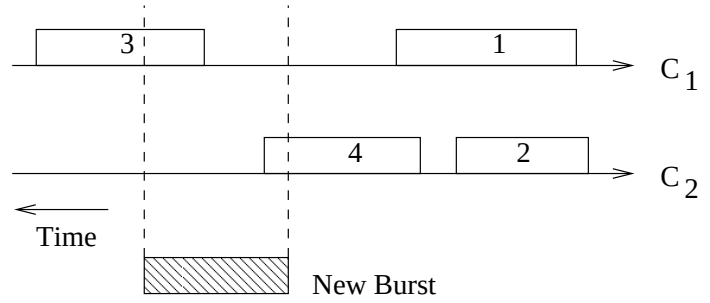
address this issue in order to be practical. To date, two batch-scheduling schemes for OBS have been proposed, which are summarised below.

In Group-Scheduling [11], the time axis is divided into successive windows. Bursts in a window are represented as vertices in an interval graph. There exists an edge to connect two vertices if and only if their corresponding bursts overlap each other. A combinatorial algorithm is used to determine sets of maximum non-overlapping bursts. Each of those sets will be scheduled on one wavelength channel. However, the authors do not address the issue of maintaining sufficient offset time for processing at downstream nodes after the headers are held up. This limits the algorithm to only bursts that have reached their last hop before the egress node.

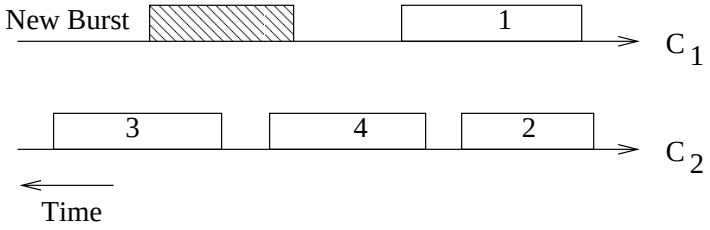
In Look-ahead Window [30], the time axis is divided into successive slots. All bursts within a moving window covering multiple slots are collected for batch processing. Each slot is represented by a vertex in a directed graph and each burst is represented by an edge connecting the starting and the ending slots. A shortest-path algorithm is used to determine the overlapping bursts. Some of them are then dropped according to some criteria such as shortest burst drop, etc. The authors suggest using FDLs to delay bursts to maintain the original offset times.

2.2.4 Burst Rescheduling

Burst rescheduling [73, 74] is a scheduling approach that achieves a better loss performance than non-void filling algorithms without requiring computationally expensive void-filling operations. It involves rescheduling a scheduled burst to another available wavelength in order to accommodate a new burst. This is possible because a header packet arrives well before the arrival of its corresponding data burst. A scenario that illustrates the benefits of burst rescheduling is shown in Figure 2.4. When a burst is rescheduled, a special NOTIFY message needs to be sent to the next downstream node to inform them of the change. To achieve low complexity, the authors only consider single-level rescheduling in which at most



(a) LAUC-VF fails to schedule the new burst



(b) Burst 3 is rescheduled to accommodate the new burst

Figure 2.4: Illustration of the benefits of burst rescheduling

one burst is rescheduled to accommodate a new one. Various variants of burst rescheduling are proposed and studied such as rescheduling only when needed (on-demand rescheduling) or attempting rescheduling for every arriving burst (aggressive rescheduling). Simulation results show that burst rescheduling has a loss performance between those of LAUC and LAUC-VF.

2.2.5 Burst-Ordered Scheduling

Recognising that scheduling bursts in a different order from their arrival order is the primary cause of channel fragmentation and the resulting scheduling inefficiency, burst-ordered scheduling algorithms attempt to go to the root of the problem by scheduling bursts in the order of their actual arrivals. As illustrated in Figure 2.5, if burst-ordered scheduling can be applied to all incoming bursts, the burst being scheduled will be to the left of all scheduled bursts. Therefore, a simple first-fit algorithm is sufficient to optimally schedule the bursts. Thus, the biggest hurdle remains how to implement burst-ordered scheduling properly.

In [10], the idea of burst-ordered scheduling is proposed for the first time in an algorithm called First Arrival First Assignment with Void Filling (FAFA-VF). When a header arrives at a node, the algorithm puts it into a priority queue

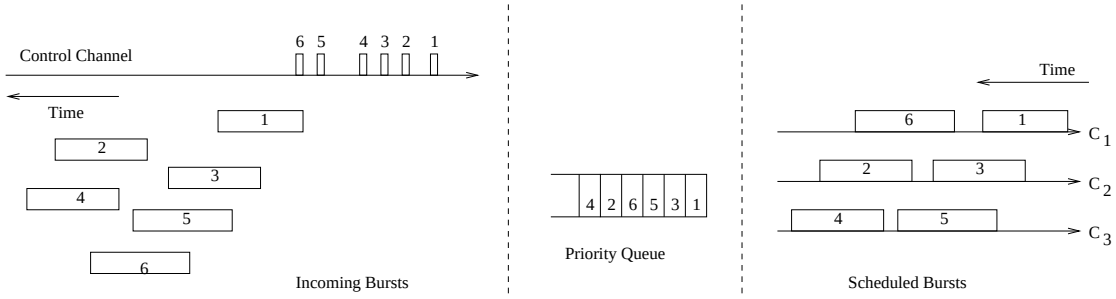


Figure 2.5: Burst-ordered scheduling concept

that sorts the headers based on their burst arrival times. The header at the front is dequeued and scheduled at time Δ before its burst arrives. However, like the Group-Scheduling proposal in [11], the authors do not address the issue of maintaining sufficient offset time for processing at downstream nodes after the headers are held up. This severely limits the algorithm's practical application.

In [48], another burst-ordered scheduling algorithm called PipeLine System is proposed. When a header arrives at a node, the algorithm puts it into an N-slot First In First Out (FIFO) queue. The header will remain in the queue either up to a maximum time T_{max} or until it is pushed out of the queue by subsequent arriving headers. The algorithm also goes through the headers in the queue in the order of their burst arrivals and tentatively assigns an outgoing channel to each of them using LAUC-VF. Such tentative assignment may change when new headers arrive. A tentative channel becomes permanent once the header exits from the queue. The authors also suggest setting the minimum offset time for a burst to $h \times T_{max}$, where h is the hop count and T_{max} is the maximum header queueing time at a node, in order to ensure that downstream nodes have sufficient processing time.

Although based on the promising concept of burst-ordered scheduling, the above two papers are unable to fully implement it for all bursts. This is due to the inherent conflict between the need to delay headers until their bursts arrive in order to carry out burst-ordered scheduling and the need to forward headers early to give downstream nodes sufficient processing times. In Chapter 3, we describe a channel scheduling algorithm that carries out scheduling in the order

of burst arrivals for all bursts. We show that it is optimal and greatly outperforms LAUC-VF [63].

2.3 QoS Differentiation Mechanisms

Quality of Service (QoS) differentiation at a node is the key to any QoS provisioning framework. Due to the bufferless nature of OBS networks, burst loss probability is a primary QoS metric of interest. Therefore, most of the existing OBS QoS proposals focus on burst loss differentiation although a few deal with delay differentiation as well [12, 52]. There are currently two main QoS models in OBS: *relative QoS* and *absolute QoS*. In the relative QoS model, the QoS performance of a traffic class is defined relative to those of other classes. For instance, the loss rate of a high priority class is guaranteed to be better or at least no worse than that of a lower priority class. However, the actual loss probability of the high priority traffic still depends on the offered load to the node, and no upper bound on its loss probability is guaranteed. On the other hand, the absolute QoS model gives worst-case quantitative QoS guarantees for each traffic class, e.g., having an end-to-end packet loss probability no greater than 1%.

The relative QoS model can be further divided according to the degree of isolation among traffic classes. The classes can be completely isolated, i.e., the performance of a high priority class is independent of lower priority classes. Examples of complete isolation models include strict priority and bandwidth partitioning. Alternatively, in non-isolated models such as proportional QoS, the loss rates of different classes are inter-dependent.

It should be noted that QoS models and QoS differentiation mechanisms do not have tight association. Each of the different QoS models above can be implemented using a variety of QoS differentiation mechanisms. Conversely, a QoS differentiation mechanism can be used to implement many QoS models with minor modifications. To date, a large number of QoS differentiation mechanisms have been proposed. In this section, we will provide a survey of representative proposals.

2.3.1 Offset-Based Approach

Offset-based differentiation is the earliest QoS approach proposed for OBS networks. In OBS, bursts are usually scheduled in the order of their header arrivals. In an output channel contention, the burst that has a longer offset time wins the contention. Therefore, in [93], QoS differentiation is achieved by adding an extra offset to the basic offsets of high priority bursts at the ingress. If the extra offset is larger than the maximum basic offset in the network, the high priority traffic class can be completely isolated from the low priority one. A performance analysis of the scheme using the $M|M|k|k$ queueing model is also provided in the paper. The analysis is refined in [29] to give more accurate estimates of the loss probabilities. In [55], a different offset allocation is proposed that takes into account the hop length of a burst. Therefore, it improves the fairness among bursts of the same class but having different hop lengths. In addition, it puts a ceiling on the maximum offset time to reduce end-to-end delay.

Although having the advantage of simplicity, offset-based approach suffers from a number of problems as pointed out in [12]. Firstly, adding extra offset times to high priority bursts increases their end-to-end delays. Secondly, by increasing offset times, the algorithm creates more channel fragmentation that degrades network performance. Such channel fragmentation also increases the loss probabilities of large bursts of low priority classes since they will have more difficulties fitting into the voids created by high priority bursts. Due to these problems, the offset-based approach is not considered in later QoS proposals.

2.3.2 Intentional Dropping Approach

In this approach, a node monitors the local QoS performance of all traffic classes. When the performance of a particular class i drops below the required level, the node selects another class j, usually a low-priority one, that has its QoS performance above the required level for dropping. Header packets of class j will be dropped before they reach the scheduler. Consequently, the offered load to the node is reduced and the performance of other classes, including class i,

will improve. In [12], this approach is employed to implement the proportional QoS model [23]. When the relative loss performance of a class pair deviates from the predefined proportional relation, all bursts from the class with better-than-expected loss probability are dropped. In [98], intentional dropping is used to implement absolute QoS differentiation. That is, when the loss probability of a class i exceeds its threshold, bursts from another class j are dropped. The dropping may be total, i.e., dropping all bursts from class j, or gradual, i.e., increasing dropping as the loss probability of class i approaches its threshold. The main drawback of this approach is that it may unnecessarily drop bursts that otherwise would not contend with any other bursts, which leads to channel under-utilisation. Preemption, which will be described next, is a more efficient and elegant implementation of the same idea.

2.3.3 Preemptive Approach

When a node that has preemption capability receives the header of a high-priority burst and fails to schedule it, the node may drop, or preempt, a scheduled low-priority burst to make room for the high-priority one. Thus, preemption employs the same concept as intentional dropping, but is more efficient and elegant in that it drops only those necessary to schedule the high-priority burst. When preemption happens, the node may send a control message to downstream nodes to clear the reservations of the preempted burst. Alternatively, it may choose to do nothing, which leads to some inefficiency as those reserved time intervals at the downstream nodes cannot be utilised by other bursts.

Preemption is a popular QoS differentiation mechanism and is used to implement many different QoS models. The key part is proper definition of the preemption policy according to the intended QoS model. The preemption policy determines in a contention which burst is the "high-priority" one, i.e., having preemption right. Preemption has been used in [50] to implement throughput differentiation. When contention happens, the algorithm searches for a class whose throughput exceeds the negotiated service level and selects one of its bursts for

preemption. Probabilistic preemption, in which the high-priority burst is granted preemption right with certain probability, is used in [91] to implement burst loss probability differentiation. Preemption is also used in [34,61] deterministically and in [59] probabilistically to implement absolute QoS differentiation. That is, bursts belonging to a class whose loss probability approaches or exceeds its threshold are able to preempt bursts from other classes.

Preemptive differentiation is often implemented in combination with burst segmentation [20, 82] to improve throughput. In that case, only the overlapping part of the preempted burst is dropped instead of the entire burst. Such implementation is considered in [7, 50]. Probabilistic preemption with segmentation is also employed in [72] to implement the proportional QoS model.

Apart from improving throughput, burst segmentation is also used to provide another level of QoS differentiation in some papers. The idea is to assemble packets with different priorities into a single burst in decreasing order or priority, i.e., high-priority packets at the head of the burst [46, 83]. In preemption, the tail-dropping segmentation [82] is used to drop the tail part of the preempted burst, which contains low-priority packets. In [2], the same idea is used but in reverse order. That is, packets are assembled in increasing order of priority and head-dropping segmentation [20] is used.

2.3.4 Header Queueing Approach

Header queueing is another popular QoS differentiation approach. It takes advantage of the fact that header packets arrive at a node and are scheduled well before the actual data bursts arrive. Being processed in the electronic domain, those headers can be put in a queue and easily manipulated. Like batch scheduling, the problem of maintaining sufficient offset times for downstream nodes can be solved by using FDL buffers at each node to delay data bursts or by setting larger offsets at edge nodes. Several variants of this approach have been proposed in the literature, which are described below.

The approach taken by most QoS differentiation proposals based on header queueing is to have one header queue for each traffic class and to implement a scheduling policy among the queues based on their corresponding QoS levels. In [92], headers are scheduled according to their priority levels. That is, a low priority header queue is scheduled only when all higher priority queues are empty. The algorithm achieves complete isolation among classes but low priority traffic may suffer excessive dropping. In [51], all headers are treated equally by the scheduler. QoS differentiation is achieved by delaying low priority headers for certain time periods depending on their priority levels before sending them to the scheduler. This implementation has the same differentiation effect as the offset-based approach without having to assign different offsets to traffic classes. In [43,53], Weighted Fair Queueing (WFQ) scheduling is used among the queues to achieve bandwidth guarantees for traffic classes. This type of QoS differentiation is similar to that in the Differentiated Services (DiffServ) framework [5] for IP networks.

2.4 Absolute QoS Model

The absolute QoS model provides worst-case quantitative QoS guarantees to applications. This type of QoS guarantee is essential to applications with stringent delay and bandwidth requirements such as multimedia and mission-critical applications. From an user's point of view, the absolute QoS model is also preferred. This is because an end user usually has a specific quantitative QoS requirement depending on the applications in use. He would prefer to receive it independent of the network load. Apart from QoS differentiation, implementing the absolute QoS model requires additional mechanisms such as admission control and end-to-end signalling. Due to these added complexities, we devote this section to survey specifically absolute QoS proposals in the OBS literature.

Implementing the QoS model generally requires two steps. In the first step, a local loss probability threshold is defined and guaranteed for each traffic class at each core node in the network. The threshold is the upper bound on the

loss probability that burst flows within the class should experience at a node. In the second step, end-to-end guarantees are split up and mapped to the local thresholds using a signalling protocol. The sum of the local thresholds assigned to an end-to-end path should be less than or equal to the end-to-end guaranteed loss probability.

Various mechanisms are used to guarantee the local thresholds. In [98], a combination of intentional dropping as described in §2.3.2 and wavelength grouping is employed. In [34, 59, 61], preemption is used. However, there are two critical features that are present only in our proposal in [61]. The first feature is admission control at core nodes. Reference [98] briefly mentions that edge-based admission control can be used but no discussion is provided. The other proposals do not mention it. Without local admission control, a core node cannot reliably guarantee the thresholds for traffic classes in heavy loading condition. The second feature is uniform loss probability among flows within a single class. Without it, flows with unfavourable traffic characteristics such as short offset times or large burst lengths may experience loss probabilities beyond the threshold even though the threshold is preserved at the class level.

The way end-to-end guarantees are split up and allocated to local thresholds is also critical to the efficiency of the whole QoS framework. This part is not considered in [34]. In [59, 98], the guaranteed end-to-end loss probability of a class is divided up and allocated equally to intermediate nodes along a path. This approach has two major problems. Firstly, it leads to inefficiency since links along a path are usually not equally loaded. Therefore, the ability to accept new traffic of a path will be limited by the most heavily loaded link. Secondly, it implies that the number of thresholds a core node has to support is many times as large as the number of end-to-end classes, which is not desirable as core nodes have to process headers at very high speed. The use of path clustering in [98] somewhat alleviates this problem but not completely solves it. In Chapter 4, we describe a threshold allocation algorithm based on Multi-Protocol Label Switching (MPLS) [60] that allocates different thresholds to a flow at different links on a path based

on the traffic condition at those links. Hence, it achieves both efficiency and low complexity for core nodes.

2.5 Load Balancing

The number of load balancing proposals for OBS networks to date is quite small. In [86, 97], when the traffic demands among various ingress/egress node pairs are known, the authors formulated the load balancing problem as an optimization problem and solved it using linear integer programming. For dynamic traffic demands, almost identical solution was proposed in [49, 78]. For each ingress/egress node pair, the proposed solution determines two link-disjoint paths, one of them being the shortest path and periodically probes them. The probe packets collect information on the loss probabilities at intermediate links along the paths. The loss probabilities are then summed up to give the cost of each path. For the longer path, a penalty term is added to its cost to account for the fact that it is less desirable to route traffic to longer paths. At the end of each probing period, a traffic portion proportional to the cost difference between the two paths is shifted from the more costly path to the other. In Chapter 5, we present a dynamic load balancing algorithm for absolute QoS traffic [62]. It is similar to those in [49, 78] but incorporates a more accurate analytical expression than the traditional Erlang B formula for calculating burst loss probability at a link in its link cost function. This enables one to show that the scheme achieves better performance than ealier formulation.

Chapter 3

Ordered Scheduling: An Optimal Channel Scheduling Algorithm for Optical Burst Switched Networks

3.1 Introduction

A major concern in Optical Burst-Switched networks is channel contention, which occurs when multiple bursts from different input ports are destined for the same output port simultaneously. Existing contention resolution proposals fall into one of the following categories: optical buffering, deflection routing, burst segmentation and wavelength conversion. Among these, wavelength conversion offers the best performance and is the preferred approach. In this approach, every OBS core node is assumed to have full wavelength conversion capability. When a burst header arrives at a node, the node invokes a channel scheduling algorithm to determine an appropriate outgoing channel to assign to the burst. Due to the large number of wavelengths per link in WDM, wavelength conversion can achieve excellent statistical multiplexing performance.

Channel scheduling plays a crucial role in the performance of the contention resolution approach based on wavelength conversion. This is especially true for JET OBS. In the JET OBS architecture, each burst occupies a fixed time interval, which is characterised by the start time and the end time carried in the burst header. Therefore, channel scheduling can be regarded as a packing problem wherein the primary objective is to pack as many incoming bursts onto the

outgoing channels as possible. This problem is complicated by the fact that the order of the burst header arrivals is not the same as the arrival order of the bursts themselves. Thus, when bursts are scheduled in the order of their header arrivals as in [77, 79, 88], those with long offset times are able to reserve a channel before those with shorter offset times. Their reservations fragment a channel's free time and produce voids among them that degrade the schedulability of bursts with shorter offset times.

A number of approaches have been proposed to solve the above problem. In burst the rescheduling approach [73,74], a node may reschedule a previously scheduled burst to another wavelength in order to accommodate a new one. However, to achieve the best result, multiple bursts may need to be rescheduled (i.e., multi-level rescheduling), which is impractical given the large number of wavelengths per link in a typical WDM network. In the batch scheduling approach [11,30], a node collects multiple burst headers and makes scheduling decisions for all of them at one go. However, the node cannot delay the headers for too long as downstream nodes need sufficient offset times to process the headers before the corresponding bursts arrive, which limits the extent to which the batch scheduling concept can be carried out in practice.

In the burst-ordered scheduling approach, a node attempts to schedule bursts in the order of their actual arrivals. If fully implemented, this approach can remove all channel fragmentation and thus achieve optimal performance. Previous proposals [10,48] attempt to implement this approach by delaying the headers and sort them according to their burst arrivals. However, like the batch scheduling proposals, the inherent conflict between the need to delay headers and the need to forward headers early to give downstream nodes sufficient processing times prevents the burst-ordered scheduling concept from being fully realised. In this chapter, we present an algorithm based on burst-ordered scheduling approach, called *Ordered Scheduling*, which does not need to delay burst headers. Thus, it is the first that can fully realise the burst-ordered scheduling concept and achieve optimal scheduling performance.

The rest of this chapter is organised as follows. In §3.2, the Ordered Scheduling algorithm is presented in detail. It includes description of two implementations of the algorithm, namely, Basic Ordered Scheduling and Enhanced Ordered Scheduling. Discussion on practical implementation and related issues is given in §3.3. The simulation study in §3.4 compares the performance of Ordered Scheduling with LAUC-VF and investigates the effects of various parameters. Finally, concluding remarks are given in §3.5.

3.2 Ordered Scheduling

3.2.1 High-Level Description

Recognising that scheduling bursts in a different order from their arrival order is the primary cause of channel fragmentation and the resulting scheduling inefficiency, Ordered Scheduling is proposed to optimise burst scheduling in OBS. We assume full wavelength conversion capability at an OBS node, as does most of the current OBS literature. In this algorithm, the scheduling of a burst consists of two phases. In the first phase, when a header packet arrives at a node, an admission control test is carried out to determine whether the burst can be scheduled. If the burst fails the test, it is dropped. Otherwise, a reservation object that contains the burst arrival time and duration is created and placed in an electronic buffer while the header is passed on to the next node. The buffer is in the form of a priority queue[1] with higher priority corresponding to earlier burst arrival time. The second phase starts just before the burst arrival time. Because of the priority queue, the reservation object of the incoming burst should be at the head of the queue. It is dequeued and a free wavelength is assigned to the burst. A special NOTIFY packet is immediately generated and sent to the next downstream node to inform it of the wavelength that the burst will travel on.

A simple example shown in Figure 3.1 helps to illustrate the main concept of Ordered Scheduling. The left section of the figure shows the order of the incoming

[1] A priority queue is a data structure that always has the highest priority element at the head of the queue.

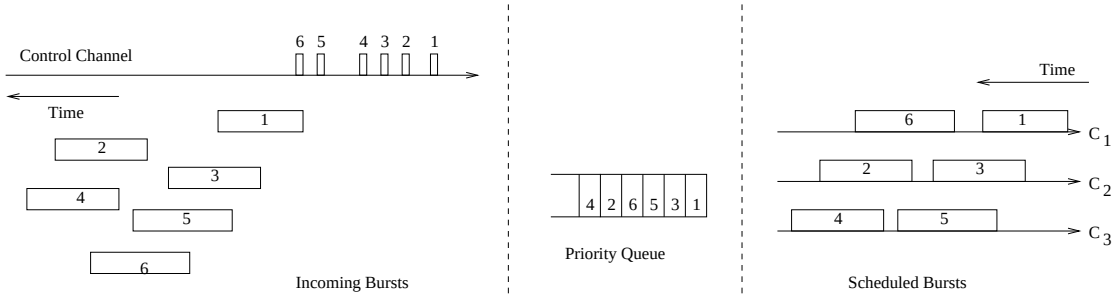


Figure 3.1: The main concept of Ordered Scheduling

bursts and the order of the header packets in the control channel. The middle section shows that the reservation objects are placed in the priority queue in the order of burst arrivals. Finally, the scheduled bursts are shown in the right section of the figure. The node simply dequeues a reservation object from the priority queue and assigns a free wavelength to it. Since the priority queue sorts the reservations according their burst arrival times, all unscheduled reservations will be to the left of the newly dequeued reservation. Therefore, any void it produces on the right has no effect on the schedulability of non-scheduled reservations. Thus, any free wavelength can be assigned to the newly dequeued reservation. In the example, we use a round robin assignment because it is the easiest way to implement.

The admission control test for an output link without an FDL buffer is given below.

A burst requesting reservation for the time interval $[t_0, t_1]$ *can be scheduled on an output link with* M *wavelengths if* $\forall t \in (t_0, t_1)$*, the number of existing reservations containing* t *is no more than* $M - 1$*.*

(Note: A reservation for interval $[t_0, t_1]$ is said to contain t if $t_0 < t < t_1$)

When an output link is equipped with an FDL buffer, which can be thought of as a collection of fibres (or FDLs) with different lengths, a node has the option of delaying a burst by routing it through one of the FDLs. In this case, the above admission control test is extended as follows. If a burst fails to reserve an output wavelength at its original arrival time t_0, the node searches through the FDLs in order of increasing length. Let the length of the FDL in consideration be

D_{FDL}. The node first checks if the FDL is free during the interval $[t_0, t_1]$. If the FDL already has another reservation overlapping that interval, the node simply proceeds to the next FDL. Otherwise, it reserves the FDL for that interval. The node then executes the admission control test for the new reservation interval $[t_0 + D_{FDL}, t_1 + D_{FDL}]$. If the test succeeds, the burst is admitted and the search stops. Otherwise, the node undoes the reservation on the FDL and proceeds to the next FDL. If all the FDLs have been searched, the burst is dropped.

It should be noted that passing the admission control test is necessary but not sufficient for a burst to be scheduled. The test guarantees that at any infinitesimal time slot δt within the reservation interval $[t_0, t_1]$ of a burst, there exists a free wavelength. However, it does not guarantee that those free time slots are located on the same wavelength for the entire reservation interval, which is required for them to be usable by the new burst. The key to ensure that they are on the same wavelength is to schedule bursts in the order of their arrival times as is done in the second phase using the priority queue.

The admission control test is important because it prevents resource wastage due to over-admitting bursts. In Ordered Scheduling, headers are passed on to the next node before scheduling takes place. Thus, without the admission control test, an incorrectly admitted burst will have its header forwarded to downstream nodes to make further reservations. However, the node that makes the incorrect admission will not be able to schedule the burst. Without having the optical switch configured for it, the burst will be lost upon arrival and resources reserved at downstream nodes will be wasted.

In comparing Ordered Scheduling with the current scheduling algorithms, we note that Ordered Scheduling is optimal in the sense that given an incoming burst pattern, it can achieve the theoretical upper bound on the number of scheduled bursts. This is because by the definition of the admission control test, if a new burst reservation fails the test, there exists a time slot within its reservation interval in which all the data wavelengths are occupied. Therefore, it is theoretically

impossible to schedule that burst. Conversely, all the bursts that pass the admission control test are scheduled by the algorithm. We will revisit the discussion on the optimality of Ordered Scheduling in §3.3.4.

3.2.2 Admission Control Test Realisation

The admission control test in §3.2.1 is presented in continuous form. However, this form may not be practical or feasible to realise. A simple solution would be to divide the time axis into slots. A burst that reserves any portion of a time slot, however small, will be considered as occupying the whole time slot. The admission control routine simply needs to keep track of the number of admitted bursts $N_{occupied}$ that occupy each time slot and compare it to the total number of data wavelengths M. A new burst will be admitted only if $N_{occupied} < M$ for all the slots it will occupy. We refer to this version as Basic Ordered Scheduling. It is suitable if the optical switches in the network also operate in a slotted fashion as mentioned in [88]. In that case, the time slots chosen by Ordered Scheduling should simply be set to be the same as the time slots of the underlying optical switches.

The basic slotted approach, however, may degrade the system performance if the underlying optical switches can operate in a truly asynchronous fashion. Due to its discrete nature, it does not consider the case where two bursts can occupy the same wavelength in a slot and thus may lead to unnecessary burst loss. This may be alleviated by having the slot size much smaller than the average burst size. However, that will increase the processing time and/or hardware complexity.

We describe here an enhanced version of the above slotted approach, which is called Enhanced Ordered Scheduling. Instead of a single number to indicate the number of bursts occupying a time slot, the admission control routine now keeps three data entities for each time slot:

1. N_{total} is the total number of bursts that occupy the slot, whether wholly or partly;

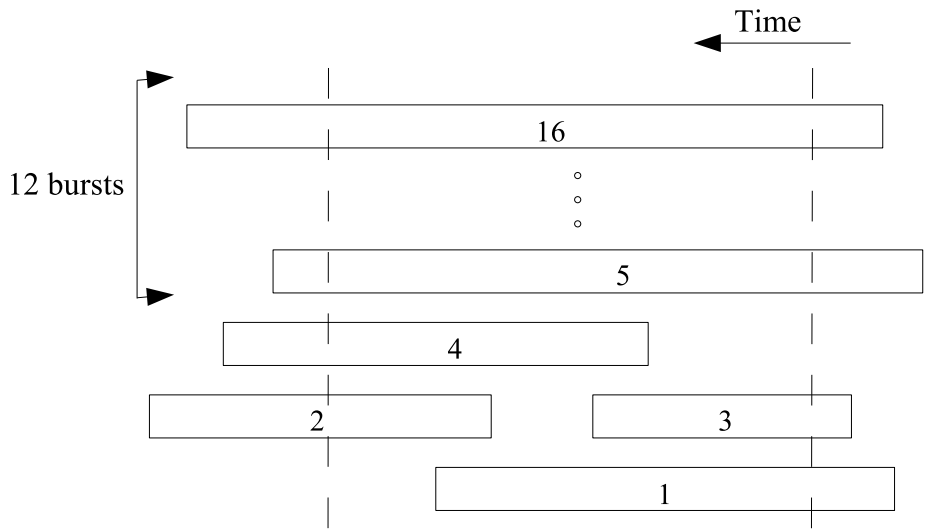


Figure 3.2: Example of the bookkeeping for a typical time slot.

