Analysis and Design of Optical Burst Switching Networks

by

Ayman Malek Kaheel

B.Sc., Cairo University, 1995
M.Sc., University of Louisville, 1998

A THESIS SUBMITTED IN PARTIAL FULFILMENT OF
THE REQUIREMENTS FOR THE DEGREE OF

DOCTOR OF PHILOSOPHY

in

THE FACULTY OF GRADUATE STUDIES

(Electrical and Computer Engineering)

THE UNIVERSITY OF BRITISH COLUMBIA

June 3, 2005

Abstract

Optical Burst Switching (OBS) is a hybrid technique between coarse grain optical circuit switching and fine grain optical packet switching. In OBS networks, user data is switched entirely in the optical domain, while control and management functions are performed in the electrical domain. This separation of the data plane and the control plane allows OBS networks to provide reasonably high levels of utilization while circumventing the need for optical buffering. In spite of OBS favorable features, several issues need to be addressed before OBS can be deployed in the Internet backbone.

The objectives of this thesis are twofold: devise new methods for quality-of-service (QoS) provisioning in OBS networks, and develop new wavelength scheduling algorithms for enhancing the blocking probability in OBS networks.

QoS provisioning is a major research problem in OBS networks. This is mainly because of the absence of the concept of "packet queues" in OBS networks. This thesis proposes two approaches for QoS provisioning in OBS networks. The first approach is a simple, yet effective scheme, called preemptive prioritized just enough time (PPJET). PPJET provides better service for high priority traffic by dropping reservations belonging to lower priority traffic using a new channel scheduling algorithm called preemptive latest available unused channel with void filling (PLAUC-VF). Simulation results show that PPJET outperforms offset-based QoS schemes both in terms of dropping probability and end-to-end delay.

As a second approach for solving the QoS problem in OBS networks, we present a detailed architecture for providing quantitative QoS guarantees with respect to end-to-end delay, throughput, and packet loss probability in labeled OBS networks. The architecture describes a novel approach for applying fair scheduling algorithms in both the data plane of labeled OBS edge nodes and the control plane of core nodes without the need for optical buffering. In addition, we present analytical results for delay, throughput, and blocking probability in the proposed architecture. Simulation results demonstrate that the proposed architecture provides accurate and controllable service differentiation in labeled OBS networks.

The absence of optical buffers in OBS nodes, coupled with the one way nature of OBS signaling protocols, drives the blocking probability to become the main performance measure in OBS networks. This give rise to the need for analytical models for calculating the blocking probability in OBS networks. In this thesis we present an approximate analytical model for calculating the blocking probability in OBS networks. The proposed analytical model takes into consideration the peculiar characteristics of OBS networks. To verify its accuracy, we compared the model results with results from a discrete-event simulation model. The proposed model results are in satisfactory agreement with simulation results.

The blocking probability at an OBS node depends to a certain degree on how efficiently can the wavelength scheduling algorithm handle voids on wavelength channels. This fact has led to a growing interest in the area of wavelength scheduling in OBS networks. In this thesis we survey all previously proposed wavelength scheduling algorithms for OBS networks. In addition, we present two new wavelength scheduling algorithms: Min-AV and Max-NGV. We compared the performance of the newly proposed algorithms to those previously proposed using discrete-event simulation. Simulation results show that, in general, the Min-AV algorithm performs better than all previously proposed algorithms.

Previously proposed wavelength schedulers, in addition to the Min-AV and Max-AV algorithms, are considered to be greedy algorithms. They are greedy in the sense that they consider every reservation request individually, and make the choice that looks best at the moment. We present in this thesis a new class of wavelength scheduling algorithms for OBS networks. The proposed wavelength scheduling algorithms process a batch of reservation requests together, instead of processing them one by one, and accept the requests that will maximize the utilization of the wavelength channels. We describe an optimal batch scheduler that serves as an upper bound on the performance of batch scheduling algorithms. Furthermore, we introduce four novel heuristic batch scheduling algorithms. Simulation results suggest that batch schedulers could significantly decrease the blocking probability in OBS networks.

Contents

Abstract ii

Contents v

List of Figures xi

List of Tables xv

List of Abbreviations xvi

Dedication xix

Acknowledgements xx

1 Introduction 1

1.1 Motivation 1

1.1.1 All-Optical Networks 2

1.1.2 Quality of Service in All-Optical Networks 5

1.2 The Thesis 7

1.2.1 Objectives 7

1.2.2 Contributions 7

1.2.3 Road Map 9

2 Overview of Optical Burst Switching Networks 11

2.1 Signaling Protocols 12

2.2 OBS Core Nodes 14

2.3 OBS Edge Nodes 16

3 Wavelength Scheduling Algorithms in OBS Networks 19

3.1 Introduction 19

3.2 Previous Work 21

3.3 Proposed Algorithms 25

3.3.1 Max-NGV 25

3.3.2 Min-AV 26

3.4 Performance Evaluation 27

3.4.1 Simulation Setup 27

3.4.2 Simulation Results 29

3.5 Concluding Remarks 34

4 A Novel Priority Scheme for Supporting QoS in OBS Networks . . . 35

4.1 Introduction 35

4.2 Overview of PJET 37

4.3 Introducing PPJET 38

4.3.1 PLAUC-VF Channel Scheduling Algorithm 39

4.4 Dropping Probability Calculation in PPJET 40

4.5 Performance Analysis 43

4.5.1 Network Setup 44

4.5.2 Burst Dropping Probability 46

4.5.3 ETE delay 48

4.6 Concluding Remarks 53

5 A New Analytical Model for Computing Blocking Probability in OBS Networks 54

5.1 Introduction 54

5.2 Blocking Probability 55

5.3 Advance Reservation Systems 56

5.4 Proposed Model 59

5.4.1 Problem Description and Assumptions 59

5.4.2 Traffic Model 60

5.4.3 State Description 62

5.4.4 Blocking Probability Calculation 64

5.5 Performance Evaluation 67

5.5.1 Effect of Offset Time 68

5.5.2 Effect of Burst Size 71

5.5.3 Comparison with Erlang's Loss Formula 73

5.5.4 Effect of the number of wavelength channels 75

5.5.5 Effect of time slot size 76

5.6 Concluding Remarks 80

6 Quantitative QoS Guarantees in Labeled OBS Networks 82

6.1 Introduction 82

6.2 Proposed Architecture 84

6.2.1 OBS Edge node 85

6.2.2 Signaling Protocol 88

6.2.3 OBS Core node 88

6.3 QoS guarantees 91

6.3.1 Weighted Fair Queueing 91

6.3.2 Bandwidth Guarantees 92

6.3.3 Delay guarantees 93

6.3.4 Burst Loss Probability 97

6.4 Performance Evaluation 99

6.4.1 Delay Guarantees 101

6.4.2 Loss Probability 102

6.4.3 Bandwidth guarantees 107

6.5 Concluding Remarks 107

7 Batch Scheduling Algorithms: A Class of Wavelength Schedulers in

OBS Networks 111

7.1 Introduction 111

7.2 Batch Scheduling 112

7.2.1 Definitions 113

7.2.2 Interval Graph Coloring 114

7.3 An Optimal Batch Scheduling Algorithm (BATCH-OPT) 116

7.4 Heuristic Batch Scheduling Algorithms 118

7.4.1 Smallest-Last Vertex Ordering (SLV) 119

7.4.2 Maximal Cliques First (MCF) 122

7.4.3 Smallest Start-time First Ordering (SSF) 125

7.4.4 Largest Interval First Ordering (LIF) 126

7.5 Performance Evaluation 126

7.5.1 Simulation Setup 127

7.5.2 Blocking Probability vs. Offered Load 128

7.5.3 Blocking Probability vs. Offset Time Range 129

7.5.4 Effect of the Acceptance Delay 134

7.6 Concluding Remarks 139

8 Conclusions and Future Research 140

8.1 Summary 140

8.2 Future Work 144

Bibliography 146

A Algorithm for Finding Maximal Cliques in Interval Graphs 158

B Offline Optimal Scheduling Algorithm 160

List of Figures

2.1 An OBS network. 12

2.2 Typical data burst format. 14

2.3 Block diagram of an OBS core node. 15

2.4 Block diagram of an OBS edge node (ingress). 17

2.5 Block diagram of an OBS edge node (egress). 17

3.1 Wavelength scheduling in OBS nodes. 21

3.2 Illustration of the LAUC algorithm. 23

3.3 Illustration of the LAUC-VF algorithm. 24

3.4 Illustration of the Min-NGV algorithm. 24

3.5 Illustration of the Max-NGV algorithm. 26

3.6 Illustration of the Min-AV algorithm. 27

3.7 Simulation model used in Section 3.4. 29

3.8 Effect of the burst size on the performance of different wavelength scheduling algorithms in terms of the burst blocking probability. 32

3.9 Effect of the maximum offset time on the performance of different wavelength scheduling algorithms in terms of the burst blocking probability. . 33

4.1 Illustration of the PLAUC-VF algorithm. 41
4.2 Simulation network model for Section 4.5. 44
4.3 Comparison between PPJET and PJET class 1 dropping probabilities in the case of constant burst sizes. 46
4.4 Comparison between PPJET and PJET class 1 dropping probabilities in the case of exponentially distributed burst sizes. 47
4.5 Effect of the mean burst size on class 1 dropping probability. 49
4.6 Class 1 ETE delay comparison in the case of exponential B_{max}. 51
4.7 Class 1 ETE delay comparison in the case of constant B_{max}. 52

5.1 Example to illustrate the difference in the loss protocol between advance reservation systems and OBS networks. 58
5.2 Simplified OBS-JET model. 60
5.3 Simulation model for Section 5.5. 68
5.4 Illustration of the FF-VF algorithm 69
5.5 Effect of the burst's offset time on its blocking probability. 70
5.6 Effect of the burst's size on its blocking probability. 72
5.7 Comparison between the proposed model and Erlang's loss formula. . . . 74

5.8 Effect of the number of wavelengths on the burst blocking blocking probability. 75
5.9 Effect of slot size on the accuracy of the proposed model. 79

6.1 Functional blocks of an OBS edge node. 86
6.2 Functional blocks of an OBS core node. 89
6.3 Illustration of the virtual queue concept. 90
6.4 Bursts arrivals to the burst queue depends on the values of σ and B_{max}. . 96
6.5 Simulation network topology for Section 6.4. 100
6.6 End-to-end delay in the case of conforming traffic flows. 103
6.7 End-to-end delay when FEC 0 is non-conforming. 104
6.8 Illustration of a sample path of the loss probability. 105
6.9 High loss probability for FEC 0 (non-conforming). 106
6.10 Throughput for conforming traffic. 108
6.11 Throughput with FEC 0 non-conforming. 109

7.1 Control burst scheduling using greedy algorithms. 112
7.2 Control burst scheduling with batch queue. 113
7.3 Mapping batch scheduling to graph coloring. 115
7.4 Interval graph example. 121
7.5 Illustration of the SLV algorithm. 122
7.6 Illustration of the MCF algorithm. 125

7.7 Simulation model for Section 7.5. 127

7.8 Blocking probability vs. offered load in case of exponentially distributed burst sizes. 128

7.9 Blocking probability vs. offered load in case of constant burst size. 130

7.10 Blocking probability vs. offset time range. 131

7.11 Effect of offset time range on the burst blocking probability. 133

7.12 Blocking probability vs. acceptance delay with exponentially distributed burst sizes. 135

7.13 Blocking probability vs. acceptance delay with constant burst sizes. . . . 136

7.14 Effect of varying the acceptance delay and the offset time range on the blocking probability. 138

B.1 Example for minimum cost flow construction 162

List of Tables

5.1 Execution time of the analytical model for different values of M 77

5.2 Slot size values and their corresponding mean burst sizes, offsets, and max offset values 78

6.1 Comparison of average and worst-case delay values between analysis and simulation. 102

List of Abbreviations

CAT	Containerization with Aggregation-Timeout
CB	Control Burst
CoS	Class of Service
DB	Data Burst
Diffserv	Differentiated Services
E/O	Electrical to Optical
ETE	End-to-End
FDL(s)	Fiber Delay Line(s)
FEC(s)	Forwarding Equivalence Class(es)
FFUC	First Fit Unscheduled Channel
FF-VF	First Fit with Void Filling
FPQ	Fair Packet Queueing
GMPLS	Generalized MPLS
GPS	Generalized Processer Sharing
ILP	Integer Linear Program
Intserv	Integrated Services

IP	Internet Protocol
JET	Just Enough Time
JIT	Just In Time
LAUC	Latest Available Unscheduled Channel
LAUC-VF	Latest Available Unused Channel with Void Filling
LIF	Largest Interval First
LOBS	Labeled OBS
Max-NGV	Maximum Newly Generated Void
MCF	Maximal Cliques First
Min-AV	Minimum Available Void
Min-NGV	Minimum Newly Generated Void
MPLS	Multi-Protocol Label Switching
O/E	Optical to Electrical
OBS	Optical Burst Switching
OPS	Optical Packet Switching
OSM	Optical Switching Matrix
OXC	Optical Cross-Connect
PJET	Prioritized JET
PLAUC-VF	Preemptive LAUC-VF
PPJET	Preemptive PJET
QoS	Quality of Service

RFD	Reserve Fixed Duration
SCFQ	Self-Clocked Fair Queueing
SCU	Switch Control Unit
SFQ	Start-time Fair Queueing
SLV	Smallest-Last Vertex
SSF	Smallest Start-time First
TAG	Tell-And-Go
TCP	Transmission Control Protocol
WDM	Wavelength Division Multiplexing
WFQ	Weighted Fair Queueing
WR	Wavelength Routed

Dedication

To my beloved parents.

الي امي الحبيبة و الي ابي الحبيب
رَبِّ ارْحَمْهُمَا كَمَا رَبَّيَانِي صَغِيرًا

Acknowledgements

First and foremost, I am obediently thankful to Allah, praise and glory be to him, for his countless blessings and mercy. Without his guidance and mercy I can do nothing.

I would like to express my heartfelt gratitude to my supervisor, Dr. Hussein Alnuweiri, for his guidance, technical advice, invaluable feedback, understanding, generous support, and friendship. I have passed through many personal and difficult circumstances during the period of my Ph.D. program. Without Dr. Alnuweiri genuine kindness, I do not think you would be reading these words today.

I am very grateful to Dr. Fayez Gebali for his valuable suggestions, and technical assistance. Although he is very busy, I always found him easy to approach and available for help.

I would like to thank Dr. Victor Leung for funding my research in the early years of the program.

I would like to express my appreciation to my thesis committee for their valuable comments which enhanced the presentation of this thesis.

I am also thankful for Dr. Will Evans for the number of discussions that we had and for his help with the minimum-cost flow problem.

I would like to thank Professors Richard Anstee, Will Evans, Nick Jaeger, Vikram Krishnamurthy, Victor Leung, and Vincent Wong for allowing me to attend their courses without registering for them.

I would like to express my sincere thanks to my friend Dr. Watheq El-Kharashi for carefully reviewing my thesis. His valuable comments and suggestions have been very useful in enhancing the presentation of this thesis.

I also would like to thank my friends and colleagues Tamer Khattab and Amr Mohamed for introducing me to OBS networks and for our fruitful collaboration that lead to building our first OBS simulation model.

I have been blessed to come in contact with many wonderful people throughout my study period at UBC. This page is definitely not enough to mention all of them, but, the least I can say to them is: thank you for making my stay at UBC such a pleasant experience.

Among the many friends that I had during my study period, the following friends had the most positive effect on my life: Ahmed, Anwar, AmrM, AmrW, Khaled, Tamer, Halfawy, Watheq, Abdo, Junaid, Tariq, and Awad.

Words cannot describe my feelings toward my parents. I can only pray to God to reward them for their constant support, persistent encouragement, and most of all for their unwavering love.

I would like to express my love and deep appreciation to the person who stood beside me through all times, my wife Faten. She was always there for me with comforting words,

encouraging thoughts, and a contagious smile.

I would like to extend my thanks to my sisters, Rabab and May, for being a source of motivation and prayers.

I would like also to thank my parents-in-law for all their prayers, encouragement, and support.

Last, but not least, I would like to thank my son and daughter, Omar and Mariam, for all the joy and happiness they have brought to my life.

Chapter 1

Introduction

1.1 Motivation

The exponential growth of the Internet traffic drives the demands for higher bandwidth and high-speed Internet protocol (IP) routers at an unprecedented rate. It is projected that Internet traffic will continue to double every year, regardless of conditions in the financial markets [63].

The major technology at hand that is promising to meet the high bandwidth demand is optical networking with wavelength-division-multiplexing (WDM) technology. The advances in WDM technology are making it possible to harness the huge potential bandwidth of optical fibers (it was shown that it is theoretically possible to send 100 terabits per second on a single fibre [49]). WDM is an optical multiplexing technique that allows better exploitation of the fiber capacity by simultaneously transmitting data over multiple wavelength channels.

In traditional optical networks, WDM was used to solve the problem of transmitting the required bandwidth on optical links. In this type of optical networks, data is transmitted between nodes in the optical domain, but all the switching and intelligent routing functions at intermediate nodes are performed in the electronic domain. Therefore, data has to go through optical-to-electrical (O/E) and electrical-to-optical (E/O) conversions at each node it traverses. The main problem with this type of network is that the switching and processing capabilities of electronic IP routers/switches lag behind the transmission capabilities of optical fibers. IP routers with capacities up to few hundred gigabits per second are currently available. At the same time, commercially available WDM fibers can achieve capacities around 1.6 Tbps (10 Gbps × 160 wavelength channels) [23]. This serious mismatch between the transmission capacity of optical fiber links and the electronic IP routers/switches stands against translating the huge bandwidth provided by the optical links into a user available bandwidth. This is one of the key drivers for the next generation of optical networks: all-optical networks.

1.1.1 All-Optical Networks

In an all-optical network, data is switched from ingress nodes to egress nodes in an optical form, without undergoing any O/E conversions along its path [61]. The transfer of the switching function from electronics to optics results in an increase in the switching speed and in the system throughput. In addition, the elimination of O/E and E/O conversions results in a major decrease in the overall system cost, since the equipment associated with these conversions contributes a significant part of the total cost of today's networks [56].

In the ultimate all-optical network, IP packets will be entirely processed in the optical domain (the header and the payload). However, it is not predicted that this kind of all-optical networks will be realized in the foreseeable future because of the limitations of the optical component technology. Furthermore, electronics will be needed to perform intelligent control functions. Therefore, current approaches to all-optical networks focus on the transition from traditional optical networks to a form of optical networks in which control operations are performed in the electrical domain and data is handled entirely in the optical domain. Three main approaches exist for the gradual transition to all-optical networks: wavelength routing, optical packet switching, and optical burst switching.

Wavelength Routing Networks

In wavelength routing networks (WR) [18], an all-optical path is established between edges of the network. This optical path is called a lightpath [6] and is created by reserving a dedicated wavelength channel on every link along the path. After a predefined period of time, the lightpath is released. WR networks consist of optical cross-connect (OXC) [9] devices connected by point-to-point fiber links in an arbitrary topology. OXC devices are capable of differentiating data streams based on the input port from which a data stream arrives and its wavelength [13]. By having a configuration table within each device, a cross connection can be setup between a certain wavelength on an input port and a wavelength (which might be different from the input wavelength) on an output port. As a result, data transmitted between lightpath endpoints requires no processing, no O/E conversion, and no buffering at intermediate nodes. However, being a form of circuit switching networks, WR networks do not use statistical sharing of resources and therefore, provides low bandwidth utilization.

Optical Packet Switching Networks

In packet switching networks, IP traffic is processed and switched at every IP router on a packet-by-packet basis. An IP packet contains a payload and header. The packet header contains the information required for routing the packet while the payload carries the actual data. The future and ultimate goal of optical packet switching (OPS) networks is to process the packet header entirely in the optical domain [53, 62]. With the current technology, it is not possible to do such processing in the optical domain. A solution for this problem is to process the header in the electronic domain and keep the payload in the optical domain [26]. This process requires the packet payload to be delayed (optically buffered) until the header is fully processed and the packet is classified, after which the packet is assigned a wavelength. The main advantage of OPS is that it can increase the networks bandwidth utilization by statistically multiplexing many packet streams together. However, because of the its synchronization requirements, many technical challenges remain to be addressed for OPS to become a viable solution [12].

Optical Burst Switching

Optical burst switching (OBS) is an optical switching technique that combines the advantages of both WR networks and OPS networks [59, 67, 74]. Similar to WR networks, there is no need for buffering and electronic processing for data at intermediate nodes.

At the same time, OBS ensures high bandwidth utilization, just like OPS, by reserving bandwidth on a link only when data is actually required to be transferred through the link. In OBS networks, packets are assembled into bursts at network ingress and disassembled back into packets at network egress. Each burst consists of a header and a payload. OBS uses separate wavelength channels to transmit the burst payload and its header. This physical separation of switching and transmission of the burst header and payload helps to facilitate the electronic processing of headers at optical core nodes, and provide an all-optical path for the burst payload.

Analysis and design of OBS networks is the focus of this thesis. OBS has received a lot of attention during the past few years and is fast becoming an important area of research. Since OBS exploits both the huge bandwidth in fibers for switching/transmission and the sophisticated processing capability of electronics, it is able to achieve cost reduction and utilize the technological advances in both optical and electronic domains. It is believed that OBS is a potential method for building the next generation optical Internet.

1.1.2 Quality of Service in All-Optical Networks

Over the past decade, a significant amount of work has been dedicated to the issue of providing QoS in non-WDM IP networks. Basic IP assumes a best-effort service model. In this model, the network allocates bandwidth to all active users as best as it can, but does not make any explicit commitment as to bandwidth, delay, or actual delivery. This service model is not adequate for many real-time applications that normally require performance assurances in terms of delay, throughput, and loss ratio. A number of enhancements have been proposed to enable offering different levels of QoS in IP networks. This work has culminated in the proposal of the Integrated Services (Intserv) [15] and the Differentiated Services (Diffserv) [16] architectures by the IETF. Intserv achieves QoS guarantees through end-to-end resource (bandwidth) reservation for packet flows and performing per-flow scheduling in all intermediate routers and switches. Diffserv, on the other hand, defines a number of per-hop behaviors that enable providing relative QoS advantages for different classes of traffic aggregates. Both schemes require sources to shape their traffic as a precondition for providing end-to-end QoS guarantees.

Since Internet traffic is eventually aggregated and carried over the core networks, it is imperative to address end-to-end QoS issues in WDM optical networks. However, previous QoS methods proposed for IP networks are difficult to apply in WDM optical networks mainly due to the fact that these approaches are based on the store-and-forward model and mandate the use of buffers for contention resolution. Currently, there is no optical memory and the use of electronic memory in an optical switch necessitates O/E and E/O conversions within the switch. Using O/E and E/O converters limits the speed of the optical switch and increases the cost of the optical switch significantly. The only means currently available for providing a limited buffering capability in optical switches is the use of fiber delay lines (FDLs). FDLs are long fiber lines used to delay the optical signal for a particular period of time. However, FDLs cannot provide the full buffering capability required by the classical QoS approaches.

1.2 The Thesis

1.2.1 Objectives

This thesis has two main objectives. The first objective is to develop new methods for QoS provisioning in OBS networks. The unique features of OBS networks, such as the lack of packet queues and the physical separation between the burst header and the burst payload, stand against applying previously developed QoS methods in OBS networks. The second objective of this thesis is to develop new algorithms for reducing the burst blocking probability in OBS networks. The absence of optical buffers in OBS nodes drives the blocking probability to become the main performance measure in OBS networks.

1.2.2 Contributions

The major contributions of this work can be highlighted as follows:

- The development of a new wavelength scheduling algorithm called *Min-AV*. The developed algorithm performs better, in terms of blocking probability, than all previously proposed algorithms. The results appear in [33].

- The proposal of a new scheme for QoS provisioning in buffer-less OBS networks called Preemptive Prioritized Just Enough Time (PPJET). PPJET provides better service for high priority traffic by dropping reservations belonging to lower priority traffic using a new channel scheduling algorithm called Preemptive Latest Available Unused Channel with Void Filling (PLAUC-VF). The results have been published in [29],[30].

- The development of a new analytical model for calculating the blocking probability in Just-Enough-Time (JET) based OBS networks. The developed analytical model takes into consideration the effects of the burst offset time and the burst length on the blocking probability. This work also appears in [37],[38],[39].
- The proposal of a new detailed architecture for providing quantitative QoS guarantees with respect to end-to-end delay, throughput, and packet loss probability in buffer-less labeled optical burst switching (LOBS) networks. Results from this work appear in [31],[32].
- The proposal of a novel class of wavelength scheduling algorithms in OBS networks. The proposed wavelength scheduling algorithms process a batch of control bursts together instead of processing them one by one. This work has been published in [34],[35],[36].
- The establishment of the relationship between OBS systems and advance reservation systems. In addition, the scheduling problem in JET-based OBS networks was formulated as instance of the well-known interval graph-coloring problem [3, 47, 52, 78].

Some other minor contributions are also summarized:

- A survey of all previously proposed wavelength scheduling algorithms in JET-based OBS networks. This survey also appears in [33],[40].

- The proposal of a novel approach for applying fair scheduling algorithms in both the data plane of LOBS edge nodes and the control plane of core nodes without the need for optical buffering.

- The description of an offline algorithm for finding the optimal schedule of bursts after receiving all the reservation requests. This optimal schedule can serve as a reference against which the performance of other algorithms can be compared.

1.2.3 Road Map

Chapter 2 gives an overview of OBS networks. It describes the architecture of OBS edge nodes, OBS signaling protocols, and the architecture of OBS core nodes.

Chapter 3 surveys all previously proposed wavelength scheduling algorithms in JET-based OBS networks. Additionally, it presents a new wavelength scheduling algorithm in JET-based OBS networks.

Chapter 4 proposes a strict priority scheme for QoS provisioning in JET-based OBS networks. This scheme is called Preemptive Prioritized JET (PPJET).

Chapter 5 presents a new analytical model for evaluating the blocking probability in JET-based OBS networks.

Chapter 6 describes a detailed architecture for providing quantitative QoS guarantees with respect to end-to-end delay, throughput, and loss probability in buffer-less LOBS networks.

Chapter 7 introduces a novel class of wavelength scheduling algorithms in JET-based

OBS networks. The proposed wavelength scheduling algorithms process a batch of bursts together instead of processing them one by one.

Chapter 8 concludes the thesis and outlines future research problems.

Chapter 2

Overview of Optical Burst Switching Networks

OBS networks consist of OBS edge and core nodes connected by WDM links. Figure 2.1 shows a block diagram of an OBS network. Packets arrive to the OBS edge nodes from IP routers using conventional interfaces (e.g. gigabit Ethernet, IP/ATM, etc.). Edge nodes assembles the packets into bursts, which are then routed through the OBS network. When bursts arrive at network egress they are disassembled into packets, which are to be forwarded to their next hops.