2. *heads* is the list of the start times of the bursts that have the start of their reservation periods fall into the slot, sorted in increasing order and
3. *ends* is the list of the end times of the bursts that have the end of their reservation periods fall into the slot, sorted in increasing order.

The bookkeeping of the two versions is illustrated in Figure 3.2. In the figure, the slot sizes are exaggerated and the unshown bursts occupy the entire slot. For the basic version, $N_{occupied} = 16$. For the enhanced version, $N_{total} = 16$ and there are two entries in each of *heads* and *ends*.

When a header arrives at a node to request reservation, the admission control routine pretends that the burst has passed the test and updates the database of all the time slots involved, i.e., the time slots of the burst corresponding to the arriving header. N_{total} is incremented by one for each of the time slots. In addition, for the time slots containing the start or the end of the burst, an entry is added to *heads* or *ends*, respectively. The admission control routine then checks all the involved time slots. For a particular slot, if N_{total} is larger than the number of wavelengths M, entries in the *heads* list will be *matched* to those in the *ends* list to reduce the number of occupied wavelengths. A pair of bursts are considered matched for a given slot if the start time of the one in the *heads* list is no greater than the end time of the other in the *ends* list. The actual number of occupied

wavelengths is N_{total} minus the number of matched pairs. If this number is smaller than M for all the involved slots, the new burst is schedulable and admitted. Otherwise, its header is dropped and its information previously inserted in the time slots' database is removed.

The matching operation is facilitated by the fact that *heads* and *ends* are kept in increasing order. The algorithm simultaneously goes from the beginning to the end on both lists, checking their entries against each other. Let i and j be the current indices on *heads* and *ends*. If $heads[i] \leq ends[j]$ then a match is recorded and i and j are incremented by one. Otherwise, only j is incremented to point to the next larger entry in *ends*. The process is repeated until either i or j passes the end of the list.

The formal description of the algorithm is presented in Algorithm 1. Denote $[t_0, t_1]$ as the requested reservation interval; *slot* as the object representing a particular time slot and s_0 and s_1 as the slots that contain t_0 and t_1, respectively. Also, let $slot.insert\_head(t)$ and $slot.insert\_end(t)$ be the functions that insert t into the sorted lists *heads* and *ends* of *slot*, respectively. The main test procedure uses three sub-functions $insert(t_0, t_1)$, $match()$ and $remove(t_0, t_1)$. The first two sub-functions are presented below the main test procedure while the last one is omitted because it is similar to $insert(t_0, t_1)$.

It is worth noting that the above algorithm will work properly even if some burst sizes are smaller than the slot size. In that case, some bursts may fall entirely within a single time slot as illustrated in Figure 3.3. For each burst falling entirely within the time slot, N_{total} is incremented by one and one entry is added to each of *heads* and *ends*. In the example shown, $N_{total} = 4$ and there are 3 entries in each of *heads* and *ends*. The $match()$ function treats each entry as a separate burst. It performs the matching operation as in Figure 3.3(b) and returns 3 matches. So the number of occupied wavelengths is $N_{total} - match() = 1$, which is the correct result.

In terms of loss performance, we observe that enhanced Ordered Scheduling is optimal since it fully implements the test in continuous form. Its superiority

Algorithm 1: Admission control test

(1) $accept \leftarrow$ **true**

(2) **for** $slot = s_0$ **to** s_1

(3) $\quad slot.insert(t_0, t_1)$

(4) $\quad$ **if** $slot.N_{total} - slot.match() > M$

(5) $\quad\quad accept \leftarrow$ **false**

(6) **if** $accept =$**false**

(7) $\quad$ **for** $slot = s_0$ **to** s_1

(8) $\quad\quad slot.remove(t_0, t_1)$

INSERT(t_0, t_1)

(1) $N_{total} \leftarrow N_{total} + 1$

(2) **if** $slot$ contains t_0 **then** $insert\_head(t_0)$

(3) **if** $slot$ contains t_1 **then** $insert\_end(t_1)$

MATCH()

(1) $i \leftarrow 0$

(2) $j \leftarrow 0$

(3) $matched \leftarrow 0$

(4) **while** Neither i nor j have passed the end of $heads$ and $ends$,respectively

(5) $\quad$ **if** $heads[i] \leq ends[j]$

(6) $\quad\quad matched \leftarrow matched + 1$

(7) $\quad\quad i \leftarrow i + 1$

(8) $\quad j \leftarrow j + 1$

(9) **return** $matched$

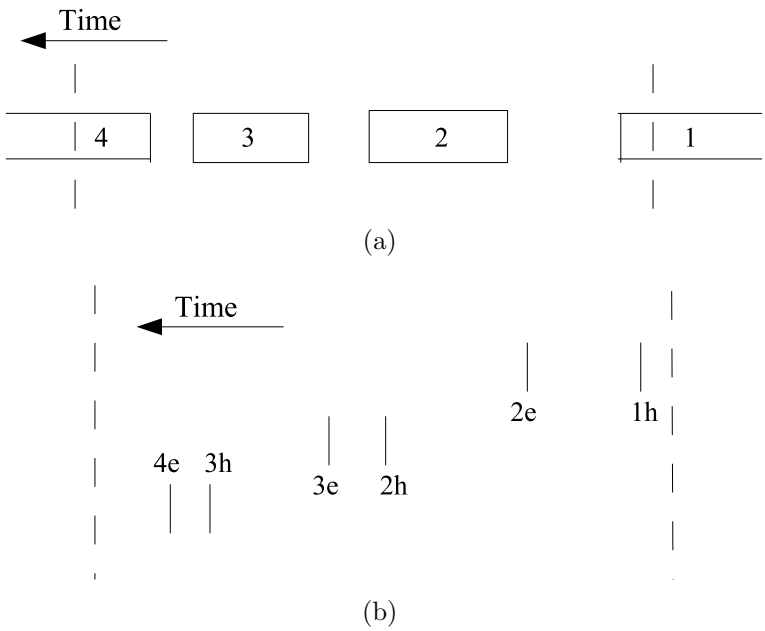


Figure 3.3: Matching operation when some bursts fall entirely within the slot: (a) Actual burst pattern, (b) Burst pattern as seen by the algorithm

compared to the basic version is illustrated in Figure 3.2 where the basic version reports that 16 wavelengths are occupied for the time slot while the enhanced version reports only 15 occupied wavelengths. The disadvantage of the enhanced version is that it is more complex. This will be explored in the next section.

3.3 Practical Implementation and Related Issues

3.3.1 Complexity Analysis

As the burst rates in optical backbone networks are very high, the scheduling of a burst has to be done very quickly. To achieve that, parallel computation is usually necessary. In addition, complexity analysis is important because it offers insights into how various parameters are best configured. In this section, we will address these issues in the context of Ordered Scheduling.

Admission Control Test

The slotted structure of the two admission control implementations is particularly suitable for parallel processing since each slot of a burst is processed independently. Let S be the maximum number of slots in the scheduling window. A simple parallel solution is to have S processing elements in the admission control unit with each

processing element responsible for one slot. When a burst header arrives, the processing elements corresponding to the slots covered by the burst will compute the admission control test simultaneously.

The time complexity analysis for the basic and enhanced versions of the admission control test is as follows. For Basic Ordered Scheduling, a processing element needs to perform at most one comparison and one update of $N_{occupied}$ per burst. Therefore, the required processing time is constant and takes less than 1 ns assuming a processing speed in the order of 10^9 operations per second. For Enhanced Ordered Scheduling, the processing element also needs to perform one comparison and one update. In addition, it needs to do the matching operation when necessary. Assuming that the slot size is smaller than the minimum burst size, the number of elements in *heads* and *ends* is M in the worst case. So the worst case complexity of the matching operation is $O(M)$. Also, the update of *heads* and *ends* at the two slots at the two ends of a burst takes $O(logM)$. Therefore, the overall worst case time complexity is $O(1) + O(M) + O(logM) = O(M)$. In a normal case, however, the size of *heads* and *ends* is about M/K where K is the average number of slots per burst. Hence, the average complexity is $O(M/K)$ per matching operation. The overall average complexity is $O(1) + O(M/K) + O(logM) = O(M/K + logM)$. Let us consider an example with $M = 256$ and $K = 16$, *heads* and *ends* will have about 16 elements on average. A worst case estimate of the processing time is 50 ns, which includes the execution of $match()$ and $remove(t_0, t_1)$. The average processing time is much smaller as $match()$ and $remove(t_0, t_1)$ are only executed in heavy loading conditions.

The required number of processing elements is inversely proportional to the slot size, or proportional to the average number of slots per burst K. Therefore, although Basic Ordered Scheduling has the advantage of fast processing compared to the enhanced version, its drawback is that it requires a much larger number of processing elements to ensure good dropping performance. For Enhanced Ordered Scheduling, there is a tradeoff between processing speed and hardware complexity.

A small value of K will reduce the required number of processing elements but will lead to longer execution time and vice versa.

For LAUC-VF, it is possible to perform a parallel search across the wavelengths to find all the unused wavelengths. Then the search results are compared to each other to find the latest available one. These operations can be performed in $O(logM)$ time, which is better than enhanced Ordered Scheduling and worse than basic Ordered Scheduling. In terms of hardware complexity, LAUC-VF requires one processing element for each wavelength with each processing element being fairly complex. If the number of wavelengths per link is large, which is usually the case, the hardware requirement for LAUC-VF will be larger than Ordered Scheduling.

Priority Queue

The queueing operations on the priority queue are common to both versions of Ordered Scheduling. Its complexity depends on the specific implementation of the underlying priority queue. Some efficient implementations of priority queues using pipelined heap are reported in the literature [4, 41]. They have $O(1)$ time complexity with regard to queue size. Implemented on conservative technologies such as 0.35-micron and 0.18-micron CMOS, they can achieve up to 200 million queueing operations per second for queues with up to 2^{17} entries. The queue size depends on the size of the scheduling window, which in turn depends on offset times and FDL buffer depth, and the burst arrival rate. We observe that the above priority queue implementations can accommodate any queue size of practical interest. The queue size only affects the amount of required memory.

Overall Time Complexity

The computational work in admission control and priority queue operations can be pipelined. That is, as soon as the admission control routine finishes with a header and passes it to the priority queue, it can handle the next header while the first header is being enqueued. Therefore, the overall complexity is the maximum of the two parts. With parallel processing, the time complexity for basic Ordered

Scheduling is $O(1)$. The worst case and average time complexities of enhanced Ordered Scheduling are $O(M)$ and $O(M/K + logM)$, respectively.

3.3.2 Timing Issues

The timing in an operation cycle of Ordered Scheduling is shown in Figure 3.4. At the beginning of an operation, when a header packet arrives at node A, the admission control test takes t_{admit}. If the burst reservation is successfully admitted, its header is sent to the next downstream node B while the reservation object is placed in the priority queue, which takes t_{queue}. At a suitable time, the object is removed from the queue, which also takes t_{queue} since for most implementations of the priority queue, enqueue and dequeue operations take approximately the same amount of time. It takes t_{NOTIFY} to transmit the NOTIFY packet at node A and to receive it at node B. At both nodes, the optical switching matrices are configured t_{config} before the burst arrival. There is a guard time t_{guard} between the receipt of the NOTIFY packet and the start of the optical switch configuration. This is to ensure that timing variations in various operations will not cause the NOTIFY packet to arrive late for the optical switch configuration. One of those timing variations may result from a large number of burst arrivals in a short interval. Without t_{guard}, this could overload the scheduler and render it unable to make scheduling decisions in time for optical switch configuration at the downstream node.

From the timing diagram, we can calculate the minimum time required to schedule a burst under Ordered Scheduling as

$$T_0 = t_{admit} + 2t_{queue} + 2t_{NOTIFY} + t_{guard} + t_{config}. \tag{3.1}$$

We attempt an estimate for T_0 here. From the previous section, we have $t_{admit} = 50$ ns in the worst case and $t_{queue} = 5$ ns. For t_{config}, according to [67], the semiconductor optical amplifier (SOA) technology can achieve a switching time t_{config} of 1 ns or less. For t_{NOTIFY}, assuming 8-byte NOTIFY packets as in §3.3.3, it will take $t_{NOTIFY} = 6.4$ ns to transmit or receive a NOTIFY packet at 10

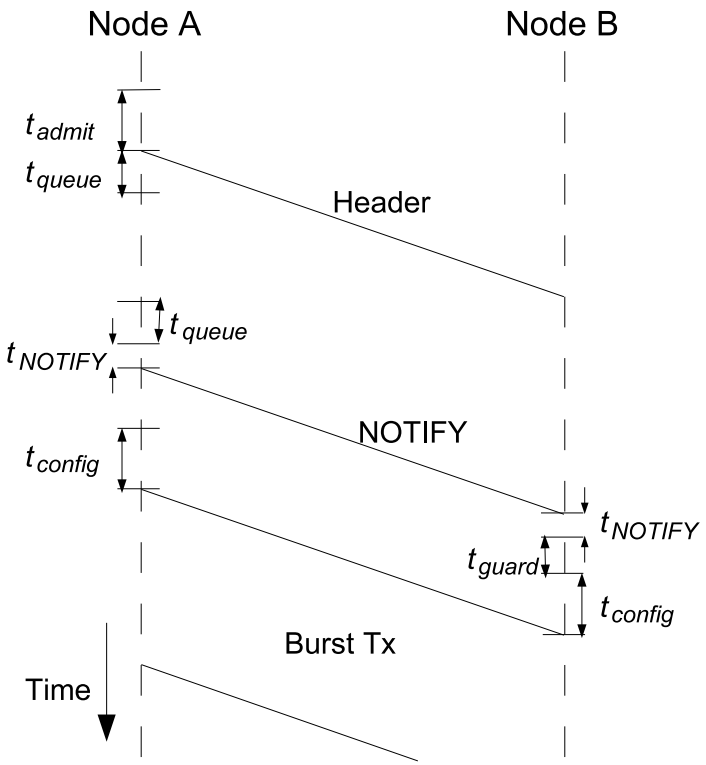


Figure 3.4: Timing diagram for scheduling a burst

Gb/s. Assuming a 10% guard time, the estimate for T_0 is 80 ns in the worst case. With rapid advances in electronics, we expect the figure to go down significantly in the near future.

T_0 can also be taken as the minimum required time offset of a burst. Of the time components on the right hand side of the above equation, t_{admit} and t_{queue} are variable. Therefore, in setting the minimum offset time, some upper bounds on t_{admit} and t_{queue} should be used. If for some reasons, t_{admit} or t_{queue} exceed their upper bounds, an arriving burst may have an offset smaller than T_0. In that case, if it passes the admission control test, the node may bypass the priority queue and go straight to assigning a wavelength to it. The reason is because its offset is so small, it would certainly end up at the head of the priority queue had it been put into the queue.

3.3.3 Signalling Overhead

Compared to traditional burst scheduling schemes, there is more signalling overhead in Ordered Scheduling due to the need to send the NOTIFY packets. One

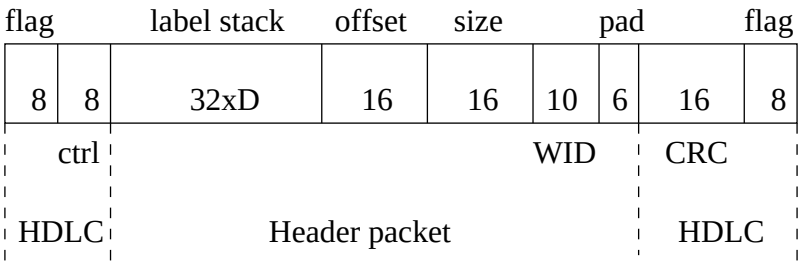
NOTIFY packet is required for every burst. Therefore, at a glance, the signalling load appears to be doubled. However, since the size of a NOTIFY packet is much smaller than that of a header packet, the increase in signalling load is much smaller. In this section, we will estimate the signalling load increase and explore ways to reduce it.

The control packet formats in normal scheduling schemes and in Ordered Scheduling are shown in Figure 3.5. In the example, High Level Data Link Control Protocol (HDLC) is used as the data link protocol. The address field in HDLC header is omitted and the control field uses 8 bits. In the header packet format, the offset and burst size fields both occupy 16 bits, which allow a resolution in the order of nanosecond to be specified. The wavelength ID (WID) field is set at 10 bits to accommodate links with up to 1024 wavelengths. Finally, the burst ID (BID) field is used to assign a unique ID to each burst in one scheduling window. Its size of 14 bits allows up to 16384 unique burst IDs.

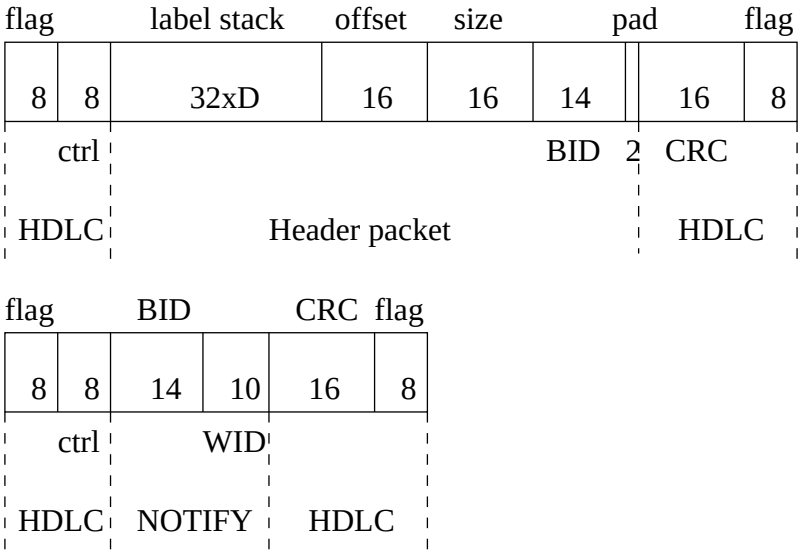
The increase in signalling load due to the use of NOTIFY packets can be calculated as follows. If we use a label stack depth $D = 3$, the length of a header packet will be $H = 23$ bytes in both normal scheduling schemes and Ordered Scheduling and the length of a NOTIFY packet will be $H_N = 8$ bytes. The percentage increase is

$$\frac{H_N}{H + H_N} = \frac{8}{23 + 8} = 0.2581 = 25.81\%.$$

It can be observed from Figure 3.5 that the HDLC header and trailer occupy a large portion of a frame carrying a NOTIFY packet. Thus, higher efficiency can be achieved if several NOTIFY packets are carried in one HDLC frame. In order to satisfy the strict timing requirement detailed in §3.3.2, we need to start dequeueing bursts at node A earlier than the deadline dictated by the timing diagram. The NOTIFY packets are then collected and sent in batch to node B. For example, assuming an average burst arrival rate of 10^7 bursts per second, bursts will need to be dequeued 1 μs earlier to collect 10 NOTIFY packets. The length of the



(a) Normal header packet format



(b) Control packet formats in Ordered Scheduling

Figure 3.5: Example of control packet formats

HDLC frame carrying 10 NOTIFY packets will be $H_N = 5 + 3 \times 10 = 35$ bytes. So the percentage of increase in signalling load is now:

$$\frac{H_N}{10H + H_N} = \frac{35}{10 \times 23 + 35} = 0.1321 = 13.21\%.$$

3.3.4 A Queueing Theory Perspective

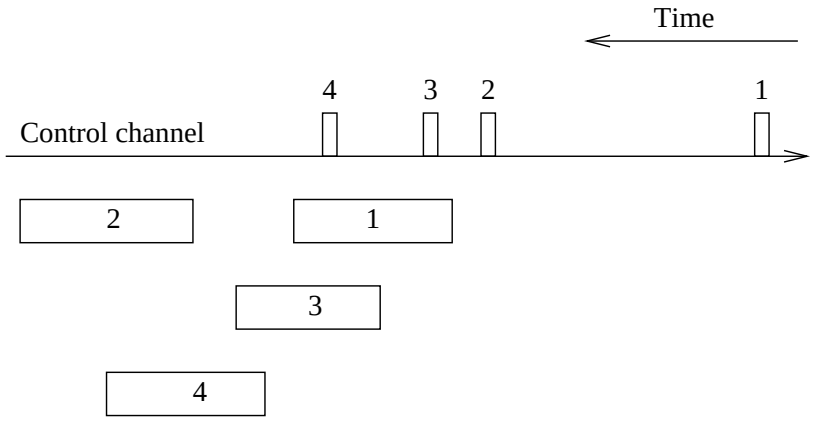
In the literature to date, most performance analyses of OBS schemes such as those in [22, 84, 93] have made use of queueing theory. In a queueing model of an OBS node, output wavelengths become servers, and when a burst is being transmitted, it is being "served". The current approach is to replace the offset times and advanced reservation in OBS with preemptive priority in the queueing model. To illustrate this approach, consider two traffic classes 1 and 2 with class 1 having a larger offset. Also, assume further that the difference in offset times is larger

than the maximum burst duration of class 2. Under these conditions, bursts from class 1 always win those from class 2 in inter-class resource contention because headers of class 1 bursts always arrive before those of class 2 bursts. The same effect can be achieved if we consider the headers do not exist and give class 1 bursts strict preemptive priority over class 2 bursts.

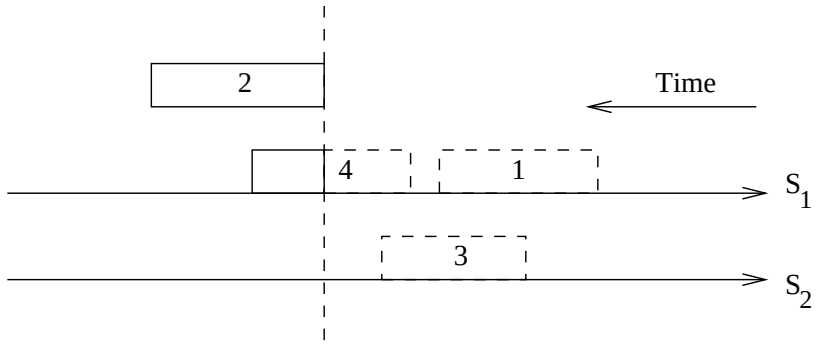
The above queueing model assumes that a burst is not tied to any particular server prior to its being served. This assumption only holds if the scheduling decision is made at the burst arrival instant, which is what our proposed algorithm does. Thus, only Ordered Scheduling can achieve the theoretical upper bound in blocking performance set by the queueing model. To illustrate this from the perspective of the queueing model, consider the burst arrival pattern in Figure 3.6 that is served by two channels. Bursts 1 and 2 have a larger offset time than bursts 3 and 4. Hence, in the corresponding queueing model, bursts 1 and 2 belong to class 1, which has preemptive priority over bursts 3 and 4 of class 2. When burst 2 arrives at the node, burst 4 is being served by server S_1 (channel 1) while server S_2 is free. In this situation, the queueing model and Ordered Scheduling would assign burst 2 to server S_2. However, the LAUC-VF algorithm assigns burst 2 to server S_1 instead and preempts burst 4. The reason for this mistake is because LAUC-VF schedules a burst when its header arrives at the node. Therefore, when burst 2 is scheduled, LAUC-VF has no knowledge about burst 4 whose header will arrive later.

3.4 Experimental Study

In this section, we study the burst dropping performance of Ordered Scheduling through simulation. Both versions of Ordered Scheduling are investigated. The slot sizes are 1 μs and 0.1 μs for the enhanced and basic versions, respectively, unless otherwise stated. The reason for the difference in the chosen slot sizes is because, as shown later in Figure 3.13, the performance of Basic Ordered Scheduling critically depends on the slot size while the performance of Enhanced Ordered Scheduling does not. We also simulate LAUC-VF under the same condition for



(a) OBS with advanced reservation



(b) Corresponding queueing model

Figure 3.6: Illustration of the queueing model approach to OBS analysis

comparison. LAUC-VF is chosen as a performance benchmark because it is one of the most efficient burst scheduling algorithms that schedule bursts as their headers arrive. Each simulation is run ten times and each run ends after 10^6 bursts are generated or 10^3 lost bursts are recorded, whichever later. We plot the results with error bars representing 95% confidence intervals; however, in some cases, the deviation from the mean is so small that the error bars are not visible.

The traffic model in use is similar to that in [88]. Specifically, each ingress node receives an IP packet stream from adjacent access networks. The IP packet length distribution is modelled according to that reported in [19] and is shown in Figure 3.7. Let l be the packet length in bytes. We have $\mathrm{P}[l = 44] = 0.6$, $\mathrm{P}[44 < l < 552] = 0.145$, $\mathrm{P}[l = 552] = 0.05$, $\mathrm{P}[552 < l < 576] = 0.005$, $\mathrm{P}[l = 576] = 0.05$, $\mathrm{P}[576 < l < 1500] = 0.05$ and $\mathrm{P}[l = 1500] = 0.1$. IP packets arrive according to a Poisson process. This choice of packet arrival distribution is justified by a study

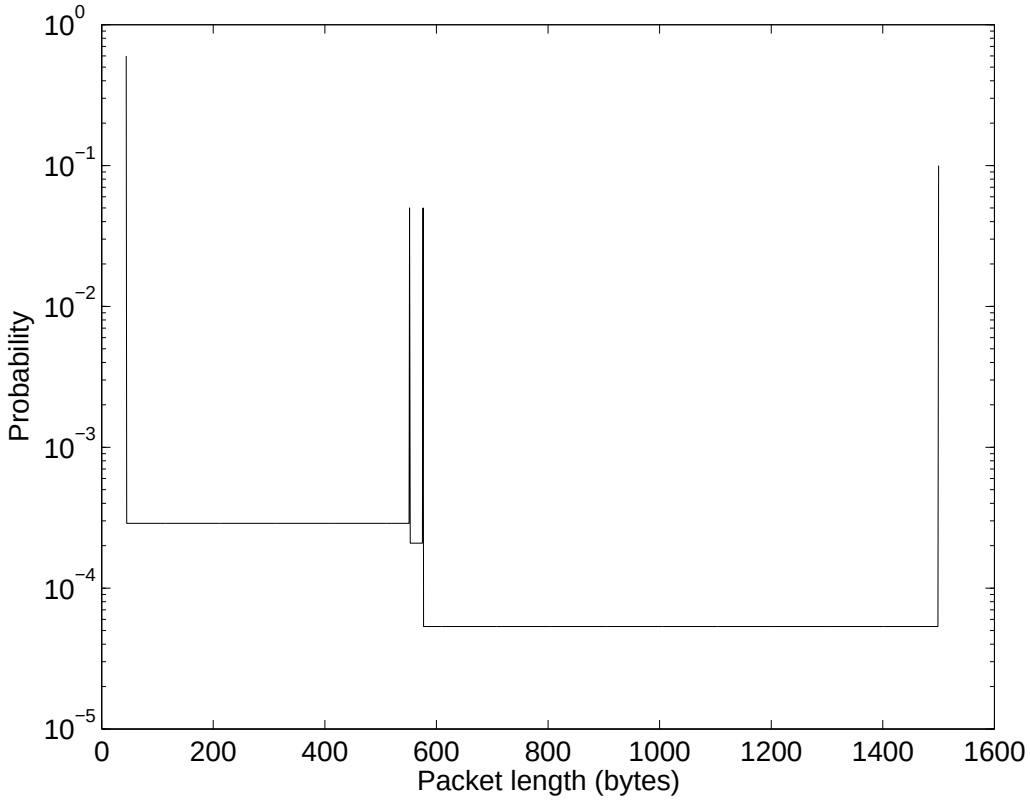


Figure 3.7: Probability distribution of packet length used in simulation study

in [8], which reports that IP traffic tends towards Poisson at very high speed links and very short time scale.

The burst assembly algorithm is a simple time-based algorithm with a time limit T_{limit}. There are separate assembly queues for each egress node and incoming IP packets choose a queue with equal probability. When the first IP packet that forms a burst arrives at an assembly queue, a timer is started from zero. Subsequent IP packets are appended to the assembly queue. A burst will be created when the timer exceeds T_{limit}. In the experiments, we set T_{limit} such that the maximum burst duration is 2.5 μs. So the average number of slots per burst are approximately 2 and 20 for Enhanced and Basic Ordered Scheduling, respectively. Another aspect of the assembly mechanism is that we assume no packet framing overhead inside a burst and guard bands at the head and the end of a burst. That is, the size of a burst is exactly the sum of all packets inside the burst.

The simulation study consists of three sets of experiments. The first two sets are carried out for a topology with a single core node. They aim to investigate

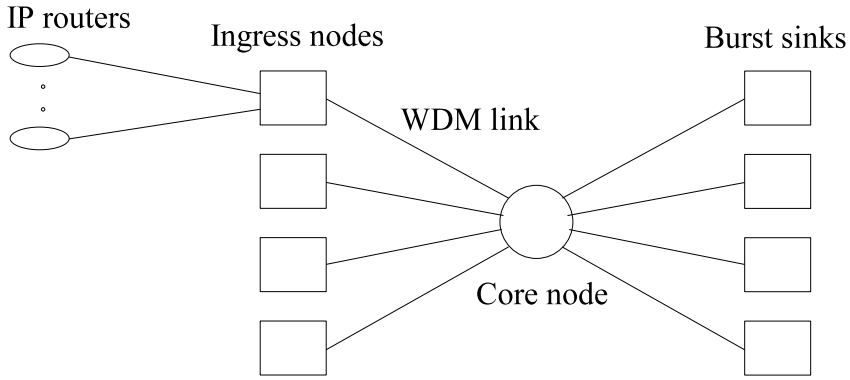


Figure 3.8: Topology for simulation study of a single core node

the effects of traffic conditions and hardware configurations on the performance of the algorithms, respectively. The final experiment set is carried out for an entire network to investigate the effect of network topology on the performance trend among the algorithms.