The basic switching entity in OBS is a burst. A burst is a train of packets moving together from one ingress node to one egress node and switched together at intermediate nodes. An optical burst consists of two parts, a header and a payload. The header is called the *control burst (CB)* and is transmitted separately from the payload, which is called the *data burst (DB)*. The CB is transmitted first on a separate signaling channel to reserve the bandwidth along the path for the corresponding DB. After a given delay, the control burst is followed by the data burst, which travels over the same path reserved by the control burst. The delay between sending the CB and sending the DB is called

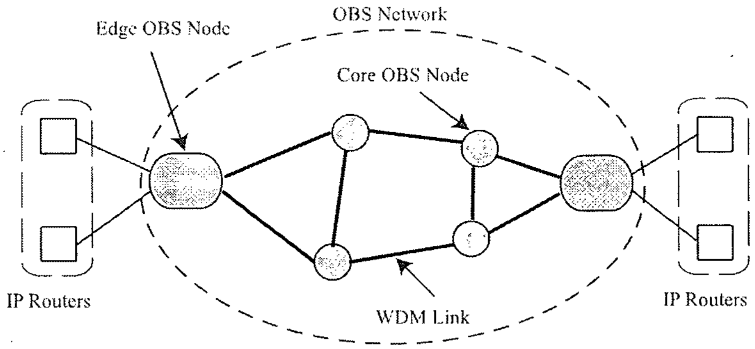


Figure 2.1: An OBS network.

the *burst offset time.* The value of the offset time is chosen to be greater than or equal to the total processing delay encountered by the CB. Let d be the queuing and processing time for a CB at a single OBS node. Assuming the CB traverses H hops, the offset time (T_o) is selected such that $T_o > Hd$.

2.1 Signaling Protocols

Signaling (reservation) protocols in OBS follow one of two schemes, namely, tell-and-go (TAG) scheme [25], and reserve-a-fixed-duration (RFD) scheme [60]. In TAG scheme, the control burst reserves the wavelength channel upon its arrival to the OBS node. On the other hand, in RFD scheme the control burst reserves the wavelength channel for a particular time interval.

Several signaling protocols have been proposed for OBS [59, 66, 72]. Among these, two protocols are widely studied in the literature: Just-In-Time (JIT) [72], and Just-

Enough-Time (JET) protocol [79]. Other protocols can be considered a variation of one of these two.

The JIT protocol follows the TAG scheme and works as follows. Upon the arrival of a CB to a core OBS node, a wavelength channel is immediately reserved if available. If no wavelength is available, the request is blocked and the corresponding DB is dropped. When the wavelength is successfully reserved, it remains reserved until the data burst transmission finishes. The only information that needs to be kept in the network nodes is whether the wavelength is currently reserved or not.

The JET protocol, on the other hand, is based on the RFD scheme. It reserves a wavelength channel exactly for the transmission time of the burst. In JET, the CB contains the destination address, the data burst length, the wavelength on which the associated data burst will arrive, and the burst offset time. When a CB arrives at a core node, a wavelength channel-scheduling algorithm is invoked to find a suitable wavelength channel on the outgoing link for the corresponding DB. The wavelength channel is reserved for a duration equal to the burst length starting from the arrival time of the DB. Therefore, nodes employing the JET protocol need to keep track of the availability of free time intervals on every wavelength channel. In this thesis we are mainly concerned with JET-based OBS networks (OBS-JET).

A typical data burst format is shown in Figure 2.2. The preamble and postamble guard bands (Pre-Guard and Post-Guard) are necessary to overcome the uncertainty of data burst arrival and data burst duration. The sync field contains synchronization

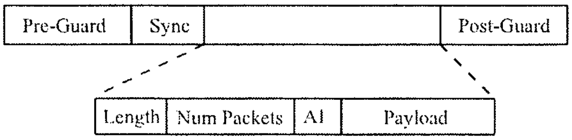


Figure 2.2: Typical data burst format.

pattern that is used to synchronize optical receivers at egress edge nodes. The length field indicates the length of the payload. The number of packets field (Num Packets) indicates the number of packets contained in the burst. The additional information (AI) field may be used to locate the start of packets in the payload field if padding is used.

The control burst contains necessary routing information to be used by core nodes to route the associated data burst hop by hop to its destination edge node. In addition to the routing information, the CB contains OBS specific information, which includes the DB duration (in RFD protocols), the wavelength on which the associated DB will arrive, DB offset time, and possibly some QoS parameters.

2.2 OBS Core Nodes

The general architecture of an 1×2 optical core OBS node is shown in Figure 2.3, which mainly consists of an optical switching matrix (OSM), a switch control unit (SCU), an optical-to-electrical (O/E) converter, and an electrical-to-optical (E/O) converter. Control bursts are carried on dedicated control channels. Generally the term *channel* is used to represent a certain unidirectional transmission capacity (in bits per second)

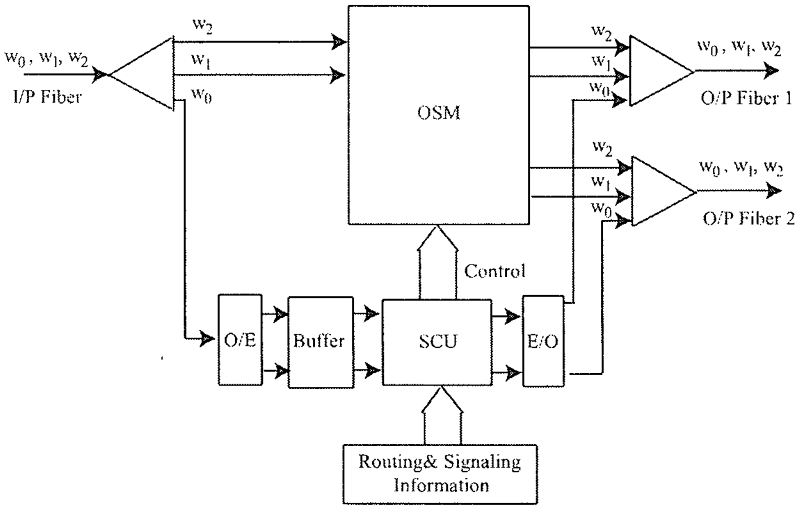


Figure 2.3: Block diagram of an OBS core node.

between two adjacent nodes [74]. A channel may consist of one wavelength, or a portion of a wavelength in case of time-division multiplexing or code division multiplexing. In this thesis a channel is equivalent to one wavelength.

When a CB arrives to a core node, it first gets converted to electrical form, then it gets time-stamped, and thereafter it gets stored in a conventional electronic buffer. For each control burst, the SCU performs a forwarding table lookup, then it decides on which outgoing link (port), data channel, and control channel to forward the CB and the corresponding DB. If there are free data channels available at the DB arrival time to the core node, the SCU configures the OSM to let the DB pass through. Otherwise, the DB is dropped. In addition, the SCU updates the burst offset time field in CB to reflect the processing delay incurred by the CB in the node. Generally, the SCU configures the OSM to allow the DB to pass only for a limited window of time, equal to the DB duration, starting from the arrival time of the DB. If the DB arrives to the OBS node outside this window of time (either earlier or later), the DB is simply dropped (blocked) since data bursts are optical analog signals. If no path is set up when a DB enters the optical switching matrix, the DB is lost.

The SCU invokes a scheduler to schedule the data burst on an outgoing data channel. The SCU keeps multiple instances of the scheduler, one scheduler instance for each output data link. Each scheduler instance keeps track of the busy/idle periods on a single outgoing data link. The scheduler works as follows. It first reads the time-stamp and the data burst duration information for the CB to determine when the corresponding DB will arrive to the OBS node, and for how long it will need the wavelength channel. Thereafter, it sends the information for the SCU. The burst offset time field is decremented at every core node by the amount of queueing, processing, and transmission delay in the node.

In this thesis we assume that tunable wavelength converters exist in OBS core nodes. A wavelength converter is a device that converts data from one incoming wavelength to another outgoing wavelength [61]. It is expected that wavelength converters will contribute a significant part to the total cost of OBS core nodes.

2.3 OBS Edge Nodes

A simplified block diagram of OBS edge nodes is shown in Figure 2.4 (ingress part) and Figure 2.5 (egress part). The main function of the ingress part is to assemble packets into bursts and send them into the network according to the OBS signaling protocol.

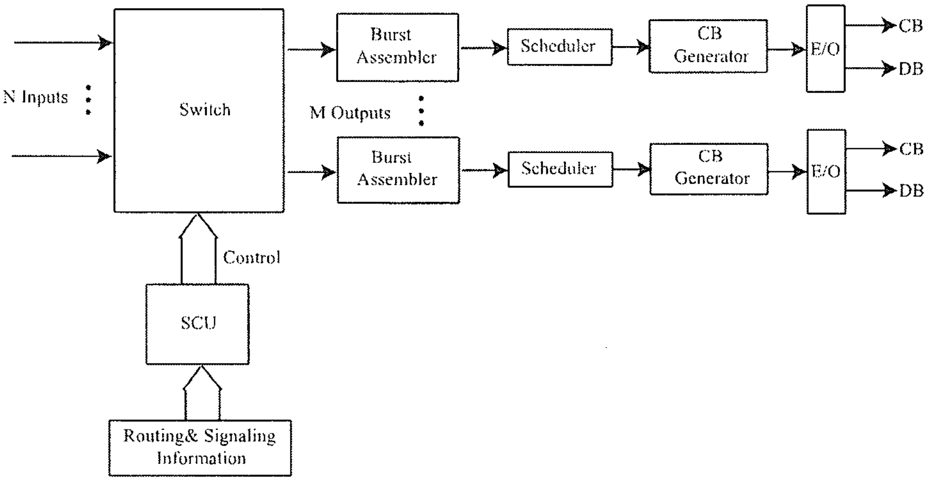


Figure 2.4: Block diagram of an OBS edge node (ingress).

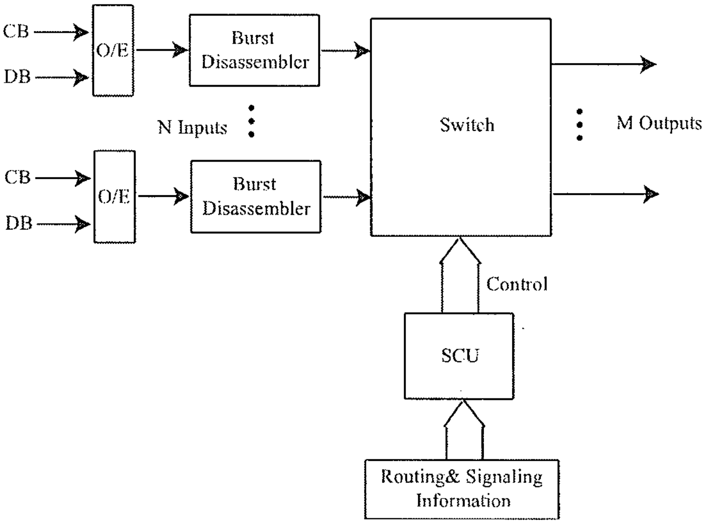


Figure 2.5: Block diagram of an OBS edge node (egress).

Packets are assembled into bursts, according to their egress edge node addresses and QoS requirements, using a burst assembly algorithm. Several burst assembly algorithms were proposed in the literature [4, 17, 41, 69, 74]. In this thesis the burst assembly process is carried out (where applicable) using the containerization with aggregation-timeout (CAT) algorithm [41][50]. In principal, the function of CAT algorithm is to assemble the arriving packets into bursts that are sent by the ingress OBS to the network's core. Two parameters control this burst assembly process in CAT, namely the maximum burst size (B_{max}) and burst assembly timeout (T_{max}). The maximum burst size controls the maximum number of bytes that can be in the same burst. If the incoming traffic intensity is high enough, the maximum burst size is reached in a reasonable time. However, if the traffic has a low arrival rate, the burst under assembly might have to wait for a relatively long time until the maximum burst size is reached. In order to avoid large burst assembling delays, an assembly timeout parameter is used.

The egress part of the OBS edge node (Figure 2.5) receives both the control burst and the data burst. Both are forwarded to the burst disassembler, where the data burst gets disassembled back into packets. These packets are then forwarded to the their next hop.

Chapter 3

Wavelength Scheduling Algorithms in OBS Networks

3.1 Introduction

As explained in Chapter 2, when a CB arrives at an OBS node, a wavelength scheduling algorithm is invoked to find a suitable wavelength channel on the outgoing link for the corresponding DB. The wavelength channel is reserved for a duration equal to the burst length starting from the arrival time of the DB. The information required by the scheduler such as the burst arrival time and its duration are obtained from the control burst. The scheduler keeps track of the availability of free time intervals (voids) on every wavelength channel.

The absence of the concept of "packet queues" in OBS nodes, coupled with the one way nature of OBS signaling protocols, drives the burst blocking probability to become the main performance measure in OBS networks. The burst blocking probability is the probability that a wavelength reservation request will not be granted due to the unavailability of a free wavelength.

Different terms have been used in the literature to express the probability of burst loss such as *blocking probability*, *dropping probability*, and *loss probability*. In this thesis, we adopt the following terminology. The term *blocking probability* is used to express burst loss that is due to unavailability of a free wavelength. The term *dropping probability* is used whenever intentional dropping of bursts is involved. The term *loss probability* is a general term that is used when there are more than one possible reason for burst loss.

The blocking probability at an OBS node depends, to a certain degree, on how efficient can the wavelength scheduling algorithm handle voids on wavelength channels. This has led to a growing interest in the area of wavelength scheduling in OBS networks. A number of wavelength scheduling algorithms have been proposed in the literature [27, 51, 66, 74, 75].

In this chapter we survey all previously proposed wavelength scheduling algorithms. In addition, we present two new scheduling algorithms and compare the performance of the two new algorithms to existing ones.

The rest of this chapter is organized as follows. Section 3.2 presents a survey of previously existing algorithms. Two new wavelength scheduling algorithms are proposed in Section 3.3. Section 3.4 presents simulation results of the proposed algorithms and their comparisons with the existing ones. Concluding remarks are provided in Section 3.5.

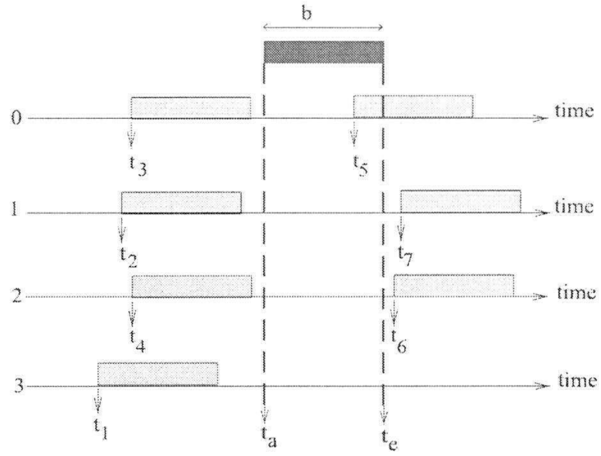


Figure 3.1: Wavelength scheduling in OBS nodes.

3.2 Previous Work

In OBS networks employing the JET protocol (OBS-JET networks), a control burst arriving at a node represents a wavelength reservation request. A reservation request consists of a pair of values (t_a, t_e). The value t_a is determined based on the offset time, whereas t_e is determined based on t_a and the length of the burst b as shown in Figure 3.1. A wavelength scheduling algorithm X is more efficient than wavelength scheduling algorithm Y, if it increases the utilization of the wavelength channels and hence decreases the burst blocking probability. The burst blocking probability is calculated as the ratio of bits blocked to bits sent.

A wavelength channel is said to be *unscheduled* at time t when no burst is using the channel at or after that time. A channel is said to be *unused* for the duration of voids between successive bursts and after the last burst assigned to the channel.

The most basic wavelength scheduling algorithm is the First Fit Unscheduled Channel (FFUC) algorithm [51]. For each of the outgoing wavelength channels, the FFUC algorithm keeps track of the unscheduled time. Whenever a control burst arrives, the FFUC algorithm searches all wavelength channels in a fixed order and assigns the burst to the first channel that has unscheduled time less than the data burst arrival time. The main advantage of the FFUC algorithm is its computational simplicity. Its main drawback is that it results in high blocking probability, since the algorithm does not consider voids between bursts scheduled.

In [66], Turner proposed the Latest Available Unscheduled Channel (LAUC) algorithm. The basic idea of the LAUC algorithm is to increase channel utilization by minimizing voids created between bursts. This is accomplished by selecting the latest available unscheduled data channel for each arriving data burst. For example, in Figure 3.2 wavelengths 0 and 3 are unscheduled at time t_a, and wavelength 0 will be selected to carry the new data burst arriving at t_a; thus, the void on wavelength 0 will be smaller than the void that would have been created if wavelength 3 were selected. Therefore, LAUC yields better burst dropping performance than FFUC and does not add any computation overhead. However, since it does not take advantage of voids between bursts, as was the case for the FFUC, it still leads to a relatively high blocking probability.

Xiong *et* al. [74] proposed a channel algorithm called *LAUC-VF* (Latest Available Unused Channel with Void Filling). The basic idea of the LAUC-VF algorithm is to minimize voids by selecting the latest available unused data channel for each arriving

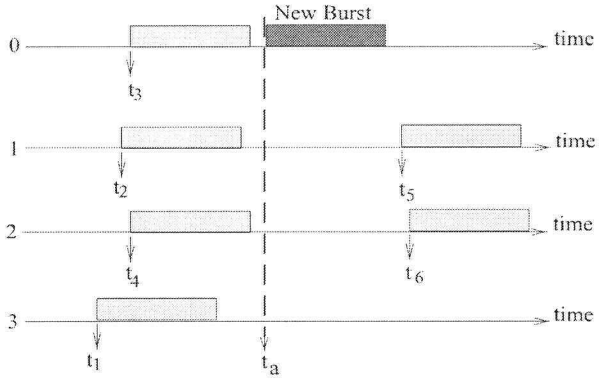


Figure 3.2: Illustration of the LAUC algorithm.

data burst. Given the arrival time t_a of a data burst with duration b to the optical switch, the scheduler first finds the outgoing data channels that are available for the time period of $(t_a; t_a + b)$. If there is at least one such data channel, the scheduler selects the latest available data channel, i.e., the channel having the smallest gap between t_a and the end of the last data burst just before t_a. Figure 3.3 illustrates that LAUC-VF chooses channel 2 in the previous example.

Iizuka *et* al. [27] proposed a scheduling algorithm that selects the data channel in which the void newly generated after the burst ending time becomes minimum (we call it *Min-NGV* algorithm). Figure 3.4 illustrates an example for the algorithm. The Min-NGV algorithm chooses wavelength channel 1 since this choice will generate the smallest void after the burst ending time.

Xu *et* al. [75] has proposed two algorithms: Minimum Starting Void (*Min-SV*) and Minimum Ending Void (*Min-EV*). Both algorithms are based on computational geometry

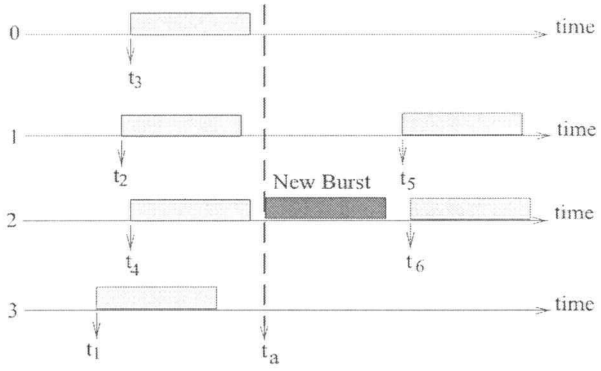


Figure 3.3: Illustration of the LAUC-VF algorithm.

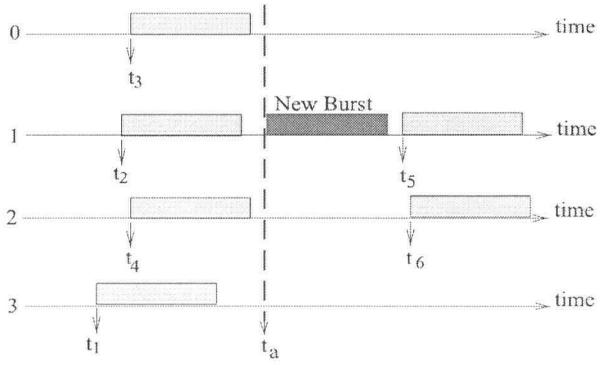


Figure 3.4: Illustration of the Min-NGV algorithm.

techniques. The Min-SV algorithm performance is close to the performance of the LAUC-VF, while the Min-EV algorithm is the same as the Min-NGV. However, Min-EV and Min-SV are claimed to be faster than Min-NGV and LAUC-VF, respectively.

3.3 Proposed Algorithms

In this section, two wavelength scheduling algorithms are proposed. The first algorithm is called Maximum Newly Generated Void (Max-NGV), and the second one is called Minimum Available Void (Min-AV).

3.3.1 Max-NGV

The idea behind the Max-NGV algorithm is to minimize the burst blocking probability by selecting the data channel that results in the largest void after the burst ending time. This way larger voids will be available for future data bursts. Given the arrival time t_a of a data burst with duration b to the optical switch, the scheduler first finds the outgoing data channels that are available for the time period of $(t_a;\ t_a + b)$. If there is at least one such data channel, the scheduler selects the channel having the largest gap between $(t_a + b)$ and the start of the data burst just after $(t_a + b)$. In other words, the algorithm assigns the burst to the channel that results in the largest newly generated void after the burst ending time. An example for the Max-NGV algorithm is illustrated in Figure 3.5.

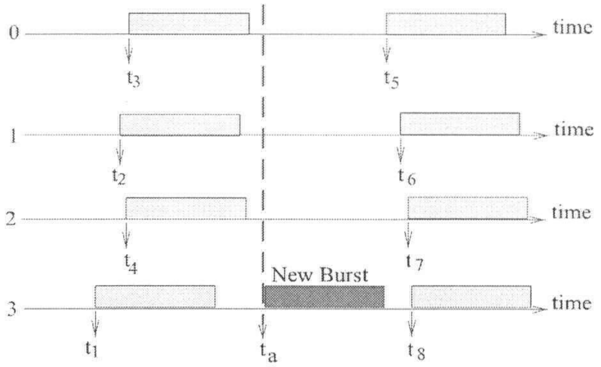


Figure 3.5: Illustration of the Max-NGV algorithm.

3.3.2 Min-AV

The basic idea of the Min-AV algorithm is to minimize voids by selecting the smallest void among all possible voids. Given the arrival time t_a of a data burst with duration b to the optical switch, the scheduler first finds the outgoing data channels that are available for the time period of $(t_a;\ t_a + b)$. If there is at least one such data channel, the scheduler selects the data channel that has the smallest available void. Accordingly, the scheduler will always minimize voids resulting from assigning the burst to the data channel. An example for the Min-AV algorithm is shown in Figure 3.6. The algorithm chooses channel 0 since it has the minimum void among all possible voids.

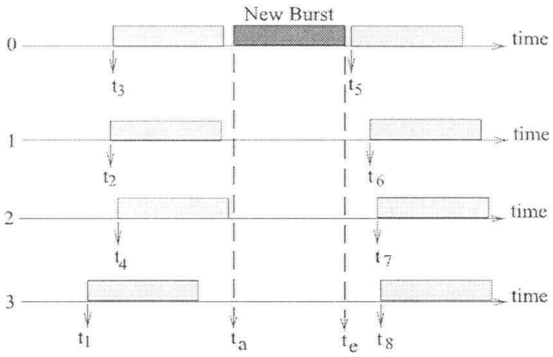


Figure 3.6: Illustration of the Min-AV algorithm.

3.4 Performance Evaluation

This section presents experimental results on the proposed algorithms, Max-NGV and Min-AV, and their comparisons with the existing algorithms, LAUC-VF, Min-NGV. Our main concern in these experiments is the blocking probability. We ignore the LAUC algorithm in our comparison since it is already known to be outperformed by the LAUC-VF algorithm in terms of blocking probability. Additionally, as indicated before, the performance of the LAUC-VF is an upper bound on the performance of the Min-SV, while Min-EV performance is similar to the performance of the Min-NGV.

3.4.1 Simulation Setup

Figure 3.7 shows the simulation model used in this section, which is based on the OPNET [54] simulation tool. It consists of a single OBS node connected to n traffic sources and a sink node. Each input link carries 2 separate wavelengths, one for data and one for control. The output link carries five wavelengths, four for data and one for control, and each wavelength has transmission speed of approximately 2.5 Gbps (OC-48). Practical WDM systems typically use larger numbers of wavelengths (e.g., 32–128 wavelengths). However, obtaining reliable results from a discrete-event simulation of such systems would require running the simulation for a prohibitively long time. Therefore, for the purpose of our research, we have used a relatively small number of wavelengths in order to model and analyze our architecture in a reasonable amount of time. Studying the simulation results of our model has revealed that the only impact of using a relatively small number of wavelengths is that the blocking probability across various classes of service has uniformly increased.

We assume that sources generate control and data bursts. Burst sizes are exponentially distributed with mean B units. The offset time has a uniform distribution over $[0, A_{max}]$. The mean burst size B and the maximum offset time A_{max} are varied to study their effect. All traffic sources used are Poisson sources.

The use of Poisson process to characterize the arrival process in OBS networks has been heavily discussed in the literature [17, 24, 28, 74, 82]. It is generally accepted that Internet traffic is statistically self-similar [44, 57, 73]. However, it has been shown that in OBS networks the burst assembly process reduces the long-range and the short-range dependency of the traffic [17, 24, 74, 82]. In addition, in buffer-less OBS networks the influence of the long-range dependency of the self-similar traffic can be neglected and the arrival process can be assumed to be Poisson process, as illustrated in [28, 82].

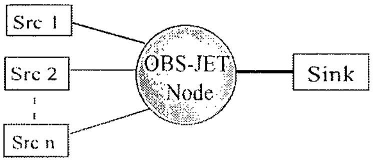


Figure 3.7: Simulation model used in Section 3.4.

The main difference between scheduling algorithms compared in this section is the wavelength selection criteria. Therefore, in the OBS node we have implemented a general wavelength scheduling algorithm that changes the void selection process depending on the particular algorithm requested. A description of the algorithm is given in Algorithm 3.1.

The function Get_Void(i, t_a, t_e) searches wavelength i for a void that can fit a burst arriving at time t_a and ending at time t_e. In general, a void can be represented by an interval with start time t_s^v and end time t_e^v. Clearly this interval can be semi-unbounded, i.e., $t_e^v = \infty$, if the channel is unscheduled at time t_s^v. If there is at least one such data channel, the function Select_Void selects a wavelength from those having available voids using the specified algorithm ($algol$).

3.4.2 Simulation Results

We study the effect of varying the maximum burst offset time and the mean burst size on the performance of the proposed algorithms, Max-NGV and Min-AV, and the existing algorithms, LAUC-VF, Min-NGV. Let Δ be the transmission time of 1024 bits on one

```
input  : $t_a$, $t_e$, $algol$
output: $wl$
initialize $i \leftarrow 0, wl \leftarrow -1$

for $i \leftarrow 0$ to $Data\_WL\_Num$ do

    $v[i] \leftarrow$ Get_Void($i$, $t_a, t_e$);

end

if $v$ is empty then

    report failure in finding an outgoing channel;

    stop;
end

$wl \leftarrow$ Select_Void($v$,$algol$);

reserve channel $wl$ for data burst;

report $wl$;
```

Algorithm 3.1: General wavelength scheduling algorithm

of the wavelengths, i.e., $\Delta = 1024/2377728000 = 4.3e^{-7}$ seconds. We express the offset time in terms of Δ. We also express B as multiples of 1024 bits. Each data point in the following results is the average of 10 runs with 10 different seeds.