The simulation topology for the first two sets of experiments is shown in Figure 3.8. There are four ingress nodes and four burst sinks connected to the core node. The burst sinks represent egress nodes in a real network. No burst dropping is assumed on the links between the ingress nodes and the core node. It only occurs on the output links of the core node. Since IP packets are destined for each sink with equal probability, the average offered loads on each output link of the core node are equal. The reported burst dropping probabilities are the average values on the four output links.

3.4.1 Effects of traffic conditions

In this set of experiments, the configuration is as follows. The links connecting the OBS nodes are made up of a single optical fibre per link. Each optical fibre has 8 data wavelengths. The number of wavelengths for control and signalling purposes is assumed to be large enough so that no control message will be lost. The core node has an FDL buffer with 6 FDLs of lengths 5 μs, 10 μs,..., 30 μs. Except for the number of wavelengths per link, this represents a typical network configuration. The number of wavelengths per link is chosen to be small so that

the simulation time will not become prohibitively long. In the next section, we will investigate the effect of this parameter on the performance of the algorithms.

Firstly, we examine the effect of varying the offered load to the core node, which is defined as the ratio between the total average rate of incoming IP packets in bits per second (bps) and the total output link capacity of the core node also in bps. For this experiment, two offset-based QoS classes as described in [93] with equal loading are used. Class 2 is given higher priority than class 1 by assigning an extra offset of 3 μs. This offset difference is larger than the maximum burst size so that full isolation between the two classes is achieved. The arrival rate ranges from 3.3 bursts per μs to 4.15 bursts per μs, or from 0.74 to 0.92 in terms of offered load. Offered loads lower than 0.74 are not considered because they would make the loss probability of class 2 too small to measure through simulation. On the other hand, offered loads larger than 0.92 would make the loss probability of class 1 too large to be of practical interest.

The simulation results are plotted in Figure 3.9. They show that the dropping probabilities increase with increasing offered load, which is expected. Among the algorithms, enhanced Ordered Scheduling has the best dropping performance followed by basic Ordered Scheduling and then LAUC-VF. This order of dropping performance among the algorithms is as expected based on the discussion in the previous sections. The order of performance is the same in virtually all of the following experiments so we will not comment on it again except when the order is different. We note that the differences in performance are greater at lower load. This is because at low load, there are more free wavelengths to choose from to assign to an incoming burst reservation and LAUC-VF is more likely to make suboptimal wavelength assignment decisions due to incomplete knowledge of other burst reservations. Between the two classes, we observe that the performance improvement of Ordered Scheduling over LAUC-VF is greater for class 2 than it is for class 1. The reason for this is also related to loading. Since full isolation is achieved between the two classes, the effective loading for class 2 traffic is only

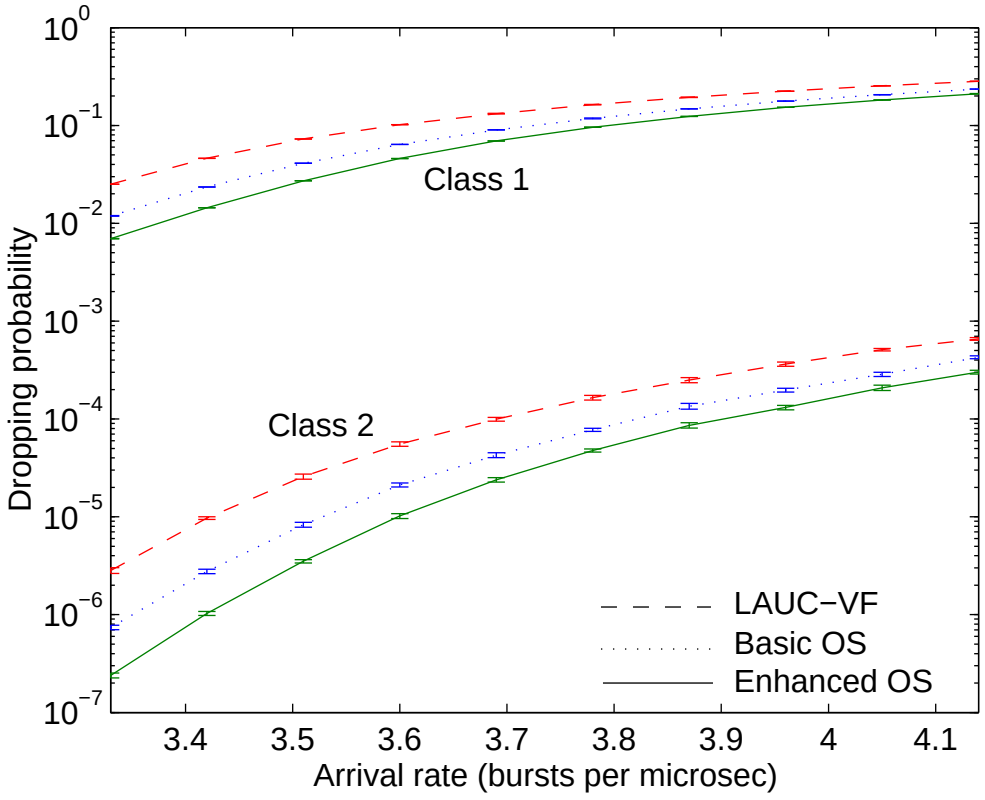


Figure 3.9: Burst loss probability versus traffic loading

half of that for class 1 traffic. So as the above reasoning goes, the improvement for class 2 is larger.

The effect of traffic class composition is considered next. We use the same traffic parameters as above except that the overall offered load is fixed at 0.9 and the offered load of each class is varied. As the proportion of class 1 traffic varies from 0 to 1, we observe in Figure 3.10(a) that the overall traffic loss rate follows a bell-shaped curve, which is slightly tilted towards the left. The dropping probabilities peak when the burst rates from the two classes are comparable and are at the lowest at the two extremities where traffic from only one class is present. This effect can be explained from the queueing model point of view. As the traffic composition becomes more balanced, more class 1 bursts are preempted by those from class 2. When burst B_1 from class 1 is preempted by burst B_2 from class 2, the effective size of B_2 is its actual size plus the portion already served of B_1. As mentioned in [84], if the burst size distribution is not exponential, that will

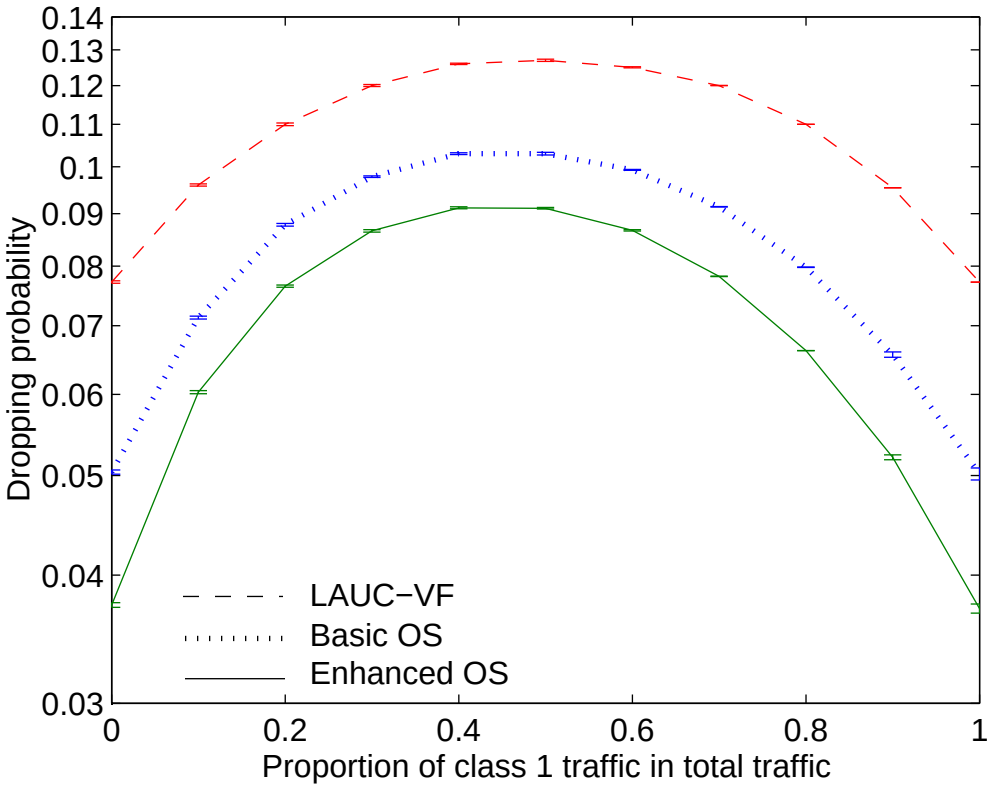
increase the effective burst size and the burst loss rates. This negative effect of burst preemption is present in all the algorithms, unlike the fragmentation of the scheduling window that only affects LAUC-VF. We also note that at the two extremes when there is only one traffic class, FDL buffers in the core node can still delay bursts and create fragmentation in the scheduling window. Therefore, there are performance differences among the algorithms even when there is only one traffic class.

The loss rates of individual classes are shown in Figure 3.10(b). It is seen that as the proportion of low priority traffic increases, the loss rates of both classes drop. For class 1, preemption by class 2 bursts make up a large part of its burst loss. Therefore, when there is less class 2 traffic, preemption occurs less frequently, which leads to the drop in class 1 burst loss. For class 2, the only cause for burst loss is intra-class contention since it is fully isolated from class 1. Thus, when its traffic rate decreases, contention rate rapidly decreases and so does the burst loss. This result implies that very low burst dropping probability can be achieved for high priority traffic even though the overall utilisation is high.

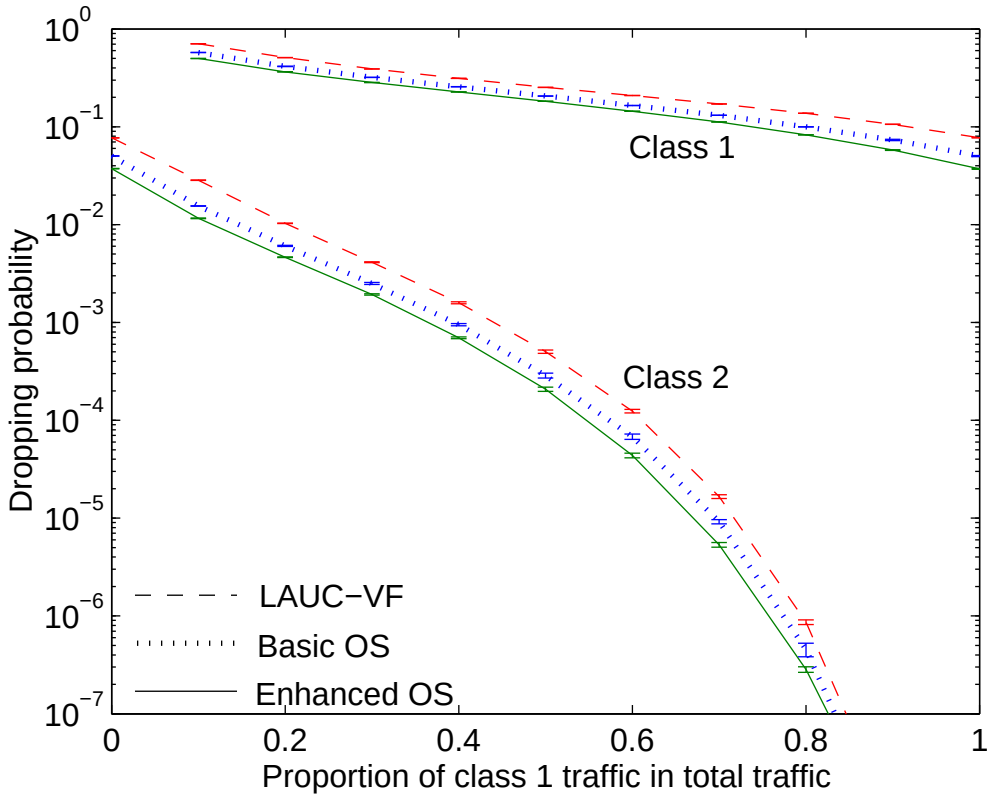
The final traffic parameter to be investigated is the number of QoS classes. In this experiment, the overall offered load is 0.8 and all traffic classes have equal loading. We plot the overall dropping probabilities in Figure 3.11. It shows that the overall dropping probability increases as the number of classes increases. This is as expected because as the number of classes increases, the scheduling window is more fragmented, which results in increasing loss probability. A notable aspect is the large increase in loss probabilities moving from one to two classes. This is caused by the large increase in the degree of fragmentation in the scheduling window when moving from one class to two classes.

3.4.2 Effects of hardware configuration

In the first experiment of this section, we study the impact of FDL buffer depth on the performance of the algorithms. Two kinds of data traffic are considered: one with a single QoS class and the other with two offset-based QoS classes. This is



(a)



(b)

Figure 3.10: Effect of traffic composition on burst loss probability: (a) Overall performance, and (b) Performance of individual traffic classes

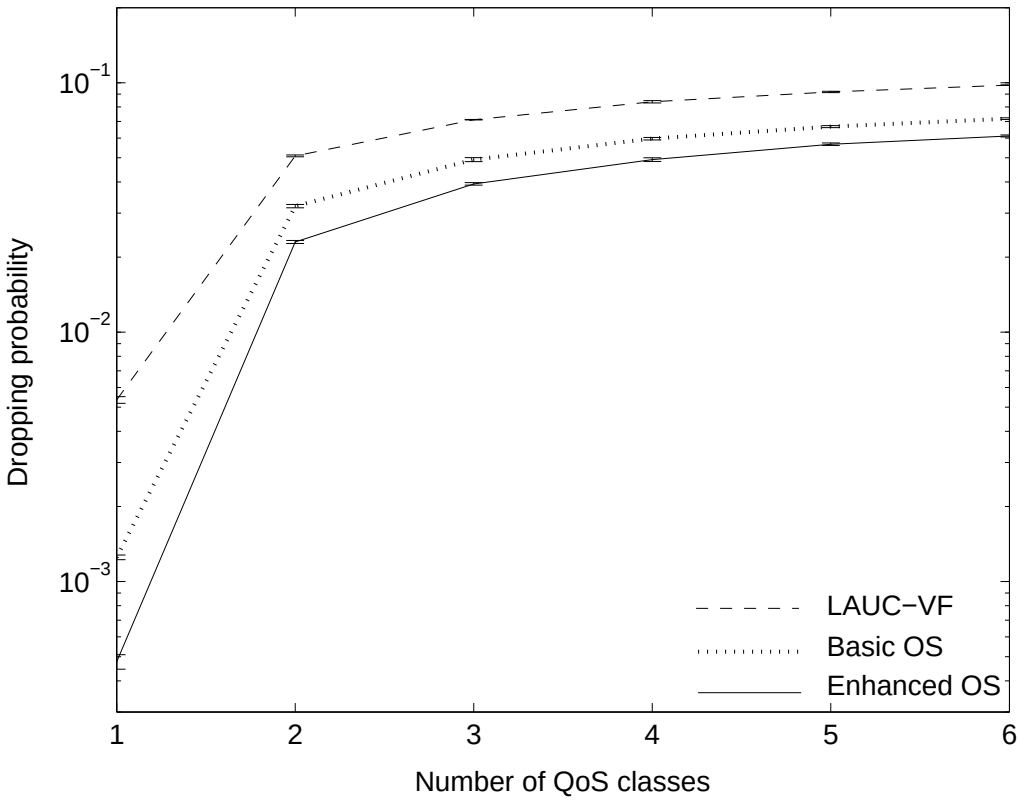


Figure 3.11: Overall burst loss probability versus number of traffic classes

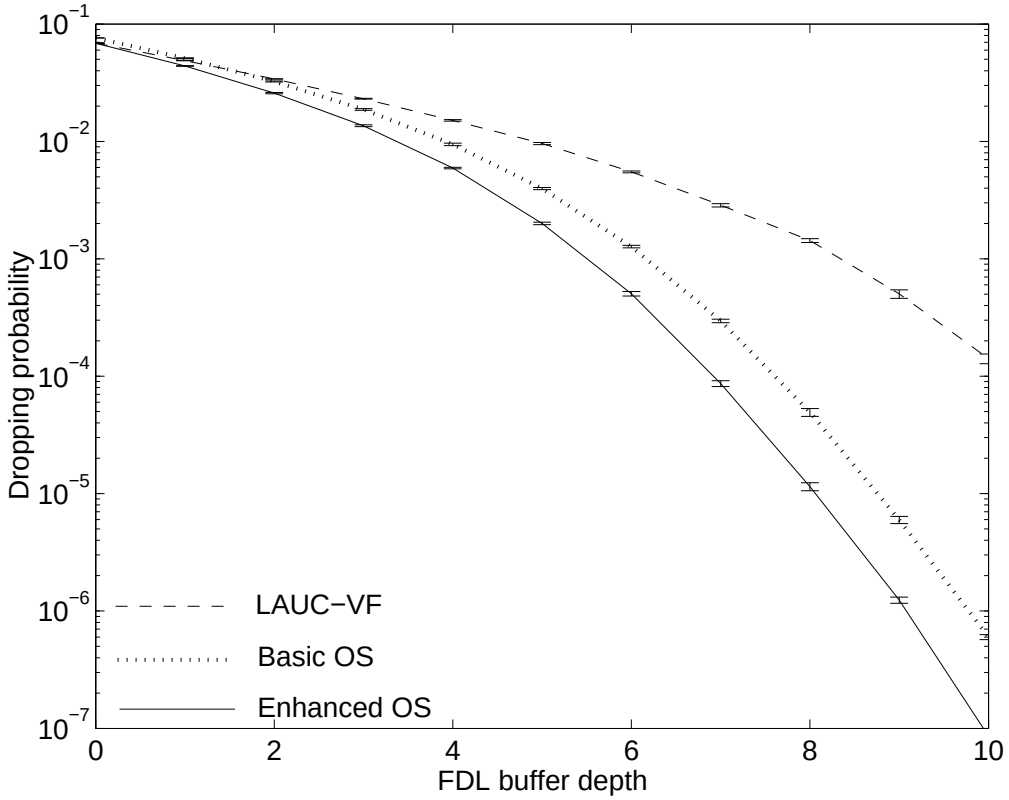
because the effects on dropping performance are slightly different between having one and two QoS classes. The offered loads for both cases are set at 0.8. The FDL buffer in use consists of a number of FDLs according to the buffer depth. The lengths of the FDLs are regularly spaced starting from 5 μs with length spacing being 5 μs.

From Figure 3.12, we see that the overall trend is improving loss performance with increasing number of FDLs. This is because when an FDL buffer is introduced, if a node cannot schedule a burst at its original arrival time, the node can try delaying the burst and scheduling it at a later time. The larger the number of FDLs there are in a buffer, the more options the node has in delaying bursts, which improves dropping performance. We also note that the curves for LAUC-VF tend to level off. This can be explained by the fact that the scheduling window is increasingly fragmented as more FDLs are introduced. For LAUC-VF, this negative effect opposes and neutralises the beneficial effect of having more FDLs, which explains the levelling off of its curve. Ordered Scheduling, on the

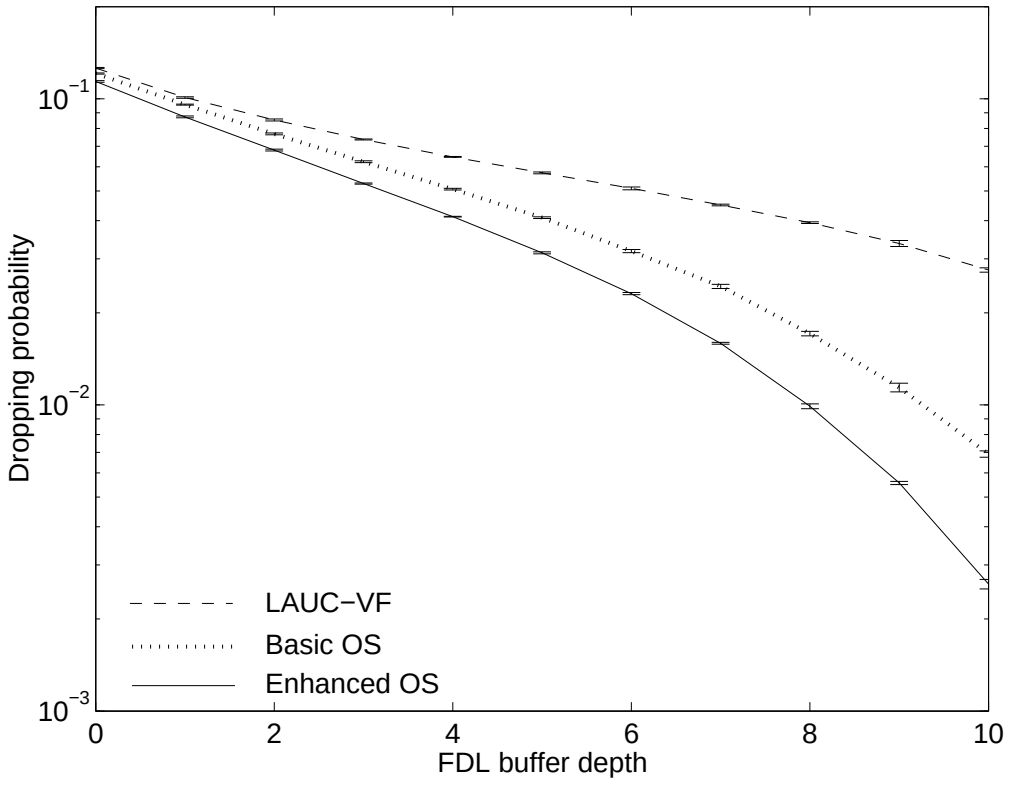
other hand, is not affected due to its deferment of scheduling decisions. The above effect is more pronounced in Figure 3.12(a) than it is in Figure 3.12(b) because having two offset-based QoS classes already introduces significant fragmentation in the scheduling window so the additional fragmentation caused by more FDLs has less effect.

The impact of slot size on the performance of basic Ordered Scheduling is studied next. For this and the remaining experiments, input traffic with two QoS classes is used. In this experiment, the loss performance of basic Ordered Scheduling with different slot sizes is measured at an overall offered load of 0.6 and compared to enhanced Ordered Scheduling and LAUC-VF. The results are plotted in Figure 3.13. Since the performance of the latter two algorithms is not affected by slot size, their loss curves show up as horizontal lines. On the other hand, as the slot size gets larger, the dropping performance of basic Ordered Scheduling rapidly worsens due to its discrete implementation of the admission control test. At a slot size of 1 μs, which is what is used by enhanced Ordered Scheduling, the dropping probability of basic Ordered Scheduling is nearly three orders of magnitude larger than enhanced Ordered Scheduling. These results confirm the necessity to use much smaller slot sizes for basic Ordered Scheduling compared to enhanced Ordered Scheduling.

The final experiment in this section investigates the effects of the number of wavelengths per link on the performance of the algorithms. We include the performance results of a non-void filling scheduling scheme called Horizon in [79] or LAUC in [88]. The purpose is to see how its performance compares to those of other void filling algorithms at different numbers of wavelengths. We measure their dropping probabilities at an overall offered load of 0.8 and different numbers of wavelengths per link and plot the results in Figure 3.14. It shows that the overall trend is decreasing loss probabilities with increasing number of wavelengths per link. This is the direct result of an OBS switch behaving like an $M|M|k|k$ loss system as described in [84, 93]. We also observe that the performance of the Horizon scheme is poor when the number of wavelengths is low but gets very close



(a)



(b)

Figure 3.12: Burst loss probability versus buffer depth: (a) Single traffic class, and (b) Two traffic classes

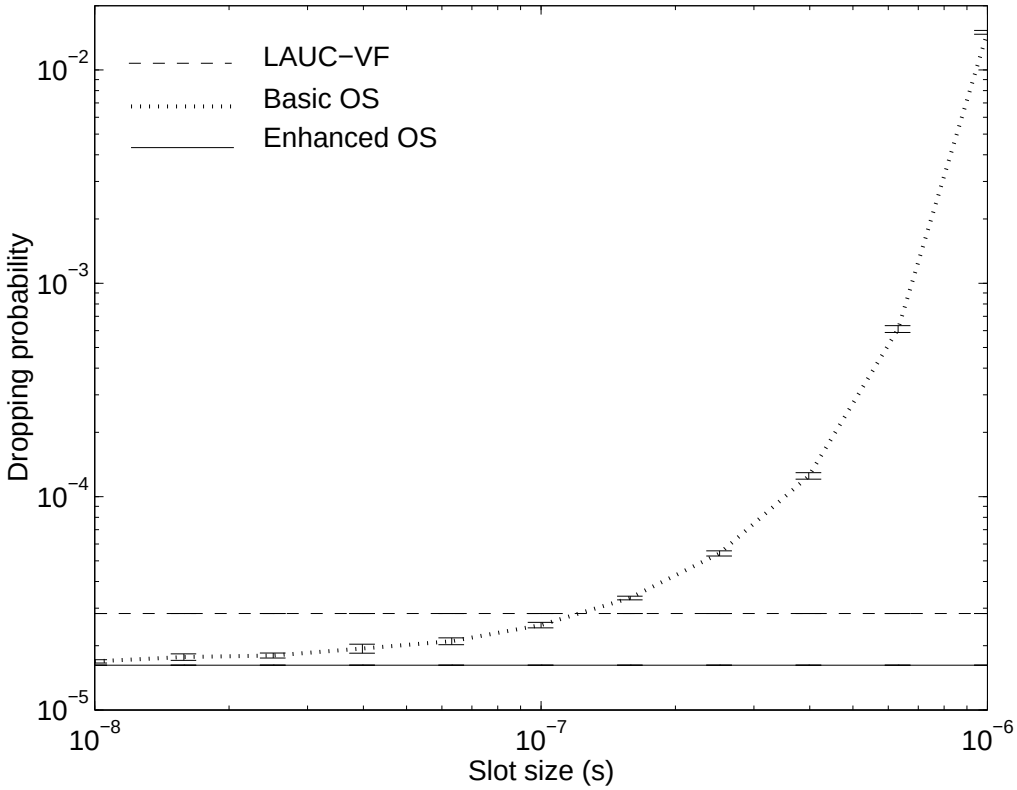


Figure 3.13: Performance of Basic Ordered Scheduling with different slot size

to that of LAUC-VF when the number of wavelengths is high. Among the void filling algorithms, we see that the relative performance between enhanced Ordered Scheduling and LAUC-VF remains the same. However, the relative performance of basic Ordered Scheduling compared to the enhanced version gradually decreases as the number of wavelengths per link increases. This is also due to the discrete nature of basic Ordered Scheduling. As a slot handles more and more bursts, the chance that basic Ordered Scheduling over-reports the number of occupied wavelengths as illustrated in Figure 3.2 increases. From this experiment and the previous one, we note that the performance of Ordered Scheduling depends on the ratio between the number of slots per burst and the number of wavelengths per link.

3.4.3 Simulation study for an entire network

We now simulate the three scheduling algorithms in a realistic network setting to see if the network topology affects the performance trend among the algorithms. For this experiment, the topology in Figure 3.15, which is a simplified topology

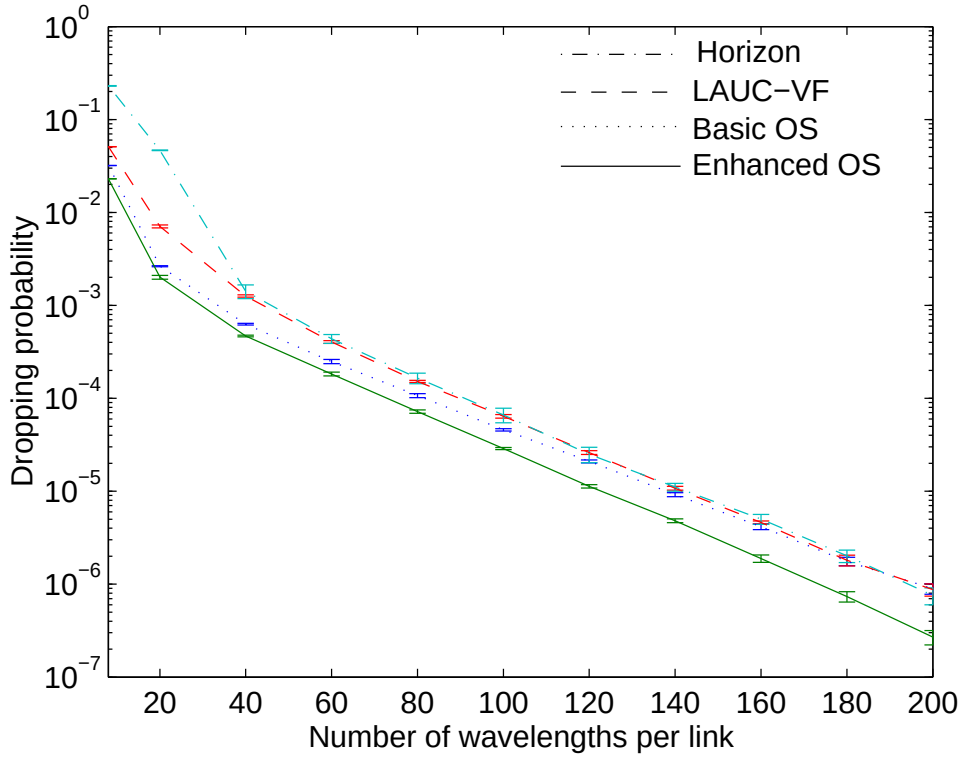


Figure 3.14: Burst loss probability versus number of wavelengths per link

for the US backbone network, is used. The topology consists of 24 nodes and 43 links. The average node degree is 3.6. Shortest path routing is used to determine the transmission paths among nodes and the average hop length of the paths is 3. For simplicity, the propagation delays between adjacent nodes are assumed to have a fixed value of 10 ms. The links are bi-directional, each implemented by two uni-directional links in opposite directions.