Effect of burst size

In this subsection, we set the maximum offset time A_{max}= 100Δ. We perform two sets of simulations. The first set corresponds to mean burst size B= 160 (i.e., B=163840 bits), while the second set corresponds to B= 20. The burst inter-arrival time was also varied in order to vary the offered load. Figure 3.8 illustrate the results.

Result set marked as (1) in Figure 3.8 corresponds to the case of B= 160. In this set of results, we see that all scheduling algorithms give approximately the same blocking probability. In the second set of results, which corresponds to B=20, the proposed Min-AV algorithm has the best performance among all algorithms, while Max-NGV and Min-NGV algorithms give the worst performance. Generally, smaller mean burst size causes more fragmentation for the wavelength channels than in the case of larger mean burst size. Higher fragmentation levels of the wavelength channel causes the blocking probability to increase. It is the job of the wavelength scheduler to decrease the level of fragmentation where possible. Max-NGV and Min-NGV are less successful in decreasing the level of fragmentation on channels when the mean burst size is small.

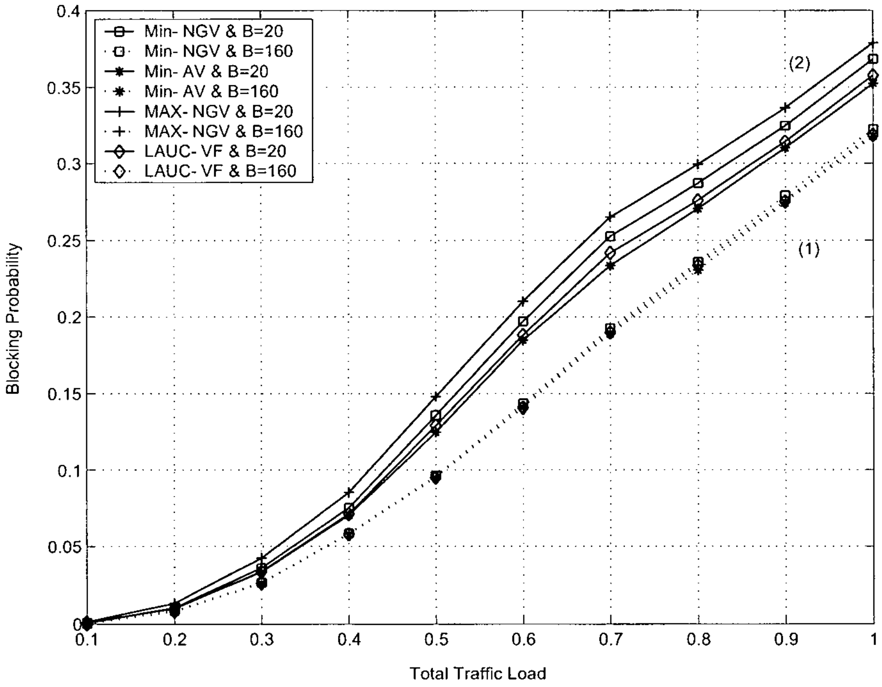


Figure 3.8: Effect of the burst size on the performance of different wavelength scheduling algorithms in terms of the burst blocking probability.

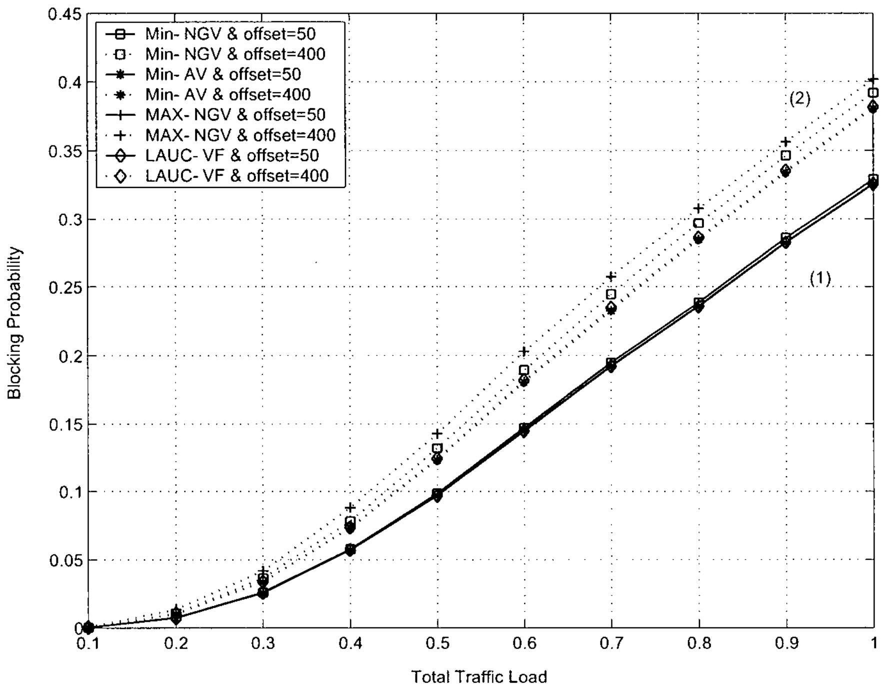


Figure 3.9: Effect of the maximum offset time on the performance of different wavelength scheduling algorithms in terms of the burst blocking probability.

Effect of offset time

In this subsection, we assign to B an arbitrary value of 80 and vary the inter-arrival time to vary the offered load. We perform two sets of simulations. The first set corresponds to a maximum offset value equal to 50Δ, whereas the second set corresponds to $A_{max}= 400\Delta$. Results are shown in Figure 3.9.

Results from the first simulations set show that for small values of A_{max} all algorithms typically result in approximately equal blocking probability. The second set of simulations shows that as A_{max} becomes larger, different algorithms result in different blocking probability values. In addition, results show that Max-NGV and Min-NGV produce higher blocking probabilities than LAUC-VF, Min-AV for larger values of A_{max}. When the maximum offset time value increases, reservations' start time values are more dispersed in time. This causes the fragmentation of the wavelength channel to increase as the maximum offset time increases. Results in Figure 3.9 show that the Min-AV and LAUC-VF algorithms pack reservations on wavelength channels better than Max-NGV and Min-NGV, which in turn decreases the fragmentation of the wavelength channels..

3.5 Concluding Remarks

In this chapter, we surveyed previously proposed wavelength scheduling algorithms. Moreover, we have introduced two new wavelength reservation algorithms: Max-NGV and Min-AV. The performance of the proposed algorithms and their comparison with previously proposed algorithms was evaluated through discrete-event simulation. Simulation results showed that the Min-AV algorithm performs at least as good as the best previously known algorithm (LAUC-VF) in terms of blocking probability. Results also showed that Min-AV is specially advantageous when mean burst size is small.

Chapter 4

A Novel Priority Scheme for Supporting QoS in OBS Networks

4.1 Introduction

In this chapter, we present a novel scheme for QoS provisioning in buffer-less OBS networks. We call this scheme Preemptive Prioritized JET (PPJET). PPJET provides service to different traffic classes based on a strict priority order. The PPJET scheme utilizes the JET signaling protocol, in conjunction with a new channel-scheduling algorithm, called Preemptive Latest Available Unused Channel with Void Filling (PLAUC-VF).

A number of related approaches appear in the literature [2, 76, 80]. A signaling protocol, called Prioritized Just-Enough-Time (PJET), has been proposed for providing priority-based service differentiation in OBS networks [80]. The main drawback of PJET is that it introduces a significant amount of delay for high priority traffic in order to provide a reasonable isolation level from lower priority traffic classes.

Two different preemptive schemes for QoS provisioning in OBS networks have been proposed in the literature. Cankaya *et al.* [2] proposed a partial preemptive scheduling scheme. This scheme combines proportional differentiation discipline with partial preemptive scheduling such that the burst dropping probability is adjusted to be proportional to the parameters set by the network service provider. The algorithm used for burst scheduling under this scheme has, however, high complexity due to the lengthly burst acceptance/rejection decision process in intermediate switching nodes. Moreover, the proportional differentiation technique itself requires maintaining statistics about each class of service in every node in the network. This information is updated with every control burst, which severely limits the scalability of the approach considering the high volume of bursts in OBS networks.

A different scheme has been described by Yang *et al.* [76]. Their scheme allows high priority bursts to preempt low priority bursts in a probabilistic manner. However, the model given in their paper to estimate the blocking probability is incomplete because it ignores completely the effects of the offset time and its distribution on the blocking probability. In addition, their scheme cannot make use of void filling channel-scheduling algorithms, leading to much higher dropping probabilities than the PJET scheme.

The rest of this chapter is organized as follows. Section 4.2 gives an overview of PJET. In Section 4.3, we present the PPJET scheme and the PLAUC-VF algorithm. We discuss an approximate method for calculating the dropping probability in PPJET in Section 4.4. We evaluate the performance of the proposed scheme in Section 4.5 and provide concluding remarks in Section 4.6.

4.2 Overview of PJET

The PJET protocol uses offset time as a way to provide different classes of service in buffer-less OBS networks. Assume we have two classes of service: class 1 and class 0. Class 1 is the real-time service class that corresponds to applications that require low delay, bandwidth guarantee, and low dropping probability, while class 0 is the best effort service class. In order for class 1 to have higher priority for bandwidth reservation, an additional offset time, denoted t_{qos}, is given to this class. The value of t_{qos} is constant and considerably larger than the original JET offset time (T_o). Additionally, t_{qos} needs to be larger than the maximum burst length in class 0. With such long offset time, the dropping probability of bursts in class 1 becomes independent of the offered load in class 0 and only a function of the offered load in class 1. On the other hand, the offered load in both classes will determine the dropping probability of class 0. The authors in [80] gave a simple analytical model to evaluate the dropping probability as a function of t_{qos} and concluded that to provide almost 100% isolation between classes 0 and 1, it is sufficient to have t_{qos} equal to 5 L_0 , where L_0 is the average burst size in class 0. The main problem with this protocol is that it introduces a significant amount of delay for high priority traffic.

4.3 Introducing PPJET

In this section we describe PPJET, a novel priority based scheme for QoS provisioning in buffer-less OBS networks. This scheme minimizes dropping of high priority traffic by preempting reservations for lower priority traffic. Consider the case of an OBS network supporting two priority classes, namely class 0 and class 1, where class 1 is given higher priority over class 0 in terms of dropping probability (i.e., class 1 has lower packet dropping probability than class 0). At the network ingress, incoming traffic is aggregated into bursts, such that each burst will belong to either class 1 or class 0. Whenever a data burst is ready to be sent, the corresponding control burst is sent to request reservations at intermediate nodes for the data burst. When class 1 reservation request arrives at an intermediate node it cannot be blocked (dropped) by a future class 0 request nor can it be blocked by an existing class 0 reservation for any duration in the future. It can only be blocked by an existing class 1 reservation, or a class 0 reservation that has a start time in the past (that is to say, the reserved duration has started already). The main advantage of PPJET is that it eliminates the need for the relatively large QoS offset period required by the PJET scheme for providing strict priority. In PPJET, reservation requests are signaled using JET with basic offset T_o. The only modification required for the JET protocol is to add to the CB a class-of-service (CoS) field indicating the class of service that the burst belongs to.

To achieve the goal of providing a strict priority for high priority reservations, we propose a new channel scheduling algorithm, called Preemptive-LAUC-VF (PLAUC-

VF), to be used at ingress and intermediate OBS nodes to allocate wavelength channels for the data burst duration.

4.3.1 PLAUC-VF Channel Scheduling Algorithm

In the PLAUC-VF algorithm, when a request belonging to a high priority traffic class arrives, the algorithm first searches the wavelength channels for voids that can accommodate the new reservation. If more than one void exist, the latest available unused data channel is selected. If no void can be found, the algorithm finds, for each channel, the original reservations that blocked the new reservation request. If these reservations belong to lower priority classes, the algorithm calculates the void that will exist if these original reservations are dropped. Then, the algorithm selects the channel that has the void with the latest start-time among the newly created voids. Thereafter, the algorithm drops the original blocking reservations on the selected channel. For example, in Figure 4.1(a) there is no void that is large enough to fit the new burst that is to arrive at time t_a. The PLAUC-VF algorithm will find bursts that might be dropped to create a new void that is large enough to fit the new arriving burst. Channels 0 and 1 are not selected by the algorithm because the bursts that need to be dropped belong to the same CoS of the new burst. Both channels 2, 3 are eligible because suitable voids can be created by dropping lower CoS bursts. However, the algorithm selects channel 3, as shown in Figure 4.1(b), since its new void has the latest start time. A formal description of the algorithm is given in Algorithm 4.1. The PLAUC-VF algorithm has similarities with the algorithm described in Algorithm 3.1. The function Get_Void is modified here to include a CoS parameter and also to return the reservations to be dropped, if any. After calculating all suitable voids, the algorithm partitions the set of voids into two subsets: the set of original voids (*ov*) and the set of new voids (*nv*). The former set includes voids that already exist on wavelength channels, while the latter set includes voids that will be created by dropping reservations belonging to lower CoS(s). The set of bursts that need to be dropped in order to accommodate the new burst on wavelength i is kept in $d[i]$. Only if *ov* set is empty, the *nv* set is considered by the algorithm. The function Find_LST_Void is used to find the void with the latest start time among *ov* if *ov* is not empty, otherwise it is used on *nv*. The algorithm terminates with the reserved wavelength if a suitable void can be found or created, otherwise an error is reported.

4.4 Dropping Probability Calculation in PPJET

One possible method for approximating the dropping probability in PPJET scheme is to model the output port as a $M/G/k/k$ queue, and calculate the dropping probability using the well-known Erlang's B-formula:

$$Bp(\rho, k) = \frac{\rho^k/k!}{\sum\limits_{i=0}^{k} \rho^i/i!} \tag{4.1}$$

In this equation, k is the number of wavelengths, and ρ is the total offered load. Consider 2 classes of service, class 0 and class 1. Class 1 corresponds to the high priority

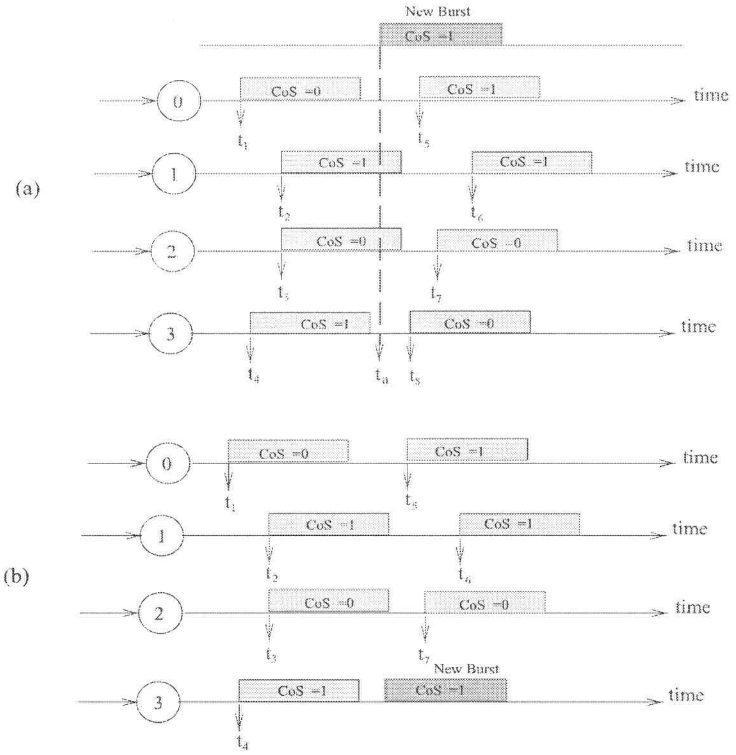


Figure 4.1: Illustration of the PLAUC-VF algorithm.

```
input  : $t_a$, $t_e$, $cos$
output: $wl$
initialize $i \leftarrow 0, wl \leftarrow -1$

for $i \leftarrow 0$ to $Data\_WL\_Num$ do
    $[v[i], d[i]] \leftarrow$ Get_Void($t_a, t_e, cos$);
end
if $v$ is empty then
    report failure in finding an outgoing channel;
    stop;
end
$ov[\,] \leftarrow \{v[i] \mid d[i]$ is empty$\}$;
$nv[\,] \leftarrow \{v[i] \mid d[i]$ not empty$\}$;
if $ov$ not empty then
    $wl \leftarrow$ Find_LST_Void($ov$);
    reserve channel $wl$ for data burst;
    report $wl$;
    stop;
else
    $wl \leftarrow$ Find_LST_Void($nv$);
    Drop ($d[wl]$);
    reserve channel $wl$ for data burst;
    report $wl$;
    stop;
end
```

Algorithm 4.1: PLAUC-VF algorithm

traffic and class 0 corresponds to the best-effort traffic. Since PPJET is a strict priority scheme, the dropping probability of class 1 ($P_{drop,1}$) depends only on its traffic load. Thus, $P_{drop,1}$ can be directly obtained by $Bp(\rho_1, k)$. Class 1 offered load can be calculated as λb, where b is the average burst size and λ is the average arrival rate. On the other hand, the burst dropping probability of class 0 ($P_{drop,0}$) can be calculated from the following conservation law:

$$\rho P_{drop,tot} = \rho_1 P_{drop,1} + \rho_0 P_{drop,0} \tag{4.2}$$

where $\rho = \rho_1 + \rho_0$ and $P_{drop,tot} = Bp(\rho, k)$. Unfortunately, this approximation is not accurate because it neglects the effect of the offset time and the burst length distribution on the dropping probability. However, this approximation may be useful for dimensioning purposes.

4.5 Performance Analysis

In this section, we compare the performance of PPJET and PJET in terms of dropping probability and end-to-end (ETE) delay by means of simulation. We restrict our analysis to two classes of service, namely, class 0 and class 1. However, PPJET and PJET can both be generalized to more than two classes of service. In this discussion, class 1 corresponds to the high priority traffic and class 0 corresponds to the best-effort traffic.

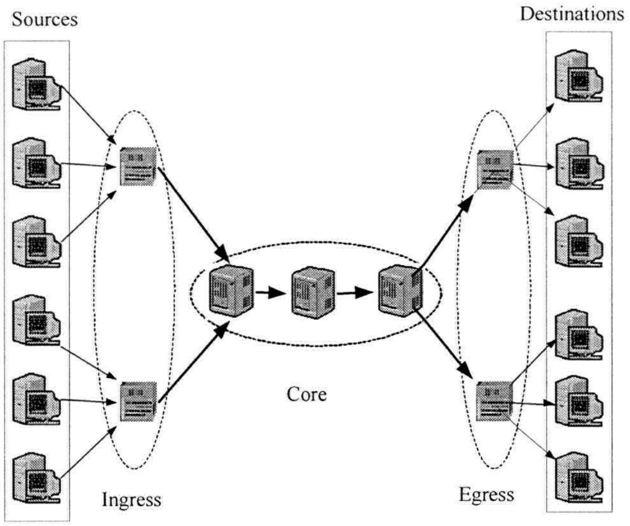


Figure 4.2: Simulation network model for Section 4.5.

4.5.1 Network Setup

The simulation is performed using a sophisticated discrete-event simulation model based on OPNET. The network topology used in the simulation is shown in Figure 4.2. The network consists of two types of OBS nodes, core OBS nodes and edge OBS nodes (ingress and egress).

Links shown as thick lines in Figure 4.2 are WDM links capable of carrying five separate wavelengths, four for data and one for control. The transmission speed per wavelength is approximately 2.5 Gbps (OC-48). Links from traffic sources to ingress OBS devices and from egress OBS devices to the destination nodes are single wavelength links with the same transmission speed. The burst assembly process is carried out using the CAT algorithm [41].The main advantage of the CAT algorithm is that it regulates the network traffic by producing bursts with sizes that are approximately constant and equal to B_{max}. Therefore, CAT enhances (i.e., reduces) burst dropping probability.

All traffic sources used in the simulation are Poisson sources. Packet sizes are exponentially distributed with a mean equal to 1,024 bytes, and the mean packet inter-arrival time is varied to demonstrate the effect of network load. Sources are configured such that class 0 and class 1 shares of the total offered load are 2/3 and 1/3, respectively.

We compare the performance of PPJET and PJET in two cases. In the first case, we use the CAT algorithm with constant B_{max}. In the second case, B_{max} is varied from one burst to another according to an exponential distribution with a mean parameter, $\bar{B}_{max}$, that is changed from one simulation scenario to another. The main purpose of varying B_{max} is to analyze the performance of PPJET in the presence of other burst assembly algorithms that may not behave as CAT.

An important parameter for evaluating the performance of PJET is the ratio of the QoS-offset (t_{qos}) to the mean burst size ($\bar{B}_{max}$). We call this ratio the *isolation ratio.* This ratio determines the degree of isolation of class 1 traffic from traffic belonging to class 0. The higher the ratio the better the isolation between the two traffic classes. Previous research has shown that in case of constant burst sizes it is sufficient to set the isolation ratio to 1 [80]. On the other hand, in case of exponentially distributed burst sizes, class 1 is sufficiently isolated when the isolation ratio is equal to 5, and that any further increase to this ratio will have minimal effect [80]. It is important to emphasize that PPJET does not require extra offset time to achieve isolation, therefore, the isolation ratio is irrelevant

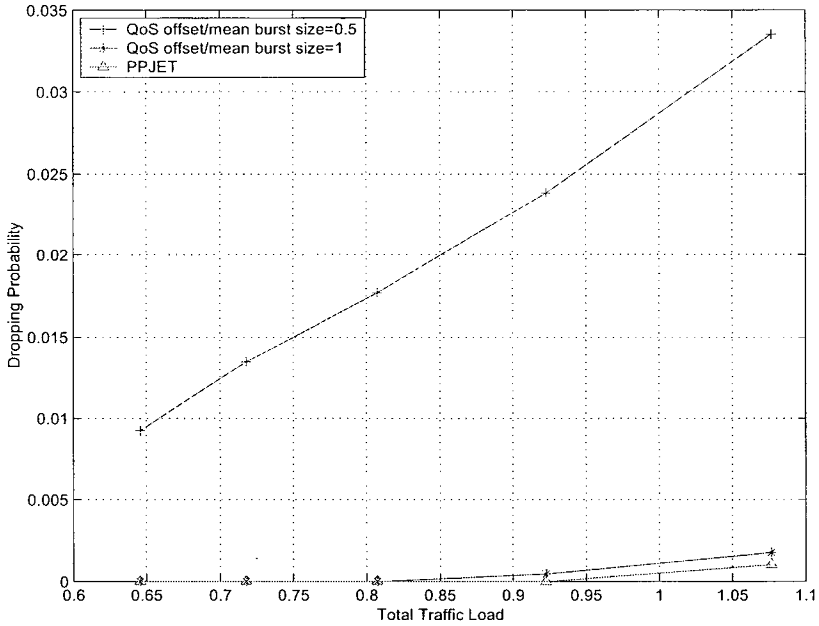


Figure 4.3: Comparison between PPJET and PJET class 1 dropping probabilities in the case of constant burst sizes.

to PPJET.

4.5.2 Burst Dropping Probability

We perform two simulation sets to compare class 1 dropping probabilities ($P_{drop,1}$) for PPJET and PJET as a function of the total offered load. In the first set of simulations, we use a constant value for B_{max} equal to 20,480 bytes. Figure 4.3 shows the results. The top two curves in Figure 4.3 show $P_{drop,1}$ for PJET when the isolation ratio is equal to 0.5 and 1. The bottom curve shows $P_{drop,1}$ for PPJET.

In the second set of simulations, we use an exponentially distributed B_{max} with mean

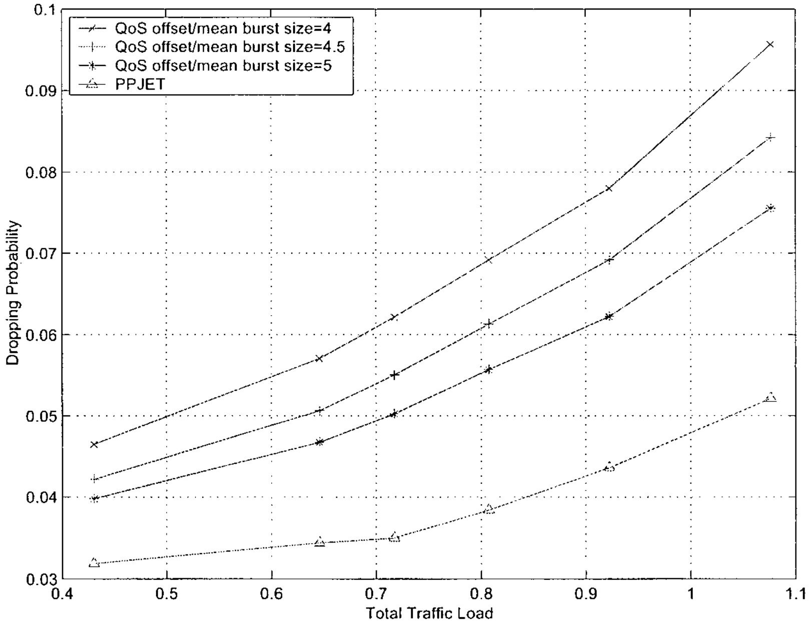


Figure 4.4: Comparison between PPJET and PJET class 1 dropping probabilities in the case of exponentially distributed burst sizes.

value = 20,480 bytes. In addition, we use three values for the PJET isolation ratio: 4, 4.5, and 5. Figure 4.4 shows the result for this set of simulations.

It is obvious from Figure 4.3 and Figure 4.4 that PPJET outperforms PJET in terms of dropping probability, especially in the case of exponentially distributed burst sizes. This can be explained by noting that the offset method used in the PJET scheme for isolation is a statistical method for providing priority. Stated otherwise: as the isolation ratio increases, the *probability* of blocking decreases. On the other hand, PPJET's method of guaranteeing priority is deterministic, since the PLAUC-VF algorithm preempts lower priority reservations.

In Figure 4.5 we illustrate the effect of $\bar{B}_{max}$ on the dropping probability of both PPJET and PJET. We fix the isolation ratio for the PJET scheme to 5. In this scenario, we use two values for $\bar{B}_{max}$: 20,480 bytes and 204,800 bytes. Results in Figure 4.5 demonstrate clearly that PPJET performs better than PJET, in terms of dropping probability, for small and large values of $\bar{B}_{max}$. In addition, it is apparent from the figure that PPJET and PJET provide better burst dropping probability as the value of $\bar{B}_{max}$ increases.

4.5.3 ETE delay

In our network model, a packet encounters four types of delays in its path from source to destination: burst assembly delay (D_a) in the ingress OBS node, offset delay (D_o), data burst transmission delay (D_t) and the queuing delay (D_q) at the output of the egress OBS node after the burst de-assembly process.

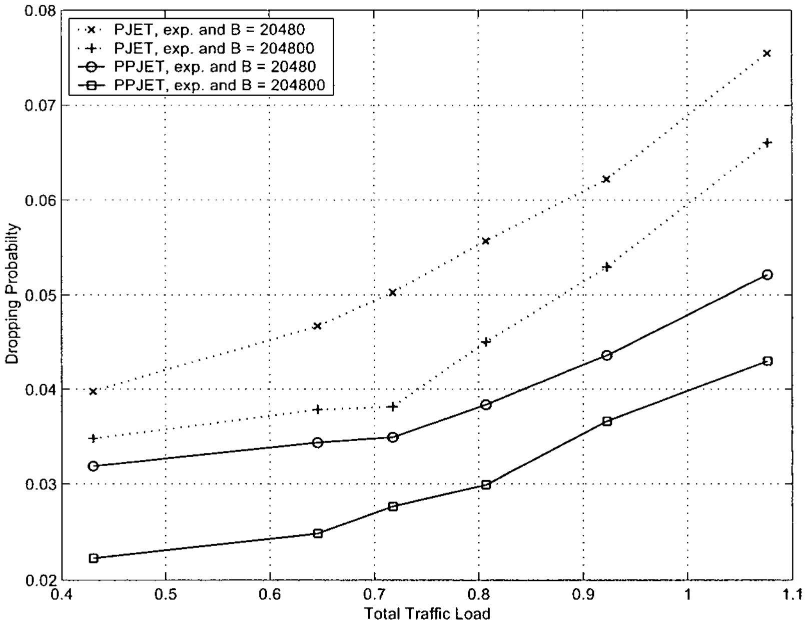


Figure 4.5: Effect of the mean burst size on class 1 dropping probability.

The offset delay D_o can be further divided into two components, JET offset time T_o and QoS offset t_{qos}. The JET offset time is constant for both PPJET and PJET, while QoS offset is equal to zero in case of PPJET.

Note that the quantity D_a is directly proportional to B_{max} and inversely proportional to the traffic load, while both D_t and D_q delay components are directly proportional to B_{max}. In case of PJET, D_o is also directly proportional to B_{max}, while D_o is constant in PPJET. Accordingly, the main difference in ETE delay between PJET and PPJET is the QoS offset. In this subsection, the isolation ratio for PJET is set to 5 for exponentially distributed B_{max}, and is set to 1 for constant B_{max}.

Figure 4.6 compares the ETE delay for PPJET and PJET as a function of the total offered load. In this scenario, B_{max} is exponentially distributed, and we use two values for $\bar{B}_{max}$: 20480 bytes and 204800 bytes. Figure 4.6 shows that ETE delay decreases in all cases as the traffic load increases. This is due to the D_a delay component that is inversely proportional to the traffic load. Furthermore, as $\bar{B}_{max}$ increases the ETE delay increases due to the fact that D_a, D_t, and D_q delay components are all proportional to $\bar{B}_{max}$. However, as explained earlier, the main advantage of PPJET over PJET with respect to the ETE delay is in the elimination of the QoS offset which can be substantially large in the case of large $\bar{B}_{max}$. This fact is evident from the simulation results in Figure 4.6, where the difference between the bottom two curves ($\bar{B}_{max}$= 20480 bytes) for PPJET and PJET is substantially smaller than the difference between the two upper curves ($\bar{B}_{max}$= 204800 bytes). Figure 4.7 shows similar results for constant B_{max}.

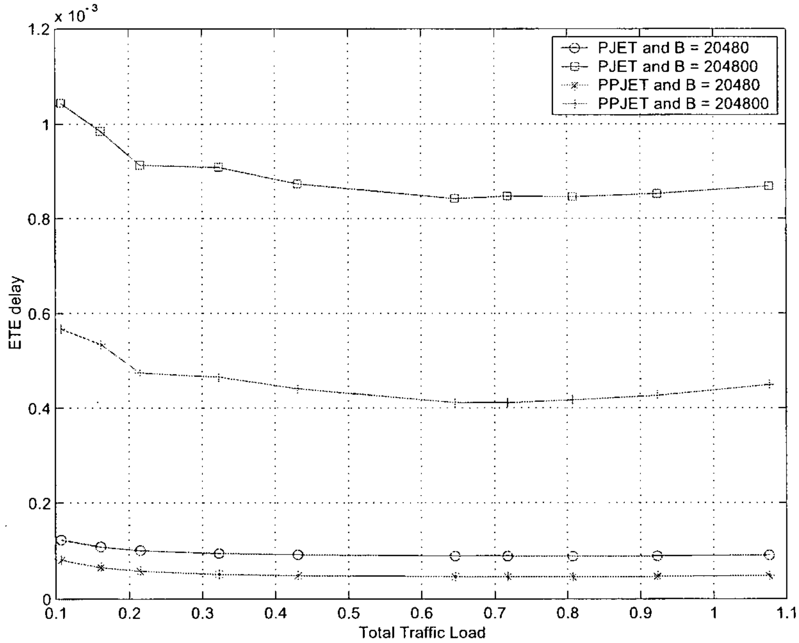


Figure 4.6: Class 1 ETE delay comparison in the case of exponential B_{max}.

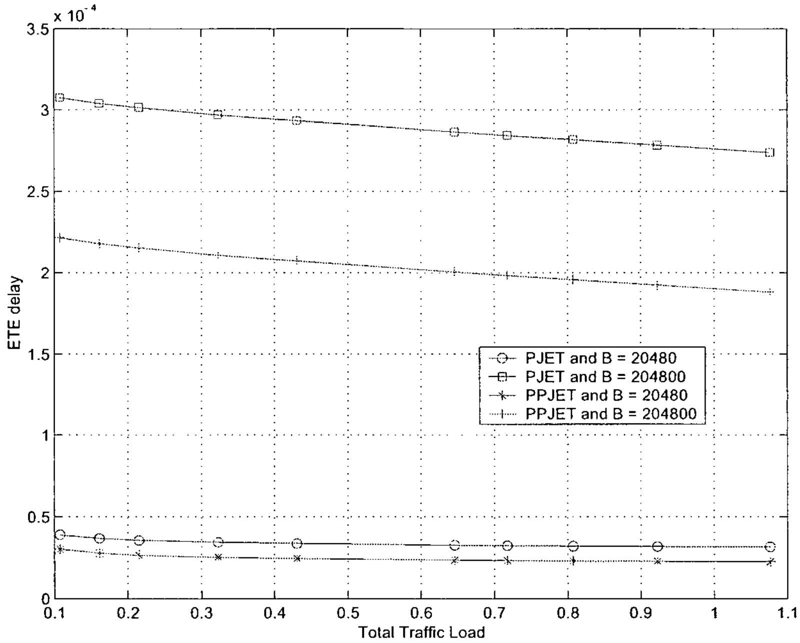


Figure 4.7: Class 1 ETE delay comparison in the case of constant B_{max}.

4.6 Concluding Remarks

In this chapter, we have presented a novel scheme for QoS provisioning in buffer-less OBS networks. This scheme assigns different priorities to traffic classes and provides service to these classes based on strict priority order. The proposed scheme achieves this goal by utilizing a new channel scheduling algorithm called PLAUC-VF that allows for dropping reservations that belong to lower priority traffic-classes in order to fit new high priority reservations. We have presented several results on the performance of the proposed scheme for the case of two priority levels. However, the scheme is general and can be used with more than two service classes. We have demonstrated that PPJET achieves significant performance enhancements over PJET in terms of dropping probability and ETE delay for high priority traffic.

Chapter 5

A New Analytical Model for Computing Blocking Probability in OBS Networks

5.1 Introduction

One of the main performance measures in OBS networks is the blocking probability, which is the probability that a wavelength reservation request will not be granted due to the unavailability of a free wavelength. To the best of our knowledge, most analytical models used for calculating the blocking probability in the JET protocol are very simple to the degree of inadequacy. These models ignore important factors that need to be considered when calculating the blocking probability, as will be explained later in this chapter.

The main objective of this chapter is to present a more elaborate analytical model that captures the essential features of JET-based OBS (OBS-JET) networks, and gives stronger insight into factors that affect the blocking probability.

In Section 5.2, we discuss key factors affecting the blocking probability in OBS-JET networks and briefly describe previous models for calculating the blocking probability. We relate the problem of calculating the blocking probability in OBS-JET networks to the problem of calculating the reservation probability in *advance reservation systems* in Section 5.3. We then propose a new analytical model in Section 5.4. We validate the accuracy of the proposed model in Section 5.5 by comparing it to simulation results and we conclude the chapter in Section 5.6.

5.2 Blocking Probability

The signaling protocol used in an OBS network is crucial in determining the blocking probability for data bursts in that network. For the JIT protocol, the blocking probability at a node can be calculated by modeling the output port as a $M/G/k/k$ queue and using the well-known Erlang's B-formula for the loss probability:

$$Bp(\rho, k) = \frac{\rho^k/k!}{\sum\limits_{i=0}^{k} \rho^i/i!} \tag{5.1}$$

In this equation, k is the number of wavelengths, and ρ is the offered load. For the JIT protocol the offered load is $\lambda(b + a)$, where λ is the mean arrival rate, a is the mean burst offset time, and b is the mean data burst length. This formula is derived using the fact that each request blocks the channel for a time duration equal to the sum of the offset time and burst transmission delay.

Erlang's loss formula has also been used to approximate the blocking probability for the JET protocol by treating the node's output link as a traditional loss system with an offered load of λb [10][71]. However, this approximation oversimplifies the problem by neglecting the effect of the offset time on the blocking probability. Moreover, if a void filling algorithm is used in conjunction with the JET protocol, the discrepancy between the approximation and the true blocking probability will become even larger. This is due to the fact that Erlang's loss formula calculates the probability that a channel will be free on the arrival of a burst, but it does not consider if the channel remains free for the service duration (length) of the burst. Therefore, Erlang's loss formula cannot be considered as a good approximation for the blocking probability because it fails to capture the effects of the offset time and the burst length.

5.3 Advance Reservation Systems

The problem of calculating the blocking probability in OBS-JET networks is similar to calculating the reservation probabilities in classical advance reservation systems. An advance reservation system can be defined as:

- A system with M servers.
- A process of arriving requests to use the servers is given.
- Each request is a pair of independent random variables:
    - an advance notice, A.

  - a service duration, B.

- Loss protocol: A reservation is made if and only if, for its whole duration at least one of the set of M servers is free. Otherwise the reservation request is blocked.

- The objective is a formula for the blocking or reservation probabilities.

This system appears in many different applications, such as hotel reservations and videoconferencing systems. Several models exist in the literature for analyzing different reservation systems [22, 45, 46, 68]. However, each of these models has a set of simplifying assumptions that are applicable to their specific system of interest. Therefore, none of these models are directly applicable to the problem at hand.

The main difficulty in modeling the OBS-JET systems, and any advance reservation systems in general, is that a reservation request can be blocked by a reservation starting after its own start time. This phenomenon of "retro-blocking" [45] becomes more significant as the advance notice dispersion becomes larger.

One notable difference between OBS-JET system and the advance reservation systems, is the above mentioned loss protocol. In OBS-JET, the reservation is accepted if and only if at least one wavelength is free at the starting time of the burst, and that particular wavelength remains free for the duration of the burst. Thus, it is not sufficient to consider only the number of free wavelengths as in the case of advance reservation systems, which further complicates the problem. The example of Figure 5.1 illustrates this difference, where a reservation request for a server arrives to a system of 4 servers.

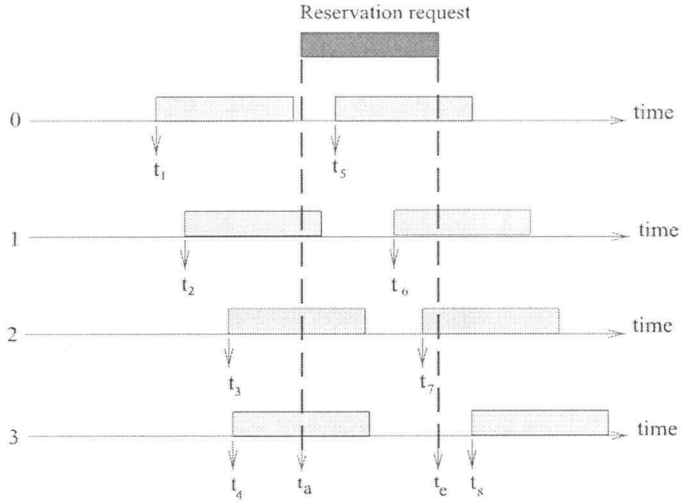


Figure 5.1: Example to illustrate the difference in the loss protocol between advance reservation systems and OBS networks.

The requested reservation is to start at time t_a and lasts for a duration equal to $(t_e - t_a)$. In classical advance reservation systems, this request will be accepted because there is a free server at any time instant in the time interval $[t_a, t_e]$. In comparison, this reservation request will be rejected (blocked) in OBS-JET systems, since there isn't any wavelength that is free on the start time of the reservation and remains free for the entire requested time interval.

5.4 Proposed Model

5.4.1 Problem Description and Assumptions

Assume that a CB requests a wavelength after a time units for a duration equal to b time units. The request is accepted if at least one wavelength is free for a period of time equal to the request duration. Stated otherwise: if t_r is the CB arrival time, then the request is granted if the request reservation interval $[t_r + a, t_r + a + b]$ does not overlap with an already reserved interval on at least one of the available wavelength channels. If there is no free interval, the CB and its corresponding DB are dropped.

In order to make the model tractable, we simplify the problem by dividing time into slots of width δ. We assume that the offset a and the burst length b are integer multiples of δ. The control burst arrival process is assumed to constitute a Poisson process with a mean of λ arrivals per time slot. The offset time A has a probability distribution $f(a) = P(A = a)$ for $a = 0, 1, ..$, and the data burst duration B has the distribution $g(b) = P(B = b)$ for $b = 1, 2, ...$ Figure 5.2 illustrates the simplified OBS-JET system. We assume that arrivals and departures occur at slot boundaries. Slots are numbered with respect to the arrival time slot of a reference CB. We use n to indicate the slot number. Our model has similar assumptions as the model developed in [45], which was developed for advance reservation systems. However, our model is significantly different from the model in [45] because of the previously mentioned difference in loss protocol between OBS-JET systems and advance reservation systems.

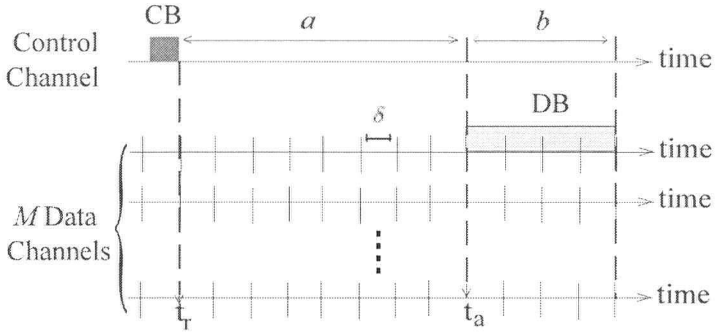


Figure 5.2: Simplified OBS-JET model.

5.4.2 Traffic Model

Assume we have a reference control burst arriving at certain time instant, and requesting to reserve a channel after n time slots from its arrival time. Without loss of generality, consider the arrival time of the control burst to be slot 0, and start of the requested duration to be slot n. In order to calculate the blocking probability of this request, we need to consider the number of arrivals at slot n seen at the time of the request. This is because any reservation request that will be generated in the future (from time slots after slot 0) for slot n will not affect the probability of accepting/blocking the reference reservation request.

The number of data bursts whose start time is within slot n, seen at the instant in which the reference control burst arrives, is Poisson distributed with mean

$$\lambda_{(n)} = \begin{cases} \lambda, & \text{for } n \leq 0 \\ \lambda\{1 - \sum\limits_{a=0}^{n-1} f(a)\}, & \text{for } n > 0 \end{cases} \tag{5.2}$$

This equation can be justified as follows. The sum of the number of requests for time slot n from all previous slots, including slot n, is λ. Given a reference control burst arriving in time slot 0, the probability that this CB will request time slot n is $f(n)$. Therefore, the number of requests from slot 0 to slot n is $\lambda f(n)$. From the point of view of the reference CB, the number of data burst arrivals in slot n is the sum of requests from all time slots before slot 0 in addition to requests from slot 0. Thus, the number of arrivals in slot n seen by the reference CB, is the number of all possible requests (λ) minus any future requests to be made in time slots 1 to n for slot n.

To further to illustrate Equation 5.2, consider the following example. Assume that the offset time can have one of four values: 0, 1, 2, or 3, each with a probability of $\frac{1}{4}$. Consider a reference control burst arriving at slot 0. The mean number of requests to slot 1, seen by the reference control burst, are the sum of requests initiated from slots -2, -1, and 0. Therefore the number of requests to slot 1 can be calculated as $\lambda\{1 - \frac{1}{4}\}$ $= \frac{3}{4}\lambda$. In the same manner, the mean number of requests to slot 3, seen by the reference control burst, are the number of requests initiated from slot 0, and can be calculated as $\lambda\{1 - \{\frac{1}{4} + \frac{1}{4} + \frac{1}{4}\}\} = \frac{1}{4}\lambda$.

5.4.3 State Description

We use two Markov chains to model our system. The first Markov chain models the state of a single wavelength channel, and it is used to calculate the probability that the wavelength channel will be free for the duration of the data burst. While the second Markov chain models the occupancy of an output link carrying multiple data wavelength channels, and it is used to calculate the probability that a wavelength channel will be free upon the arrival of the data burst. The main idea behind our model is to calculate the transitional probabilities with non-homogenous arrivals with mean given by Equation (5.2), while ignoring the retro-blocking phenomenon. Thereafter, we use the calculated transitional probabilities to express the blocking probability taking into consideration the retro-blocking phenomenon. We assume that the data burst duration has a geometric distribution $g(b) = q(1-q)^{b-1}$ with mean $\bar{B}$ and $q = 1/\bar{B}$.

Model for a single wavelength channel

Let $S^{(n)}$ represent the state of a wavelength in time slot n. If the wavelength is occupied then $S^{(n)} = 1$, otherwise $S^{(n)} = 0$. We assume that burst arrivals are uniformly distributed among wavelength channels on the output link. We denote arrivals to a single wavelength channel by $\tilde{\lambda}$. Accordingly, $\tilde{\lambda} = \lambda/M$, where M is the number of wavelength channels on the output link. A wavelength channel can be modeled as a non-homogenous

Markov chain with transitional probability matrix $Q^{(n)}$:

$$Q^{(n)} = \begin{bmatrix} p_{0,0}^{(n)} & p_{0,1}^{(n)} \\ p_{1,0}^{(n)} & p_{1,1}^{(n)} \end{bmatrix} \tag{5.3}$$

The transitional probabilities are defined as:

$$p_{i,j}^{(n)} = P(S^{(n+1)} = j | S^{(n)} = i), \qquad i,j = 0,1 \tag{5.4}$$

$$= \begin{cases} e^{-\tilde{\lambda}_{(n+1)}}, & i = j = 0 \\ 1 - e^{-\tilde{\lambda}_{(n+1)}}, & i = 0, j = 1 \\ qe^{-\tilde{\lambda}_{(n+1)}}, & i = 1, j = 0 \\ 1 - qe^{-\tilde{\lambda}_{(n+1)}}, & i = j = 1 \end{cases} \tag{5.5}$$

Model for output link

Let $Z^{(n)}$ denote the number of channels reserved on the output link in slot n ($Z^{(n)} = 0, \ldots, M$) . The output link can be modeled as a non-homogenous Markov chain with transitional probability matrix $X^{(n)}$:

$$X^{(n)} = \begin{bmatrix} x_{0,0}^{(n)} & x_{0,1}^{(n)} & \cdots & x_{0,M}^{(n)} \\ x_{1,0}^{(n)} & x_{1,1}^{(n)} & \cdots & x_{1,M}^{(n)} \\ \vdots & \vdots & \ddots & \vdots \\ x_{M,0}^{(n)} & x_{M,1}^{(n)} & \cdots & x_{M,M}^{(n)} \end{bmatrix} \tag{5.6}$$

The transitional probabilities are defined as:

$$x_{i,j}^{(n)} = P(Z^{(n+1)} = j | Z^{(n)} = i), \qquad i, j = 0, 1 \ldots, M \tag{5.7}$$

$$= \begin{cases} \sum_{z=0}^{i} \binom{i}{z} q^z (1-q)^{i-z} e^{-\lambda_{(n+1)}} \frac{\lambda_{(n+1)}^{j-i+z}}{(j-i+z)!}, & j \leq M-1, i \leq j \\ \sum_{z=0}^{j} \binom{i}{i-j+z} q^{i-j+z} (1-q)^{j-z} e^{-\lambda_{(n+1)}} \frac{\lambda_{(n+1)}^{z}}{z!}, & j \leq M-1, i \geq j \\ 1 - \sum_{z=0}^{M-1} x_{i,z}^{(n)}, & j = M \end{cases} \tag{5.8}$$

The stationary probabilities for $Z^{(n)}$ are given by

$$\Pi = \begin{bmatrix} \pi_0 & \pi_1 & \ldots & \pi_M \end{bmatrix} \tag{5.9}$$

We obtain Π by solving the following set of equations:

$$\pi_j = \sum_{i=0}^{M} x_{i,j}^{(-1)} . \pi_i, \qquad 0 \leq j \leq N \tag{5.10}$$

5.4.4 Blocking Probability Calculation

Let the probability distribution of $Z^{(n)}$ be

$$\nu_z^{(n)} = P(Z^{(n)} = z), \qquad z = 0, 1, \ldots, M.$$

A reservation request for a unit length data burst with offset equal to a slots will be blocked with probability $\nu_M^{(a)}$. To calculate the blocking probability for a reference data burst with duration b, we need to consider the state of the wavelength for all slots $n = a, a+1, .., a+b-1$. We can express the blocking probability in terms of the *first-passage-time* distributions [42]. Let $T_s(a, b)$ be the probability that, starting at state s

in time slot a, $S^{(n)}$ remains in state s for a duration equal to b time slots. Accordingly, the blocking probability for a data burst with offset a and duration b can be written as:

$$BP(a,b) = 1 - T_0(a,b) \sum_{z=0}^{M-1} \nu_z^{(a)} \tag{5.11}$$

What Equation (5.11) says is that the probability the reservation request will be accepted is equal to the probability that at least one wavelength will be free at the arrival time of the data burst, and will remain free for the duration of the data burst. The probability $\nu_z^{(a)}$ can be calculated as

$$\nu_z^{(a)} = \Pi X^{(0)} X^{(1)} \dots X^{(a-2)} X_{(:,z)}^{(a-1)} \tag{5.12}$$

The symbol $X_{(:,z)}^{(a-1)}$ denotes the z^{th} column of the matrix $X^{(a-1)}$. Equation 5.12 describes possible system evolution paths from being in one of the initial states before slot 0 to the final state of having z wavelength channels occupied in slot a. The probability that the wavelength will be unused for the duration of the data burst can be expressed as:

$$T_0(a,b) = p_{0,0}^{(a)} p_{0,0}^{(a+1)} \dots p_{0,0}^{(a+b-2)} \tag{5.13}$$

$$= e^{-\tilde{\lambda}_{(a+1)}} e^{-\tilde{\lambda}_{(a+2)}} \dots e^{-\tilde{\lambda}_{(a+b-1)}} \tag{5.14}$$

Equation 5.13 describes the probability that starting at state 0 in slot a the channel will remain in state 0 for b slots. Note that $p_{0,0}^{(a)} = P(S^{(a+1)} = j | S^{(a)} = i)$. If the offset time has a uniform distribution in $[0, A_{max}]$ then Equation (5.2) can be expressed as

$$\lambda_{(n)} = \begin{cases} \lambda, & \text{for } n \leq 0 \\ \lambda\{1 - \frac{n}{1+A_{max}}\}, & \text{for } 0 < n \leq A_{max} \\ 0, & \text{for } n > A_{max} \end{cases} \tag{5.15}$$

Under the condition that $a + b - 1 \leq A_{max}$, Equation (5.14) can be rewritten using Equation (5.15) as

$$T_0(a, b) = e^{-\tilde{\lambda}(b-1)(1-k(a+b/2))}, \qquad \text{where } k = \frac{1}{1 + A_{max}}. \tag{5.16}$$

The proof is as follows. Under the condition that $a + b - 1 \leq A_{max}$, Equation (5.14) becomes

$$T_0(a, b) = e^{-\tilde{\lambda}\{1-\sum_{i=0}^{a} \frac{1}{1+A_{max}}\}} e^{-\tilde{\lambda}\{1-\sum_{i=0}^{a+1} \frac{1}{1+A_{max}}\}} \ldots e^{-\tilde{\lambda}\{1-\sum_{i=0}^{a+b-2} \frac{1}{1+A_{max}}\}}$$

$$\text{Let } k = \frac{1}{1 + A_{max}}$$

$$\begin{aligned} T_0(a, b) &= e^{-\tilde{\lambda}((1-k(a+1))+(1-k(a+2))+\ldots+(1-k(a+b-1)))} \\ &= e^{-\tilde{\lambda}\sum_{j=1}^{b-1}[1-k(a+j)]} \\ &= e^{-\tilde{\lambda}[(1-ka)(b-1)-k(b-1)b/2]} \\ &= e^{-\tilde{\lambda}(b-1)[1-k(a+b/2)]} \qquad \square \end{aligned}$$

5.5 Performance Evaluation

In this section, we present several key results to illustrate the effect of offset-time and burst size parameters on the blocking probability in OBS-JET networks. In addition, we compare results obtained from the approximate model developed in the previous section with results from a discrete-event simulation model. Moreover, we compare our model results with results obtained from Erlang's loss formula. Finally, we study the impact of the time slot size on the proposed model accuracy.

Figure 5.3 shows the simulation model used in this section, which is based on the OPNET simulation tool. It consists of a single OBS node connected to traffic sources and a sink node. Each input link carries 2 separate wavelengths, one for data and one for control. The output link carries five wavelengths, four for data and one for control, and each wavelength channel has transmission speed of approximately 2.5 Gbps (OC-48). We assume that sources generate control and data bursts. Sources generate bursts according to Poisson process and burst sizes are exponentially distributed with mean $\bar{B} = 81920$ bits. The offset time has a uniform distribution over $[0, A_{max}]$. The maximum offset time A_{max} was arbitrary chosen to be 180 μs. In each experiment we specify the chosen time slot size in bits. The slot size in seconds (δ) can be calculated by dividing the slot size expressed in bits by the data channel bandwidth (2,377,728,000 bits/sec in this example). The blocking probability is calculated from the analytical model for offered load values ranging from 0.1 to 0.9. On the other hand, the blocking probability values obtained from the discrete-event simulation correspond to offered load values ranging from 0.4 to

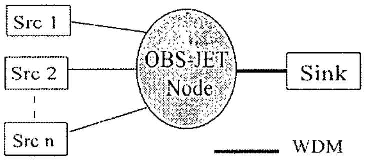


Figure 5.3: Simulation model for Section 5.5.

0.9. The smaller range is due to the fact that in order to get an accurate value for the blocking probability from the discrete-event simulation at low loads, the simulation needs to be executed for prohibitively long times.

The wavelength scheduling algorithm employed here is First Fit with Void Filling (FF-VF)[74]. The FF-VF algorithm finds the first available data channel for each arriving data burst. Given the arrival time t_a of a data burst with duration b to the optical switch, the scheduler finds the first data channel that is available for the time period $[t_a, t_a + b]$. Figure 5.4 illustrates the FF-VF algorithm. A new burst arrives at time t_a. At time t_a, wavelength 0 is ineligible because the void on this wavelength is too small to fit the new burst. The algorithm chooses the first available wavelength channel, which is channel 1 in this example.