The input traffic and hardware configuration for each node remain the same, i.e., two offset-based QoS classes, eight wavelengths per link and six FDLs per output link at each node. Each node has 23 separate assembly queues, one for every other node. An incoming IP packet to a node enters one of the queues with equal probability. The IP packet arrival rates are the same for every node. The header processing time per node is assumed to be $\Delta = 1$ μs. For a path with H hops, the initial offset times assigned for bursts of classes 1 and 2 are $\Delta \cdot H$ μs and $\Delta \cdot H + 3$ μs, respectively.

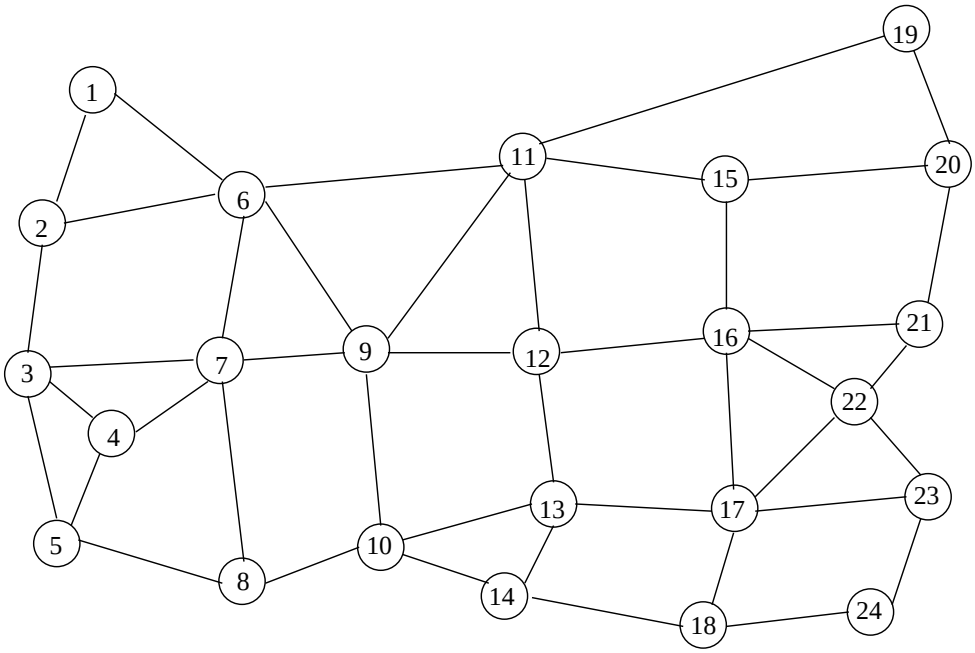


Figure 3.15: 24-node NSF network topology

We plot the dropping probabilities for the algorithms against the offered load to the network. The offered load is measured in terms of the number of departing bursts per node per μs. The simulation results are shown in Figure 3.16. We observe that the performance trend is similar to that in Figure 3.9. The order among the algorithms remains the same, i.e., enhanced Ordered Scheduling has the best performance, followed by basic Ordered Scheduling and LAUC-VF. We also see that the arrival rates used in this experiment are much smaller than those used in Figure 3.9 but the ranges of the dropping probabilities are approximately the same. This is because in a network environment, many paths may converge at some nodes, causing bottlenecks. The offered loads to those bottlenecked nodes are much larger than the average offered load to the network and most of the burst loss in the network is concentrated there.

3.5 Conclusion

In this chapter, we have presented a new scheduling algorithm called Ordered Scheduling for Optical Burst Switching networks with the objective of improving burst dropping performance while keeping the computational and signalling overheads manageable. Two versions of Ordered Scheduling are available: the basic

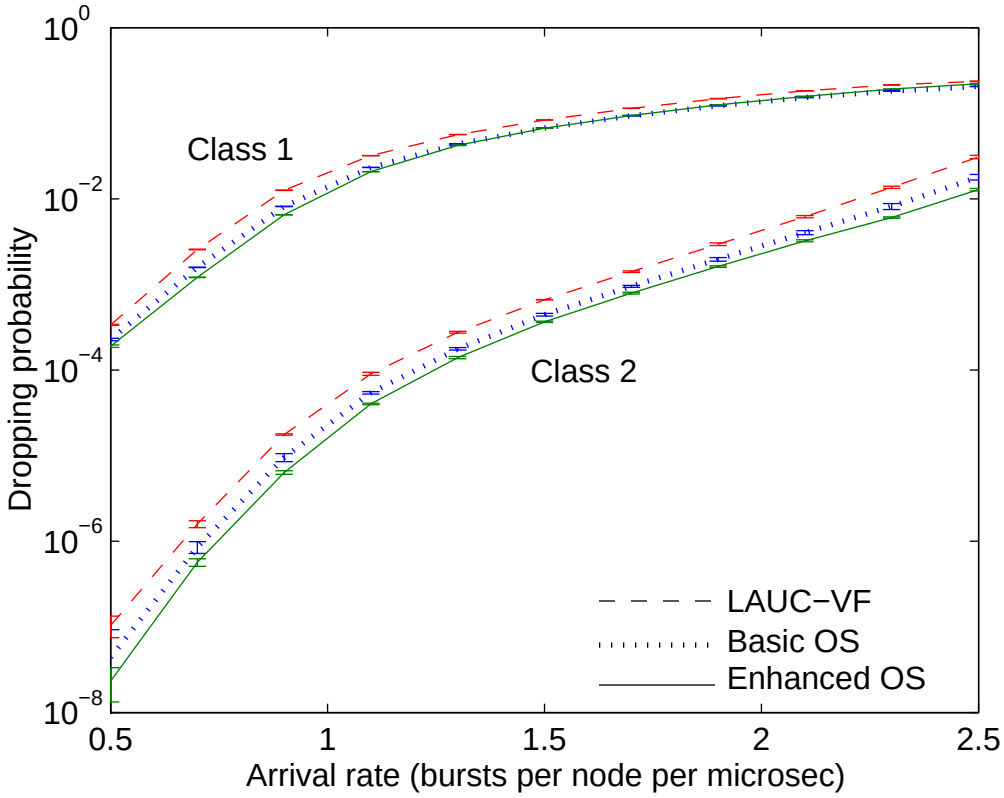


Figure 3.16: Average burst loss probability for NSFNET versus average network load

version with discrete operation and the enhanced version with continuous operation. Extensive simulation studies show that the two versions of Ordered Scheduling outperform LAUC-VF, which is one of the most efficient existing scheduling algorithms, when the scheduling window is large and fragmented. Between the two versions, enhanced Ordered Scheduling always achieves good dropping performance in all traffic conditions and hardware configurations; the performance of basic Ordered Scheduling is dependent on the number of slots per bursts and the number of wavelengths per link.

Implementation and complexity issues are also discussed in the chapter. We show that the proposed algorithm is particularly suitable for parallel implementation due to its slotted structure. When the number of wavelengths per link is large, the computational overhead is shown to be less than LAUC-VF. The proposed algorithm does introduce additional signalling overhead but this is small compared to the gain in dropping performance. Finally, the proposed algorithm

is implemented entirely in the control and signalling domain. Since this is an electronic domain, no significant problem is expected when migrating from other scheduling algorithms.

Chapter 4

An Absolute Quality of Service Framework for Edge-to-Edge Loss Guarantees in Optical Burst-Switched Networks

4.1 Introduction

An important issue in OBS network is how to provide Quality of Service at the optical layer due to the proliferation of multimedia and mission critical Internet applications that require stringent QoS guarantees. In OBS networks, there is only very limited buffering capability in the form of fibre delay lines (FDLs). Therefore, burst loss probability is a primary QoS metric of interest. There are currently two main QoS models in OBS: *relative QoS* and *absolute QoS*. In the relative QoS model, the QoS performance of a traffic class is defined relative to those of other classes. For instance, the loss rate of a higher priority class is guaranteed to be better or at least no worse than that of a lower priority class. On the other hand, the absolute QoS model provides quantitative QoS guarantees for each traffic class, e.g., having an end-to-end packet loss probability no greater than 1%.

From a user's point of view, the absolute QoS model is preferred. This is because an end user usually has a specific quantitative QoS requirement depending on the applications in use and would prefer to receive it independent of the network load. Although it is possible for an Internet Service Provider (ISP) to

emulate this absolute QoS behaviour in a relative QoS environment by continuously adjusting the user's subscribed QoS class, it may not be feasible to do this for all users under heavy load conditions. Besides, the required performance and accounting management functions would be very complex. By placing absolute QoS mechanisms at core nodes, which have the most updated information about their own traffic conditions, the absolute QoS model can offer more accurate and robust quantitative QoS guarantees in simpler ways than the relative QoS model.

To date, a number of absolute QoS proposals have been put forward [34,59,98]. However, they primarily focus on providing absolute QoS differentiation, which allows absolute loss thresholds to be maintained for traffic classes at each core node. This is only a first step towards realising edge-to-edge (e2e) absolute loss guarantees. In addition to absolute QoS differentiation, a full-fledged absolute QoS framework requires at least an admission control mechanism to limit input traffic to the network and a signalling protocol to coordinate the reservation along an e2e path.

In this chapter, we present a novel absolute QoS framework to provide quantitative e2e loss guarantees to burst flows in an OBS network. It defines a limited number of per-hop QoS classes and assigns each class a loss threshold. The framework employs a preemptive differentiation mechanism and an admission control mechanism at each output link of a core node. The differentiation mechanism shifts burst loss from classes in danger of breaching their thresholds to other classes while the admission control mechanism limits the link's offered load to a certain level. They work together to guarantee the loss threshold for each class at the link. Using these classes as building blocks, the framework employs a signalling and reservation mechanism to assign each burst flow to a certain class at each intermediate link such that the flow's e2e loss probability request is satisfied. The framework is shown to achieve reliable bounds on e2e loss probabilities under all traffic loads.

The rest of the chapter is organised as follows. In §4.2, we present an overview of the QoS framework. The preemptive differentiation scheme and the node-based

admission control scheme are presented in detail in §4.3. This section includes an analysis of the differentiation scheme. The e2e signalling and reservation scheme is presented next in §4.4. In §4.5, we compare our framework with existing absolute QoS proposals. The differentiation scheme and the entire system are evaluated through simulation in §4.6. Finally, concluding remarks are given in §4.7.

4.2 Overview of the Framework

Our absolute QoS framework is applied to a burst flow with a required maximum e2e loss probability over a pre-determined path. It aims to reserve resources over the path so that the required e2e loss probability of the flow is guaranteed. If the reservation process fails for that particular path, a routing protocol with route-pinning capability may be used to select another path and the reservation process is applied again. As such, the framework should be considered as part of a final solution to the QoS provisioning problem in OBS networks.

The key idea of the proposed framework is to define a limited number of per-hop absolute QoS classes[1] and enforce their loss thresholds at each link. The network then divides the required e2e loss probability of the flow into a series of small loss probabilities and maps them to the available thresholds at the intermediate links on the path. When each intermediate node guarantees that the actual loss probability at its link is below the allocated loss probability, the overall e2e loss guarantee is fulfilled.

The proposed QoS framework includes two mechanisms to enforce per-hop thresholds for individual flows, i.e., a preemptive absolute QoS differentiation mechanism and an admission control mechanism. The differentiation mechanism allows bursts from classes that are in danger of breaching their thresholds to preempt bursts from other classes. Thus, burst loss is shifted among the classes based on the differences between the thresholds and the measured loss probabilities of the classes. The differentiation mechanism is also designed such that individual

[1]In the rest of this chapter, the term "class" refers to per-hop QoS class unless otherwise specified.

flows within a single class experience uniform loss probability. Hence, even though it works at the class level, its threshold preserving effect extends to the flow level. The admission control mechanism limits the link's offered load to an acceptable level and thereby makes it feasible to keep the loss probabilities of all classes under their respective thresholds.

For the mapping of classes over an e2e path, we assume a label switching architecture such as Multi-Protocol Label Switching (MPLS) [69] to be present in the OBS network. In this architecture, each burst header carries a label to identify the LSP that it belongs to. When a header arrives at a core node, the node uses the header's label to look up the associated routing and QoS information from its Label Information Base (LIB). The old label is also swapped with a new one. Label information is downloaded to the node in advance by a Label Distribution Protocol (LDP). Such label switching architecture enables an LSP to be mapped to different QoS classes at different links.

An e2e signalling and reservation mechanism is responsible for probing the path of a new LSP and mapping it to a class at each intermediate link. When the LSP setup process begins, a reservation message that contains the requested bandwidth and the required e2e loss probability of the LSP is sent along the LSP's path toward the egress node. The message polls intermediate nodes on their available capacity and conveys the information to the egress node. Based on this information, the egress node decides whether the LSP's request can be accommodated. If the result is positive, an array of QoS classes whose elements correspond to the links along the path is allocated to the LSP. The class allocation is calculated such that the resulting e2e loss probability is not greater than that required by the LSP. It is then signalled to the intermediate core nodes by a returned acknowledgment message.

Finally, existing LSPs are policed for conformance to their reservations at ingress nodes. When the traffic of an LSP exceeds its reserved traffic profile, its generated bursts are marked as out of profile. Such bursts receive best-effort service inside the network.

4.3 A Preemptive Scheme for Absolute QoS Differentiation

4.3.1 Description

In this section, we describe a preemptive absolute QoS differentiation scheme for the framework. To quantify the risk of breaching the threshold of a local QoS class, we introduce a metric called the *distance to threshold*, which is defined as the difference between the predefined loss threshold and the measured loss probability. In order to achieve the objectives set out in the previous section, i.e., loss shifting among classes and uniform loss probability for flows within a class, the differentiation scheme must meet the following requirements:

- *Inter-class requirement:* It must ensure that as the offered load to a link increases, the distances to thresholds of all classes present at the link converge to zero. This implies that burst loss from classes that are in danger of breaching their thresholds is shifted to other classes by the differentiation scheme.
- *Intra-class requirement:* It must ensure that bursts belonging to the same class experience the same loss probability at a particular link regardless of their offsets and burst lengths. In OBS networks, it is well-known that burst lengths and offsets have significant impacts on burst loss probability. Hence, without intervention from the differentiation scheme, some flows with unfavourable burst characteristics may experience loss probabilities above the threshold even though the overall loss probability of the class is still below the threshold.

Having set out the requirements and their rationales, we will now describe the proposed differentiation scheme. For scalability reasons, the scheme uses class-level differentiation. The scheme only requires a core node to keep per-class information, which includes the predefined loss threshold, the amount of admitted traffic and the current average loss probability. The average loss probability is continuously updated using an exponentially weighted averaging algorithm.

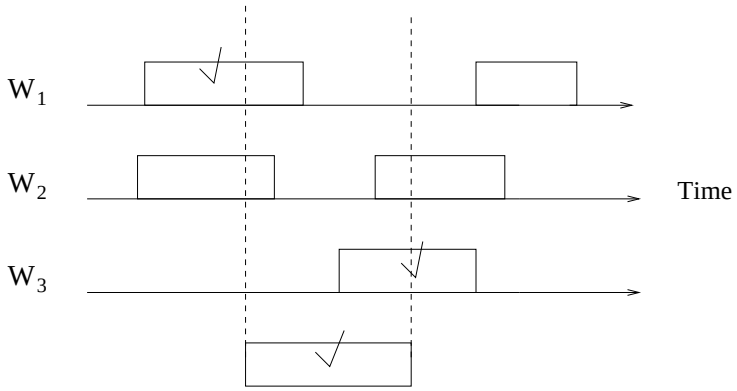


Figure 4.1: Construction of a contention list

The scheme works as follows. When a burst header arrives at a node and fails to reserve an output wavelength, the node constructs a *contention list* that contains the incoming burst reservation and scheduled burst reservations that overlap (or contend) with the incoming one. Only one scheduled reservation on each wavelength is included if its preemption helps to schedule the new reservation. This is illustrated in Figure 4.1 where only the ticked reservations among the ones overlapping with the incoming reservation on wavelengths W_1, W_2 and W_3 are included in the contention list. The node then selects one reservation from the list to drop according to some criteria described later. If the dropped reservation is a scheduled one then the incoming reservation will be scheduled in its place. In that case, we say that the incoming reservation *preempts* the scheduled reservation.

When preemption happens, a special NOTIFY packet will be immediately generated and sent on the control channel to the downstream nodes to inform them of the preemption. The downstream nodes then remove the burst reservation corresponding to the preempted burst. Although one NOTIFY packet is required for every preemption, the rate of preemption is bounded by the loss rate, which is usually kept very small. Therefore, the additional overhead by the transmission of NOTIFY packets is not significant.

There are two criteria for selecting a burst reservation from the contention list to drop. The first criterion is that the selected reservation belongs to the class with the largest distance to threshold in the contention list. This criterion

ensures that all the distances to thresholds of the classes present at the node are kept equal, thereby satisfying the first requirement above. The second criterion is applied when there are more than one reservation belonging to the class with the largest distance to threshold. In that case, only one of them is selected for dropping. Let the length of the ith reservation be $l_i, (1 \leq i \leq N)$, where N is the number of reservations belonging to the class with the largest distance to threshold in the contention list. The probability of it being dropped is

$$p_i^d = \frac{\frac{1}{l_i}}{\sum_{j=1}^{N} \frac{1}{l_j}}. \tag{4.1}$$

The rationale is that the probability that a reservation is involved in a contention is roughly proportional to its length, assuming Poisson burst arrivals. So p_i^d is explicitly formulated to compensate for that burst length selection effect. In addition, the selection is independent of burst offsets. That is, although a large-offset burst is less likely to encounter contention when its header first arrives, it is as likely to be preempted as other bursts in subsequent contention with shorter-offset bursts. Therefore, the second requirement is achieved.

The above description assumes that no FDL buffer is present. It can be trivially extended to work with FDL buffers by repeating the preemption procedure for each FDL and the new reservation interval.

4.3.2 Analytical Model

In this section, we derive the overall loss probability for the preemptive differentiation scheme. Both the lower and upper bounds and an approximate formula for the loss probability will be derived. Depending on the application's requirement, one can choose the most suitable formula to use.

The following assumptions are used in the analysis. Firstly, for the sake of tractability, only one QoS class is assumed to be active, i.e., having traffic. The simulation results in Figure 4.3 indicate that the results obtained are also applicable to the case with multiple classes. Secondly, burst arrivals follow a Poisson

process with mean rate λ. This is justified by the fact that a link in a core network usually has a large number of traffic flows and the aggregation of a large number of independent and identically distributed point processes results in a Poisson point process. Thirdly, the incoming traffic consists of a number of traffic components with the ith component having a constant burst length $1/\mu_i$ and arrival rate λ_i. This assumption results from the fact that size-triggered burst assembly is a popular method to assemble bursts. This method produces burst lengths with a very narrow dynamic range, which can be considered constant. Finally, we assume that no FDL buffer is present and the offset difference among incoming bursts is minimal.

The lower bound on loss probability is easily derived by observing that preemption itself does not change the total number of lost bursts in the system. Thus, it is determined using Erlang's loss formula for an $M|G|k|k$ queueing model [44] as follows:

$$P_l = B(k, \rho) = \frac{\frac{r^k}{k!}}{\sum_{i=0}^{k} \frac{r^i}{i!}}, \tag{4.2}$$

where k is the number of wavelengths per output link; ρ is the total offered load and $r = k\rho$.

Although preemption does not directly affect the number of lost bursts, it affects the probability that a burst whose header arrives later is successfully scheduled. Depending on the reservation intervals of later bursts, the preemption may have detrimental or beneficial effects. Consider a preemption scenario as illustrated in Figure 4.2 where burst 1 is preempted by burst 2. Let bursts 3 and 4 be two bursts whose headers arrive after the preemption. For burst 3, the preemption is detrimental because had there been no preemption, burst 3 would be successfully scheduled. On the other hand, the preemption is beneficial to burst 4. However, for that to happen, burst 4 has to have a considerably shorter offset than other bursts, which is unlikely due to our assumption that the offset difference among bursts is minimal. For other preemption scenarios, it can also be demonstrated that a considerable offset difference is required for a preemption to have

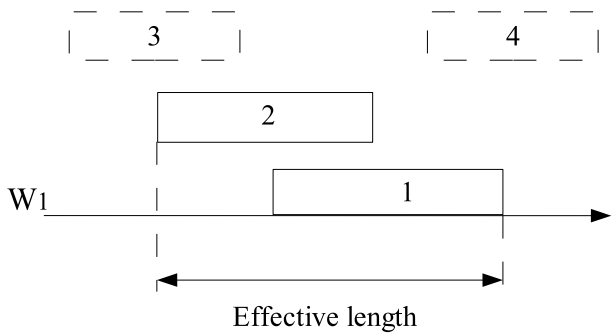


Figure 4.2: Example of a preemption scenario.

beneficial effects. Therefore, it can be argued that preemption generally worsens the schedulability of later bursts.

To quantify that effect, we observe that from the perspective of burst 3, the preemption is equivalent to dropping burst 2 and extending the effective length of burst 1 as in Figure 4.2. Therefore, it increases the time that the system spends with all k wavelengths occupied. The upper bound on burst loss probability is derived by assuming that the loss probability is also increased by the same proportion. Denote

$$\delta = \frac{(1/\mu' - 1/\mu)}{1/\mu} = \frac{\mu}{\mu'} - 1$$

where $1/\mu'$ is the new effective length and $1/\mu$ is the actual length of the preempted burst. The upper bound on loss probability is then given as

$$P_u = \begin{cases} (1+\delta\rho)B(k,\rho) & \text{if } B(k,\rho) < \frac{1}{1+\delta\rho} \\ 1 & \text{otherwise} \end{cases} . \tag{4.3}$$

An approximate formula for the loss probability can be derived based on (4.3) by observing that the increase in effective length of a preempted burst will increase the overall loss probability only if another incoming burst contends with it again during the extended duration. The probability that this does not happen is

$$p = \sum_{i=0}^{\infty} \frac{e^{-\delta r}(\delta r)^i}{i!} \left(\frac{k}{k+1}\right)^i = e^{-\frac{\delta r}{k+1}}. \tag{4.4}$$

From (4.3) and (4.4), the loss probability is given as

$$P = P_u - e^{-\frac{\delta r}{k+1}} \delta \rho B(k, \rho). \tag{4.5}$$

We will now derive δ. Suppose the incoming traffic has N_c traffic components with N_c different burst lengths. Let a and b denote the component indices of the incoming burst and the preempted burst, respectively. The probability of a particular combination (a, b) is given by the formula

$$P(a,b) = \frac{\lambda_a}{\sum_{i=1}^{N_c} \lambda_i} \cdot \frac{\rho_b}{\sum_{j=1}^{N_c} \rho_j} \cdot \frac{\mu_b}{\sum_{k=1}^{N_c} \mu_k} \qquad (1 \le a, b \le N_c). \tag{4.6}$$

The first and second factors are the probabilities that an incoming burst and a scheduled burst belong to components a and b, respectively. The third factor accounts for the length selective mechanism of the preemption scheme. For a preemption situation (a,b), the effective length is increased by $\frac{1}{\mu_a} - \frac{1}{2\mu_b}$. Therefore, it follows that

$$\delta = \sum_{a=1}^{N_c} \sum_{b=1}^{N_c} P(a,b) \left(\frac{\mu_b}{\mu_a} - \frac{1}{2} \right). \tag{4.7}$$

4.3.3 Local Admission Control at a Link

Since the distances to thresholds of the classes at a node are kept equal by the differentiation scheme, the admission control routine only needs to keep the average of these greater than zero. In other words, it needs to keep the overall loss probability smaller than the weighted average threshold. Suppose there are M QoS classes at the node and let T_i and B_i be the predefined threshold and the total reserved bandwidth of the ith class, respectively. The weighted average threshold is calculated as

$$T = \frac{\sum_{i=1}^{M} T_i B_i}{\sum_{j=1}^{M} B_j}. \tag{4.8}$$

The overall loss probability P can be calculated using (4.5) in the previous section. Alternatively, the upper bound given by (4.3) may be used for better protection

against threshold violation. In case that the analytical formulas cannot be used, e.g., due to non-Poisson traffic, an empirical graph of the overall loss probability versus the total offered load may be used.

A reservation request will contain the amount of bandwidth to be reserved b_0 and the QoS class c to accommodate b_0 in. When a request arrives, the admission control routine substitutes B_c with $B'_c = B_c + b_0$ and recalculates the weighted average threshold T' and the overall loss probability P' as above. If $P' \leq T'$, the request is admitted. Otherwise, it is rejected.

4.3.4 Per-Hop QoS Class Definition

Since per-hop QoS classes are used as building blocks by the network to construct quantitative e2e loss guarantees, their definition is an important part of configuring the system. Usually, the number of classes M, which is directly related to the complexity of a core node's QoS differentiation block, is fixed. Hence, in this process, one only decides on where to place the available thresholds, namely the lowest and highest loss thresholds T_l and T_h and those between them.

Consider an OBS network in which LSPs have a maximum path length of H hops and a required e2e loss guarantee between P_l and P_h (not counting best-effort and out-of-profile traffic). The case requiring the lowest loss threshold T_l occurs when an LSP over the longest H-hop path requires P_l. Thus, T_l can be calculated as follows

$$T_l = 1 - (1 - P_l)^{1/H}.$$

Similarly, the highest threshold is $T_h = P_h$ for the case when a one-hop LSP requires P_h.

When considering how to place the remaining thresholds between T_l and T_h, it is noted that since the potential required e2e loss probability P_0 is continuous and the threshold values are discrete, the e2e loss bound P_{e2e} offered by the network will almost always be more stringent than P_0. This "discretization error" reduces the maximum amount of traffic that can be admitted. Therefore, the thresholds need to be spaced so that this discretization error is minimised. A simple and

effective way to do this is to distribute the thresholds evenly on the logarithmic scale. That is, they are assigned the values $T_l,\ \gamma T_l,\ \gamma^2 T_l, \ldots, \gamma^{M-1} T_l$, where $\gamma = (T_h/T_l)^{1/(M-1)}$.

4.4 Edge-to-Edge Signalling and Reservation

4.4.1 Description

Edge-to-edge signalling and reservation mechanisms, as the name implies, are responsible for coordinating the QoS reservation setup and teardown for LSPs over the e2e paths. During the reservation process of an LSP, the signalling mechanism polls all the intermediate core nodes about the remaining capacity on the output links and conveys the information to the egress node. Using that information as the input, the egress node produces a class allocation that maps the LSP to an appropriate class for each link on the path. The signalling mechanism then distributes the class allocation to the core nodes. As a simple illustration, let us consider an LSP with an e2e loss requirement of 5% that needs to be established over a 4-hop path and the second hop is near congestion. The network allocates the LSP a threshold of 3.2% for the second hop and 0.4% for the other hops to reflect the fact that the second node is congested. The resulting guaranteed upper bound on e2e threshold will roughly be 4.4%, satisfying the LSP's requirement.

The QoS requirements of an LSP consists of its minimum required bandwidth and its maximum e2e loss probability. As new IP flows join an LSP or existing IP flows terminate, a reservation or teardown process needs to be carried out for the LSP. The reservation scenarios for an LSP can be categorised as follows.

1. A new LSP is to be established with a specified minimum bandwidth requirement and a maximum e2e loss probability. This happens when some IP flow requests arrive at the ingress node and cannot be fitted into any of the existing LSPs.
2. An existing LSP needs to increase its reserved bandwidth by a specified amount. This happens when some incoming IP flows have e2e loss requirements compatible with that of the LSP.

3. An existing LSP needs to decrease its reserved bandwidth by a specified amount. This happens when some existing IP flows within the LSP terminate.
4. An existing LSP terminates because all of its existing IP flows terminate.