5.5.1 Effect of Offset Time

In this subsection, we study the effect of the burst offset time on its blocking probability. The blocking probability is calculated with burst duration $b = \bar{B}$ and slot size = 20480 bits. Figure 5.5 shows the corresponding blocking probability results. There are three

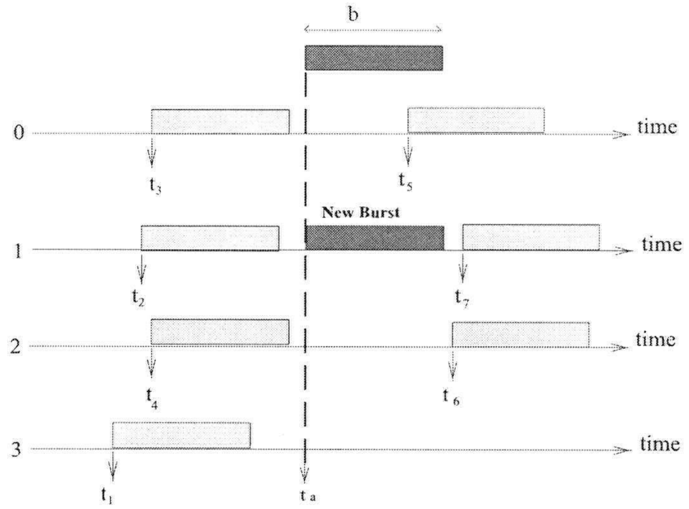


Figure 5.4: Illustration of the FF-VF algorithm

sets of plots in Figure 5.5, each corresponds to a different offset time value. Each set consists of two plots, one corresponding to simulation results and one corresponding to results obtained using the approximate analytical model we developed in Section 5.4.

Result sets 1, 2 and 3 correspond to offset time values $a = 0.2A_{max}$, $a = 0.5A_{max}$ and $a = A_{max}$, respectively. From the three sets, we observe that as the offset time value increases, the blocking probability decreases (i.e., $BP(0.2A_{max}) > BP(0.5A_{max}) > BP(A_{max})$) and approaches zero in the case of $a = A_{max}$. This behavior is expected, since the further in the future the requested interval is, the less the number of requests for intervals it will intersect with. Given a particular request for interval $[t_s, t_e]$, only those requests that have finishing times larger than t_s can block this particular request. This fact has been exploited in [80] to provide different levels of quality-of-service.

The effectiveness and accuracy of our approximate analytical model is demonstrated

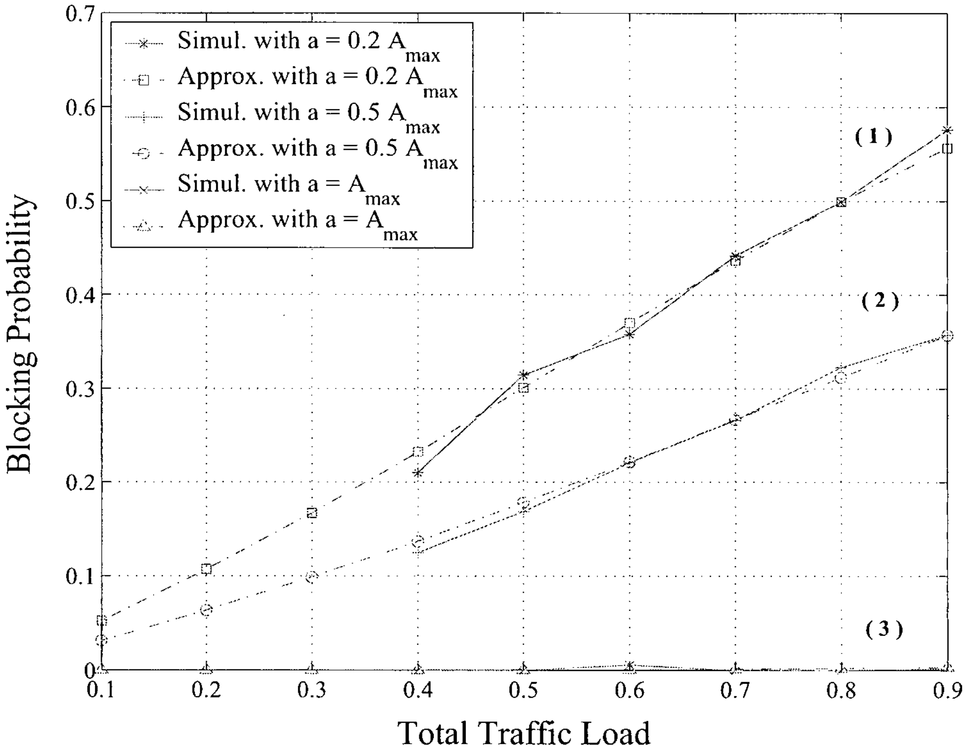


Figure 5.5: Effect of the burst's offset time on its blocking probability.

by its satisfactory agreement with the simulation results in Figure 5.5. The effect of the offset time on the blocking probability in our model can be understood from Equation (5.2). As the value of a increases, the arrival rate $\lambda_{(n)}$ decreases which, in turn, causes the blocking probability to drop. In addition, Equation (5.16) shows the effect of the burst offset time on its blocking probability in the case of uniformly distributed offset times. As the value of a increases, the value of $T_0(a, b)$ increases and as a result, the blocking probability decreases.

5.5.2 Effect of Burst Size

In this subsection we study the effect of varying the burst size on the burst blocking probability. The blocking probability is calculated with offset time value $a = 0.5A_{max}$ and slot size = 5120 bits. We use two values for the burst size b in this experiment: b_1= 20480 bits ($0.25\bar{B}$) and b_2= 122880 bits ($1.5\bar{B}$).

Figure 5.6 plots the resulting blocking probability in two sets of plots. Set 1 corresponds to using burst size equal to b_1 and set 2 corresponds to using burst size equal to b_2. It is clear from the figure that the blocking probability value for b_1 is less than its value for b_2. This is due to the fact that the value of b affects the probability that the wavelength scheduler algorithm will be successful in finding a suitable time gap (void) on a wavelength to fit the incoming burst. Intuitively, the larger the burst size the lower the probability of success. As a result, the larger the value of b the larger the blocking probability.

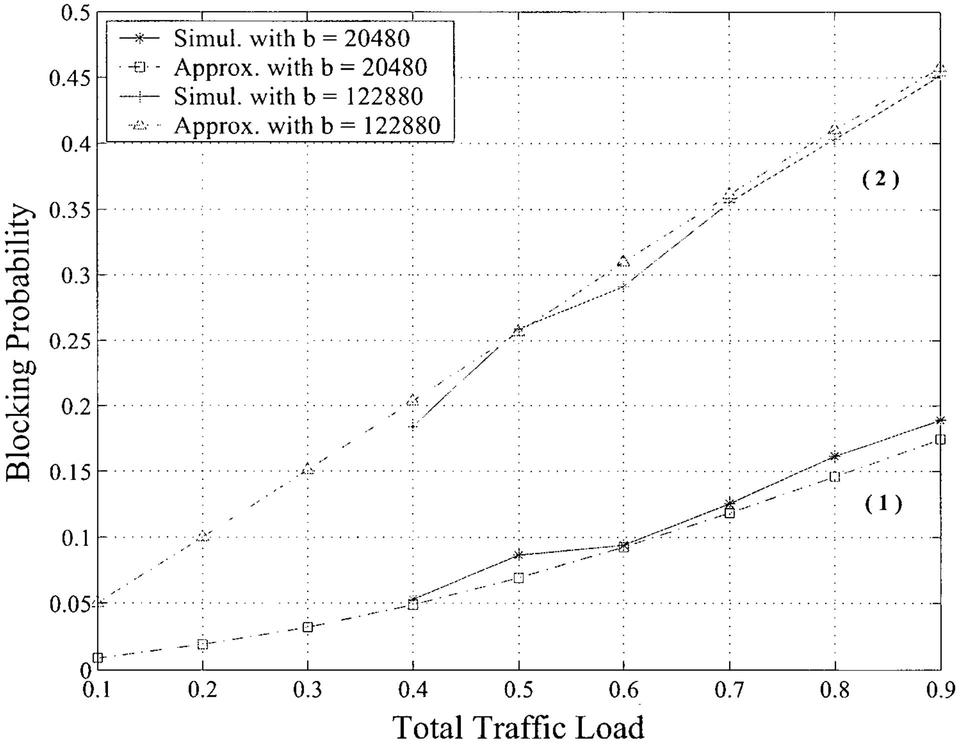


Figure 5.6: Effect of the burst's size on its blocking probability.

As in the case of the previous subsection, the results plotted in Figure 5.6 exemplify the good agreement between our analytical model and simulation results. The effect of the burst size can be understood from Equation (5.14). Generally, the longer the burst the larger the number of probability terms that get multiplied in Equation (5.14). Therefore, a larger value of b causes the probability $T_0(a, b)$ to decrease which, in turn, causes the blocking probability $BP(a, b)$ to increase.

5.5.3 Comparison with Erlang's Loss Formula

In this subsection, we compare results obtained from Erlang's loss formula (see Section 5.2) with results from our proposed model. The blocking probability is calculated with the mean value for both the offset time and burst length ($a = 0.5A_{max}$, $b = \bar{B}$.) and slot size = 20480 bits. Results from our model, simulation, and Erlang's B-formula are shown in Figure 5.7.

Figure 5.7 confirms that our model is much more accurate than Erlang's loss formula. In addition, Figure 5.7 shows that the curve produced from our approximate model follows the same trend as the Erlang's loss formula, thus providing strong evidence on the accuracy of our model. It is important to reemphasize that Erlang's loss formula does not consider the effect of either the offset time nor the burst length on the blocking probability, and only depends on the average load. Accordingly, changing the offset time value or the burst length will not affect the results obtained from Erlang's loss formula, as long as the offered load is the same.

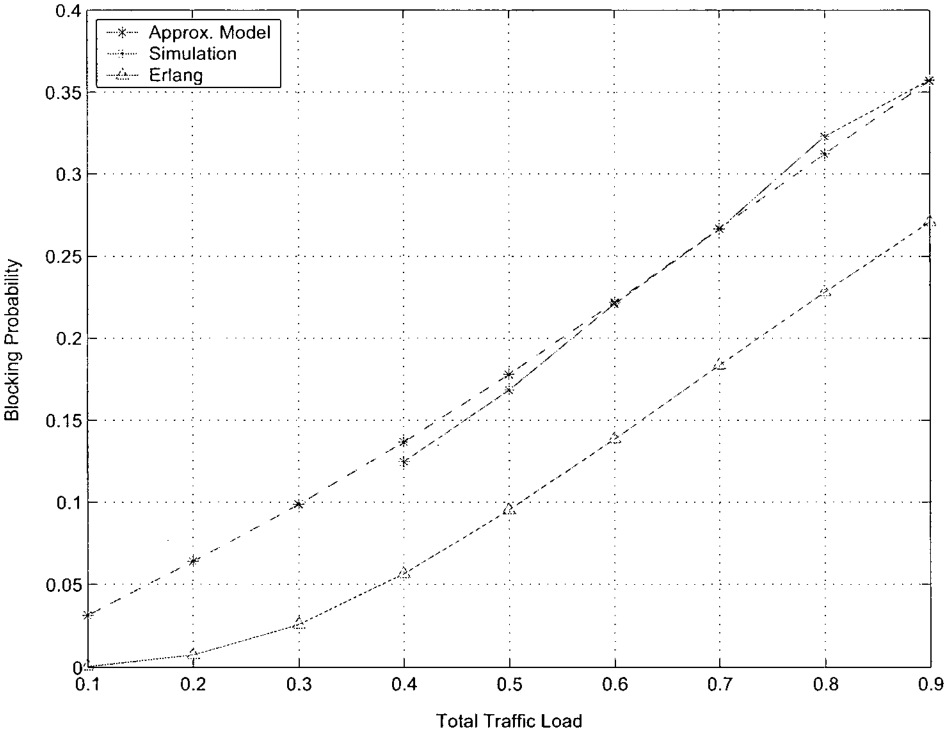


Figure 5.7: Comparison between the proposed model and Erlang's loss formula.

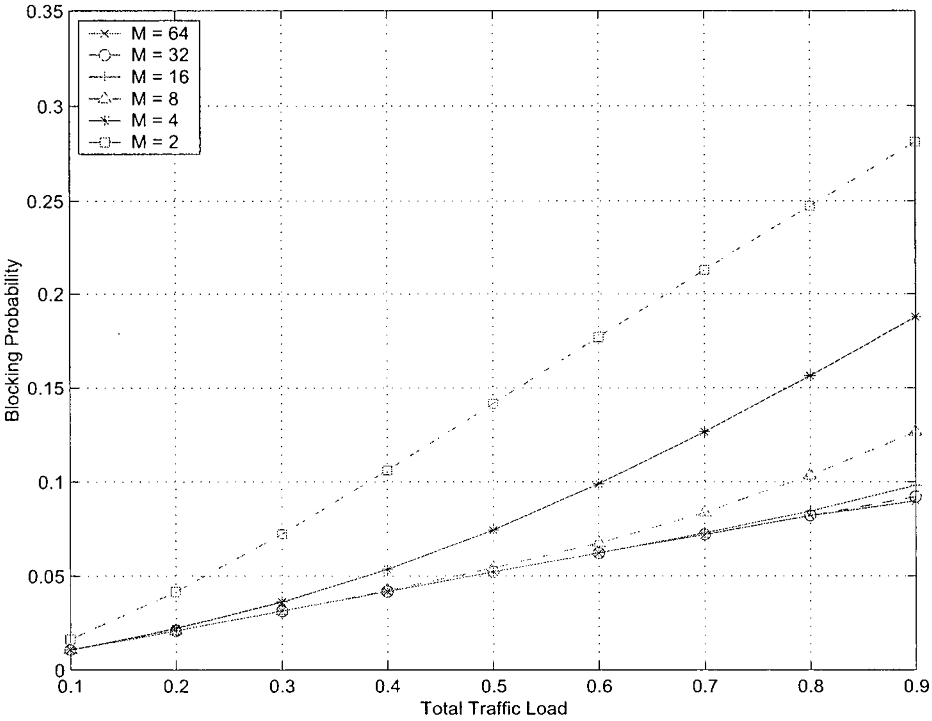


Figure 5.8: Effect of the number of wavelengths on the burst blocking blocking probability.

5.5.4 Effect of the number of wavelength channels

We study in this section the effect of the number of wavelength channels at the output link on the blocking probability. We assign different values for M (the number of wavelength channels): 2, 4, 8, 16, 32, and 64. The blocking probability is calculated from the analytical model with the mean value for the offset time ($a = 0.5A_{max}$), burst length $b = 0.5\bar{B}$ and slot size = 10240 bits. The offered load value is varied between 0.1 and 0.9 for every value of M. Results are shown in Figure 5.8.

Results in Figure 5.8 show that when M increases the value of the blocking probability decreases. However, results also show that the effect of M on the blocking probability decreases as M increases. In other words, increasing M beyond a certain value, $M = 32$ in this case, have minimal effect on the blocking probability, and the offered load becomes the sole controlling factor. It can be seen from the figure that the blocking probability for the case of $M = 64$ is almost indistinguishable from the blocking probability in the case of $M = 32$.

It is important to note here that the execution time of our analytical model increases as the number of wavelength channels increase. This due to the fact that the size of the transitional probability matrix $X^{(n)}$, defined in Equation 5.6, increases with the number of wavelength channels. Therefore, the execution time of Equation 5.12 increases with M. To further illustrate the growth in the execution time of our model, Table 5.1 shows the execution time needed to calculate the results in Figure 5.8. The table also shows the running time of the discrete-event simulation for the case of $M = 4$.

It is apparent from Table 5.1 that the execution time of the analytical model increases significantly as M increases. Nevertheless, the execution time of the analytical model is by far less than the execution time of the simulation model.

5.5.5 Effect of time slot size

The OBS-JET network is not a time slotted network. However, our model assumes that time is slotted for simplicity. Therefore, time slotting in our model is an artifact of the

Table 5.1: Execution time of the analytical model for different values of M

M	**Analytical Model**	**Simulation**
4	1 second	$\approx$ 54 hours
16	$\approx$ 15 minutes	-
32	$\approx$ 2.6 hours	-
64	$\approx$ 23 hours	-

modeling process. In this subsection, we study the impact of the slot size on the accuracy of our model and thereafter provide simple guidelines for the selection of the slot size. The blocking probability is calculated with the mean value for both the offset time and burst length. The offset time value used is $a = 0.5A_{max}$ and the value of the burst size b is $\bar{B}$. The values used for the slot size in bits are: 81920, 40960, 20480, 10240, 5120, and 4096 bits. The corresponding values expressed in slots for the mean burst size, the burst offset value, and the maximum offset value are shown in Table 5.2. The value of the offset time expressed in slots can be calculated by dividing its value (in seconds) by δ (the slot size in seconds). Results from our analytical model are shown in Figure 5.9.

In Figure 5.9, the blocking probability corresponding to slot size equal to 81920 is much lower from the rest of the curves and very far from the values obtained from the discrete-event simulation of Subsection 5.5.3. This observation can be explained by noting that the burst size in this case is 1 slot, which means that the effect of burst size is ignored.

Table 5.2: Slot size values and their corresponding mean burst sizes, offsets, and max offset values

Slot size	$\bar{B}$	**Offset value**	A_{max}
81920	1	3	5
40960	2	5	10
20480	4	10	20
10240	8	20	40
5120	16	40	80
4096	20	50	100

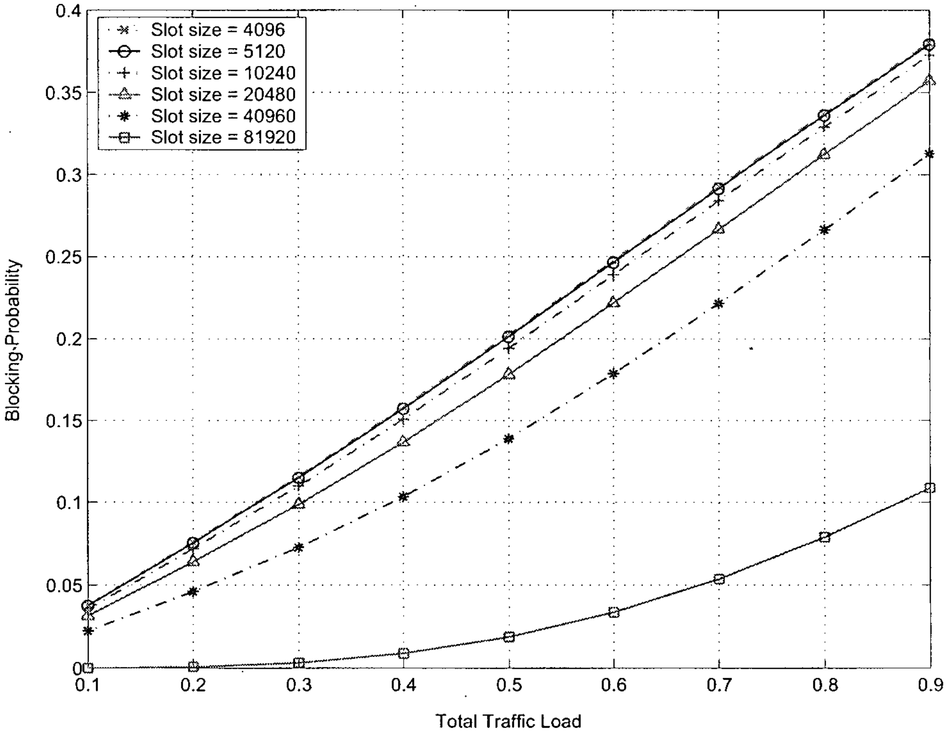


Figure 5.9: Effect of slot size on the accuracy of the proposed model.

This is evident from Equation (5.16) where the value of $T_0(a, b)$ becomes equal to 1 when the value of b is equal to 1. As the slot size value decreases (i.e., value of b translates to a larger number of slots), the value of the blocking probability becomes closer to values obtained from simulation in Figure 5.7. This continues to be the case until the value of b reaches 16 slots. Any further decrease in the slot size results in identical blocking probability values. Accordingly, our model produces accurate blocking probability values when the slot size is chosen such that the value of b is larger than 4 slots and this applies for any value of b and $\bar{B}$.

5.6 Concluding Remarks

This chapter presented an approximate analytical model for calculating the blocking probability in OBS-JET networks. The proposed model takes into account the effects of the offset time and burst duration on the blocking probability. To describe the data burst arrival process we used a non-homogenous Poisson process whose mean value decreases with the data burst arrival time. Moreover, we expressed the blocking probability in terms of first passage time distributions to account for the transient behavior of the wavelength channel for the duration of the data burst. We verified the accuracy of the proposed model by comparing it with the results of a sophisticated discrete-event simulation model. The model results were found to be in very good agreement with simulation results. In addition to being accurate, our approximate model is orders of magnitude faster than the discrete-event simulation: computing one point for the plots of Figures 5.5 to 5.7 take many hours of simulation time, while it takes about one second only for the same computation using our model.

Chapter 6

Quantitative QoS Guarantees in Labeled OBS Networks

6.1 Introduction

Labeled Optical Burst Switching (LOBS), proposed in [58], is an integration of multiprotocol label switching (MPLS)[64], or its generalized version (GMPLS) [14], and optical burst switching (OBS). MPLS is a forwarding technique that uses labels associated with packets to make packet forwarding decisions in network nodes. In MPLS, the entire set of possible packets are partitioned into a set of *Forwarding Equivalence Classes* (FECs). A FEC is defined as a group of IP packets which are forwarded in the same manner (e.g., over the same path, with the same forwarding treatment).

A number of approaches for QoS provisioning in OBS networks have been proposed in the literature. These approaches can be classified into offset-based [81], strict priority [77][29], segmentation-based [70], or proportional QoS [5].

In offset-based mechanisms different burst loss probabilities are obtained by assigning different offset values to different classes. The highest priority class gets the largest offset and thus is able to reserve resources ahead of other classes. The main disadvantage of this protocol is that it introduces a significant amount of delay for high priority traffic.

Strict priority schemes provide strict priority for high priority traffic class by dropping future reservations belonging to lower priority traffic class. Therefore, a reservation request belonging to high priority class can be only blocked by an existing reservation in the same priority level, or lower priority reservation that has a start time in the past (that is to say, the reserved duration has started already). Thus, dropping of high priority traffic is minimized. However, this scheme is not flexible, since it is not possible to control the bandwidth share or loss probability for different service classes. Furthermore, if the highest priority traffic is misbehaving, the lower priority traffic can starve.

Bursts in segmentation-based mechanisms are composed of segments, each segment belonging to a certain priority class. Contention is resolved in the core by dropping the lower priority segments. Accordingly, the low priority traffic will get higher dropping probability than the high priority traffic. The drawback of this scheme is the increased complexity of burst assembly and scheduling.

In proportional QoS schemes, the burst loss probability and average packet delay are adjusted to be proportional to the factors that the network service provider sets. Although this scheme is the most flexible scheme mentioned in literature, the proportional differentiation technique is not scalable as mentioned before in Section 4.1.

The main goal of the aforementioned proposals is to provide relative service differentiation with regards to packet loss probability.

In this chapter, we describe a detailed architecture for providing *quantitative* QoS guarantees with respect to end-to-end delay, throughput, and packet loss probability in buffer-less LOBS networks. The proposed architecture achieves its goal by deploying fair scheduling algorithms in edge and core LOBS nodes. The use of fair scheduling algorithms in the data plane of optical nodes has generally been avoided in the literature. This is due to the absence of the concept of "packet queues" in optical nodes, beyond the number of packets that can be buffered (while in-flight) in fiber delay lines. We will present a novel approach for applying fair scheduling algorithms in both the data plane of LOBS edge nodes and the control plane of core nodes without the need for optical buffering. In addition, we present analytical expressions for the end-to-end delay and the blocking probability in the proposed architecture.

The rest of this chapter is organized as follows. Section 6.2 introduces the proposed architecture and explains its components. We give a concrete example of how this architecture may provide QoS guarantees in Section 6.3. In Section 6.4, we evaluate the performance of the architecture through simulation. Concluding remarks are provided in Section 6.5.

6.2 Proposed Architecture

The proposed architecture is composed of a number of functional elements implemented in network nodes, including packet classifier, burst assembler, signaling protocol, and traffic conditioning functions that include shaping and policing. We assume in this architecture that data bursts are assembled at the network ingress and delivered to the egress in all-optical domain.

The network operator defines a set of FECs in the network, each with its own QoS parameters (e.g., bandwidth share, delay, and loss probability). Constraint-based routing and label distribution protocol are used to establish a path between ingress and egress nodes. In this section, we describe the edge node architecture, the employed signaling protocol, and the core node architecture.

6.2.1 OBS Edge node

The functional blocks of an OBS edge node are illustrated in Figure 6.1. Packets arriving at the ingress node are classified into one of the FECs already defined by the network operator. These packets are then shaped and policed to conform with the QoS requirements of their FEC. Packets are then assembled into bursts in the burst assembly queue. The burst assembly process is carried out using the *CAT* algorithm [41]. Thereafter, data bursts are inserted into the bursts queue and scheduled for processing by the reservation manager according to their QoS requirements.

We propose the use of *Fair Packet Queueing* (FPQ) [65] algorithms for scheduling the processing of data bursts by the reservation manager. The FPQ scheduling discipline has three desirable properties: (a) it can guarantee an upper bound on delay to token-bucket constrained sessions; (b) it guarantees the upper bound on delay regardless of the behavior of other sessions (isolation); (c) it can ensure relative fairness in bandwidth allocation

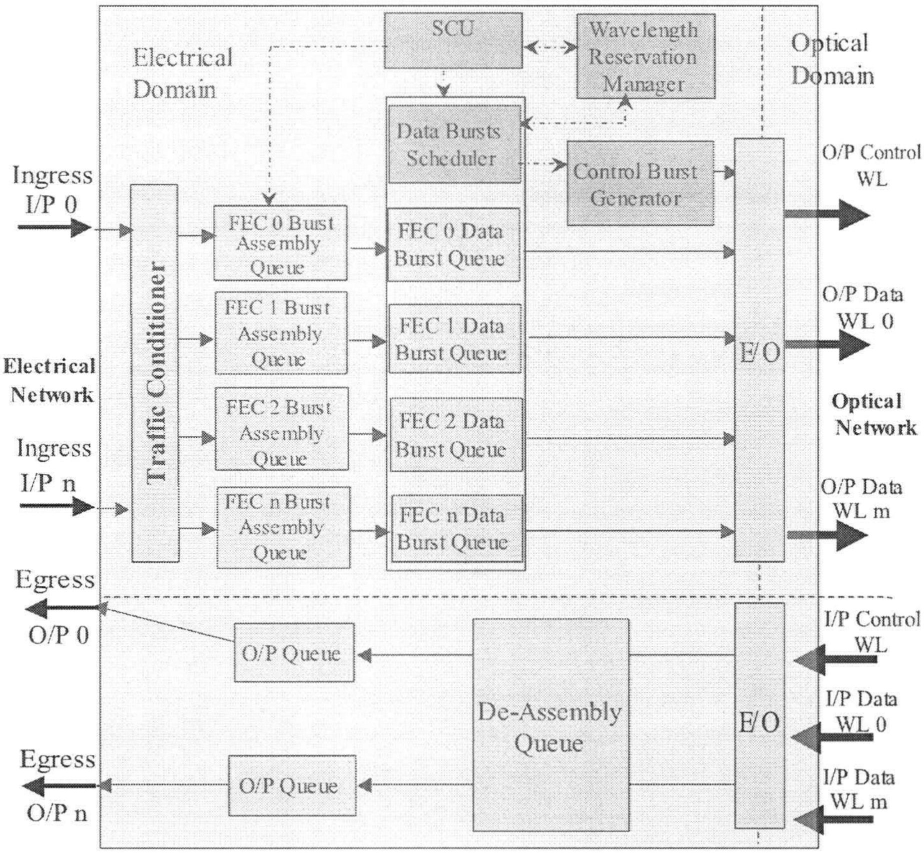


Figure 6.1: Functional blocks of an OBS edge node.

among backlogged sessions. The particular choice of the scheduler is not imposed by the architecture. Several FPQ algorithms have been proposed in the literature, including *Weighted Fair Queueing* (WFQ) [8], *Self-Clocked Fair Queueing* (SCFQ) [19], and *Start-time Fair Queueing* (SFQ) [21].