The detailed reservation process for the first scenario is as follows. The ingress node sends a reservation message towards the egress node over the path that the LSP will take. The message contains a requested bandwidth b_0 and a required e2e loss probability P_0. When a core node receives the message, its admission control routine checks each class using the method described in §4.3.3 to see if the requested bandwidth can be accommodated in that class. The check starts from the lowest index class, which corresponds to the lowest threshold, and moves up. The node stops at the first satisfactory class and records in the message the class index c and a parameter κ calculated as follows

$$\kappa = \left\lfloor log_\gamma \left(\frac{T'}{P'} \right) \right\rfloor \tag{4.9}$$

where T' and P' are as described in §4.3.3 and γ is the ratio between the thresholds of two adjacent classes. These parameters will be used by the egress node for the final admission control and class allocation. The message is then passed downstream. The node also locks in the requested bandwidth by setting the total reserved bandwidth B_c of class c as $B_c(new) = B_c(old) + b_0$ so that the LSP will not be affected by later reservation messages. On the other hand, if all the classes have been checked unsuccessfully, the request is rejected and an error message is sent back to the ingress node. Upon receiving the error message, the upstream nodes release the bandwidth locked up earlier.

The final admission control decision for the LSP is made at the egress node. The received reservation message contains two arrays $\boldsymbol{c}$ and $\boldsymbol{\kappa}$ for the intermediate links of the path. Assuming burst blocking at each link is independent, the lowest

possible e2e loss probability P^0_{e2e} given as

$$P^0_{e2e} = 1 - \prod_{i=1}^{n}(1 - p^0_i) \tag{4.10}$$

where p^0_i is the lowest threshold offered by the ith node on a n-node path. If $P^0_{e2e} \leq P_0$, the request is admitted. The egress node then allocates each core node one of the predefined classes in which to support the LSP such that

$$\begin{cases} p_i \geq p^0_i \\ P_{e2e} = 1 - \prod_{i=1}^{n}(1 - p_i) \leq P_0 \end{cases} \tag{4.11}$$

where p_i is the threshold of the class allocated to the LSP at the ith node and P_{e2e} is the corresponding e2e loss probability. The class allocation algorithm will be described in the next section. This class allocation is signalled back to the intermediate core nodes using a returned acknowledgment message that contains the old index array $\boldsymbol{c}$ and an allocated index array $\boldsymbol{c_a}$. Upon receiving the acknowledgment message, a core node moves the reserved bandwidth of the LSP from class c to class c_a. The new LSP is allowed to start only after the ingress node has received the successful acknowledgment message. If $P^0_{e2e} > P_0$, the request is rejected and an error message is sent back to the ingress node. The intermediate core nodes will release the locked bandwidth upon receiving the error message.

The reservation process for the second scenario is relatively simpler. In this case, the ingress node sends out a reservation message containing the requested bandwidth b_0 and the LSP's label. Since there is already a QoS class associated with the LSP at each of the core nodes, a core node on the path only needs to check if b_0 can be supported in the registered class. If the outcome is positive, the node locks in b_0 and passes the reservation message on. Otherwise, an error message is sent back and the upstream nodes release the bandwidth locked previously. If the reservation message reaches the egress node, a successful acknowledgment message

is returned to the ingress node and the LSP is allowed to increase its operating bandwidth.

In the last two scenarios, the reservation processes are similar. The ingress node sends out a message carrying the amount of bandwidth with a flag to indicate that it is to be released and the LSP's label. The released bandwidth is equal to the reserved bandwidth of the LSP if the LSP terminates. At intermediate core nodes, the total reserved bandwidth of the class associated with the LSP is decreased by that amount. No admission control check is necessary. Since the core nodes do not keep track of bandwidth reservation by individual LSPs, the processing at core nodes is identical for both scenarios. It should be noted that when an LSP terminates, there is a separate signalling process to remove the LSP's information from core nodes' LIBs. However, it is not considered part of our QoS framework.

4.4.2 Dynamic Class Allocation

When the egress node has determined that the LSP request is admissible using (4.10), it uses a dynamic class allocation algorithm to find the bottleneck link and allocate the class with the highest possible threshold to it while still satisfying (4.11). This shifts some of the loss guarantee burden from the bottleneck link to other lightly loaded links. Since the remaining capacity of the path is determined by the bottleneck link, the algorithm will maximise the path's remaining capacity and allows more future QoS traffic to be admitted.

For this purpose, the egress node has at its disposal two arrays $\boldsymbol{c}$ and $\boldsymbol{\kappa}$. Note that $\boldsymbol{\kappa}[i] > 0$ only if $\boldsymbol{c}[i] = 0$. We can see from (4.9) that $\boldsymbol{\kappa}[i]$ indicates the distance between T' and P' for node i in logarithmic scale when $\boldsymbol{c}[i] = 0$ and cannot be decreased further. In other words, $\boldsymbol{c} - \boldsymbol{\kappa}$ indicates the remaining capacity at the intermediate links. The higher $\boldsymbol{c}[i] - \boldsymbol{\kappa}[i]$, the lower the remaining capacity at link i and vice versa. Based on this observation, the class allocation algorithm is detailed in Algorithm 2. In executing this algorithm, negative class indices in $\boldsymbol{c_a}$ are counted as zero. In the first two lines, the algorithm sets $\boldsymbol{c_a}$ such that the

maximum element is $M-1$ and the differences among the elements are the same as in the array $\boldsymbol{c}-\boldsymbol{\kappa}$. Next, it repeatedly decrements all the elements of $\boldsymbol{c_a}$ until $P_{e2e} \leq P_0$. Finally, the elements of $\boldsymbol{c_a}$ are incremented one by one until just before $P_{e2e} > P_0$ in order to push P_{e2e} as close as possible to P_0 without exceeding it.

Algorithm 2: Class allocation algorithm

(1) $d_{max} \leftarrow \textsc{Max}(\boldsymbol{c}-\boldsymbol{\kappa})$

(2) $\boldsymbol{c_a} \leftarrow (\boldsymbol{c}-\boldsymbol{\kappa}) - d_{max} + M - 1$

(3) **while** $P_{e2e} > P_0$

(4) $\quad \boldsymbol{c_a} \leftarrow \boldsymbol{c_a} - 1$

(5) Increment elements of $\boldsymbol{c_a}$ one by one until just before $P_{e2e} > P_0$

As an illustrative example, consider an OBS network that has 8 predefined QoS classes with indices $\{0, 1, \ldots, 7\}$. The lowest threshold is $T_l = 0.05\%$ and the ratio between two adjacent thresholds is $\gamma = 2$. An LSP with an e2e loss requirement of 1% is to be set up over a three-hop path. Its required bandwidth is assumed to be very small compared to the link capacity. The utilisation levels at the intermediate links are $\{0.3, 0.6, 0.35\}$. Suppose the received message at the egress node contains $\boldsymbol{c} = \{0, 0, 0\}$ and $\boldsymbol{\kappa} = \{50, 2, 35\}$. Going through the algorithm, we have $\boldsymbol{c_a} = \{-41, 7, -26\}$ on line (3) and $\boldsymbol{c_a} = \{-44, 7, -29\}$ on line (5). The final result is $\boldsymbol{c_a} = \{1, 4, 1\}$ corresponding to thresholds of $\{0.1\%, 0.8\%, 0.1\%\}$. It shows that the algorithm successfully allocates the maximum possible class index to the bottleneck node.

4.5 Comparison with Existing Proposals

In this section, we will compare our absolute QoS framework with existing proposals on two parts, namely, the local mechanisms at a core node to guarantee per-hop thresholds and the e2e mechanisms to achieve e2e absolute guarantees.

For the first part, we have specified in §4.3.1 two requirements that must be met by an absolute QoS differentiation scheme and their rationales. Almost all of the differentiation approaches surveyed in §2.3 except the offset-based approach

can satisfy the inter-class requirement with minor modifications. For example, the intentional dropping approach in [98] and the preemption approach in [34, 59] are used to implement absolute QoS differentiation. However, none of them except our proposed scheme considers the intra-class requirement, which specifies that all LSPs within a single class should experience a uniform loss probability. Therefore, LSPs with unfavourable traffic characteristics such as short offset times or large burst lengths may experience loss probabilities beyond the threshold even though the threshold is preserved at the class level.

Link-based admission control is also missing from all existing absolute QoS differentiation proposals [34, 59, 98]. Reference [98] briefly mentions that edge-based admission control can be used but no further elaboration is provided. The other proposals do not mention it at all. Without local admission control, a core node cannot reliably guarantee the thresholds for traffic classes in heavy loading conditions.

For the second part, references [59,98] attempt to achieve e2e absolute guarantees. In these schemes, e2e QoS classes with absolute loss thresholds are defined first. New LSPs will choose a suitable e2e class in which to transmit. For each e2e threshold P_0^{end}, core nodes are given equal per-hop thresholds, which are calculated as

$$p_0^i = 1 - (1 - P_0^{end})^{1/H} \tag{4.12}$$

where H is the maximum hop length of a network path. The scheme in [98] further improves on this by grouping possible network paths into clusters based on their hop lengths and uses the maximum hop length H_c in a cluster instead of H in equation (4.12).

The above approach has two major problems. Firstly, it leads to inefficiency since links along a path are usually not equally loaded. Therefore, the ability to accept new traffic of a path will be limited by the most heavily loaded link. Secondly, it implies that the number of thresholds a core node has to support is $M = N \times H$, where N is the number of e2e loss guarantees. Hence, M is many

times as large as N, which is not desirable as core nodes have to process headers at very high speeds. The use of path clustering in [98] somewhat alleviates this problem but does not completely solves it. Our proposal solves both problems by defining per-hop classes first and using these as building blocks to provide e2e loss guarantees. Together with a label switching architecture, this enables the network to tailor the threshold allocation based on traffic loading at intermediate links. Also, a small number of predefined per-hop classes can be arbitrarily combined to generate a large number of e2e loss guarantees.

4.6 Experimental Study

4.6.1 Absolute QoS Differentiation

In this section, we evaluate the proposed absolute differentiation algorithm against the two criteria in §4.3.1 and verify the analytical results obtained in §4.3.2 through simulation at the node level.

The node in the simulation has an output link with 64 data wavelengths, each having a transmission rate of 10 Gbps. We assume that the node has full wavelength conversion capability and no buffering. Bursts arrive at the link according to a Poisson process with rate λ. This Poisson traffic assumption is valid for core networks due to the aggregation effect of a large number of flows per link. The burst lengths are generated by a size-limited burst assembly algorithm with a size limit of 50 kB. Thus, the generated bursts have lengths between 50 kB and 51.5 kB, or between 40 μs and 41.2 μs.

In the first experiment, we wish to verify the accuracy of the analysis in §4.3.2. For this purpose, we plot the overall loss probabilities of traffic with one QoS class, traffic with seven QoS classes and the analytical value against the overall loading. In the case with seven classes, the classes are configured with thresholds ranging from $T_l = 0.0005$ to $T_h = 0.032$ and the ratio between two adjacent thresholds is $\gamma = 2$. The traffic of the highest threshold class takes up 40% of the total traffic. For each of the remaining classes, their traffic takes up 10% of the total traffic.

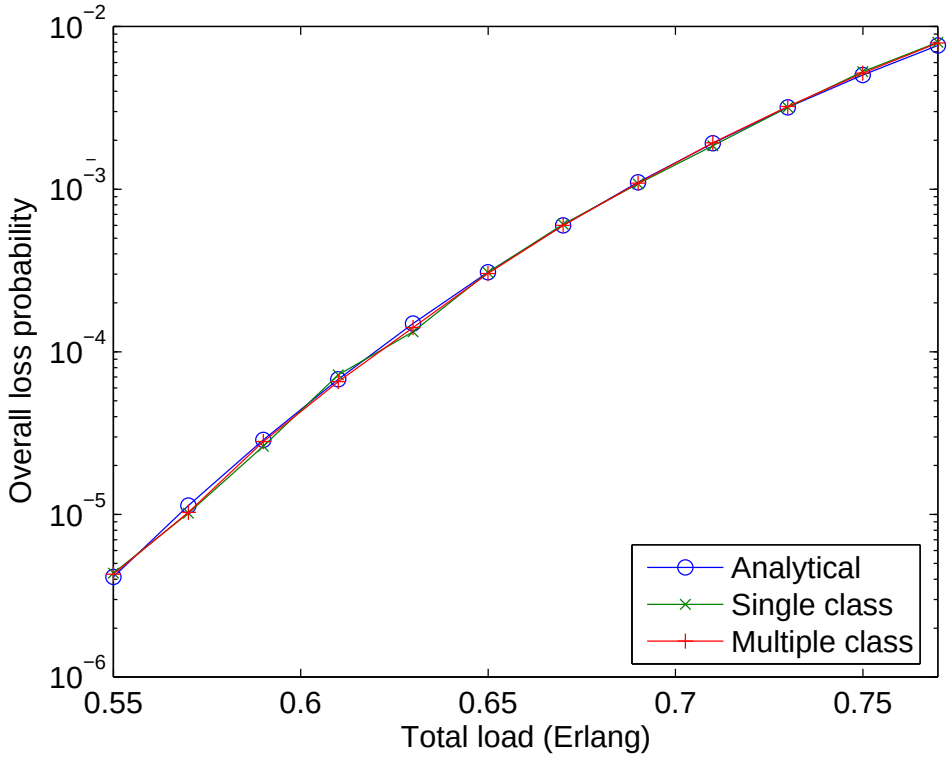


Figure 4.3: Validation of analytical result for different number of traffic classes

From the plot in Figure 4.3, we observe that all the three loss curves match one another very well. It shows that the analysis is accurate and its assumption is valid, i.e., the traffic mix does not affect the overall loss probabilities. The reason is that in our differentiation scheme, preemption potentially happens whenever there is a contention between bursts, regardless of whether they are of different classes or of the same class. Therefore, the number of lost bursts depends only on the number of burst contentions but not the traffic mix.

In the next experiment, the loss probabilities of individual classes are plotted against the overall loading in Figure 4.4. For easy visualisation, we use only two QoS classes. Class 0 has a threshold of 0.005 and takes up 20% of the overall traffic. Class 1 has a threshold of 0.01. We observe that as the loading increases, the loss probabilities of both classes approach their corresponding thresholds. It shows that the algorithm satisfies the first criterion set out in §4.3.1. In addition, the distances to thresholds are always kept equal except when the loss probability of class 0 becomes zero. Although this feature is not required by the model, it is

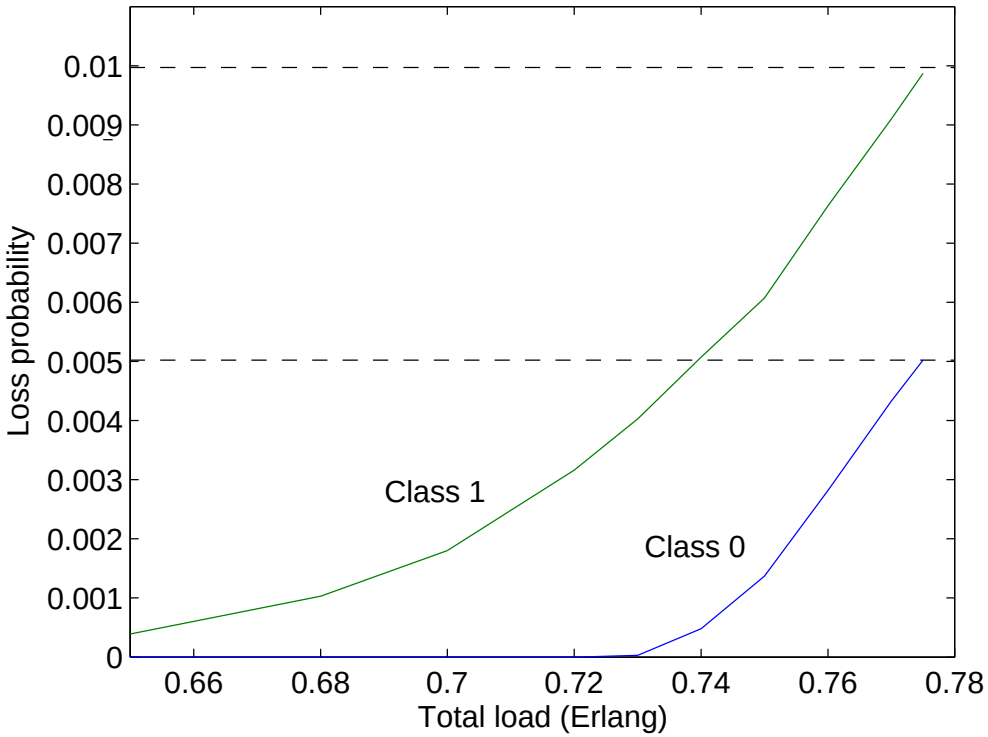


Figure 4.4: Burst loss probabilities of individual classes vs total offered load

introduced to keep the minimum of the distances to thresholds as large as possible, thereby reducing the possibility that a sudden increase in incoming traffic could take the loss probability of a class beyond its threshold.

In Figure 4.5, the loss performance of traffic components with different traffic characteristics within a class is investigated. The overall loading is 0.77 and the class configuration is the same as in the previous experiment. Each class has two traffic components in equal proportion. The plot in Figure 4.5(a) is for the situation where the traffic components have different offsets. The offset difference is 40 μs, which is approximately one burst length. In Figure 4.5(b), each class has two traffic components with different burst lengths. The size limits for the two components in the burst assembly algorithms are set at 50 kB and 100 kB, respectively. These settings would cause major differences in loss performance of the traffic components in a normal OBS system. Nevertheless, both figures show that the loss performance of different components within a class follows each other very closely despite the difference in their burst characteristics. It can

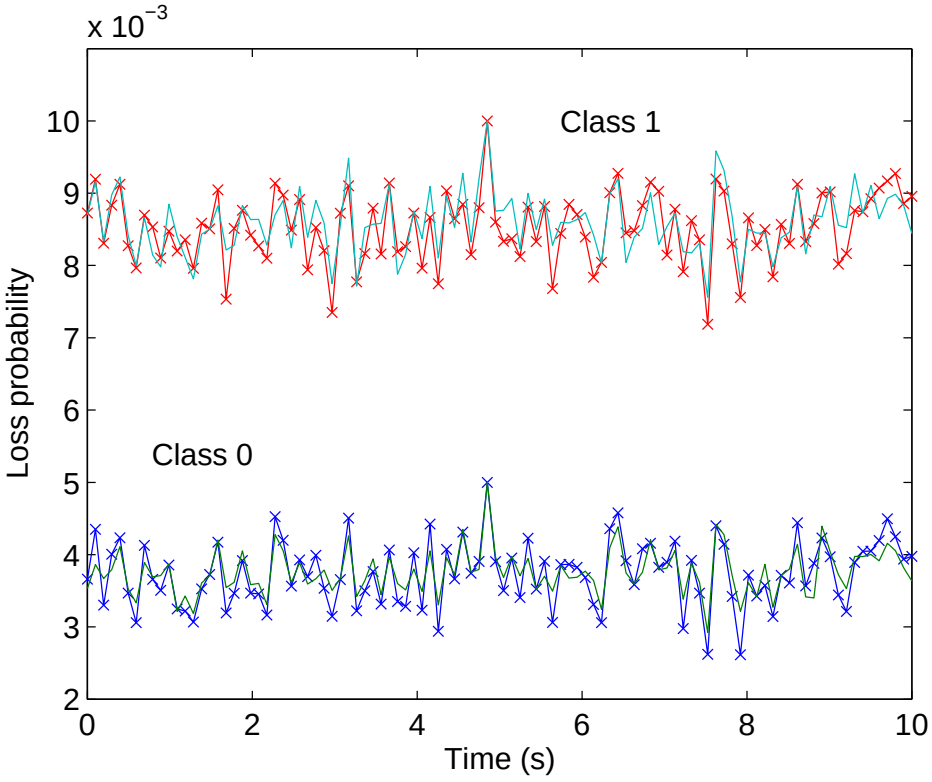
be concluded that the proposed differentiation scheme can achieve uniform loss performance for individual flows within the same class as required by the second criterion in §4.3.1.

4.6.2 Edge-to-Edge Reservation

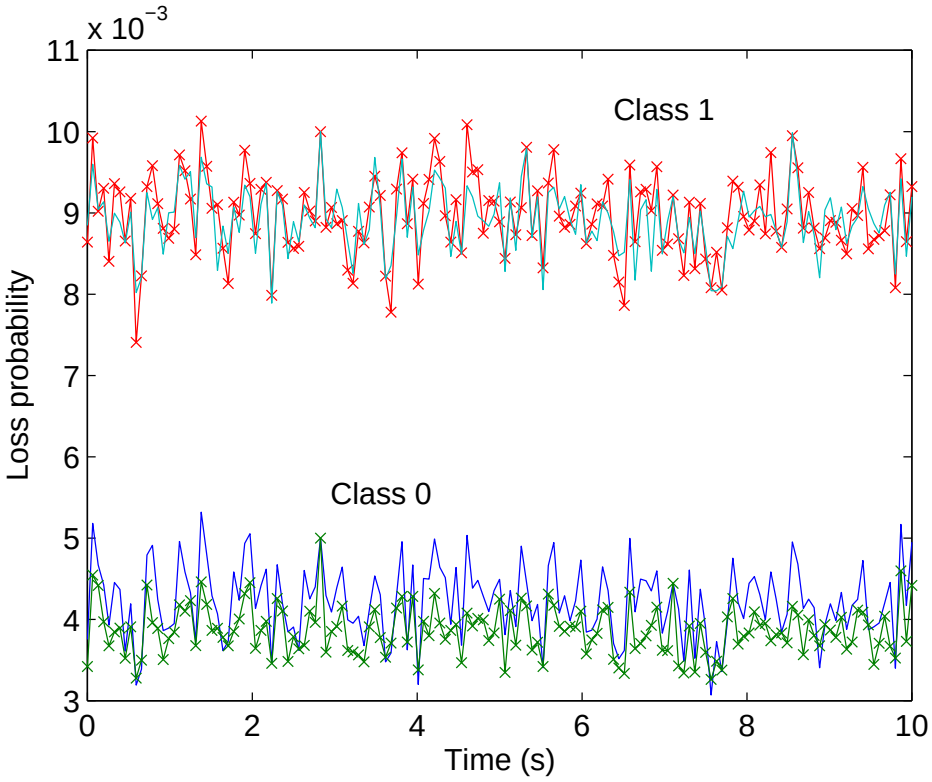
In this section, we present the simulation results for the whole framework over an end-to-end path. The topology in Figure 4.6, which is a simplified topology for the US backbone network, is used. It consists of 24 nodes and 43 links. Fixed shortest path routing is used and the maximum path length is 6. For simplicity, the propagation delays between adjacent nodes are assumed to have a fixed value of 5 ms. The links are bi-directional, each implemented by two uni-directional links in opposite directions. Link parameters and burst characteristics are the same as in the previous section. Seven per-hop QoS classes are defined at each link with the lowest threshold $T_l = 0.0005$ and the ratio between two adjacent thresholds $\gamma = 2$.

New LSPs are generated at each node according to a Poisson process with rate λ_{LSP} and have exponentially distributed durations. For simplicity, we assume that LSPs do not change their bandwidth requirements. Two groups of LSPs are considered: group 0 and group 1 with required e2e loss probabilities of 0.01 and 0.05, respectively. A new LSP is destined to a random node and falls into one of the two groups with equal probability.

In the first experiment, we investigate the temporal loss behaviour of the framework. To do this, we run the simulation for 11 s and monitor the e2e loss rate of traffic between node pair (1, 24). The path between this node pair is 6 hops long, which is the longest path in the network, and it runs through the bottleneck link (9,10). The data in the first second, which is the system warm-up period, is discarded. During the first 6 s, the total network load is 15 Erlang, which is equally distributed among all node pairs. After that, the offered load between node pair (1,24) is increased 10 folds. The loss rates of the two groups are plotted against time in Figure 4.7. It is observed that the loss probabilities increase in



(a)



(b)

Figure 4.5: Comparison of transient burst loss probabilities of traffic components with different characteristics: (a) Different offset times, and (b) Different burst lengths

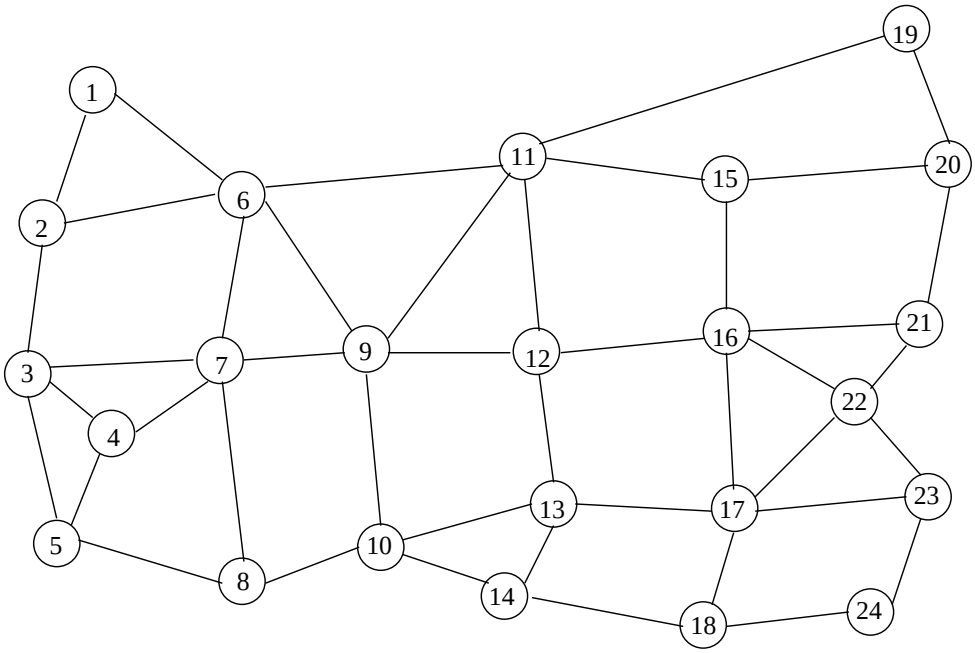


Figure 4.6: 24-node NSF network topology

response to the increase in the offered load at $t = 6$ s. Nevertheless, they are always kept below the respective thresholds. It shows that the reservation process is able to guarantee the loss bounds to admitted LSPs in real time regardless of the traffic load.

Another observation from Figure 4.7 is that the maximum loss probabilities of the two traffic groups are 0.004 and 0.03, which are well below the required e2e loss probabilities. This is due to the fact that almost all of the burst loss on the path occurs at a single bottleneck link. Hence, the e2e loss probabilities are limited by the maximum thresholds that can be allocated to the bottleneck links. In this case, they are 0.004 and 0.032, respectively. If more per-hop classes are available, the gaps between adjacent thresholds will be reduced and the e2e loss probabilities can be pushed closer to the targets.

In Figure 4.8, we plot the e2e loss probabilities of LSPs with different hop lengths and at two different loads of 15 and 30 Erlang. The same loss probabilities of the path clustering scheme proposed in [98] are also plotted for comparison. Since no admission control implementation is provided in the paper, we do not include any admission control mechanism for the path clustering scheme. The

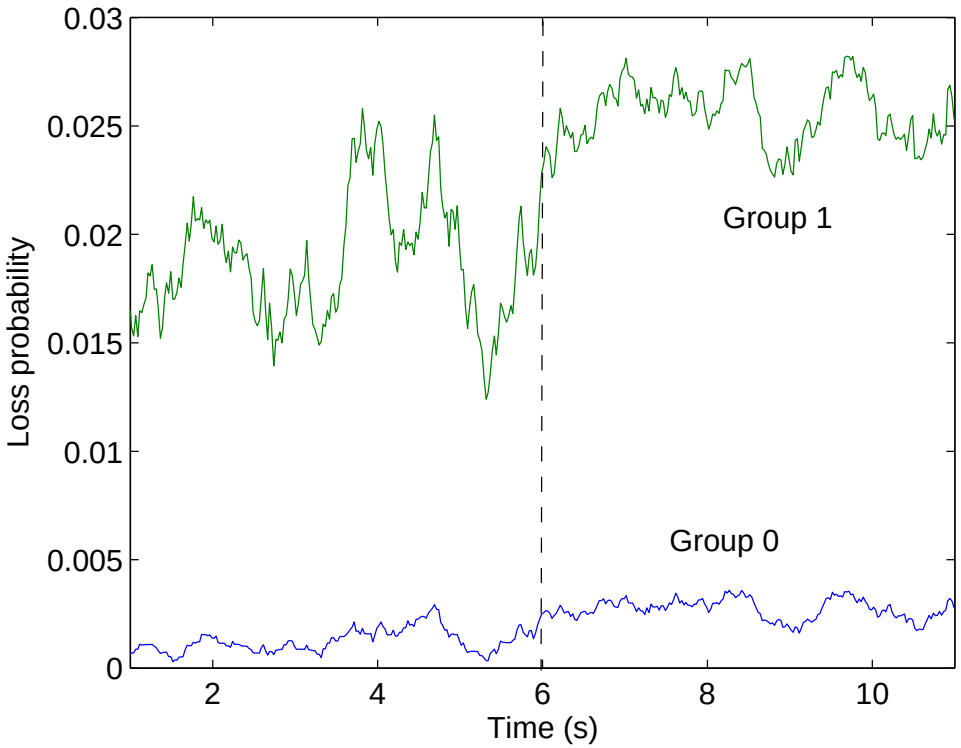


Figure 4.7: Transient edge-to-edge burst loss probability of two traffic groups with e2e loss requirements of 0.01 and 0.05

cluster combination {1,2}{3,4,5,6} is used as it is the best performing one. It groups LSPs with one or two hop lengths into one cluster and all the remaining LSPs into the other cluster.