In our OBS architecture, the FPQ scheduler is used to select, from the bursts queue, the data burst eligible for processing by the wavelength reservation manager. In this context, there is a significant difference between the proposed implementation of the FPQ scheduler in OBS edge nodes and its usage as a conventional packet scheduler. In the latter case, the packet selected by the FPQ scheduler is directly sent for transmission, whereas in our architecture the FPQ scheduler regulates the access to the reservation manager. This difference appears more clearly when calculating the queuing delay in the burst queue, since the processing delay in the reservation manager is independent of the burst's length.

The switch control unit (SCU) invokes a wavelength reservation manager to decide on which outgoing data channel to forward the data burst. The primary job of the wavelength reservation manager is to process wavelength reservation requests and to keep track of the availability of free time intervals (voids) on every data wavelength channel on the link. The reservation manager exploits a wavelength scheduling algorithm which can be any of the algorithms mentioned in Chapter 3. The wavelength scheduling algorithm tries to find a void on any of the output wavelengths, such that the void starts after the offset time and has a duration equal to the DB transmission time. If the reservation is successful, a control burst is generated. The generated CB is updated with the FEC label and the reserved wavelength, then it is sent out into the network. If the reservation fails, the DB is dropped.

6.2.2 Signaling Protocol

We adopted the JET protocol as the CB signaling protocol in our architecture with minor modification. Basically, the destination field is replaced with the MPLS label field. The label is assigned to the control burst at creation time according to the data burst's FEC. In addition to implicitly pointing to the QoS information for the burst, the label simplifies the forwarding operation of the control burst by encoding the route of the burst, and thus specifying the output port in intermediate nodes.

6.2.3 OBS Core node

Figure 6.2 depicts the functional architecture of the OBS core node. A separate queue is established for each of the configured FECs. When a CB arrives at a core node, it gets time-stamped with its arrival time, then classified based on its label and inserted in the corresponding queue. In our architecture, we use a FPQ scheduling discipline to select the CB to be processed by the reservation manager. The fashion in which the FPQ scheduler is used here has two major differences from how it is used in the OBS edge node or in electronic packet networks. First, the actual data bursts are absent from the core node at the time of scheduling, therefore, they are scheduled through the processing

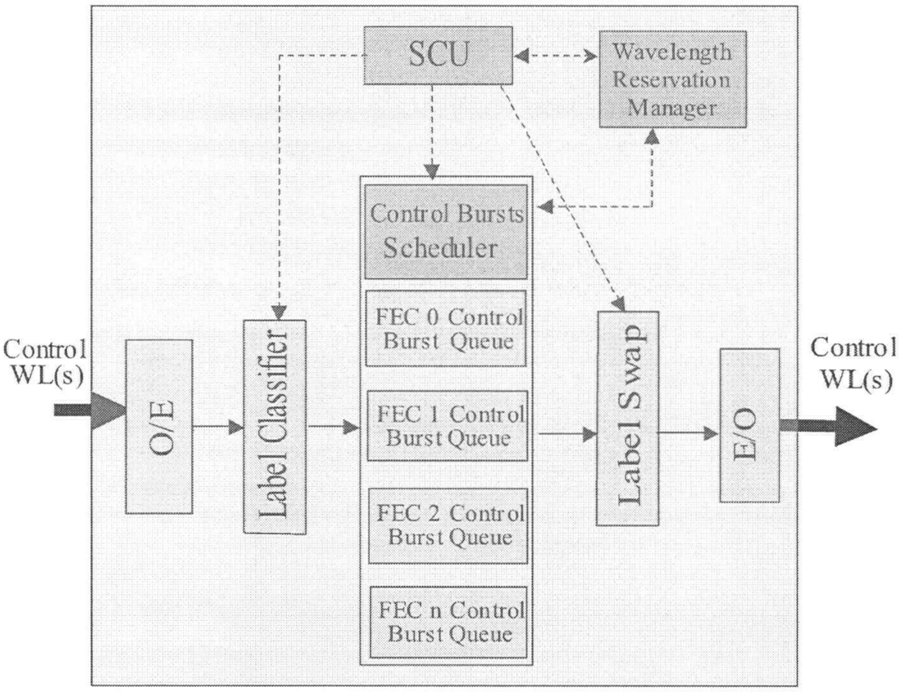
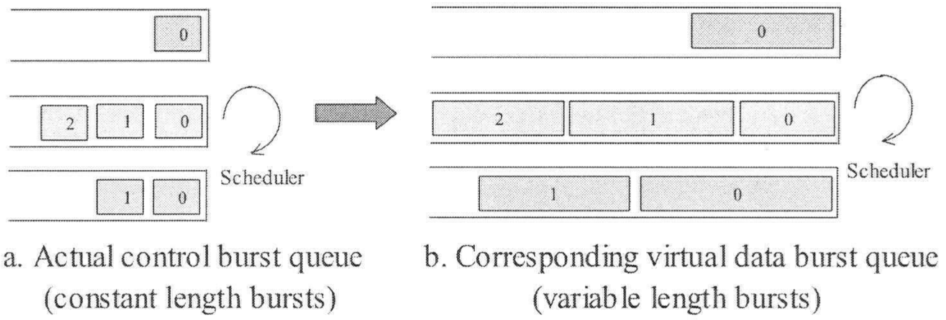

SCU
Wavelength Reservation Manager
Control Bursts Scheduler
FEC 0 Control Burst Queue
FEC 1 Control Burst Queue
FEC 2 Control Burst Queue
FEC n Control Burst Queue
Control WL(s)
O/E
Label Classifier
Label Swap
E/O
Control WL(s)

Figure 6.2: Functional blocks of an OBS core node.



a. Actual control burst queue (constant length bursts)

b. Corresponding virtual data burst queue (variable length bursts)

Figure 6.3: Illustration of the virtual queue concept.

of the "representative" control bursts. Second, the length used in calculating the eligible control burst is *not* the control burst length, rather, it is the *data burst length* which is specified in the control burst. This implies that the scheduler is working on a "virtual" queue structure of data bursts. This concept is illustrated in Figure 6.3. The actual control burst queue, shown in Figure 6.3a, contains control bursts which have constant size. Using the data burst length field in the control bursts, the FPQ scheduler creates a virtual queue for variable-sized data bursts, as shown in Figure 6.3b, and uses this virtual queue to select the eligible control burst. Once a CB is selected it will be processed by the reservation manager which functions in the same manner as the reservation manager in the OBS edge node described earlier. The offset time field (T_o) contained in the CB is then updated, based on the CB arrival time-stamp, and label swapping is performed if necessary. The SCU keeps track of reservations on all wavelengths on all ports. When the arrival time of a data burst is reached, the switching fabric is configured so that the data burst is switched transparently on its reserved wavelength.

6.3 QoS guarantees

In this section, we describe how the proposed architecture provides guarantees for bandwidth, end-to-end delay, and loss probability. The upper bound on the delay in our architecture depends on the particular choice of the FPQ algorithm, since each FPQ has its own worst-case delay bound. To illustrate how the upper bound on the delay can be derived, we choose the well known Weighted-Fair-Queuing (WFQ) algorithm for scheduling data bursts in ingress nodes and for scheduling control bursts in core nodes. We start by giving a brief review of WFQ calculations, then explain how WFQ can be adapted to OBS networks.

6.3.1 Weighted Fair Queueing

WFQ belongs to the Generalized Processor Sharing (GPS) scheduling family [8]. In this algorithm, each packet arriving to the output port is stamped with a finish time according to the following formula

$$F_i^k = \max\{F_i^{k-1}, V(a_i^k)\} + \frac{L_i^k}{r_i}, \tag{6.1}$$

where F_i^k is the finish time of the k^{th} packet of the i^{th} session, L_i^k is the packet length, a_i^k is its arrival time, and r_i is the reserved rate for session i. $V(t)$ is the virtual time function representing the progression of virtual time in GPS. Packets are then serviced in the increasing order of their finish times.

In WFQ, the worst case delay bound for session i that is constrained by a *token bucket* with parameters (σ_i, r_i) [7] is

$$d_{max,i} = \frac{L_{max}}{r} + \frac{\sigma_i + L_{max}}{r_i} \qquad (6.2)$$

where r is the output link capacity, σ_i is the token bucket depth for session i, and r_i is the token bucket rate for session i.

WFQ is claimed to have high computational complexity, which can be an issue for OBS nodes that are required to process control bursts at high speed. However, there are a number of WFQ variants that are computationally more efficient than WFQ (e.g., SCFQ and SFQ). It is important to reemphasize that the particular choice of the scheduler is not enforced by our architecture, and any FPQ algorithm can be used.

6.3.2 Bandwidth Guarantees

Each of the configured FECs has a weight (ϕ_i) associated with it. This weight represents its required share of bandwidth. Using WFQ, or any FPQ algorithm, and having a separate queue for each FEC, the number of wavelength reservation requests generated from a FEC will be proportional to it's weight, and this applies both to ingress and core nodes. Accordingly, the rate (r_i) received by FEC i at an output link is

$$r_i = (1 - bp_i)\frac{\phi_i}{\phi}r, \qquad (6.3)$$

where bp_i is the blocking probability of FEC i at the output link, ϕ is the sum of all backlogged FECs' weights at the output link, and r is the output link capacity. Admission control policy need to be applied to insure that the sum of rates of all flows belonging to a certain FEC does not exceed its reserved bandwidth share. If it exceeds its bandwidth share, only the misbehaving FEC will be affected due to the isolation property of the FPQ algorithms.

6.3.3 Delay guarantees

In our architecture, the total delay that a packet, belonging to FEC i, encounters in its path from ingress to egress is composed of six components: burst assembly delay ($d_{a,i}$) in ingress OBS node, queueing delay in ingress node before being processed by the reservation manager ($d_{q,i}$), burst offset time ($T_{o,i}$), packet transmission delay ($d_{t,i}$), light propagation delay ($d_{l,i}$) and the queuing delay at the output of the egress OBS after the burst de-assembly process ($d_{e,i}$). Thus the total delay (D_i) for packets belonging to FEC i is

$$D_i = d_{a,i} + d_{q,i} + T_{o,i} + d_{t;i} + d_{l,i} + d_{e,i}. \tag{6.4}$$

Note that the propagation delay $d_{l,i}$ depends only on the total light path length, while $T_{o,i}$ is proportional to the number of OBS core nodes on that path. Therefore, $d_{l,i}$ and $T_{o,i}$ are uncontrollable quantities in our architecture. In the following, we derive expressions for each of the delay components.

The packet arrival process to FEC i is assumed to be Poisson process with mean value λ_i packets/sec, and the packet size is assumed to have an average length of L bits. The average burst assembly delay can then be calculated as follows:

$$d_{a,i} = \frac{1}{N_i}\sum_{k=1}^{N_i} d_k \tag{6.5}$$

$$= \frac{1}{N_i}\sum_{k=1}^{N_i}(N_i - k)\frac{1}{\lambda_i} \tag{6.6}$$

$$= \frac{1}{2\lambda_i}(N_i - 1) \tag{6.7}$$

where d_k is the assembly delay for packet number k in the assembly queue, and N_i is the average number of packets in a data burst belonging to FEC i. The variable N_i can have one of two possible values depending on the arrival rate λ_i, the FEC's maximum burst size ($B_{max,i}$) and the FEC's burst assembly timeout ($T_{max,i}$). In order to calculate the value of N_i we need to consider two possibilities: one is that $T_{max,i}$ is reached first, and the other is that $B_{max,i}$ is reached first. If the timeout is reached first, then $N_i = \lambda_i T_{max,i}$. If the maximum burst size is reached first, then $N_i = \lfloor \frac{B_{max,i}}{L} \rfloor$. Thus, Equation 6.5 can be rewritten as:

$$d_{a,i} = \begin{cases} \frac{1}{2\lambda_i}(\lfloor \frac{B_{max,i}}{L} \rfloor - 1)\ ,\ \lambda_i \geq \frac{1}{T_{max,i}}\frac{B_{max,i}}{L} \\ \\ \frac{1}{2\lambda_i}(\lambda_i T_{max,i} - 1)\ ,\ \lambda_i < \frac{1}{T_{max,i}}\frac{B_{max,i}}{L} \end{cases} \tag{6.8}$$

It is clear from Equation 6.8 that $d_{a,i}$ is directly proportional to $B_{max,i}$, the maximum burst size in the CAT algorithm, and inversely proportional to the traffic load.

The maximum burst queueing delay d_q can be derived by mapping Equation 6.2 to our specific problem. Recall from the discussion in Subsection 6.2.1 that the data burst scheduler regulates the access to the reservation manager (not the output link).

Therefore, the processing delay per DB is constant (we call it t_{db}) and does not depend on the DB length. The first term in Equation 6.2 models the situation where a packet arrives at a busy scheduler and may have to wait up to $\frac{L_{max}}{r}$ time before it is served. This corresponds to the processing delay t_{db} in our problem. The second term σ_i/r_i reflects the maximum delay for a burst of packets of length σ_i. It can be seen that this term can be substituted in our case with $\frac{N_i t_{db}}{\phi_i/\phi}$, where N_i is the maximum number of data bursts arriving to the bursts queue of FEC i, and ϕ is the sum of all backlogged FECs' weights. Similarly the third term $\frac{L_{max}}{r_i}$ can be mapped to $\frac{t_{db}}{\phi_i/\phi}$. Thus, the worst case maximum queueing delay for FEC i is

$$d_{q,i} = t_{db} + \frac{\phi N_i t_{db}}{\phi_i} + \frac{\phi t_{db}}{\phi_i}. \tag{6.9}$$

The value of N_i can be calculated from the token bucket depth of FEC i as follows.

$$N_i = \begin{cases} 1 & \text{if } \sigma_i \leq B_{max,i}, \\ \lfloor \frac{\sigma_i}{B_{max,i}} \rfloor & \text{if } \sigma_i > B_{max,i} \end{cases} \tag{6.10}$$

This equation can be explained by looking at Figure 6.4. In Figure 6.4a, where $\sigma_i \leq B_{max,i}$, after the arrival of σ_i bits to FEC i queue, the burst assembler has to wait extra time $t = \frac{B_{max,i} - \sigma_i}{r_i}$ until the maximum burst size is reached, and thereafter one data burst is generated. While in Figure 6.4b, on the arrival of σ_i bits, a number of bursts equal to $\lfloor \frac{\sigma_i}{B_{max,i}} \rfloor$ will be assembled. Hence Equation 6.9 becomes

$$d_{q,i} = t_{db} + \frac{\phi \lfloor \frac{\sigma_i}{B_{max,i}} \rfloor t_{db}}{\phi_i} + \frac{\phi t_{db}}{\phi_i}. \tag{6.11}$$

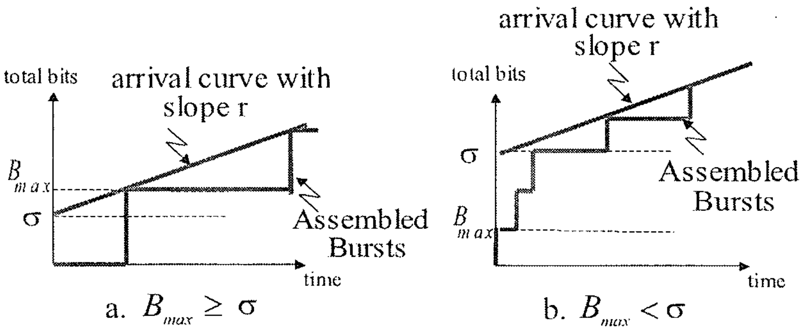


Figure 6.4: Bursts arrivals to the burst queue depends on the values of σ and B_{max}.

According to the above analysis, bounding the delay for traffic belonging to a FEC depends on shaping the input traffic so that all traffic flows belonging to that FEC conform to a token bucket with parameters (σ_x, r_x), where $\sum r_x$ does not exceed the total FEC bandwidth reservation, and the ratio $\frac{\sum \sigma_x}{B_{max}}$ does not cause data burst queuing to be unacceptability large. The average burst queueing delay is approximately half the maximum value.

The average queuing delay at the output of the egress OBS after the burst de-assembly process $(d_{e,i})$ can be calculated in a manner similar to that of $d_{a,i}$, however, we need to consider that multiple bursts could arrive to the output queue in the same time. Therefore, $d_{e,i}$ can be calculated as

$$d_{e,i} = \frac{1}{M}\sum_{k=1}^{M}\frac{d_{t,i}}{2}(kN_i - 1) \tag{6.12}$$

$$= \frac{d_{t,i}}{2}\{\frac{N_i(M+1)}{2} - 1\} \tag{6.13}$$

$$= \begin{cases} \frac{d_{t,i}}{2}(\lfloor\frac{B_{max,i}}{L}\rfloor\frac{(M+1)}{2} - 1)\,,\ \lambda_i \geq \frac{1}{T_{max,i}}\frac{B_{max,i}}{L} \\ \frac{d_{t,i}}{2}(\frac{\lambda_i T_{max,i}(M+1)}{2} - 1)\,,\ \lambda_i < \frac{1}{T_{max,i}}\frac{B_{max,i}}{L} \end{cases} \tag{6.14}$$

where M is the maximum number of bursts arriving to the same output queue at the egress node, which is equal to $\lfloor\frac{\sigma_i}{B_{max,i}}\rfloor$. The packet transmission delay $d_{t,i}$ is equal to $\frac{L}{C}$, where C is the wavelength channel speed. The worst case value of $d_{e,i}$ is given by $d_{t,i}(M\lfloor\frac{B_{max,i}}{L}\rfloor - 1)$.

6.3.4 Burst Loss Probability

Two factors control burst loss probability for a FEC in our architecture: conformance of the FEC traffic to the negotiated bandwidth share and B_{max} for that FEC. Consider the case where all FECs have the same value for B_{max}, if all FECs traffic are well behaving, then all FECs will have the same loss probability value and that value will be a function of network utilization only. Assuming there is no policing at ingress, if a FEC is sending traffic more than its reserved share, then that FEC's control bursts will be queued for a longer time because of the isolation property of the scheduler. If a control burst get queued for a time period longer than it's remaining offset time value, then it will be dropped with it's corresponding data burst.

The value of B_{max} affects the probability that the reservation manager will be successful in finding a suitable time gap (void) on a wavelength to fit the incoming burst. Intuitively, the larger the burst size, the lower the probability of success. We have presented in Chapter 5 an analytical expression for calculating the burst blocking probability at the output port of an OBS node in terms of the burst's length, offset time value and the total offered load. We have shown that for a given offset time value and traffic load, the blocking probability increases as the value of B_{max} increases, which confirms our intuition.

If a particular FEC traffic conforms to the its negotiated bandwidth share, then the blocking probability for data bursts belonging to that FEC can be approximated as follows. Let the blocking probability for a data burst with length b, offset time T and arriving at node n be $BP(b, \rho_n, T)$, where ρ_n is the offered load at node n. $BP(b, \rho_n, T)$ is the blocking probability at a single OBS node, and can be calculated using the model given in Chapter 5.

The blocking probability for a data burst with offset time T and duration b can be written as:

$$BP(b, \rho_n, T) = 1 - S_0(T, b, \rho_n) \sum_{z=0}^{W-1} \nu_z^{(T)}(\rho_n) \tag{6.15}$$

here, $\nu_z^{(T)}(\rho_n)$ is the probability that wavelength z will be free at the arrival time of the data burst, $S_0(T, b, \rho_n)$ is the probability that the wavelength will be unused for the duration of the data burst, and W is the number of wavelength channels on the output link.

Assume that the control burst traverses h nodes and that the offset-time is decremented at each node by an amount equal to a, the total blocking probability BP_{tot} can be expressed as

$$\begin{aligned} BP_{tot} = 1 - ((1 - BP(b, \rho_1, T))(1 - BP(b, \rho_2, T - a)) \\ \dots (1 - BP(b, \rho_h, T - (h-1)a))) \qquad (6.16) \end{aligned}$$

This equation serves as an approximation of the blocking probability for a particular FEC, if that FEC traffic conforms to its negotiated bandwidth share. This is true even in the existence of a non-conforming traffic belonging to other FECs. Results from this equation will be compared to simulation results in the following section.

6.4 Performance Evaluation

In this section, we use results from extensive simulation to demonstrate the performance of the proposed architecture in terms of bandwidth guarantees to each traffic class, as well as loss probability and end-to-end delay. The simulation is performed using a sophisticated discrete-event simulation model based on OPNET.

The network topology used in the simulation is shown in Figure 6.5. The network consists of both edge and core OBS nodes. Internal links in the network model are wavelength-division-multiplexing (WDM) links capable of carrying five separate wavelengths. The transmission speed per wavelength is approximately equal to 2.5 Gbps

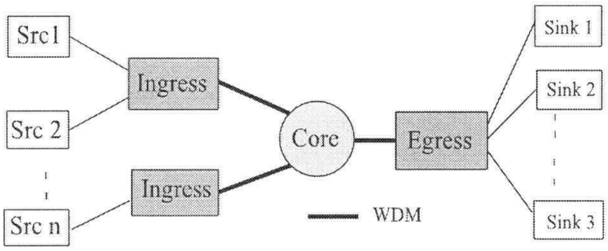


Figure 6.5: Simulation network topology for Section 6.4.

(OC-48). One of the wavelengths is used for control, while the remaining are used for data.

Links from sources to ingress OBS devices and from egress OBS devices to the destination nodes (sinks in Figure 6.5) are single wavelength links with the same transmission speed. All traffic sources used are Poisson sources. The mean packet inter-arrival time is varied in order to study the effect of the network load. The packet size has exponential distribution with a mean equal to 1024 bits. Furthermore, we set the maximum burst size B_{max} to 20480 bits.

We define three traffic classes, or FECs, in the simulation network, namely, FEC 0, FEC 1, and FEC 2. We assign FEC 2 a weight of 3, FEC 1 a weight of 2, and FEC 0 a weight of 1. In addition, we use WFQ as the DB scheduler in the ingress and as the CB scheduler in core nodes. The wavelength scheduling algorithm used here is the LAUC-VF algorithm.

6.4.1 Delay Guarantees

From the delay components explained in Section 6.3, only three components are FEC-dependent, namely, burst assembly delay (d_a), burst queueing delay in ingress node (d_q), and de-assembly delay (d_e). Since the value of B_{max} here is constant for all FECs, the value of d_a is only dependent on the arrival rate, and the value of d_e is equal for all FECs. On the other hand, the value of d_q is dependent only on the FEC weight.

Two simulation experiments were performed. In the first experiment, all traffic sources conform to the contracted rate and total offered load is equal to unity, while in the second experiment traffic belonging to FEC 0 exceed its share. Results for the first experiment are shown in Figure 6.6. Results in Figure 6.6 confirm that the proposed architecture provides delay differentiation according to the FEC weight.

Table 6.1 compares the average and maximum delay values in Figure 6.6 against average and worst case delay values calculated using the analysis in Section 6.3. It can be seen from the table that the average delay values from simulation match those calculated using equations of Subsection 6.3.3. On the other hand, there is a large discrepancy between simulation and analysis when it comes to the worst case delay values. This is expected since the worst-case delay values are intended to serve as an *upper bound* on the delay, and are generally hard to produce using simulation.

Figure 6.7 illustrate that when FEC 0 sends at a higher rate than the reserved rate, the end-to-end delay for FEC 0 increases significantly. This increase is solely due to the d_q delay component. In addition, both FEC 1 and FEC 2 are unaffected by the overload

	Average Delay		**Worst Case Delay**	
	Analysis	**Simulation**	**Analysis**	**Simulation**
FEC 0	3.7E-05	3.6E-05	6.2E-05	3.7E-05
FEC 1	2.8E-05	2.8E-05	4.3E-05	2.9E-05
FEC 2	2.5E-05	2.5E-05	3.6E-05	2.60E-05

Table 6.1: Comparison of average and worst-case delay values between analysis and simulation.

situation. This clearly shows the power of the architecture in terms of traffic isolation and delay control.

6.4.2 Loss Probability

In this subsection, we compare the loss probability of the configured FECs in two situations: when all sources conforming to the reserved rate and when the sources of FEC 0 send more than the configured rate. The loss probability is calculated as the ratio of bits lost to bits sent.

For the first situation, we vary the offered load of each of the configured FECs in proportional to their bandwidth share, while keeping the total network utilization below the value of one. Results shown in Figure 6.8 illustrate a sample path of the loss probability, and the corresponding values calculated from the analysis provided in Section 6.3.4 (Equations 6.15 and 6.16). As shown, the loss probabilities for all FECs are approximately equal to each other with a value that is a function of the total offered load only.

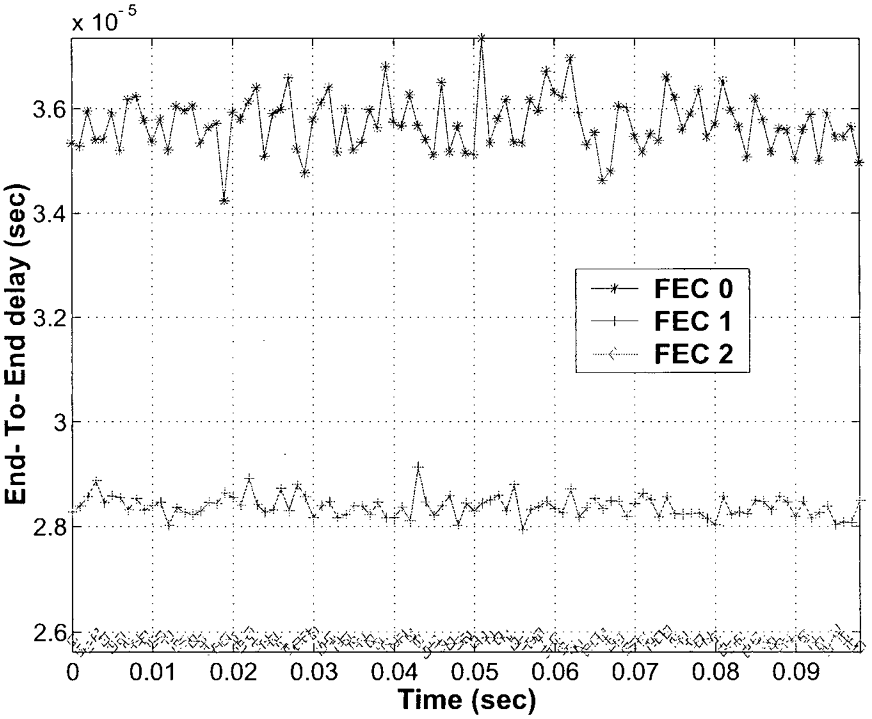


Figure 6.6: End-to-end delay in the case of conforming traffic flows.

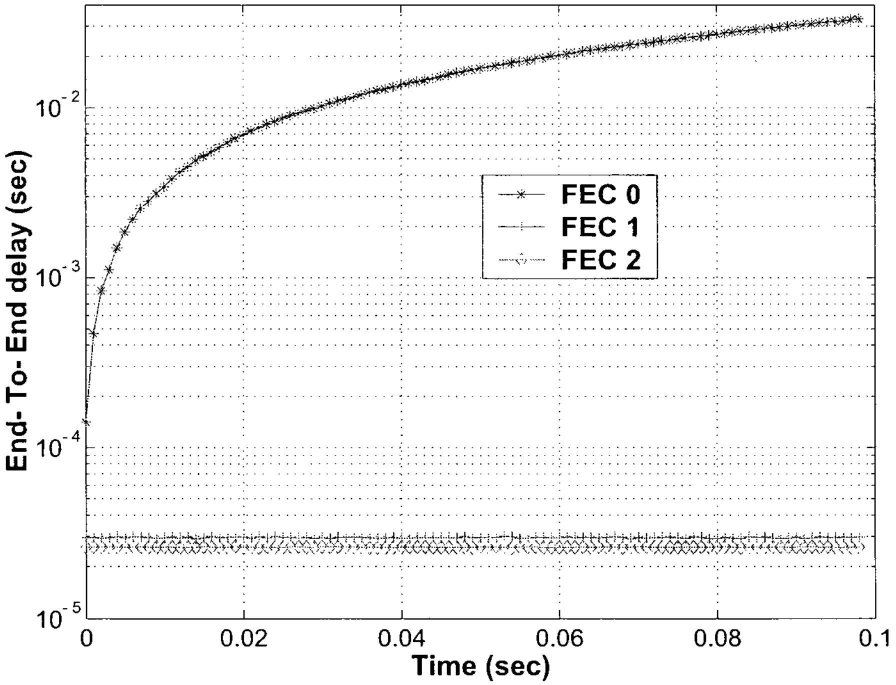


Figure 6.7: End-to-end delay when FEC 0 is non-conforming.

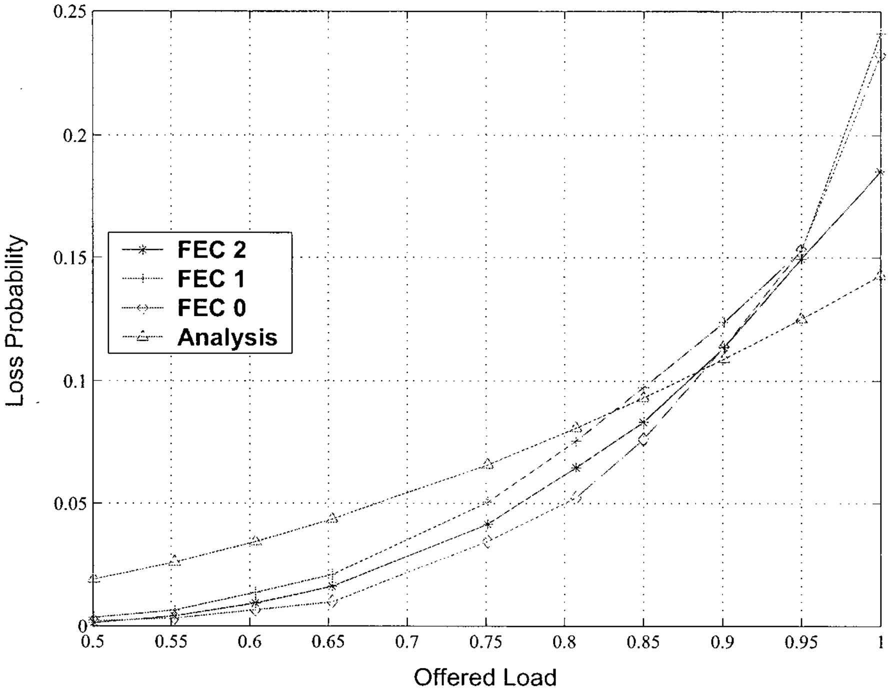


Figure 6.8: Illustration of a sample path of the loss probability.

In addition, Figure 6.8 shows that values obtained from our analysis can be considered a reasonably close approximation to the blocking probability derived from our model.

In the second experiment, we vary the offered load of FEC 0 beyond its reserved bandwidth share. Figure 6.9 shows the resulting loss probability. While FECs 1 and 2 keep the same value of loss probability as in Figure 6.8, the loss probability of FEC 0 is increased significantly. This due to the number of bursts sent exceeding the reserved rate, which causes the corresponding control bursts to suffer longer queueing delay in the core nodes, which in turn causes them to miss the arrival time of their associated data bursts and, consequently, both get dropped.

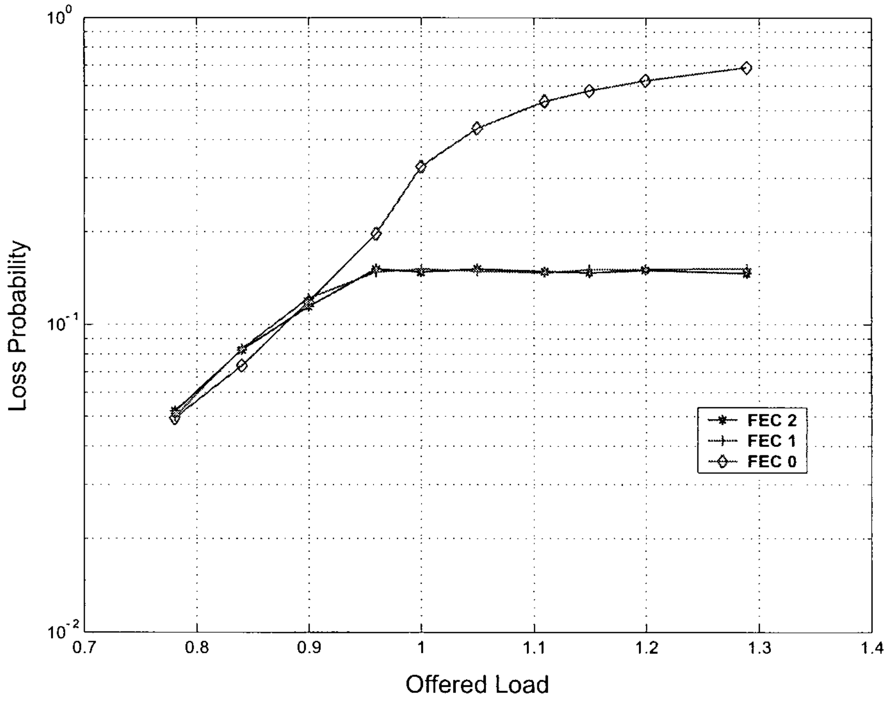


Figure 6.9: High loss probability for FEC 0 (non-conforming).

6.4.3 Bandwidth guarantees

In this subsection, we perform two experiments to study the performance of the proposed architecture in terms of bandwidth guarantees. In the first experiment sources are configured such that the network utilization is equal to unity. Moreover, traffic is shaped at the ingress to conform to the reserved bandwidth share of each FEC. Figure 6.10 shows that the throughput for the three configured FECs is proportional to their configured bandwidth share. Moreover, it can be seen from Figure 6.8 and Figure 6.10 that the throughput for each FEC and the total throughput is in conformance with Equation 6.3. In the second experiment, traffic belonging to FECs 1 and 2 conforms to their negotiated bandwidth share, while sources for FEC 0 send more than the contracted share such that the network utilization exceeds unity. It can be seen from Figure 6.11 that the three FECs get their contracted bandwidth share only.

Results in this section verify that the proposed architecture is capable of providing the contracted bandwidth share, even in the presence of misbehaving sources, due to the fairness and isolation properties of the scheduler used.

6.5 Concluding Remarks

This chapter introduced a new service differentiation architecture with a powerful control-plane scheduling mechanisms for delivering concrete QoS guarantees in LOBS networks. Service differentiation is achieved by a combined use of a burst assembly algorithm and a

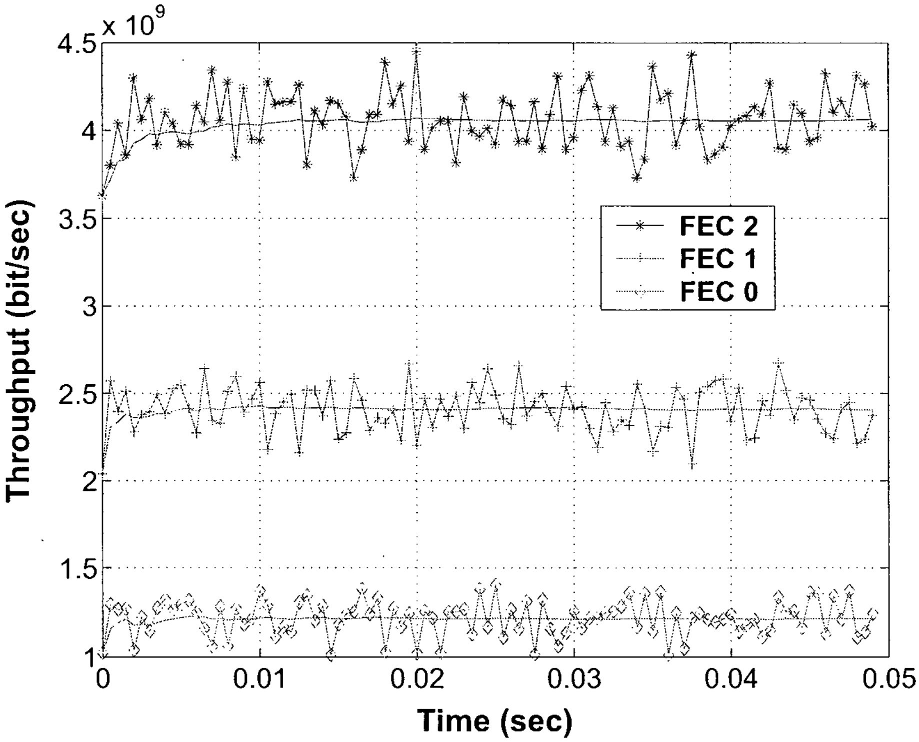


Figure 6.10: Throughput for conforming traffic.

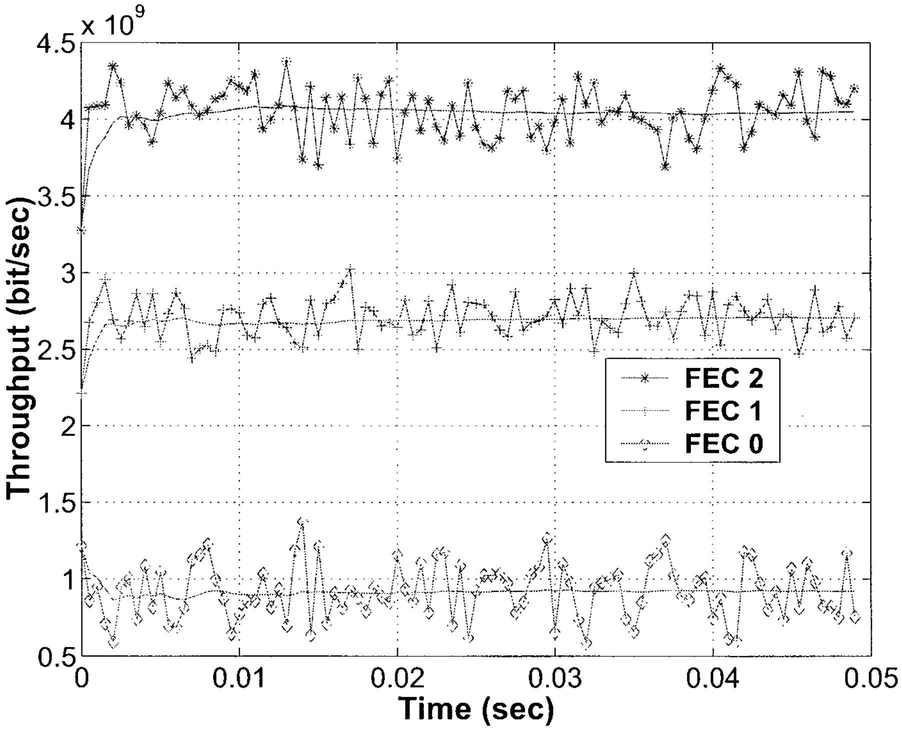


Figure 6.11: Throughput with FEC 0 non-conforming.

fair queueing scheduler. The burst assembly algorithm controls the blocking probability and assembly delay using two parameters: the maximum burst size and the assembly timeout. A FPQ scheduler is used in edge nodes to regulate the access to a wavelength reservation algorithm. In addition, the chapter proposed a novel use of FPQ algorithms in core OBS nodes to provide fair bandwidth allocation. We used the WFQ algorithm as an example to show how the end-to-end delay can be derived. We presented analytical results for the end to end delay and the blocking probability in our architecture. Simulation experiments were conducted to verify the performance of the proposed architecture. Results illustrated that the proposed architecture is able to flexibly support a wide range of service guarantees with regards to throughput, end-to-end delay and packet loss probability.

Chapter 7

Batch Scheduling Algorithms: A Class of Wavelength Schedulers in OBS Networks

7.1 Introduction

In this chapter, we propose a novel class of wavelength schedulers. The proposed schedulers process a batch of bursts together instead of processing them one by one as in the case of previously proposed algorithms.

The rest of this chapter is organized as follows. We introduce the batch scheduling class of schedulers in Section 7.2. Section 7.3 describes the optimal batch scheduler. In Section 7.4, we introduce four new heuristic batch scheduling algorithms. The proposed algorithms are evaluated in Section 7.5 using a discrete event simulation model. We conclude the chapter in Section 7.6.

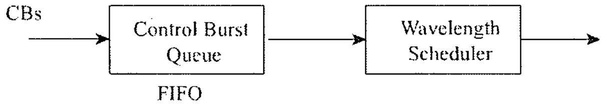


Figure 7.1: Control burst scheduling using greedy algorithms.

7.2 Batch Scheduling

Wavelength schedulers previously proposed in the literature are considered to be greedy algorithms. They are greedy in the sense that they consider every request individually and make the choice that looks best at the moment. Therefore, if the request can be accepted (unblocked), it will be indeed accepted. When a CB arrive to an OBS node employing a greedy wavelength scheduling algorithm, it is inserted into a FIFO queue. When the wavelength scheduler is free, it dequeues the first packet on the queue and processes it. Figure 7.1 describes this process.

In this work, we pose the following question: What if we defer the acceptance of some unblocked requests until we see more requests? The intuition behind this question is that it may be advisable to reject an initially unblocked request if later requests will make better use of wavelength channels.

To further illustrate the idea. Let the acceptance delay be d. Consider the case of $d = \infty$. In this case the system will wait until all reservation requests have been received, then it will select requests that will maximize the utilization. Obviously the case of $d = \infty$ is impractical. Our proposal is to use an acceptable value of d, such that we schedule a batch of reservation requests together, instead of using $d = 0$ as in the case of

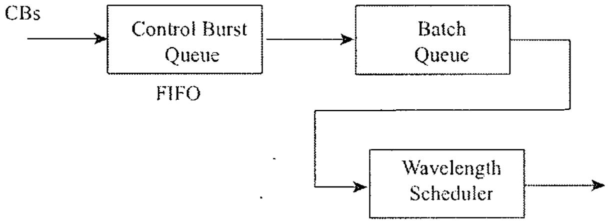


Figure 7.2: Control burst scheduling with batch queue.

greedy algorithms.

To implement the above concept, we modify the burst scheduling process by inserting a queue before the wavelength scheduler and after the control burst queue, we call it the *batch queue*, as shown in Figure 7.2. When a control burst arrives to an OBS node, it gets inserted into the control burst queue. When the wavelength scheduler becomes free, it will wait for a small amount of delay, viz. the *acceptance delay* d, then it moves all bursts that are in the control burst queue to the batch queue, then processes all the CBs in the batch queue together instead of the previously used greedy approach.

The benefit of the batch queue, as will be seen later, is to limit the delay that a control burst can incur during the batch processing.

7.2.1 Definitions

A graph $G = (V, E)$ consists of a finite set V of elements called *vertices*, and a set E of pairs of vertices called *edges*. Let $V(G)$ represent the vertex set of G, and $E(G)$ represent the edge set of G. We call $adj(v)$ the *adjacency set* of vertex v, and we call the pair (v, w) an edge. Clearly $(v, w) \in E$ if and only if $w \in adj(v)$.

A subgraph H of G, denoted $H \subset G$, is a graph with $V(H) \subset V(G)$ and $E(H) \subset E(G)$. The degree of v in the subgraph H for any $v \in V(H)$, denoted $deg(v|H)$, is the number of vertices of H adjacent to v, i.e., $deg(v|H) = |adj(v|H)|$. An undirected graph G is called an *interval graph* if its vertices can be put into one-to-one correspondence with a set of intervals on the real line, such that two vertices are connected by an edge of G if and only if their corresponding intervals have nonempty intersection (have a common point). In other words, G is an interval graph provided that one can assign to each $v \in V(G)$ an interval I_v such that $I_u \cap I_v$ is nonempty if and only if $(u, v) \in E$.

An interval is defined by two points: left point (start of the interval) and right point (end of the interval). Let $l(I)$ and $r(I)$ correspond to the left and right points of interval I respectively. Intervals I_v and I_u overlap if $l(I_v) \leq l(I_u) < r(I_v)$ or if $l(I_u) \leq l(I_v) < r(I_u)$.

A subgraph is called a *clique*, if every pair of vertices in the subgraph is connected by an edge. A clique C is maximal if there is no clique of G that properly contains C as a subset. Let $size(C)$ be the number of vertices in the clique, the maximum clique is the clique of greatest size among all maximal cliques.

7.2.2 Interval Graph Coloring

One of the most important applications of interval graphs is job scheduling. Consider a set of n jobs to be scheduled on k servers. Finding a feasible schedule is equivalent to finding a proper k-coloring of the corresponding interval graph, such that no two adjacent vertices can have the same color (overlapping jobs are assigned to different

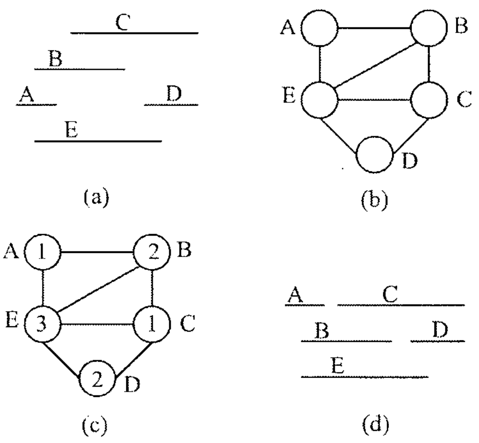


Figure 7.3: Mapping batch scheduling to graph coloring.

servers). Interval graphs and graph coloring problems have been studied intensively in the literature (see [3, 47, 52, 78] and references within).

Batch scheduling in OBS networks can be treated as a subset of the interval graph coloring problem. We have a set of data bursts that we want to assign to a number of wavelength channels. Each data burst corresponds directly to an interval on the real line, and assigning wavelength channels to bursts is similar to assigning colors to intervals, where no two overlapping bursts can be assigned to the same wavelength channel. Figure 7.3 gives an example for mapping batch scheduling to graph coloring. In Figure 7.3a, a batch consisting of five bursts to be assigned to three wavelength channels. Figure 7.3b shows the corresponding interval graph. In Figure 7.3c, the interval graph is colored using three colors: 1, 2 and 3. Figure 7.3d shows the wavelength assignment of the batch.

Each control burst represents a reservation request for an interval. The reservation request for data burst b_i specifies a start time of the reservation $l(b_i)$, corresponding to the arrival time of the data burst, and an end time of the reservation $r(b_i)$. The weight $w(b_i)$ of burst b_i is equal to $(r(b_i) - l(b_i))$.

7.3 An Optimal Batch Scheduling Algorithm (BATCH-OPT)

The batch scheduling problem can be stated as follows: Given k wavelengths and a set of n bursts $\{b_1, .., b_n\}$ in the batch, find a feasible schedule that maximizes the value of the data bursts accepted on the k wavelength channels. Note that n varies from one batch to another depending on the arrivals to the control burst queue. The value of the accepted bursts is simply the sum of their weights. The weight of the burst is defined as the length of its corresponding interval.

A number of algorithms exist in the literature for finding an optimal feasible schedule of intervals while maximizing the total weight of the selected intervals [1, 3, 78]. In [78], the authors formulated the problem as an instance of the interval graph coloring problem as explained next. Given k colors and n intervals, the objective is to maximize the value of the legally colored graph. The problem is then reformulated as the following binary integer linear program (ILP):

$$\begin{aligned} \text{Maximize} \quad & w^T x \\ \text{subject to} \quad & Ax \leq ek, \\ & x \in \{0,1\}, \end{aligned}$$

where w is an n-vector representing the weights of the intervals, x is a binary n-vector in which $x_j = 1$ implies that interval j is to be selected, $x_j = 0$ otherwise. A is $m \times n$ clique matrix [20], in which m is the number of the maximal cliques in the interval graph, where $A_{ij} = 1$ if interval j is in the i^{th} maximal clique, and $A_{ij} = 0$ otherwise. Also, e is an m-vector of 1's. ILP problems are generally NP-hard. However, for our particular problem, the A matrix has the *consecutive ones* property, and A is thus *totally unimodular*[55]. Therefore, we can relax the integrality constraint and solve the problem as a linear program in polynomial time.

This ILP formulation can be used to find an offline optimal schedule of bursts after receiving all the reservation requests. This optimal schedule can serve as a reference against which the performance of other algorithms can be compared.

Although it is tempting to use the same ILP formulation for finding the optimal schedule for a batch of bursts, it is not possible. The reason is that zero or more of the n bursts may not be assignable to one or more of the k wavelength channels due to already existing assignments of bursts belonging to previous batches. This adds extra constraints to the problem.

To find an optimal batch schedule, we formulate our problem as an instance of the *scheduling with non-identical machines problem* [1], in which we associate with each interval (burst) a subset of servers (wavelength channels) on which it can be processed. Some bursts can not be processed on any wavelength channel, because all wavelength channels are busy in the time intervals corresponding to these bursts, these bursts will be dropped and not included in the optimization process. The authors in [1] have given an algorithm that finds the optimal feasible schedule for the non-identical machines problem. The algorithm (BATCH-OPT) works as follows. First, build a list of events corresponding to the start and end times of the intervals being optimized. Then, for each event construct the vertices corresponding to legal combination of intervals and servers at the time of the event. The constructed vertices are used to build a directed graph. It then can be shown that the interval assignments along the longest path in the directed graph represent the optimal interval assignment to servers.

Unfortunately, the above algorithm has an $O(n^{k+1})$ computational complexity, which is high when k is large, rendering this algorithm impractical for use as an online batch scheduler. However, the algorithm is useful for comparison purposes since it serves as an upper bound on the performance of batch scheduling algorithms in terms of blocking probability.

7.4 Heuristic Batch Scheduling Algorithms

Because finding the optimal batch schedule has high computational complexity, we have to turn our attention to heuristic algorithms. In this section we propose a number of batch heuristic algorithms for wavelength channel scheduling in OBS networks. All of the proposed heuristic algorithms are based on performing the following two steps:

1. Impose a certain linear order on the control bursts in the batch queue.

2. Traverse the bursts in this linear order, and assign the corresponding data bursts

to wavelength channels using a greedy void filling wavelength scheduling algorithm.

Although the ordering process imposes a small additional delay on the bursts in the batch queue, this delay is limited since the number of bursts in the batch queue is bounded by the acceptance delay value. If we were to order the bursts directly in the FIFO queue shown in Figure 7.1, then it is possible that a control burst will be delayed past the arrival time of the corresponding data burst because the number of bursts involved in the ordering process will not be bounded, and this shows the importance of the batch queue.

In the following we propose four batch ordering algorithms: *Smallest-Last Vertex Ordering (SLV), Maximal Cliques First Ordering (MCF), Smallest Start-time First Ordering (SSF),* and *Largest Interval First Ordering (LIF).* The first two algorithms rely on the structure of the interval graph corresponding to bursts under scheduling, while the last two algorithms utilize the properties of the individual bursts.

7.4.1 Smallest-Last Vertex Ordering (SLV)

The SLV heuristic algorithm orders the vertices of the interval graph according to the smallest-last ordering [47]. The vertices $v_1, v_2, \ldots, v_n$ of a graph are said to be in smallest last order whenever v_i has minimum degree in the maximal subgraph on the vertices $v_1, v_2, \ldots, v_i$ for all i. Let $\delta(H) = min_{v \in V(H)}\{deg(v|H)\}$ be the minimum degree of graph H. A formal description of the SLV algorithm is given in Algorithm 7.1.

The SLV algorithm works as follows. Let v_n be chosen to have minimum degree in

```
input  : $G$ on $n$ vertices
output: An ordering $v_1, v_2, \ldots, v_n$ of vertices of G,
        where $deg(v_i|H_i) = \delta(H_i)$ for $1 \leq i \leq n$;
initialize $i \leftarrow n, H \leftarrow G$
while $i \geq 1$ do
    Let $v_i$ be a vertex of minimum degree in $H$;
    $H \leftarrow H - v_i$, $i \leftarrow i - 1$;
end
Report sequence $v_1, v_2, \ldots, v_n$;
```

Algorithm 7.1: SLV algorithm

G. For $i = n-1, n-2, \ldots, 2, 1$, let v_i be chosen to have minimum degree in $H_i = \langle V(G) - v_n, v_{n-1}, \ldots, v_{i+1} \rangle$. From the resulting sequence $v_1, v_2, \ldots, v_n$, where v_n has the minimum degree in G, vertex v_1 will be colored first. Which means that the burst corresponding to v_1 will be assigned first to the first available wavelength. The process repeats until each burst is either assigned to a wavelength channel or dropped. The algorithm does not consider the weight of the bursts, and it only attempts to minimize the *number* of dropped bursts.

The intuition behind the SLV algorithm is that if a graph has only a few nodes of very large degree, then coloring these nodes early will avoid the need for using a very large set of colors. This means that assigning bursts which overlap the largest number of other bursts first to wavelength channels will generally lead to minimizing the number of bursts to be dropped. The fact that any smallest-last vertex ordering minimizes the number of colors needed over $n!$ possible ordering is shown in [48].

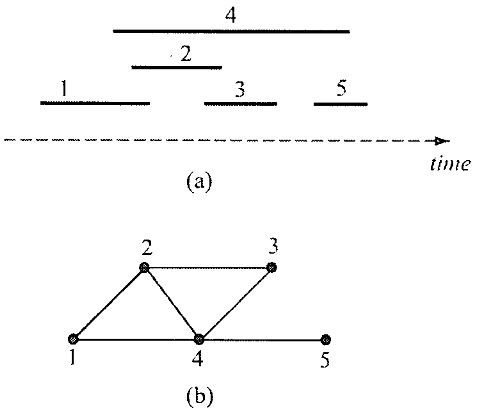


Figure 7.4: Interval graph example.