A number of observations can be made from Figure 4.8. Firstly, we observe that the e2e loss probabilities of all LSP groups in our scheme are below their required e2e levels. This is true under both medium and heavy loading conditions. Secondly, the loss probabilities in our scheme increase from 1-hop group to 3-hop group but level off after that. The loss probability increase is due to the fact that burst traversing more hops will experience more loss. However, at a certain level, the effect of admission control dominates and the loss probabilities stay nearly constant. For the path clustering scheme, it is observed that it can keep the e2e loss probabilities for group 0 LSPs below the required level. However, it is achieved at great cost to group 1 LSPs, which experience very large loss probabilities. This happens because there is no admission control present. So core nodes must

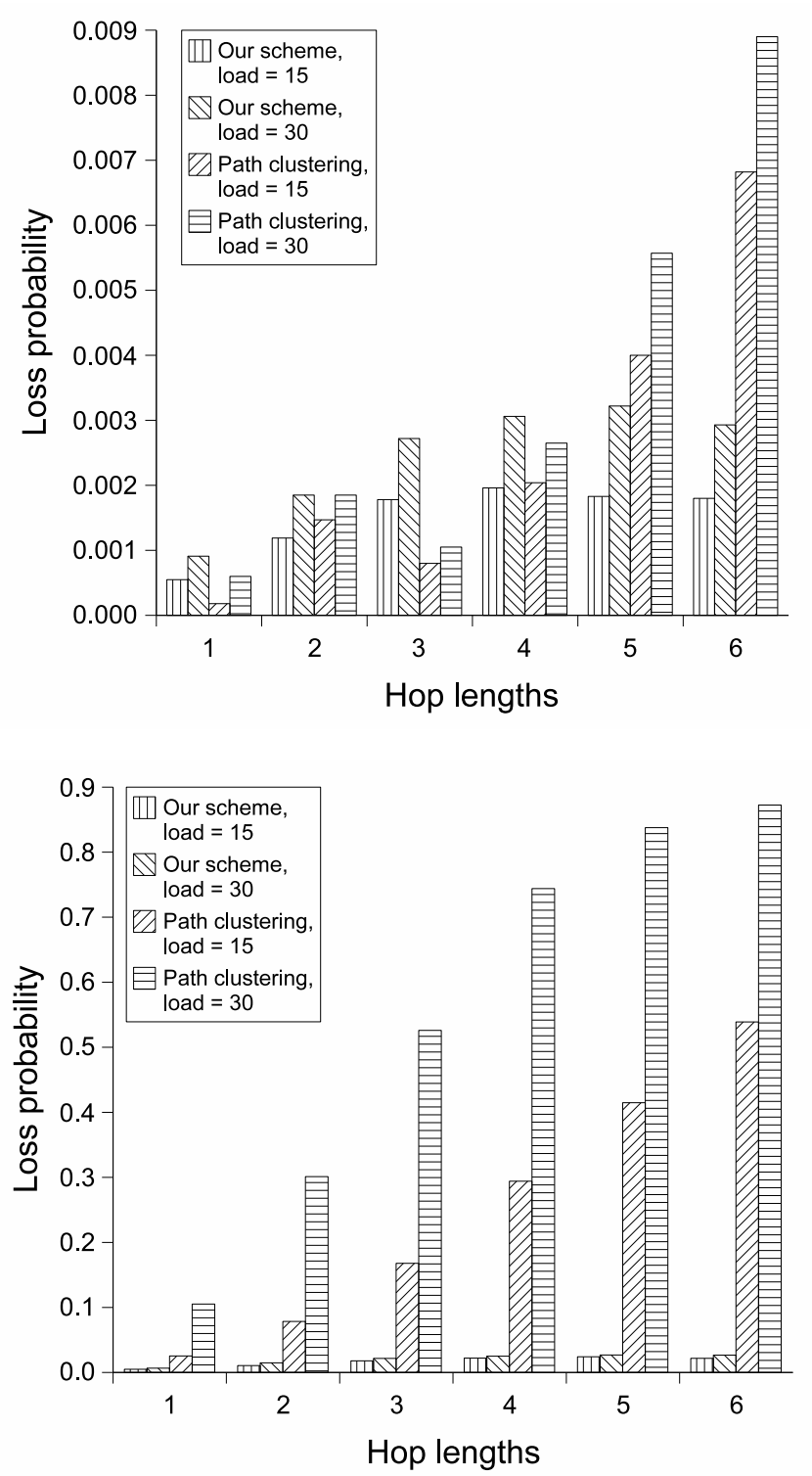


Figure 4.8: Average e2e loss probability of LSPs with different hop lengths for our scheme and path clustering scheme: (a) Traffic group 0 (required e2e loss probability of 0.01), and (b) Traffic group 1 (required e2e loss probability of 0.05)

drop low priority bursts excessively in order to keep the loss guarantees for high priority bursts. Another observation is that the loss probabilities of group 0 LSPs in the path clustering scheme vary significantly with hop lengths. This is because the scheme allocates the same per-hop threshold to LSPs within a cluster. Therefore, LSPs in a cluster with many different hop lengths such as {3,4,5,6} will experience significantly different e2e loss probabilities, some of which are far below the required level.

In the final experiment, the acceptance percentage of LSP groups with different hop lengths are plotted against the network loads in Figure 4.9. It shows that the acceptance percentage of all LSP groups decrease with increasing load, which is expected. Among the groups, the longer the hop length, the worse the performance. There are two reasons for that. Firstly, the network must allocate more stringent per-hop thresholds to longer LSPs compared to shorter LSPs that have the same required e2e loss probability. Secondly, longer LSPs are more likely to encounter congestion on one of their intermediate links. This situation can be remedied by a fairness scheme that gives more favourable treatment to longer LSPs in the reservation process. However, that is beyond the scope of this thesis.

4.7 Conclusion

We have proposed a novel absolute QoS framework that can offer quantitative e2e loss guarantees for individual LSPs in OBS networks. The framework consists of two parts. The first part includes a preemptive differentiation mechanism and a hop-based admission control mechanism. They enable core nodes to offer per-hop loss guarantees to individual LSPs. The second part includes e2e signalling and reservation mechanisms. For each new LSP, the framework divides its required e2e loss probability into small loss probabilities that are allocated to the intermediate links through the signalling and reservation process. When each core node fulfills its loss guarantee, the e2e loss guarantee is achieved. We have analyzed the differentiation scheme and verified it through simulation. The whole framework

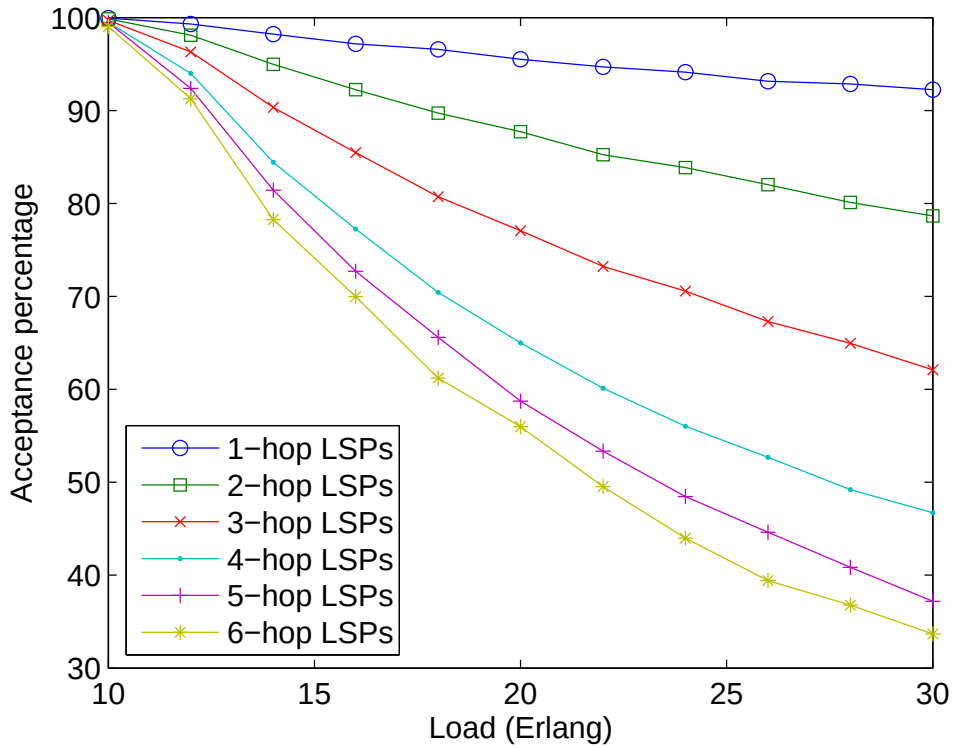


Figure 4.9: Overall acceptance percentage of LSPs with different hop lengths versus average network load

has also been evaluated through network level simulation. It has been shown to be able to reliably guarantee e2e loss probabilities under different network scenarios.

Chapter 5

The Streamline Effect in OBS Networks and Its Application in Load Balancing for Absolute QoS Traffic

5.1 Introduction

Besides node-based performance improvement mechanisms, network-wide performance optimisation is also very important in large data networks in general and in OBS networks in particular. This is achieved primarily through load balancing. A load balancing algorithm attempts to spread traffic across a network. In doing so, traffic is diverted away from bottleneck links and bottleneck congestion is reduced. An important part of a load balancing algorithm is to collect information about the level of congestion at various parts of the network.

In an OBS network, congestion level at a link is indicated through burst loss probability. If the load balancing algorithm is online and a majority of network traffic is best-effort, it is sufficient to measure the burst loss probability at a link to know its congestion level. However, this is not possible if the algorithm is offline or if a majority of network traffic is reservation-based QoS traffic. In the latter case, what is important is the *future* burst loss probability when the traffic corresponding to a reservation is transmitted. In these scenarios, the load balancing algorithm must *estimate* the burst loss probability at a link.

Traditionally, the $M|M|k|k$ queueing model is adopted in performance evaluation of OBS networks. This model assumes that the input traffic to an OBS core

node is Poisson, which is equivalent to having an infinite number of independent input streams. However, the number of input streams to a core node is bounded by the small number of input links. Therefore, the actual burst loss probability at an OBS link, as we will show later, is smaller than that obtained from the $M|M|k|k$ model and is strongly dependent on the number of input streams and their relative rates. It is caused by the fact that bursts within one input stream are streamlined and only inter-stream contentions happen at the link. We refer to this phenomenon as the *streamline effect*.

In this chapter, we study the streamline effect in OBS networks and derive an analytical formula to accurately estimate the burst loss probability at a link. We then apply the result to a dynamic load balancing scheme for absolute QoS traffic in OBS networks. The load balancing scheme considers multiple paths between an ingress/egress node pair and uses the absolute QoS scheme in the previous chapter to make reservations over each path. By including the streamline effect in the link cost function of the load balancing scheme, network traffic will be better distributed and congestion reduced. The absolute QoS scheme also benefits from a tighter burst loss probability estimation, which will allow more QoS traffic to be admitted.

The rest of this chapter is organised as follows. In §5.2, the streamline effect is presented in detail. This includes a loss analysis of an OBS core node with the effect taken into consideration and a simulation evaluation. Some existing analyses of OBS networks are also briefly described. In §5.3, the dynamic load balancing scheme for absolute QoS traffic is presented. The proposed scheme is evaluated through simulation in §5.3.3. Finally, concluding remarks are given in §5.4.

5.2 The Streamline Effect

5.2.1 Description

Consider an OBS core node with a number of input links connected to an output link as shown in Figure 5.1. To facilitate discussion, we call the bursts within one

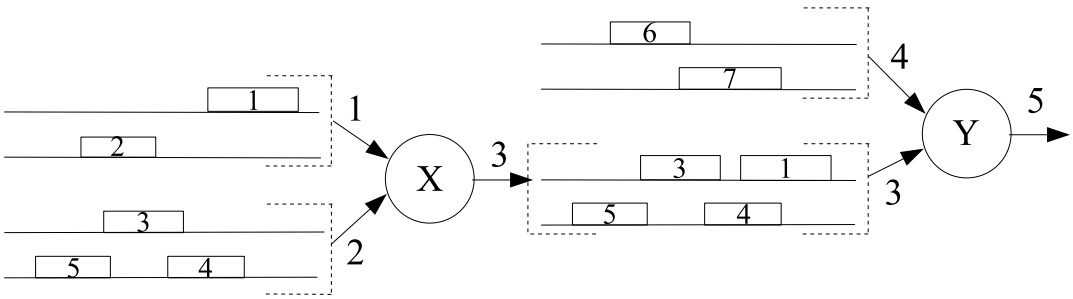


Figure 5.1: Illustration of the streamline effect

input link a burst stream as opposed to a burst flow since one burst stream may consists of many burst flows carried by different LSPs.

The streamline effect is the phenomenon wherein bursts travelling in a common link are streamlined and do not contend with each other at downstream links. The reason is because there is no buffer inside an OBS network. Therefore, once the contentions among them are resolved at the first link where they merge, no intra-stream contention will occur thereafter. This effect is illustrated in Figure 5.1. Burst streams 1 and 2 merge at node X. After any burst loss that might happen at node X, the remaining bursts are streamlined in output stream 3 and no further contention will happen among them. However, they may still experience contention with other burst streams that merge at downstream nodes. In this example, the bursts in link 3 merge with those in link 4 at node Y.

The significance of this streamline effect is two folds. Firstly, since bursts within an input stream only contend with those from other input streams but not among themselves, their loss probability is lower than that obtained from the $M|M|k|k$ queueing model. This model assumes that the input traffic to an OBS core node is Poisson, which is equivalent to having an infinite number of independent input streams. However, in practice, there are often only a small number of input streams leading to an output link. This has major implications for network algorithms that need to estimate burst loss probability at a link such as QoS or load balancing algorithms. Secondly, as we will later observe, the burst loss probability is not uniform among the input streams. The higher the burst rate of the input stream, the lower its loss probability. Therefore, if traffic within

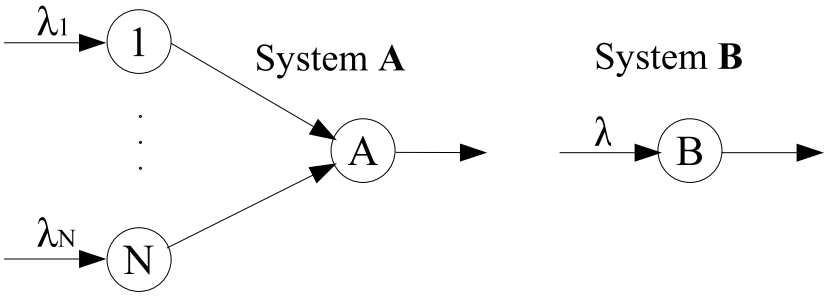


Figure 5.2: Two equivalent systems used to analyze the streamline effect

an OBS network is encouraged to form major flows with fewer merging points, the overall loss rate will be reduced.

5.2.2 Analytical Model

Consider two systems **A** and **B** as shown in Figure 5.2. Each node in the figure has W wavelengths per link with full wavelength conversion capability and no FDL buffer. The two systems receive identical input traffic with rate λ. The input traffic to system **A** is split into N streams with rates $\lambda_1, \ldots, \lambda_N$ coming into nodes 1,...,N. That is, $\lambda = \sum_{i=1}^{N} \lambda_i$. On the other hand, the entire input traffic λ is fed into node B in system **B**. We wish to determine the burst loss probability of each input stream to node A. To do this, we will prove below that the two systems are equivalent in terms of burst loss. That is, for any burst that is lost in one system, there is a corresponding burst lost in the other.

Let M_i, M_A and M_B be the number of bursts arriving simultaneously at node i ($1 \leq i \leq N$), node A and node B, respectively. For a burst to be dropped at a node, the number of overlapping bursts must be greater than W.

- Consider a burst lost in system **A**. There are two possibilities. It is dropped either at a node i ($i \in [1, \ldots, N]$) or node A. Thus, either $M_i > W$ or $M_A > W$. Also, observe that M_B is greater than both M_i and M_A. Therefore, $M_B > W$, i.e., there is a burst dropped at node B of system **B**.
- Consider a burst lost in system **B**. Thus, $M_B = \sum_{i=1}^{N} M_i > W$. In system **A**, there are two possibilities. First, one particular term $M_i > W$, which implies that a burst is dropped at node i in system **A**. Alternatively, no

term in the sum is larger than W, which implies that no burst is dropped at any of the nodes 1,...,N, so $M_A = \sum_{i=1}^{N} M_i > W$, i.e., there is a burst dropped at node A in system **A**.

Having established that the two systems are equivalent, the loss probability of input stream i to node A can be easily calculated by equating the number of lost bursts for that stream in both systems. Let P_i, P_i^A and P^B be the loss probabilities of stream i at node i, node A and node B, respectively. We have

$$\lambda_i P^B = \lambda_i P_i + \lambda_i (1 - P_i) P_i^A \Rightarrow P_i^A = \frac{P^B - P_i}{1 - P_i}. \tag{5.1}$$

The overall loss probability at node A is the weighted average of the loss probabilities of individual streams.

$$P^A = \frac{\sum_{i=1}^{N} \lambda_i (1 - P_i) P_i^A}{\sum_{j=1}^{N} \lambda_j (1 - P_j)} = \frac{\lambda P^B - \sum_{i=1}^{N} \lambda_i P_i}{\lambda - \sum_{j=1}^{N} \lambda_j P_j}. \tag{5.2}$$

If the input streams to node i ($1 \leq i \leq N$) and node B have Poisson arrival distribution, which is a reasonable assumption in backbone networks according to [8], P_i and P^B can be determined from the Erlang B formula [44] with W wavelengths and total load ρ as follows

$$P = B(\rho, W) = \frac{\rho^W / W!}{\sum_{n=0}^{W} \rho^n / n!}. \tag{5.3}$$

It is also noted that in a real network, true Poisson traffic like the inputs to nodes 1,...,N and node B does not exist and neither do λ and λ_i. Nevertheless, λ and λ_i can be derived from the input rates to node A using (5.3).

The above results hold well in the limiting cases. If $N = 1$, $P_1 = P^B \Rightarrow P_1^A = P^A = 0$. On the other hand, if $N \to \infty$, $\lambda_i \to 0 \Rightarrow P_i = 0$. Thus, $P_i^A = P^A = P^B$. In this second special case, the assumption of the $M|M|k|k$ model is valid and

it produces the same result as our model. Therefore, the $M|M|k|k$ model is a special case of our model when $N \to \infty$.

5.2.3 Previous Performance Analyses for OBS

In the current literature, loss performance analyses of OBS networks have been reported in a few papers. Most of them, however, adopt the $M|M|k|k$ model [84, 93]. Reference [68] studies loss performance in OBS networks with several OBS core nodes connecting a pair of source and destination. However, the input burst traffic is assumed to have an exponential length distribution and bounded interarrival time. The work in [90] models the traffic arrival process using a Markov process. However, the results are only applicable to OBS edge nodes.

The work in [94] is closest to our work here. It analyzes an OBS core node that directly receives input traffic from multiple burst generators. The paper recognises that bursts generated by a common assembly process are streamlined and added a factor to the Erlang B formula to account for the fact. However, there are a number of important differences compared to our work. First, the fact that bursts travelling in a common link inside the network are also streamlined is not recognised by the paper. Therefore, its results are not applicable to a general OBS core node. Next, the final formula has a number of parameters that must be determined through simulation. Finally, the loss probabilities of individual input streams, which are different from the overall loss probability of the output link, are not given.

Another performance analysis of an OPS node in [58] might be applicable here. It used the Engset traffic model to analyze an OPS node with finite inputs. However, unlike our analysis, the paper does not offer a closed form expression for packet loss probability. Therefore, it does not scale well when the number of wavelengths per link becomes large, which is common for core networks.

5.2.4 Experimental Study

In this section, we present numerical results for the loss analysis in the previous section. The measured loss probabilities of input streams at node A and those

given by the $M|M|k|k$ model and our model are plotted against various parameters. Apart from validating our analysis, an additional goal is to determine which parameters affect the accuracy of the $M|M|k|k$ model. This information will be useful in determining the scenarios under which the Erlang B formula can be used in OBS networks. In the simulations, we use the same topology as system **A** in Figure 5.2. Unless otherwise stated, each link has 8 wavelengths and each wavelength has a capacity of 40 Gbps. In our simulations, the input burst streams have Poisson arrival pattern and the burst lengths are uniformly distributed between 50 and 51.5 KB. The burst length distribution is chosen to emulate the output produced by a size-limited burst assembly algorithm such as in [88].

In Figure 5.3, we plot the overall loss probability versus the number of input streams N. In this simulation, the total offered load at the output link of node A is 0.6. All input streams have equal rates. The results show that our model always gives accurate results. The $M|M|k|k$ model, on the other hand, is only accurate when $N \geq 3$. There is a slight deviation when $N = 2$, which is tolerable. However, when $N = 1$, the measured loss probability falls to zero but the $M|M|k|k$ model still gives the same result. This special case is predicted by our analysis in the previous section.

In the next experiment, the number of input streams is kept constant at 2 and the traffic contribution of each stream is varied. The results in Figure 5.4 show very different curves for the loss probabilities of the two streams. For the smaller stream, the loss probability is relatively constant. This is expected from our analysis since for the small stream, P_i is very small and hence, $P_A^i \approx P_B$. For the dominant stream, its loss probability falls rapidly as its traffic share gets closer to unity. This can also be explained from our analysis since $P_i \rightarrow P_B$ as the dominant flow gets larger. Thus, $P_A^i \rightarrow 0$. However, this result cannot be predicted by the $M|M|k|k$ model, which gives a constant loss probability.

The results in Figure 5.4 can also be explained intuitively if we recall that the bursts within one input stream do not contend with each other and only contend with those from other streams. For the dominant stream, their bursts have few

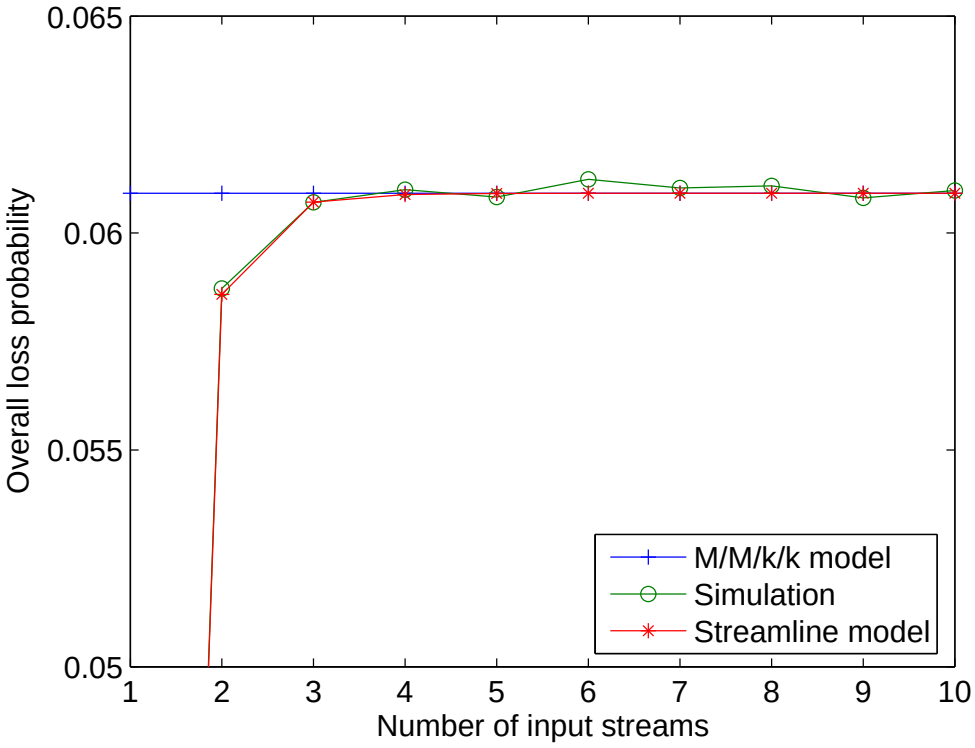


Figure 5.3: Overall burst loss probability versus number of input streams

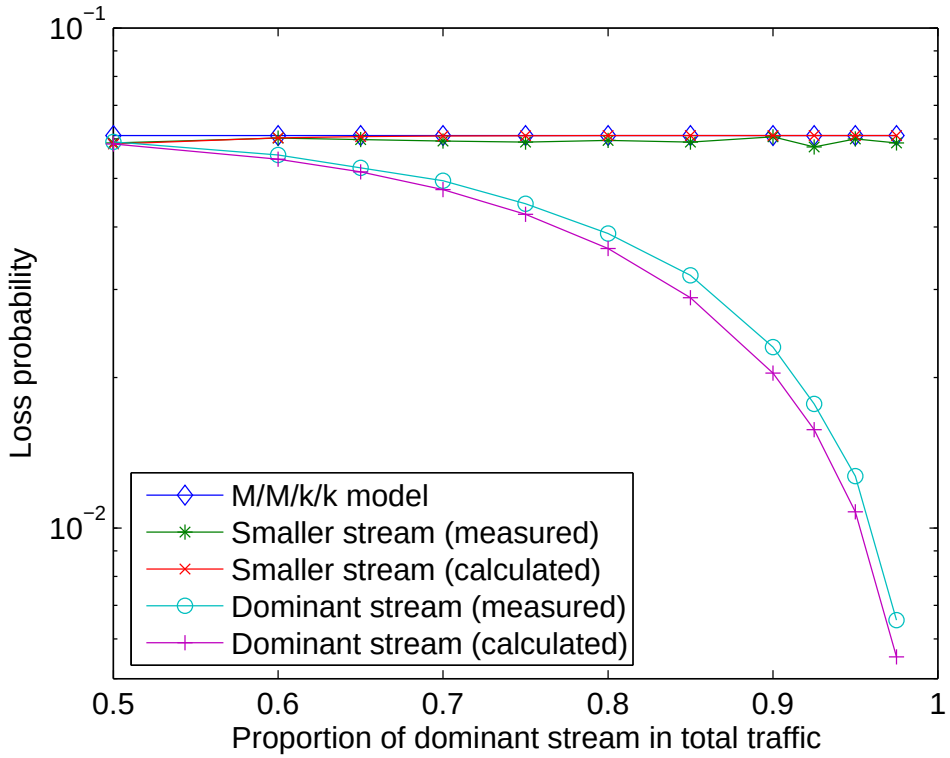


Figure 5.4: Loss probabilities of individual streams versus traffic proportion of dominant stream

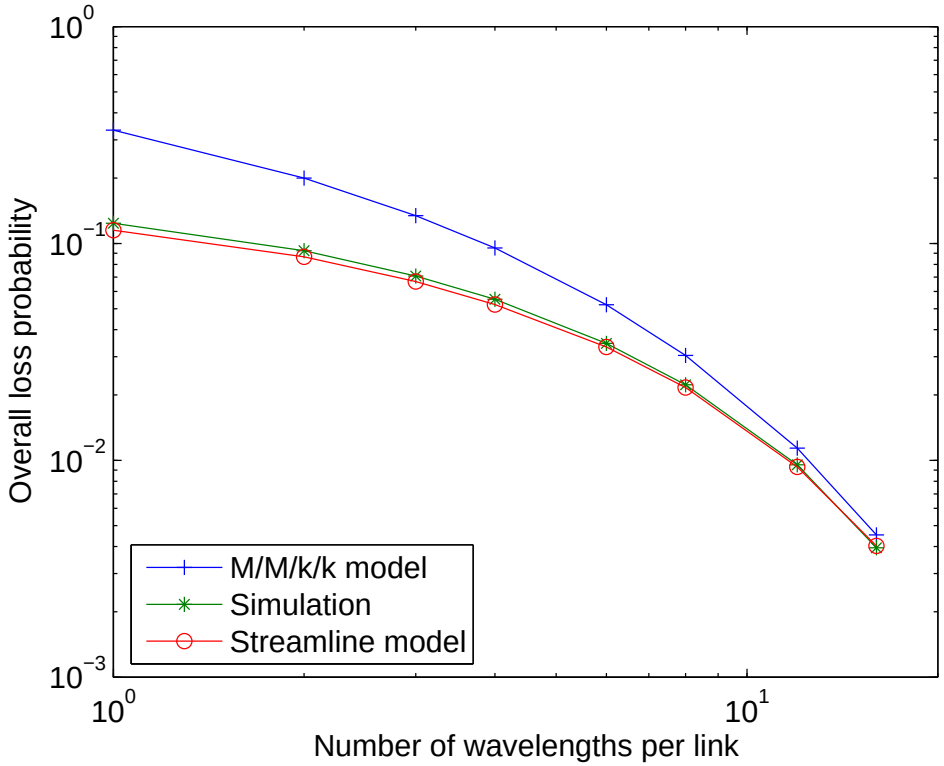


Figure 5.5: Overall loss probability versus number of wavelengths per link

competitors from the small stream and hence their small loss probability. On the other hand, the bursts from the small stream have almost as many competitors from the dominant streams as in the case of a large number of input streams. Therefore, their loss probability is close to that given by the $M|M|k|k$ model.

The effect of the number of wavelengths per link on the accuracy of the models is examined next. There are two input streams with rates of 0.4 and 0.1 to the node. As shown in Figure 5.5, the streamline model always gives accurate results. On the other hand, the $M|M|k|k$ model is only accurate when the number of wavelength is large.

In the last experiment, we plot the overall loss probability versus the total offered load in Figure 5.6. The number of input streams is set at 2 and both streams have equal rates. We observe that our analysis and the $M|M|k|k$ model both give results that closely match the simulation results at all offered loads. In other words, the offered load does not play a role in the accuracy of the $M|M|k|k$ model.

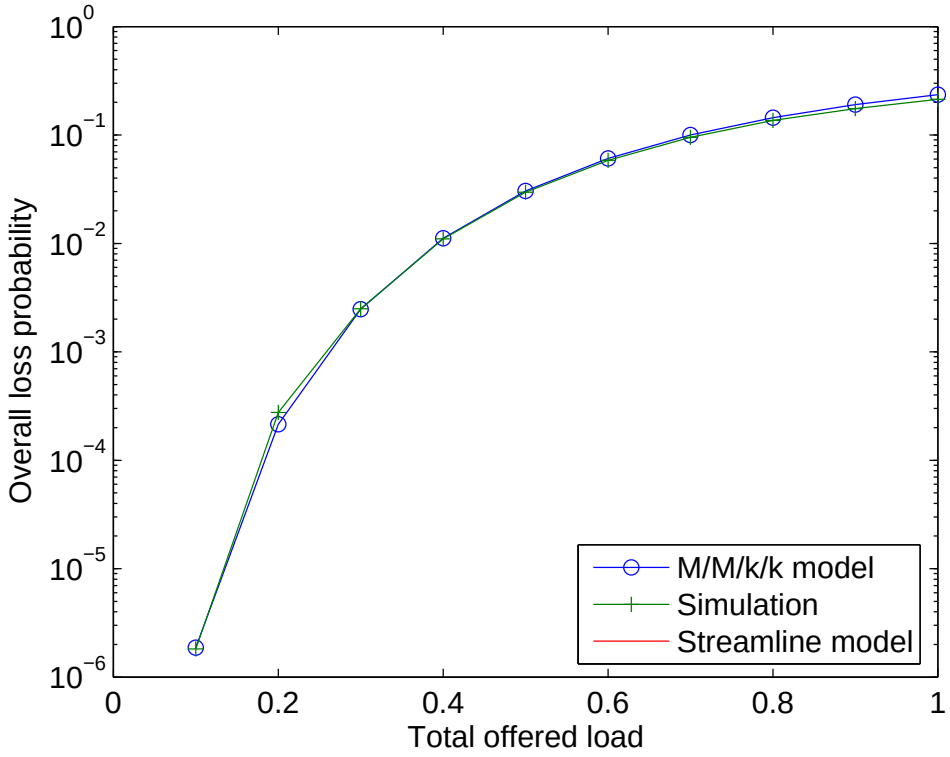


Figure 5.6: Overall loss probability versus total offered load

5.3 Application in Load Balancing for Absolute QoS Traffic

Hereafter, we apply the results obtained from analysing the streamline effect in §5.2.2 to a load balancing scheme for absolute QoS traffic. When a new LSP needs to be established, the absolute QoS framework in chapter 4 reserves resource along the path of the LSP to satisfy the LSP's request or denies the request if insufficient resource is available. However, the framework only operates over a predetermined path for each LSP. In this section, we extend it to consider multiple paths between an ingress/egress node pair and choose a path based on the calculated cost of the paths. Since the future burst loss probability of a link if/when a request is accepted cannot be measured, the scheme needs to estimate it based on the current offered load to the link and the amount of bandwidth the new LSP requests. Hence, the results in the previous sections are especially relevant to this scheme.

5.3.1 *Multipath Extension to the Absolute QoS Framework Taking into Account the Streamline Effect*

In this section, we describe the key ideas of the absolute QoS framework in the previous chapter and examine the implication of the streamline effect on the framework.

The absolute QoS framework comprises two main parts. The first part includes local mechanisms at a core node. The framework defines a number of local QoS classes with corresponding loss thresholds. The loss threshold of a class is the upper bound on the loss probability experienced by bursts in that class. A core node is responsible for ensuring that these thresholds are not violated. It does this through a preemptive differentiation mechanism and an admission control mechanism. The differentiation mechanism allows bursts in the classes that are in danger of breaching their thresholds to preempt those from other classes. This in effect shifts burst loss from the classes in danger of breaching their thresholds to other classes. The admission control mechanism limits the total offered load to a link to make it feasible for the differentiation mechanism to keep burst loss probabilities of all classes under their respective thresholds. Since each class has a different threshold, both the requested bandwidth of a new request and the class that it wants to be fitted in are relevant in whether the request can be accepted.

The second part of the framework deals with signalling and reservation over the entire path. When a new LSP needs to be set up, the ingress node sends a request message carrying the required edge-to-edge loss probability P_0 and the requested bandwidth along the path to the egress node. At each intermediate node, the local admission control routine tries to fit the request in one of its local classes, starting from the one with the lowest threshold. If the request is accepted at the node, the index of the class with the lowest threshold that can accommodate the request is recorded in the message, which continues to be forwarded towards the egress node.

The final admission control decision is made at the egress node. Assuming burst blocking at each link is independent, the lowest possible edge-to-edge loss probability P^0_{e2e} is.

$$P^0_{e2e} = p^0_1 + \sum_{i=2}^{n} p^0_i \prod_{j=1}^{i-1}(1 - p^0_j) = 1 - \prod_{i=1}^{n}(1 - p^0_i) \tag{5.4}$$

where p^0_i is the lowest threshold offered by the ith node on a n-link path. If $P^0_{e2e} \leq P_0$, the request is admitted. The egress node then allocates each intermediate link one of the predefined classes in which to support the LSP such that

$$\begin{cases} p_i \geq p^0_i \\ P_{e2e} = 1 - \prod_{i=1}^{n}(1 - p_i) \leq P_0 \end{cases} \tag{5.5}$$

where p_i is the threshold allocated to the LSP at the ith link and P_{e2e} is the corresponding edge-to-edge loss probability. This class allocation is signalled back to the core nodes using a returned acknowledgment message.

The implication of the streamline effect to the above absolute QoS framework is two folds. The first obvious and beneficial implication is that a more accurate loss probability formula will enable tighter estimates and better admission control decisions for the admission control routine at each core node. The second implication can be understood by looking at Figure 5.4. It shows that the loss probabilities of dominant and non-dominant streams are vastly different if the traffic share of the dominant stream is large. Therefore, there are some issues in guaranteeing thresholds for individual classes at a core node. We will divide them into the following two categories.

- *Inter-class contention:* This is the case where the traffic classes in the dominant stream are different from those in the small streams. There is no problem in this scenario since the absolute QoS framework uses preemptive differentiation to resolve contention among classes. The preemptive differentiation will override the streamline effect.

- *Intra-class contention:* This is the case where one or more traffic classes exist in both the dominant stream and the small streams. Within such classes, the traffic portion from the dominant stream will experience lower loss probability than those from other streams. This is not desirable since some LSPs in the small streams may experience loss probability over the guaranteed threshold of their class. There are two solutions for this. The first solution is to build the streamline effect into the preemption mechanism to compensate for it, i.e., to mark the bursts from the dominant stream and over-drop them. The second solution is to divide the affected class into two subclasses: one for the traffic portion from the dominant stream and one for the rest. It in effect converts intra-class contentions into inter-class contentions which the preemptive differentiation mechanism can handle. This is preferred since it does not introduce additional complexity into the preemption mechanism.

5.3.2 Dynamic Load Balancing Algorithm

In this section, we describe a dynamic route selection algorithm for the above absolute QoS framework. The algorithm is similar to that in [49, 78]. However, there are differences due to the fact that absolute QoS traffic is involved and the streamline effect is considered.

At an ingress node, two link-disjoint paths are determined in advance for each egress node. One of the paths is the shortest path. At any time, a cost is associated with each of the paths, which is the sum of the costs of the intermediate links along the path. For the longer path, a penalty term is also added to its cost. Let Δh be the hop difference between the two paths. The penalty term is given by $PA = \Delta h \times \tau$ where τ is a system control parameter.

When a new LSP needs to be set up, the less costly path will be selected to initiate the QoS reservation process. Since path cost is calculated based on the loss probability on that path, the less costly path will offer a better chance of acceptance. However, it is possible that changes in traffic conditions at the

intermediate links have not been updated in the path costs. Therefore, when the selected path does not accept the reservation request, the reservation process is started again on the other path. During the reservation process, the costs of the intermediate links of the path involved are also collected. This is signalled back to the ingress node using the acknowledgment message to update the cost of the path involved. If there is no reservation process on a path within a time period T (e.g., because its cost is too high and therefore it is not selected), a probe packet is sent out to update its cost. Note that we choose not to shift existing LSPs between the two paths. This is to avoid out-of-order burst arrivals and the synchronisation problem, which is caused by the feedback between traffic shifting and path costs. The number of LSPs in backbone networks should be sufficiently high to effectively balance traffic between the paths.

For each output link at a core node, the node needs to keep track of the traffic contribution λ_i from each input link in addition to the total traffic λ at the output link. When a reservation request with bandwidth b_0 arrives from input link i, the node substitutes λ_i with $\lambda_i' = \lambda_i + b_0$ and calculates the estimated loss probability of traffic from that input link using equation (5.1). This is recorded in the reservation message as the link cost. On the other hand, if a probe packet arrives, no substitution of λ_i is necessary.

The use of equation (5.2) in calculating the link cost differentiates reservation requests from dominant and non-dominant streams. That is, it gives lower costs to requests coming from the dominant stream. This will encourage the dominant stream to grow, which in turn lowers its cost. Thus, our proposed algorithm does not only balance but also judiciously redistribute traffic in an OBS network to further reduce burst loss.

5.3.3 Experimental Study

In this section, we investigate the performance of our proposed load balancing algorithm through simulation on a randomly generated network topology as shown

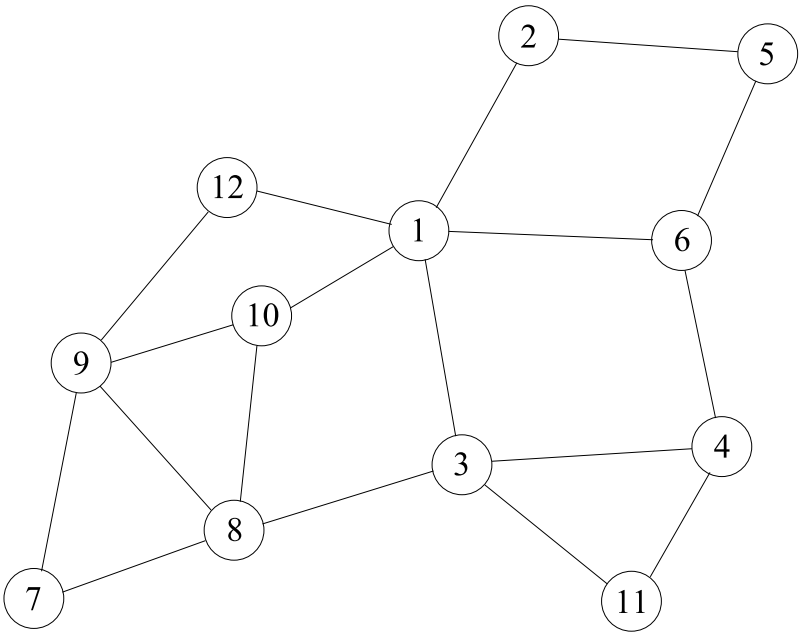


Figure 5.7: Random network topology with 12 nodes

in Figure 5.7. The network has 12 nodes, 18 bidirectional links and 8 data wavelengths per link. We use a smaller network topology instead of the standard NSF network topology as in the previous chapters since the time required for simulation of the proposed load balancing scheme on the latter topology would be prohibitively long. The capacity of a wavelength is 10 Gbps. We assume full wavelength conversion and no FDL buffer at all nodes. We use the LAUC-VF scheduling algorithm presented in [88] to schedule bursts. The bursts are generated from a size-limited assembly algorithm with the size limit at 50 kB. Thus, the mean burst length is about 10 μs.

Six ingress/egress node pairs are selected to transmit in our simulation, namely (7;5), (7;11), (9;5), (9;11), (12;5), (12;11). There are two paths between each node pair. One path is computed using a shortest path routing algorithm, while the other is the link-disjoint next-shortest path. New LSPs arrive at each ingress node according to a Poisson process. For simplicity, we assume all LSPs have the same burst rate of 4000 bursts/s. Hence, it takes 200 concurrently active LSPs to have 100% offered load. A new LSP will request an edge-to-edge loss probability of either 0.02 or 0.05 at random.

We set system control parameter $\tau = 0.01$. This is used to calculate the penalty cost that is added to the longer path to avoid excessive traffic being routed to that path, which may result in performance degradation due to increased consumption of network resources. We simulate three routing algorithms for comparison: shortest path routing (SP); load balancing using the Erlang B formula in estimating burst loss probability for LSP admission control and link cost calculation (LB-E) and load balancing using the formula obtained from analysing the streamline effect (LB-S). The percentage of LSPs being accepted into the network over the total arriving LSPs is used as the performance metric.

In the first experiment, we investigate the case wherein all six node pairs have identical offered loads. The LSP acceptance percentage of the three schemes is plotted against the offered load per node pair in Figure 5.8. The results show that LB-S has the best performance followed by LB-E and SP. Since SP always chooses the shortest path to use, congestion may easily develop at links where some shortest paths join. On the other hand, LB-S and LB-E may spread traffic to longer paths and hence perform better. Between the two load balancing algorithm, LB-S takes into consideration the streamline effect when calculating path costs. Therefore, major streams are encouraged to form, which reduces burst loss probability and increases LSP acceptance probability. In addition, LB-S uses the streamline formula in LSP admission control, which gives added improvement over LB-E.

The percentage improvement of LB-S and LB-E over SP is plotted against the offered load per node pair in Figure 5.9. We observe that the improvement increases as the network load increases. When the traffic load is light, congestion at bottleneck nodes is not so severe and there is not much room for improvement by load balancing. However, when the traffic load becomes heavy, the two load balancing schemes are able to mitigate congestion and hence their increased percentage of improvement.

Figure 5.10 shows the mean hop length traversed by a burst against the offered load per node pair. The mean hop length could affect delay, signalling overhead

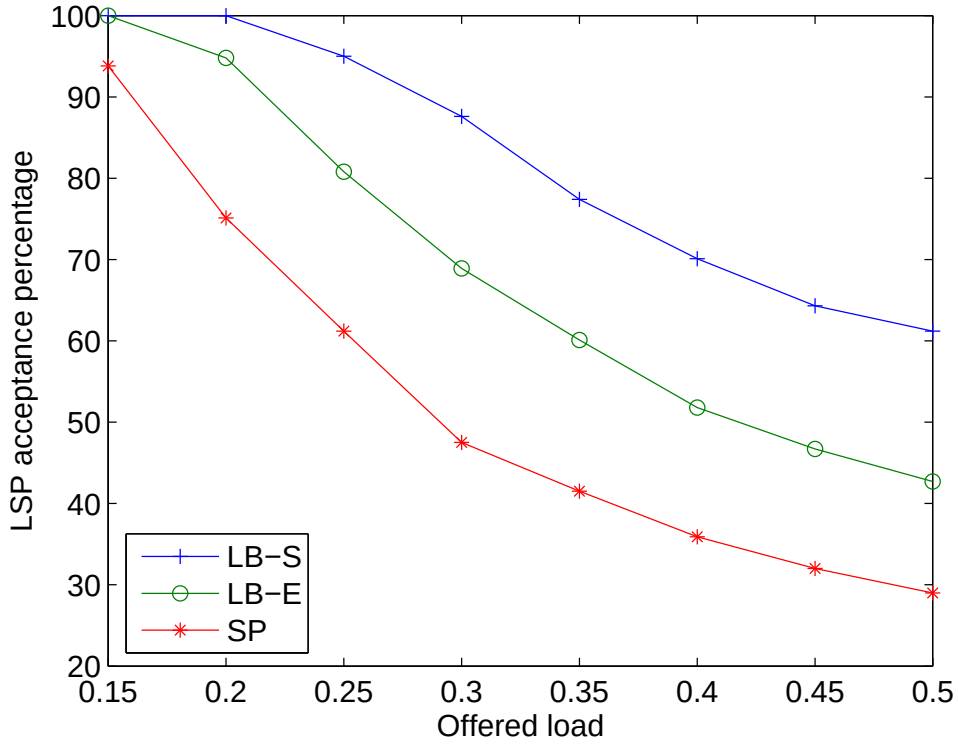


Figure 5.8: Percentage of LSP accepted versus offered load per node pair for identical traffic demands

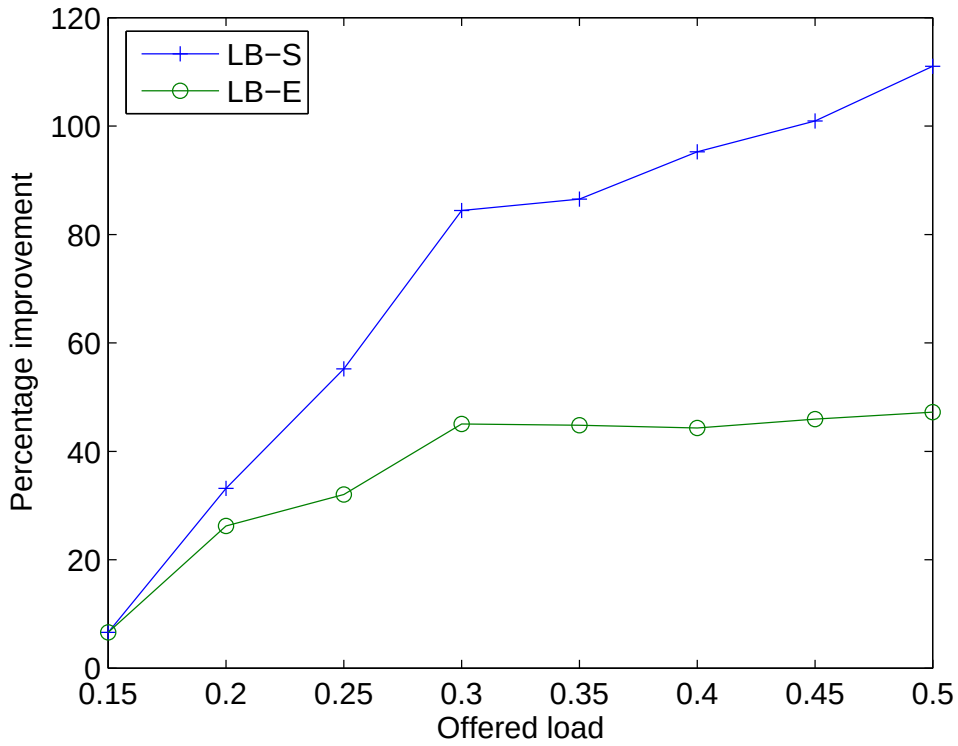


Figure 5.9: Percentage improvement over shortest path routing for identical traffic demands

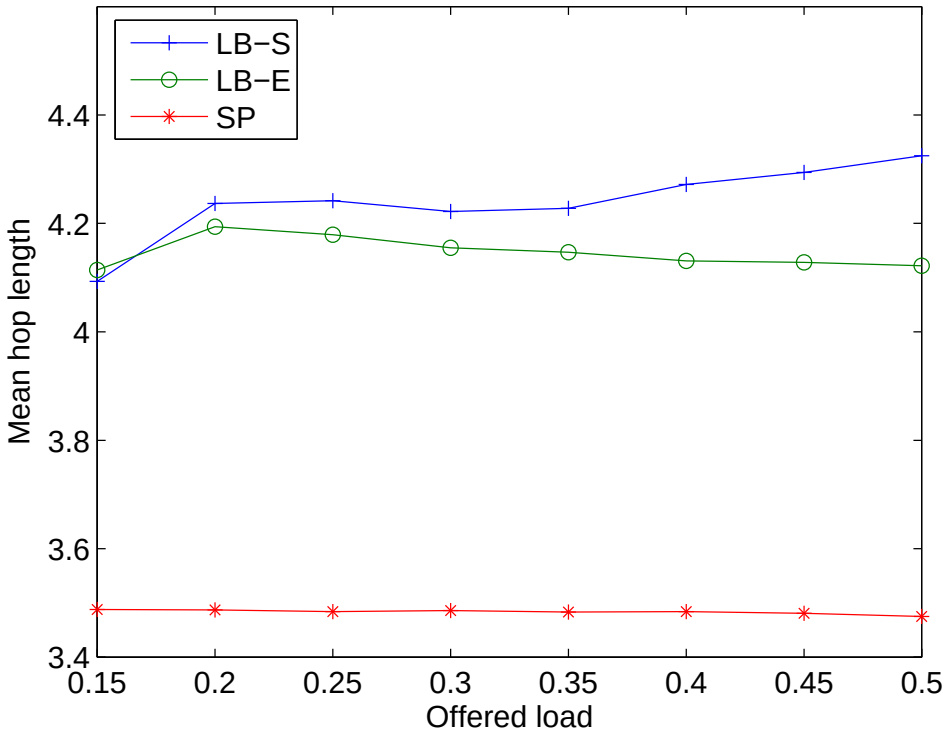


Figure 5.10: Mean hop length versus offered load per node pair for identical traffic demands

and initial offset time in the network. We observe that SP has the lowest mean hop length followed by LB-E and LB-S. This is natural since it only routes traffic through shortest paths. Between the two load balancing schemes, LB-S has slightly larger mean hop lengths. It implies that the additional delay, signalling overhead and initial offset time introduced by LB-S are rather low when compared to the achieved performance improvement.

In the next experiment, we investigate the applicability of LB-S in balancing non-identical traffic demands. In this experiment, the LSP arrival rates are taken randomly between $l-0.1$ and $l+0.1$, where l is the mean offered load. Figure 5.11 shows the LSP acceptance percentage of the three schemes against the mean offered load per node pair. Figure 5.12 shows the percentage improvement of LB-S and LB-E over SP. Figure 5.13 shows the mean hop lengths of the three schemes. From these figures, we make similar observations as in the case of identical traffic demands. The only difference is that in this experiment, LB-S has slightly lower

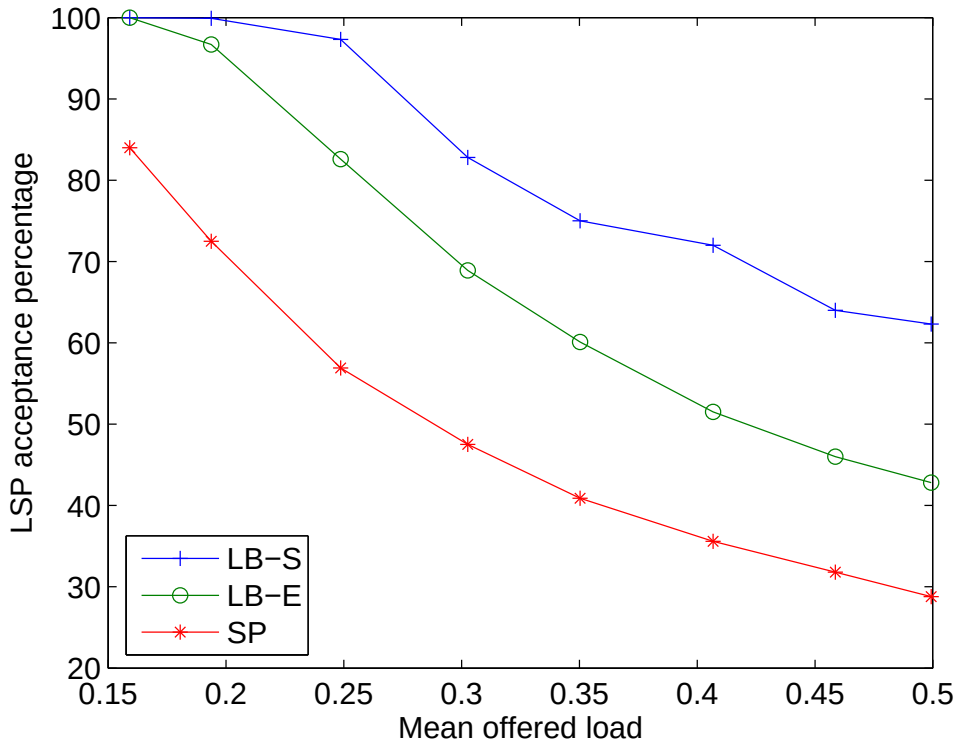


Figure 5.11: Percentage of LSP accepted versus mean offered load per node pair for non-identical traffic demands

mean hop lengths than LB-E. These observations prove that LB-S works well under different traffic scenarios.

5.4 Conclusion

In this chapter, we have described and analyzed the streamline effect in OBS networks. The simulation study shows that the formula obtained gives more accurate loss probability estimates than the traditionally used Erlang B formula. When applied to a load balancing scheme for absolute QoS traffic, it gives much better acceptance probability than shortest path routing and the load balancing scheme that does not consider the streamline effect.

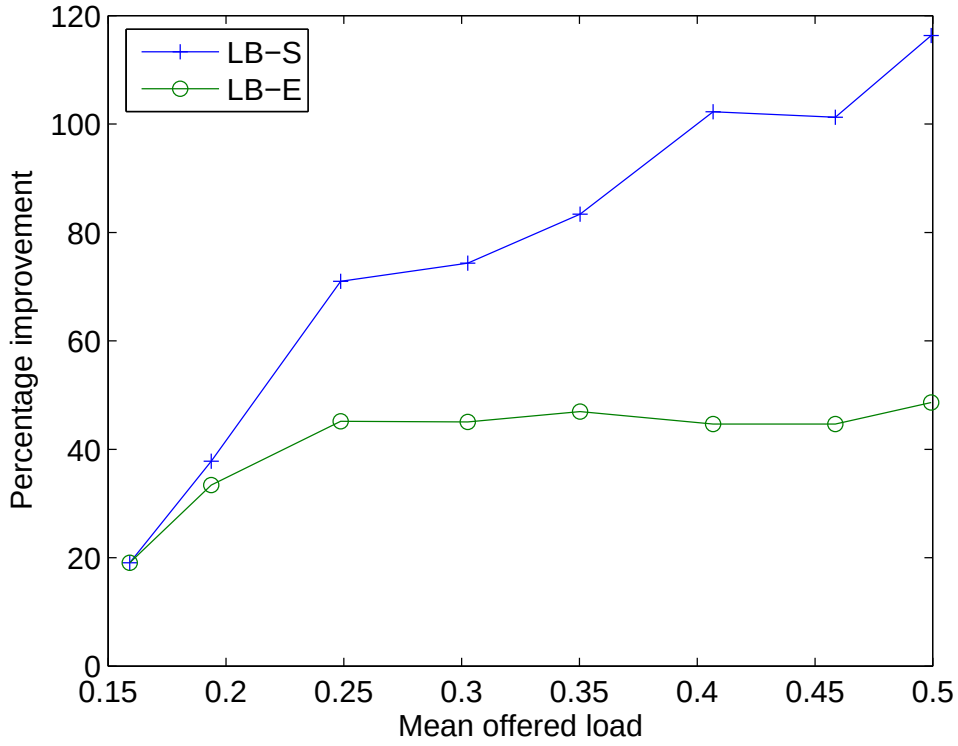


Figure 5.12: Percentage improvement over shortest path routing for non-identical traffic demands

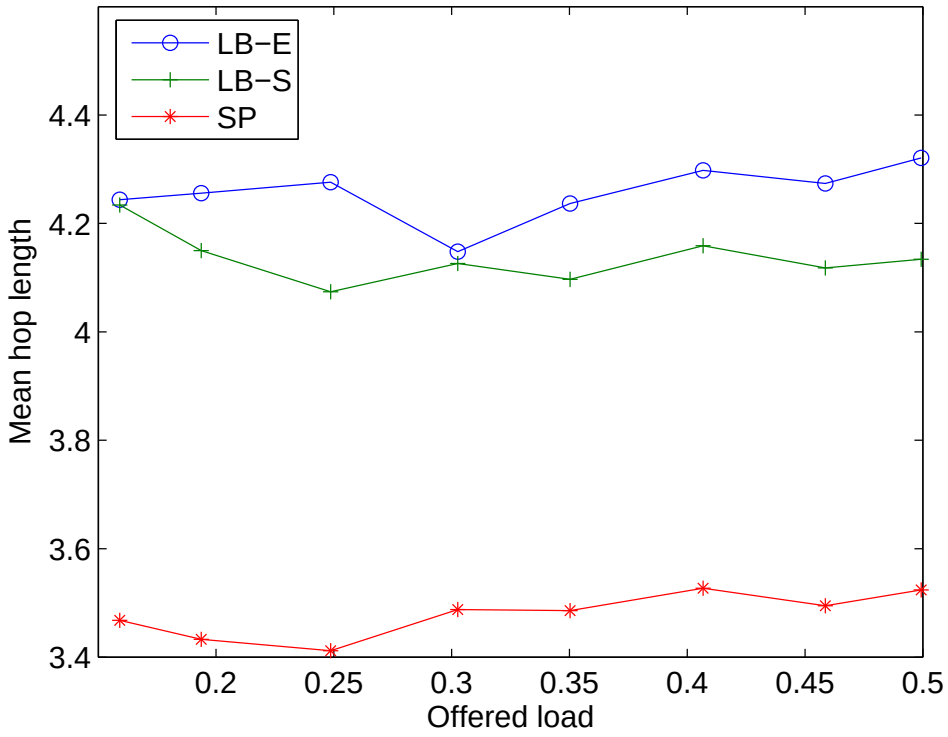


Figure 5.13: Mean hop length versus mean offered load per node pair for non-identical traffic demands

Chapter 6

Summary and Future Work

Quality of Service support is becoming a crucial part of today's data networks. Due to the proliferation of real-time applications, network users have come to expect not only connectivity but also adequate network performance. At the same time, network operators would like to maximise network resource utilisation to increase profits. As Optical Burst-Switching (OBS) is generally regarded as the transport technology of choice in the Internet backbone in the medium term, it is important that QoS mechanisms be developed for OBS networks. In this thesis, we have presented several algorithms for provisioning and enhancing QoS in OBS networks at different levels of operational levels, from link level to path and network levels. In this chapter, we will summarise the contributions of this work and suggest some directions for future research.

6.1 Summary of Contributions

In Chapter 2, we provided a survey of the current literature on QoS provisioning and performance enhancement algorithms in OBS. At the node level, we described available contention resolution approaches and existing channel scheduling algorithms. We then surveyed current proposals on QoS differentiation mechanisms and discussed in detail current absolute QoS proposals. Finally, at the network level, we surveyed existing load balancing algorithms for OBS.

In Chapter 3, we introduced an optimal channel scheduling algorithm for OBS networks called Ordered Scheduling and described two realisations of the

algorithm, namely, Basic Ordered Scheduling and Enhanced Ordered Scheduling. Basic Ordered Scheduling uses a purely slotted implementation and thus is simpler but has poorer loss performance than Enhanced Ordered Scheduling, which uses a hybrid implementation to fully realise the Ordered Scheduling algorithm. We discussed various practical implementation issues such as timing, complexity, signalling overhead, etc. The performance of Basic and Enhanced Ordered Scheduling was evaluated through simulations against various traffic and hardware parameters such as offered load, number of traffic classes, slot size and number of wavelengths per link. Their performance was also compared with that of LAUC-VF, which is a popular channel scheduling algorithm. They were found to significantly outperform LAUC-VF in almost all simulation scenarios.

In Chapter 4, we presented an absolute QoS framework for OBS networks that can provide edge-to-edge burst loss guarantees. The QoS framework consists of two parts. The first part includes a preemptive differentiation mechanism and a node-based admission control mechanism, which are responsible for providing guaranteed per-hop loss thresholds. Using these guaranteed thresholds as building blocks, a signalling mechanism in the second part coordinates reservation along the edge-to-edge path to achieve edge-to-edge loss guarantees. We gave an analytical model for the preemptive differentiation mechanism. The framework was evaluated through simulations at both the node level and network level and was found to be able to provide reliable loss guarantees under all network scenarios.

In Chapter 5, we analyzed the streamline effect in OBS networks and applied the results to a dynamic load balancing algorithm for absolute QoS traffic. The streamline effect is caused by the bufferless property of OBS networks. It makes the burst loss probability at an OBS link smaller than that obtained from the traditionally used $M|M|k|k$ model and strongly dependent on the number of input streams and their relative rates. We derived a more accurate analytical formula for the burst loss probability at a link, which was validated through simulation. The formula was incorporated into the link cost function of the load balancing

algorithm. Simulation results showed that the new load balancing algorithm performed better than one that uses the traditional $M|M|k|k$ model.

6.2 Suggestions for Future Work

The following are some of the possible areas of future work based on individual chapters in the thesis.

In Chapter 3, a possible area of future work would be to investigate more into parallel implementation and complexity issues of the algorithm. Although we gave some rough ideas on parallel implementation, it would take considerable research and development effort to come up with a detailed switch architecture.

There are two interesting areas of future work for the absolute QoS framework in Chapter 4. The first area is concerned mainly with the interface between an OBS network and access networks at ingress and egress nodes. Since LSP setup is a time consuming process and the allowed maximum number of LSPs is likely to be limited due to complexity reasons, a node has to map the QoS requirements of IP flows to those of LSPs in an efficient manner. For example, a new LSP may be set up with more reserved bandwidth than the sum of its component IP flows so as to accommodate new IP flows without needing to initiate new reservation requests. Alternatively, a node may decide to delay IP flow requests in order to collect more of them before initiating an LSP request. Also of interest is to investigate efficient policing mechanisms, which should minimise the loss probability of out-of-profile traffic without affecting in-profile QoS traffic. The second area of future work involves introducing some fairness mechanisms into the edge to edge reservation process. In the current scheme, LSPs spanning long paths have low successful reservation probabilities compared to LSPs with shorter paths. The future fairness scheme may give more favourable treatment to longer LSPs in the reservation process to remedy this situation.

In Chapter 5, the analytical loss formula provided has a wide range of possible applications in traffic engineering. We have considered only dynamic load bal-

ancing for reservation-based QoS traffic in this thesis. Future work may consider offline traffic engineering, protection routing and so on.

References

[1] A. S. Acampora and S. I. A. Shah, "Multihop lightwave networks: A comparison of store-and-forward and hot-potato routing," *IEEE Transactions on Communications*, vol. 40, no. 6, pp. 1082–1090, June 1992.

[2] Y. Arakawa, M. Sakuta, and I. Sasase, "QoS scheme with Burst Dropping in Optical Burst Switching," in *Proc. IEEE Pacific RIM Conference on Communications, Computers, and Signal Processing*, vol. 1, Aug. 2003, pp. 397–400.

[3] J. Bannister, F. Borgonovo, L. Fratta, and M. Gerla, "A performance model of deflection routing in multifiber networks with nonuniform traffic," *IEEE/ACM Transactions on Networking*, vol. 3, no. 5, pp. 509–520, Oct. 1995.

[4] R. Bhagwan and B. Lin, "Fast and scalable priority queue architecture for high-speed network switches," in *Proc. IEEE INFOCOM*, 2000.

[5] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, and W. Weiss, "An architecture for Differentiated Services," RFC 2475, Dec. 1998.

[6] C. Brackett, "Dense Wavelength Division Multiplexing networks: Principles and applications," *IEEE Journal on Selected Areas in Communications*, vol. 8, no. 6, pp. 948–964, Aug. 1990.

[7] H. C. Cankaya, S. Charcranoon, and T. S. El-Bawab, "A preemptive scheduling technique for OBS networks with service differentiation," in *Proc. IEEE GLOBECOM*, 2003.

[8] J. Cao, W. S. Cleveland, D. Lin, and D. X. Sun, "The effect of statistical multiplexing on the long-range dependence of Internet packet

traffic," Bell Labs, Tech. Rep., 2002. [Online]. Available: http://cm.bell-labs.com/cm/ms/departments/sia/doc/multiplex.pdf

[9] G. Castañón, L. Tančevski, and L. Tamil, "Optical Packet Switching with multiple path routing," *Computer Networks*, vol. 32, no. 5, pp. 653–662, May 2000.

[10] J.-B. Chang and C.-S. Park, "Efficient channel-scheduling algorithm in Optical Burst Switching architecture," in *Proc. IEEE Workshop on High Performance Switching and Routing*, May 2002, pp. 194–198.

[11] S. Charcranoon, T. S. El-Bawab, H. C. Cankaya, and J.-D. Shin, "Group scheduling for Optical Burst Switched (OBS) networks," in *Proc. IEEE GLOBECOM*, vol. 5, Dec. 2003, pp. 2745–2749.

[12] Y. Chen, M. Hamdi, and D. Tsang, "Proportional QoS over OBS networks," in *Proc. IEEE GLOBECOM*, 2001, pp. 1510–1514.

[13] Y. Chen, C. Qiao, and X. Yu, "Optical Burst Switching: a new area in optical networking research," *IEEE Network*, vol. 18, no. 3, pp. 16–23, 2004.

[14] Y. Chen, H. Wu, D. Xu, and C. Qiao, "Performance analysis of Optical Burst Switched node with deflection routing," in *Proc. IEEE ICC*, vol. 2, May 2003, pp. 1355–1359.

[15] T. Chich, J. Cohen, and P. Fraigniaud, "Unslotted deflection routing: A practical and efficient protocol for multihop optical networks," *IEEE/ACM Transactions on Networking*, vol. 9, no. 1, pp. 47–59, Feb. 2001.

[16] I. Chlamtac *et al.*, "CORD: Contention resolution by delay lines," *IEEE Journal on Selected Areas in Communications*, vol. 14, no. 5, pp. 1014–1029, June 1996.

[17] I. Chlamtac and A. Fumagalli, "Quadro-Star: A high performance optical WDM star network," *IEEE Transactions on Communications*, vol. 42, no. 8, pp. 2582–2591, Aug. 1994.

[18] I. Chlamtac, A. Ganz, and G. Karmi, "Lightpath communications: An approach to high bandwidth optical WAN's," *IEEE Transactions on Communications*, vol. 40, no. 7, pp. 1171–1182, July 1992.

[19] K. Claffy, G. Miller, and K. Thompson, "The nature of the beast: Recent traffic measurements from an Internet backbone," in *INET'98*, 1998. [Online]. Available: http://www.caida.org/outreach/papers/1998/Inet98/Inet98.pdf

[20] 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)," *IEEE/OSA Journal of Lightwave Technology*, vol. 20, no. 2, pp. 154–165, Feb. 2002.

[21] L. Dittmann *et al.*, "The European IST project DAVID: A viable approach toward Optical Packet Switching," *IEEE Journal on Selected Areas in Communications*, vol. 21, no. 7, pp. 1026–1040, Sept. 2003.

[22] K. Dolzer, C. Gauger, J. Späth, and S. Bodamer, "Evaluation of reservations mechanisms for Optical Burst Switching," *AEÜ International Journal of Electronics and Communications*, vol. 55, no. 1, Jan. 2001.

[23] C. Dovrolis, D. Stiliadis, and P. Ramanathan, "Proportional differentiated services: Delay differentiation and packet scheduling," *IEEE/ACM Transactions on Networking*, vol. 10, no. 1, pp. 12–26, Feb. 2002.

[24] M. Düser and P. Bayvel, "Analysis of a dynamically wavelength-routed optical burst switched network architecture," *IEEE/OSA Journal of Lightwave Technology*, vol. 20, no. 4, pp. 574–585, Apr. 2002.

[25] T. S. El-Bawab and J.-D. Shin, "Optical Packet Switching in core networks: Between vision and reality," *IEEE Communications Magazine*, vol. 40, no. 9, pp. 60–65, Sept. 2002.

[26] V. Eramo and M. Listanti, "Packet loss in a bufferless optical WDM switch employing shared tunable wavelength converters," *IEEE/OSA Journal of Lightwave Technology*, vol. 18, no. 12, pp. 1818–1833, Dec. 2000.

[27] V. Eramo, M. Listanti, and P. Pacifici, "A comparison study on the number of wavelength converters needed in synchronous and asynchronous all-optical switching architectures," *IEEE/OSA Journal of Lightwave Technology*, vol. 21, no. 2, pp. 340–355, Feb. 2003.

[28] V. Eramo, M. Listanti, and M. Spaziani, "Resources sharing in optical packet switches with limited-range wavelength converters," *IEEE/OSA Journal of Lightwave Technology*, vol. 23, no. 2, pp. 671–687, Feb. 2005.

[29] P. Fan, C. Feng, Y. Wang, and N. Ge, "Investigation of the time-offset-based QoS support with Optical Burst Switching in WDM networks," in *Proc. IEEE ICC*, 2002, pp. 2682–2686.

[30] F. Farahmand and J. P. Jue, "Look-ahead window contention resolution in Optical Burst Switched networks," in *Proc. IEEE Workshop on High Performance Switching and Routing*, 2003, pp. 147–151.

[31] F. Forghieri, A. Boroni, and P. R. Prucnal, "Analysis and comparison of hot-potato and single-buffer deflection routing in very high bit rate optical mesh networks," *IEEE Transactions on Communications*, vol. 43, no. 1, pp. 88–98, Jan. 1995.

[32] A. M. Glass *et al.*, "Advances in fiber optics," *Bell Labs Technical Journal*, vol. 5, no. 1, pp. 168–187, 2000.

[33] A. G. Greenberg and B. Hajek, "Deflection routing in hypercube networks," *IEEE Transactions on Communications*, vol. 40, no. 6, pp. 1070–1081, June 1992.

[34] X. Guan, I. L.-J. Thng, Y. Jiang, and H. Li, "Providing absolute QoS through virtual channel reservation in Optical Burst Switching networks," *Computer Communications*, vol. 28, no. 9, pp. 967–986, June 2005.

[35] Z. Haas, "The "staggering switch": An electronically controlled optical packet switch," *IEEE/OSA Journal of Lightwave Technology*, vol. 11, no. 5/6, pp. 925–936, 1993.

[36] C.-F. Hsu, T.-L. Liu, and N.-F. Huang, "Performance analysis of deflection routing in optical burst-switched networks," in *Proc. IEEE INFOCOM*, 2002, pp. 66–73.

[37] D. K. Hunter and I. Andonovic, "Approaches to optical Internet packet switching," *IEEE Communications Magazine*, vol. 38, no. 9, pp. 116–122, Sept. 2000.

[38] D. K. Hunter, M. C. Chia, and I. Andonovic, "Buffering in optical packet switches," *IEEE/OSA Journal of Lightwave Technology*, vol. 16, no. 12, pp. 2081–2094, Dec. 1998.

[39] D. K. Hunter, W. D. Cornwell, T. H. Gilfedder, A. Franzen, and I. Andonovic, "SLOB: A switch with large optical buffers for packet switching," *IEEE/OSA Journal of Lightwave Technology*, vol. 16, no. 10, pp. 1725–1736, Oct. 1998.

[40] M. Iizuka, M. Sakuta, Y. Nishino, and I. Sasase, "A scheduling algorithm minimizing voids generated by arriving bursts in optical burst switched WDM network," in *Proc. IEEE GLOBECOM*, Nov. 2002.

[41] A. Ioannou and M. Katevenis, "Pipelined heap (priority queue) management for advanced scheduling in high-speed networks," in *Proc. IEEE ICC*, vol. 7, June 2001, pp. 2043–2047.

[42] *Traffic Control and Congestion Control in B-ISDN*, Recommendation I.371, ITU-T, 1995.

[43] A. Kaheel and H. Alnuweiri, "Quantitative QoS guarantees in Labeled Optical Burst Switching networks," in *Proc. IEEE GLOBECOM*, vol. 3, 2004, pp. 1747–1753.

[44] L. Kleinrock, *Queueing Systems, Volume 1: Theory.* New York: Wiley Interscience, 1975.

[45] R. Langenhorst *et al.*, "Fiber loop optical buffer," *IEEE/OSA Journal of Lightwave Technology*, vol. 14, no. 3, pp. 324–335, Mar. 1996.

[46] A. Lazzez, N. Boudriga, and S. G. E. Fatmi, "Segments-priorities based contention resolution technique for QoS support in Optical Burst-Switched networks," in *Proc. 12th IEEE Mediterranean Electrotechnical Conference*, vol. 2, May 2004, pp. 527–530.

[47] S. Lee, L. Kim, J. Song, D. Griffith, and K. Sriram, "Dynamic deflection routing with virtual wavelength assignment in Optical Burst-Switched networks," *Photonic Network Communications*, vol. 9, no. 3, pp. 347–356, May 2005.

[48] H. Li, H. Neo, and L.-J. I. Thng, "Performance of the implementation of a PipeLine buffering system in Optical Burst Switching networks," in *Proc. IEEE GLOBECOM*, 2003.

[49] J. Li, G. Mohan, and K. C. Chua, "Load balancing using adaptive alternative routing in IP-over-WDM Optical Burst Switching networks," in *Proc. International Conference on Optical Networking and Communications*, Oct. 2003.

[50] W. Liao and C.-H. Loi, "Providing service differentiation for Optical-Burst-Switched networks," *IEEE/OSA Journal of Lightwave Technology*, vol. 22, no. 7, pp. 1651–1660, July 2004.

[51] D. Q. Liu and M. T. Liu, "Burst scheduling for differentiated services in Optical Burst Switching WDM networks," *International Journal of Communication Systems*, vol. 17, no. 2, pp. 127–140, 2004.

[52] J. Liu, N. Ansari, and T. J. Ott, "FRR for latency reduction and QoS provisioning in OBS networks," *IEEE Journal on Selected Areas in Communications*, vol. 21, no. 7, pp. 1210–1219, Sept. 2003.

[53] K. Long, R. S. Tucker, and C. Wang, "A new framework and burst assembly for IP DiffServ over Optical Burst Switching networks," in *Proc. IEEE GLOBECOM*, 2003.

[54] E. Modiano and P. J. Lin, "Traffic grooming in WDM networks," *IEEE Communications Magazine*, vol. 39, no. 7, pp. 124–129, July 2001.

[55] G. Mohan, K. Akash, and M. Ashish, "Efficient techniques for improved QoS performance in WDM Optical Burst Switched networks," *Computer Communications*, vol. 28, no. 7, pp. 754–764, May 2005.

[56] A. R. Moral, P. Bonenfant, and M. Krishnaswamy, "The Optical Internet: Architectures and protocols for the global infrastructure of tomorrow," *IEEE Communications Magazine*, vol. 39, no. 7, pp. 152–159, July 2001.

[57] M. J. O'Mahony, D. Simeonidou, D. K. Hunter, and A. Tzanakaki, "The application of Optical Packet Switching in future communication networks," *IEEE Communications Magazine*, vol. 39, no. 3, pp. 128–135, Mar. 2001.

[58] H. Overby, "Performance modelling of optical packet switched networks with the Engset traffic model," *OSA Optics Express*, vol. 13, no. 5, pp. 1685–1695, Mar. 2005.

[59] H. Øverby and N. Stol, "Quality of Service in asynchronous bufferless Optical Packet Switched networks," *Telecommunication Systems*, vol. 27, no. 2–4, pp. 151–179, 2004.

[60] M. H. Phùng, K. C. Chua, G. Mohan, M. Motani, and T. C. Wong, "Absolute QoS signalling and reservation in Optical Burst-Switched networks," in *Proc. IEEE GLOBECOM*, Nov. 2004.

[61] ——, "A preemptive differentiation scheme for absolute loss guarantees in OBS networks," in *Proc. IASTED International Conference on Optical Communication Systems and Networks*, July 2004.

[62] ——, "The streamline effect in OBS networks and its application in load balancing," in *Proc. 2nd International Conference on Broadband Networks*, Oct. 2005.

[63] M. H. Phùng, K. C. Chua, G. Mohan, M. Motani, T. C. Wong, and P. Y. Kong, "On Ordered Scheduling for Optical Burst Switching," *Computer Networks*, vol. 48, no. 6, pp. 891–909, Aug. 2005.

[64] C. Qiao, "Labeled Optical Burst Switching for IP-over-WDM integration," *IEEE Communications Magazine*, vol. 38, no. 9, pp. 104–114, Sept. 2000.

[65] C. Qiao and M. Yoo, "Optical Burst Switching – a new paradigm for an optical Internet," *Journal of High Speed Network*, vol. 8, no. 1, pp. 69–84, Jan. 1999.

[66] B. Ramamurthy and B. Mukherjee, "Wavelength conversion in WDM networking," *IEEE Journal on Selected Areas in Communications*, vol. 16, no. 7, pp. 1061–1073, Sept. 1998.

[67] R. Ramaswami and K. N. Sivarajan, *Optical Networks: A Practical Perspective*, 2nd ed. Morgan Kaufmann Publishers, 2002.

[68] Z. Rosberg, H. L. Vu, M. Zukerman, and J. White, "Performance analyses of Optical Burst Switching networks," *IEEE Journal on Selected Areas in Communications*, vol. 21, no. 7, pp. 1187–1197, 2003.

[69] E. Rosen, A. Viswanathan, and R. Callon, "Multiprotocol label switching architecture," RFC 3031, Jan. 2001.

[70] D. B. Sarrazin, H. F. Jordan, and V. P. Heuring, "Fiber optic delay line memory," *Applied Optics*, vol. 29, no. 5, pp. 627–637, Feb. 1990.

[71] G. Shen, S. K. Bose, T. H. Cheng, C. Lu, and T. Y. Chai, "Performance study on a WDM packet switch with limited-range wavelength converters," *IEEE Communications Letters*, vol. 5, no. 10, pp. 432–434, Oct. 2001.

[72] C.-W. Tan, G. Mohan, and J. C.-S. Lui, "Achieving proportional loss differentiation using probabilistic preemptive burst segmentation in Optical Burst Switching WDM networks," in *Proc. IEEE GLOBECOM*, vol. 3, 2004, pp. 1754–1758.

[73] S. K. Tan, G. Mohan, and K. C. Chua, "Algorithms for burst rescheduling in WDM Optical Burst Switching networks," *Computer Networks*, vol. 41, no. 1, pp. 41–55, Jan. 2003.

[74] ——, "Burst rescheduling with wavelength and last-hop FDL reassignment in WDM Optical Burst Switching networks," in *Proc. IEEE ICC*, vol. 2, May 2003, pp. 1448–1452.

[75] W. Tan, S. Wang, and L. Li, "Burst segmentation for void-filling scheduling and its performance evaluation in Optical Burst Switching," *Optics Express*, vol. 12, no. 26, pp. 6615–6623, Dec. 2004.

[76] H. Tanida, K. Ohmae, Y.-B. Choi, and H. Okada, "An effective BERN/CRN typed deflection routing for QoS guaranteed Optical Burst Switching," in *Proc. IEEE GLOBECOM*, 2003, pp. 2601–2606.

[77] L. Tančevski, S. Yegnanarayanan, G. Castanon, L. Tamil, F. Masetti, and T. McDermott, "Optical routing of asynchronous, variable length packets," *IEEE Journal on Selected Areas in Communications*, vol. 18, no. 10, pp. 2084–2093, Oct. 2000.

[78] G. P. Thodime, V. M. Vokkarane, and J. P. Jue, "Dynamic congestion-based load balanced routing in Optical Burst-Switched networks," in *Proc. IEEE GLOBECOM*, 2003.

[79] J. S. Turner, "Terabit burst switching," *Journal of High Speed Network*, vol. 8, no. 1, pp. 3–16, Jan. 1999.

[80] E. A. Varvarigos and V. Sharma, "The ready-to-go virtual circuit protocol: A loss-free protocol for multigigabit networks using FIFO buffers," *IEEE/ACM Transactions on Networking*, vol. 5, no. 5, pp. 705–718, Oct. 1997.

[81] V. M. Vokkarane, , G. P. Thodime, V. U. Challagulla, and J. P. Jue, "Channel scheduling algorithms using burst segmentation and FDLs for optical burst-switched networks," in *Proc. IEEE ICC*, 2003, pp. 1443–1447.

[82] V. M. Vokkarane and J. P. Jue, "Burst segmentation: an approach for reducing packet loss in Optical Burst Switched networks," *SPIE/Kluwer Optical Networks*, vol. 4, no. 6, pp. 81–89, 2003.

[83] ——, "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, no. 7, pp. 1198–1209, Sept. 2003.

[84] H. L. Vu and M. Zukerman, "Blocking probability for priority classes in optical burst switching networks," *IEEE Communications Letters*, vol. 6, no. 5, pp. 214–216, May 2002.

[85] J. Y. Wei and R. I. McFarland Jr., "Just-in-time signaling for WDM Optical Burst Switching networks," *IEEE/OSA Journal of Lightwave Technology*, vol. 18, no. 12, pp. 2019–2037, Dec. 2000.

[86] H. Wen, H. Song, L. Li, and S. Wang, "Load-balancing contention resolution in LOBS based on GMPLS," in *Proc. 4th International Conference on Parallel and Distributed Computing, Applications and Technologies*, Aug. 2003, pp. 590–594.

[87] I. Widjaja, "Performance analysis of burst admission-control protocols," *IEE Proceedings – Communications*, vol. 142, no. 1, pp. 7–14, Feb. 1995.

[88] Y. Xiong, M. Vandenhoute, and H. C. Cankaya, "Control architecture in Optical Burst-Switched WDM networks," *IEEE Journal on Selected Areas in Communications*, vol. 18, no. 10, pp. 1838–1851, Oct. 2000.

[89] J. Xu, C. Qiao, J. Li, and G. Xu, "Efficient burst scheduling algorithms in Optical Burst-Switched networks using geometric techniques," *IEEE Journal on Selected Areas in Communications*, vol. 22, no. 9, pp. 1796–1811, Nov. 2004.

[90] L. Xu, H. G. Perros, and G. N. Rouskas, "A queueing network model of an edge Optical Burst Switching node," in *Proc. IEEE INFOCOM*, 2003.

[91] L. Yang, Y. Jiang, and S. Jiang, "A probabilistic preemptive scheme for providing service differentiation in OBS networks," in *Proc. IEEE GLOBECOM*, 2003.

[92] M. Yang, S. Zheng, and D. Verchere, "A QoS supporting scheduling algorithm for Optical Burst Switching DWDM networks," in *Proc. IEEE GLOBECOM*, 2001, pp. 86–91.

[93] M. Yoo, C. Qiao, and S. Dixit, "QoS performance of Optical Burst Switching in IP-over-WDM networks," *IEEE Journal on Selected Areas in Communications*, vol. 18, no. 10, pp. 2062–2071, Oct. 2000.

[94] X. Yu, J. Li, Y. Chen, X. Cao, and C. Qiao, "Traffic statistics and performance evaluation in Optical Burst Switched networks," *IEEE/OSA Journal of Lightwave Technology*, vol. 22, no. 12, pp. 2722–2738, Dec. 2004.

[95] A. H. Zaim, I. Baldine, M. Cassada, G. N. Rouskas, H. G. Perros, and D. Stevenson, "Jumpstart just-in-time signaling protocol: A formal description using extended finite state machines," *Optical Engineering*, vol. 42, no. 2, pp. 568–585, Feb. 2003.

[96] A. Zalesky, H. L. Vu, Z. Rosberg, E. W. M. Wong, and M. Zukerman, "Modelling and performance evaluation of Optical Burst Switched networks with

deflection routing and wavelength reservation," in *Proc. IEEE INFOCOM*, Mar. 2004, pp. 1864–1871.

[97] J. Zhang *et al.*, "Explicit routing for traffic engineering in Labeled Optical Burst-Switched WDM networks," in *Proc. 4th International Conference on Computational Science*, June 2004.

[98] Q. Zhang, V. M. Vokkarane, J. Jue, and B. Chen, "Absolute QoS differentiation in Optical Burst-Switched networks," *IEEE Journal on Selected Areas in Communications*, vol. 22, no. 9, pp. 1781–1795, Nov. 2004.

[99] Z. Zhang and Y. Yang, "Performance modeling of bufferless WDM packet switching networks with wavelength conversion," in *Proc. IEEE GLOBECOM*, vol. 5, 2003, pp. 2498–2502.

[100] S. Q. Zheng, Y. Xiong, and H. C. Cankaya, "Hardware design of a channel scheduling algorithm for Optical Burst Switching routers," in *Proceedings of SPIE*, vol. 4872, July 2002, pp. 199–209.

[101] W. D. Zhong and R. S. Tucker, "A new wavelength-routed photonic packet buffer combining traveling delay lines with delay line loops," *IEEE/OSA Journal of Lightwave Technology*, vol. 19, no. 8, pp. 1085–1092, Aug. 2001.

Author's Publications

1. M. H. Phùng, K. C. Chua, G. Mohan, M. Motani, and T. C. Wong, "A preemptive differentiation scheme for absolute loss guarantees in OBS networks," in *Proc. IASTED International Conference on Optical Communication Systems and Networks*, July 2004.
2. M. H. Phùng, K. C. Chua, G. Mohan, M. Motani, and T. C. Wong, "Absolute QoS signalling and reservation in Optical Burst-Switched networks," in *Proc. IEEE Globecom*, Nov. 2004.
3. M. H. Phùng, K. C. Chua, G. Mohan, M. Motani, T. C. Wong, and P. Y. Kong, "On Ordered Scheduling for Optical Burst Switching," *Computer Networks*, vol. 48, no. 6, pp. 891–909, Aug. 2005.
4. M. H. Phùng, K. C. Chua, G. Mohan, M. Motani, and T. C. Wong, "The streamline effect in OBS networks and its application in load balancing," in *Proc. 2nd International Conference on Broadband Networks*, Oct. 2005.
5. M. H. Phùng, D. Shan, K. C. Chua, G. Mohan, "Performance Analysis of a Bufferless OBS Node Considering the Streamline Effect," *IEEE Communications Letters*, vol. 10, no. 4, pp. 293–295, Apr. 2006.
6. M. H. Phùng, K. C. Chua, G. Mohan, M. Motani, and T. C. Wong, "An Absolute QoS Framework for Loss Guarantees in Optical Burst-Switched Networks," *IEEE Transactions on Communications*, to appear.