To illustrate how the SLV algorithm works, we use the simple graph given in Figure 7.4. Figure 7.4a shows a batch of intervals and Figure 7.4b shows the corresponding interval graph. Intervals are numbered in an ascending order according to their finish time. The SLV algorithm selects the node with the smallest degree, which is node 5 in Figure 7.4b. Then it updates the graph and repeats, as shown in Figure 7.5. If there is a tie, the node with the smallest number is chosen. The final sequence is 5, 1, 2, 3, 4. Interval coloring will be done in the reverse order (i.e., 4, 3, 2, 1, 5).

The SLV algorithm has a computational complexity of $O(|E| + |V|)$ [47]. For dense graph the complexity becomes $O(n^2)$. The algorithm requires the input to be a graph. Building the graph from the interval list takes $O(n^2)$ time. The LAUC-VF algorithm require $O(nk \log N)$ time to assign n bursts to wavelength channels. Therefore, the overall complexity of the SLV algorithm is $O(n^2 + nk \log N)$. Since the number of wavelength channels k is in the same order of magnitude as n, the overall complexity of the SLV

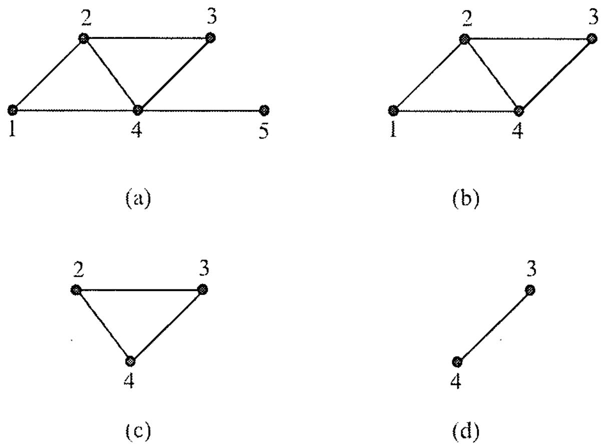


Figure 7.5: Illustration of the SLV algorithm.

algorithm can be reduced to $O(nk \log N)$.

7.4.2 Maximal Cliques First (MCF)

The basic idea behind the MCF algorithm is that since the maximum clique that can be colored is of size k, then our problem is equivalent to that of deleting a subset of intervals such that all remaining cliques are of size k or less. To this end, the MCF algorithm finds all the maximal cliques in the interval graph, then orders them in an increasing order according to time. Let $\{C_1, C_2 ... C_m\}$ be the set of maximal cliques in G ordered such that $C_i \prec C_j$ for $i < j$. The algorithm processes intervals belonging to C_j before intervals belonging to C_i for $j > i$. A formal description of the algorithm is given in Algorithm 7.2.

The MCF algorithm works as follows. It finds the list of maximal cliques in the

```
input  : $G$ on $n$ vertices, $k$
output: A list of vertices $L$
initialize $H \leftarrow G$

while true do
    Let $C$ = list of all maximal cliques in $H$ with size $> k$;
    if $C$ is empty then
        break;
    end
    Let $c_{max}$ = clique with latest occurrence time in $C$;
    $z = size(c_{max}) - k$;
    for $i \leftarrow 0$ to $z - 1$ do
        $v_i$ = vertex with smallest finish time in $c_{max}$;
        $H \leftarrow H - v_i$;
        $c_{max} \leftarrow c_{max} - v_i$;
        $S \leftarrow S \cup v_i$;
    end
end
$L \leftarrow L \cup v_j, \forall v_j$ remaining in $G$;
Append $S$ to the end of $L$;
Report sequence $L$;
```

Algorithm 7.2: MCF algorithm

interval graph with size larger than the number of available wavelengths. The algorithm then finds the clique with the latest occurrence time among this list. Thereafter, the algorithm removes the intervals with the smallest finish time from the maximal clique (and from the graph) such that the size of the maximal clique becomes equal to the number of the available wavelengths. This process is repeated until the size of all maximal cliques in the graph are less than or equal to the number of wavelengths. All vertices remaining in G are appended to the output list first, then the vertices removed by the algorithm are appended to the end of the output list. Figure 7.6 shows how the MCF algorithm is applied to the interval graph of Figure 7.4, for $k = 2$. In Figure 7.6b, the maximal cliques list of the interval graph is shown. The maximal clique that has the greatest finish time has two intervals only, then no updates are required here. The next maximal clique has three intervals, therefore, one interval needs to removed. The interval to be removed is the interval with smallest finish time, which is interval 2. Figure 7.6c and Figure 7.6d shows the resulting interval graph and its maximal cliques list after removing node 2. The resulting node sequence is: 5, 4, 3, 1, 2, and interval 5 will be assigned first.

Finding the maximal cliques in the interval graph requires end points of intervals to be sorted first, which can done in $O(n \log n)$ time. Then we apply the MAXIMAL_CLIQUES algorithm described in Appendix A, which requires $O(n)$ time. The MCF algorithm performs $O(n)$ updates per maximal clique. Since we have $O(n)$ maximal cliques, the complexity of the MCF algorithm is $O(n^2)$. When taking the LAUC-VF into consideration, the overall complexity becomes $O(n^2 + nk \log N)$. Since k is in the same order of

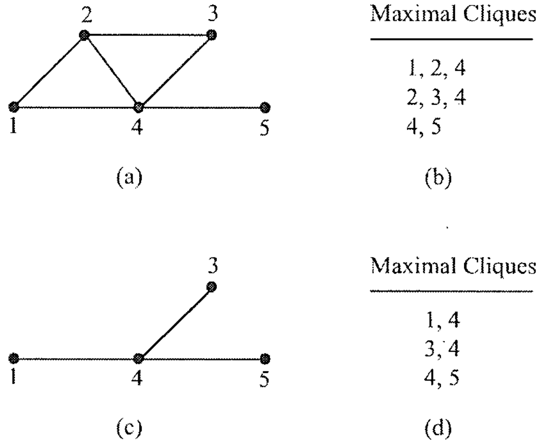


Figure 7.6: Illustration of the MCF algorithm.

magnitude as n, the final complexity reduces to $O(nk \log N)$.

7.4.3 Smallest Start-time First Ordering (SSF)

In SSF ordering, intervals are ordered according to their left most point (start time). Accordingly, a set of intervals $\{i_1, i_2, \dots, i_n\}$ are said to be in SSF ordering if $l(i_1) \leq l(i_2) \leq ... \leq l(i_n)$. According to the SSF algorithm, bursts that start earlier in time are assigned to wavelength channels first. The idea behind the SSF algorithm is to minimize the effect of the offset dispersion on the blocking probability.

Finding the SSF ordering can be done in $O(n \log n)$ time, where n is the number of bursts in the batch queue. Therefore, the overall complexity is dominated by the complexity of the LAUC-VF algorithm, and it becomes $O(nk \log N)$.

7.4.4 Largest Interval First Ordering (LIF)

LIF algorithm orders Intervals according to their weight. The LIF algorithm outputs a sequence of intervals $i_1, i_2, \ldots, i_n$, where $w(i_j) \geq w(i_k)$ for $j \leq k$. Using LIF ordering, bursts are sorted in a descending order according to their size, and the burst with the largest size is assigned first. The rational behind the LIF algorithm can be explained as follows. Our aim is to maximize the number of bits accepted, therefore, if more large bursts are accepted then we have moved closer to our aim. Intuitively, the probability that a large burst will be accepted is higher if we try to assign it to a wavelength channel earlier in the process.

The LIF ordering can be found in $O(n \log n)$ time, where n is the number of bursts in the batch queue. Accordingly, the final complexity will be $O(nk \log N)$.

7.5 Performance Evaluation

This section presents experimental results on the proposed algorithms: SLV, MCF, SSF and LIF, and their comparisons with the optimal batch algorithm (BATCH-OPT), and the greedy LAUC-VF algorithm. Our main focus in the following simulations is the blocking probability.

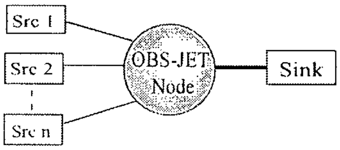


Figure 7.7: Simulation model for Section 7.5.

7.5.1 Simulation Setup

Figure 7.7 shows the simulation model used in this section, which is based on the OPNET simulation tool. It consists of a single OBS node connected to traffic sources and a sink node. Each input link carries two separate wavelengths, one for data and one for control. The output link carries five wavelengths, four for data and one for control, and each wavelength has transmission speed of approximately 2.5 Gbps (OC-48). We assume that sources generate control and data bursts. Sources generate bursts according to Poisson process and the burst size has a mean value equal to B bits. The offset time has a uniform distribution over $[A_{min}, A_{max}]$. Let Δ be the transmission time of 1024 bits on one of the wavelengths, i.e., $\Delta = 1024/2377728000 = 4.3\text{e-}7$ seconds. We express the acceptance delay and the offset time in terms of Δ. In the following simulations, unless otherwise stated, we set the acceptance delay d to an arbitrary value of 100Δ second, which is approximately 40μsec. The greedy wavelength algorithm used in conjunction with the batch algorithms is the LAUC-VF algorithm.

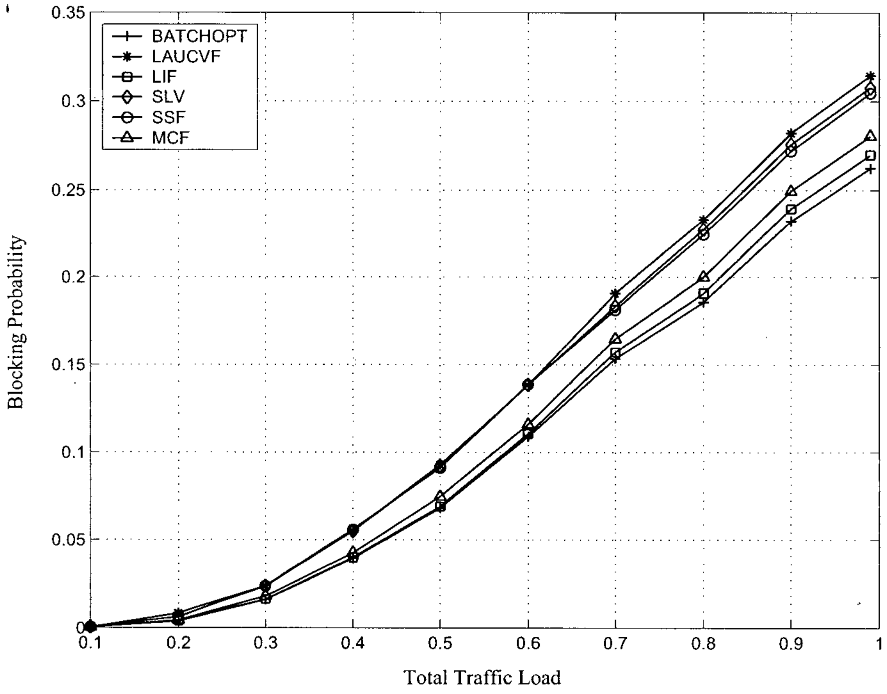


Figure 7.8: Blocking probability vs. offered load in case of exponentially distributed burst sizes.

7.5.2 Blocking Probability vs. Offered Load

In this subsection, we study the performance of the proposed algorithms while varying the OBS node load. We study two scenarios. In the first scenario, we use exponentially distributed burst sizes with mean value $B = 81920$ bits. The values of A_{max} and A_{min} are set to 150Δ seconds and 130Δ seconds, respectively.

Figure 7.8 shows that the performance of the batch algorithms is upper bounded by the optimum batch algorithm as expected, and lower bounded by the greedy LAUC-VF algorithm. The figure shows that the LIF algorithm is the best performing algorithm in this scenario, and its performance is very close to the BATCH-OPT algorithm. Moreover, the figure shows that SLV and SSF algorithms perform slightly better than the LAUC-VF algorithm, however their results are not as significant when compared to the MCF and LIF algorithms.

In the second scenario we use constant burst size equal to 81920 bits. Figure 7.9 shows that the LIF algorithm in this scenario performs exactly as the LAUC-VF algorithm. This result is actually intuitive since all intervals will have the same size, thus no ordering will actually take place. Additionally, the results show that the SSF and SLV algorithms are the best performing algorithms in this scenario. Moreover, the MCF algorithm in this scenario performs significantly better than the LAUC-VF algorithm, but not as good as the SSF and SLV algorithms.

7.5.3 Blocking Probability vs. Offset Time Range

We define the offset time range to be $A_{max} - A_{min}$. In this subsection, we vary the value of A_{min} to study the effect of the offset time range on the performance of the batch scheduling algorithms. We use exponentially distributed burst sizes with mean B = 81920 bits. The value of A_{max} is set to 200Δ, and offered load is set to 99% of the link capacity. The value of A_{min} is varied between 10Δ and 150Δ. Figure 7.10 plots the blocking probability against the offset time range value.

Figure 7.10 illustrates that, generally, the blocking probability increases as the offset time range increases. Additionally, the rate of the increase in the blocking probability

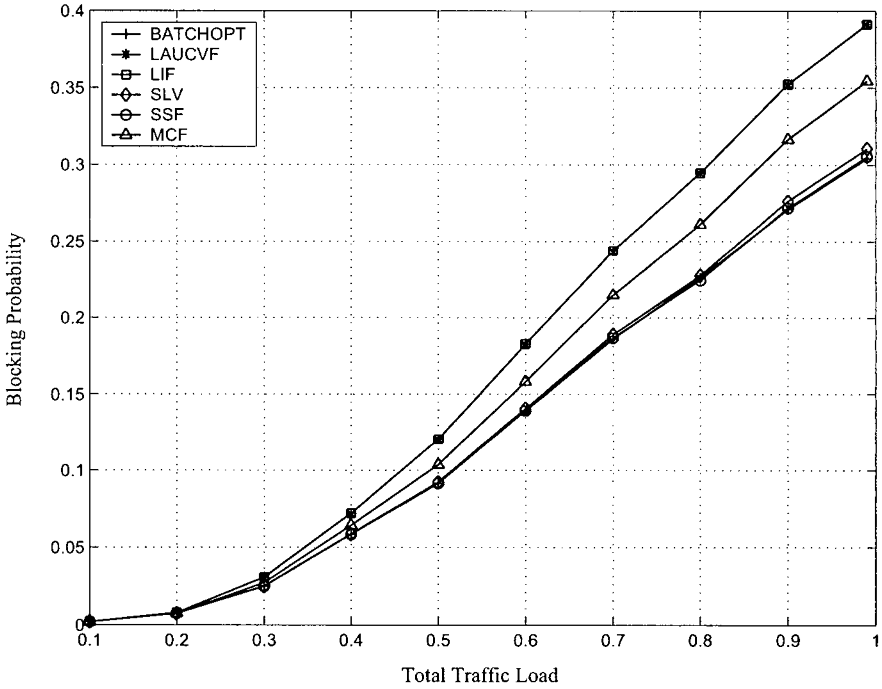


Figure 7.9: Blocking probability vs. offered load in case of constant burst size.

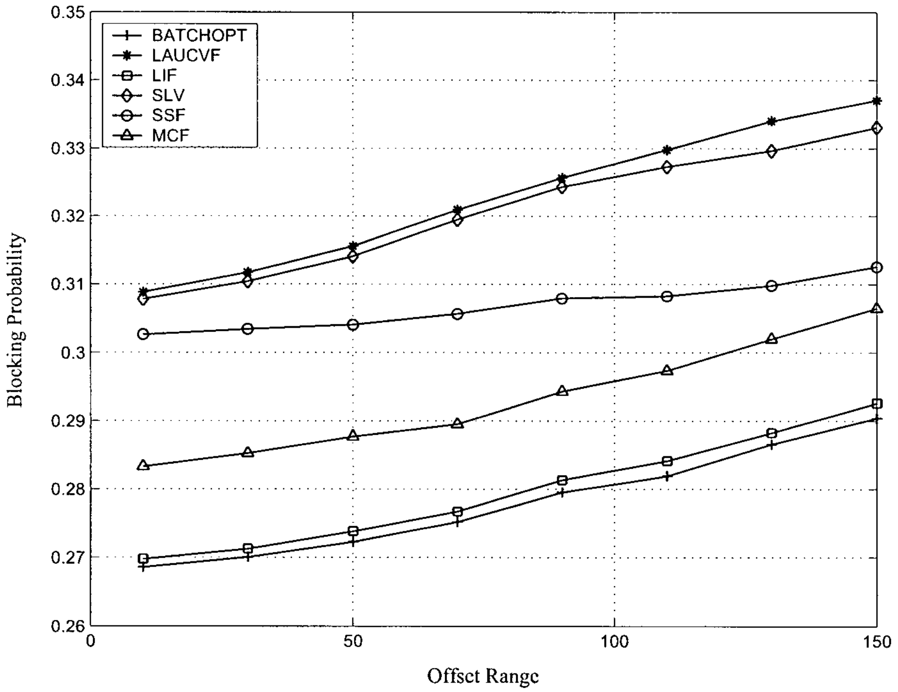


Figure 7.10: Blocking probability vs. offset time range.

is approximately equal for all algorithms, except for the SSF algorithm whose blocking probability increases at a lower rate. This increase is due to the "retro-blocking" phenomena of the JET signaling protocol (refer to Chapter 5), in which, a reservation request can be blocked by another reservation starting after its own time. Obviously this phenomena becomes more significant as the offset time range becomes larger.

The blocking probability of the SSF algorithm increases at a lower rate because it sorts the bursts according to their starting time, therefore, it minimizes the effect of the offset time range.

To evaluate the effect of the offset time range on the performance of the batch scheduling algorithms, we implemented the offline optimal (OPTIMUM) algorithm given in [1] (described in appendix B), and compared the BATCH-OPT algorithm against it. Figure 7.11 re-plots the blocking probability curves for the LAUC-VF, BATCH-OPT algorithms shown in Figure 7.10, in addition to the blocking probability curve for the OPTIMUM algorithm.

Figure 7.11 shows that the performance of the OPTIMUM algorithm is not affected by the offset time range. This is due to the fact that the OPTIMUM algorithm waits until all bursts are received before deciding on the bursts to be rejected, at which time the offset will have minimal effect, because burst arrivals will be equally distributed over time. Furthermore, Figure 7.11 clearly shows that the difference between the performance of the BATCH-OPT algorithm and performance of the OPTIMUM algorithm becomes very small when the offset time range is small (5Δ). This difference starts to increase as the

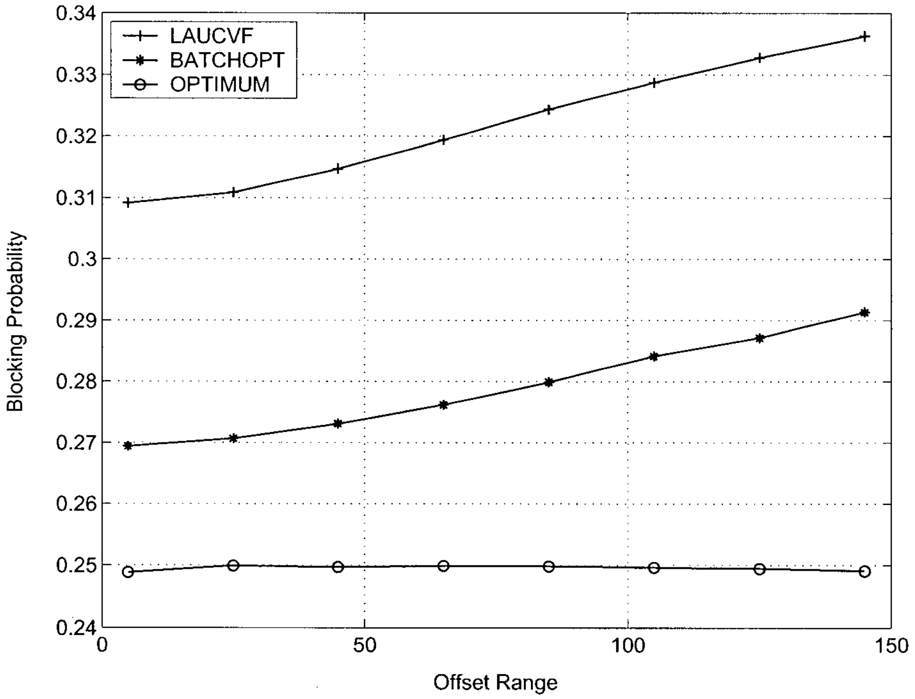


Figure 7.11: Effect of offset time range on the burst blocking probability.

offset time range increases. This can be explained when we note that as the offset time range increases, the probability of overlapping between intervals requested by bursts in the batch queue decreases (refer to Chapter 5). In other words, as the offset time range increases, the requested intervals are more spread out in time. This will decrease the effect of the optimization done by the batch scheduling algorithm and subsequently will lead to higher blocking probability.

7.5.4 Effect of the Acceptance Delay

In this subsection, we investigate the effect of the acceptance delay parameter (d) on the performance of the batch scheduling algorithms. We fix the value of the offset time range to 50Δ and the offered load to 99% of the link capacity. Figure 7.12 and Figure 7.13 plot the blocking probability of the BATCH-OPT algorithm versus the acceptance delay for exponentially distributed burst sizes and constant burst sizes, respectively. The acceptance delay is varied between 10Δ and 150Δ. In addition, the figure shows the blocking probability for the LAUC-VF and OPTIMUM algorithms. It is important to note that the acceptance delay is not a parameter of either the LAUC-VF or the OPTIMUM algorithms, therefore they have constant values in the figure.

Figure 7.12 and Figure 7.13 show that increasing the acceptance delay has the effect of enhancing the performance of the BATCH-OPT algorithm, and bring its performance closer to the performance of the OPTIMUM algorithm. On the other hand, when the acceptance delay is decreased, the blocking probability of the BATCH-OPT algorithm

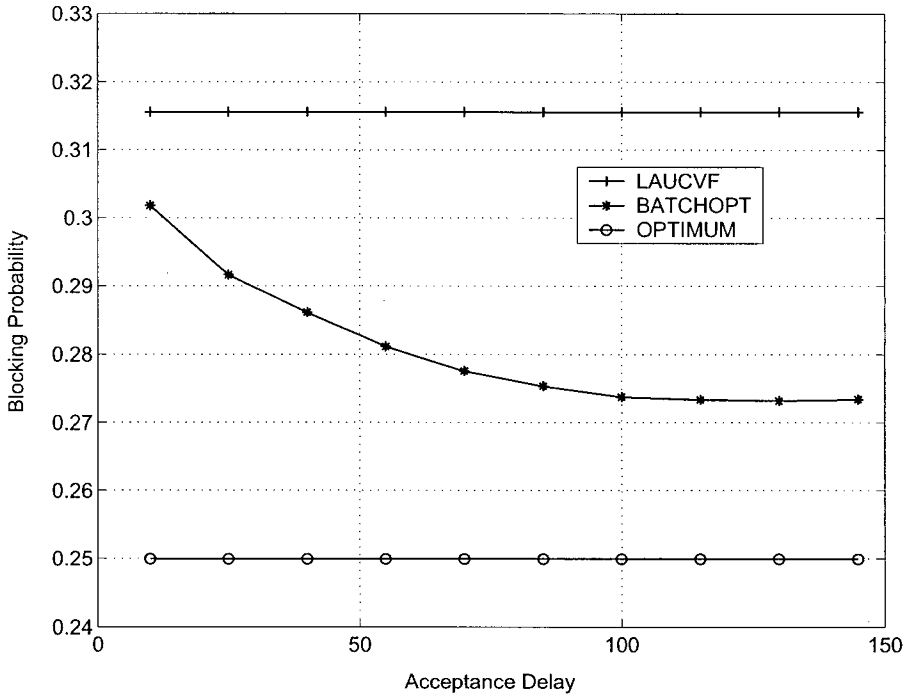


Figure 7.12: Blocking probability vs. acceptance delay with exponentially distributed burst sizes.

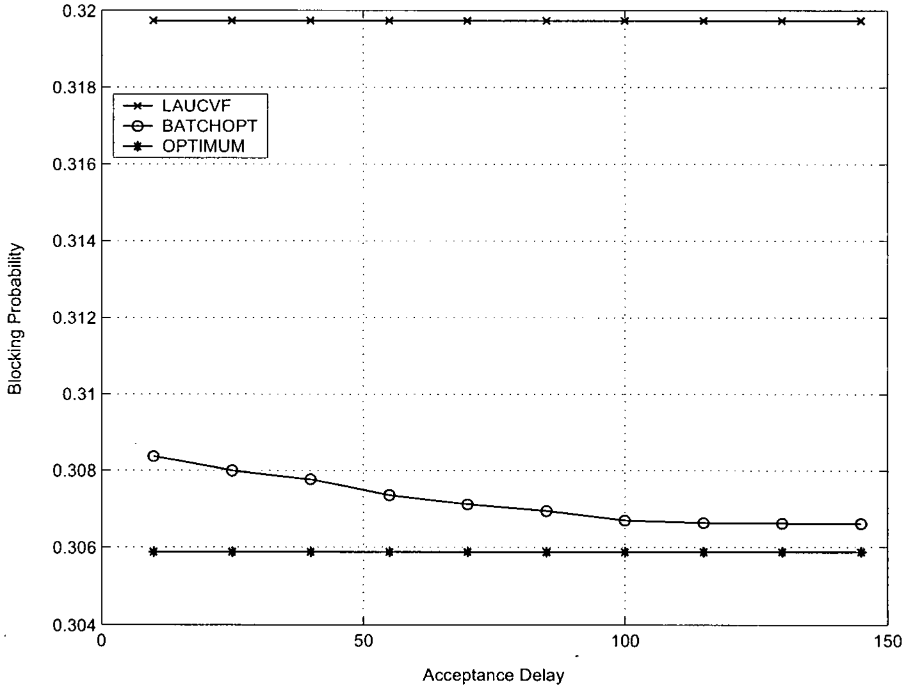


Figure 7.13: Blocking probability vs. acceptance delay with constant burst sizes.

decreases and becomes closer to the LAUC-VF algorithm. This effect is attributed to the number of bursts available in the batch queue. As the acceptance delay increases, the number of bursts involved in the optimization process also increases, which gives better results in terms of the blocking probability.

It should be noted that a control burst will incur a maximum of one acceptance delay per hop, as a result, it is necessary to increase the offset time per hop by an amount equal to the acceptance delay. Accordingly, increasing the acceptance delay will cause an increase in the data burst end-to-end delay. Therefore, there is a tradeoff between the end-to-end delay and the blocking probability. However, Figure 7.12 and Figure 7.13 show that even a small processing delay yields a considerable performance enhancement over greedy algorithms.

Figure 7.14 plots the blocking probability of the BATCH-OPT algorithm when both the offset time range and the acceptance delay are varied. It is clear from the figure that the worst blocking probability value occurs when the offset time range has a large value and in the same time the acceptance delay has a small value. As the offset time range decreases and the acceptance delay increases, the blocking probability decreases. Both the offset time range and the acceptance delay affect the quality of the optimization process carried out by the BATCH-OPT algorithm. Increasing the offset time range decreases the probability that the intervals requested are overlapping and accordingly decreases the effect of the optimization process. On the other hand, increasing the acceptance delay increases the number of bursts involved in the optimization process and subsequently

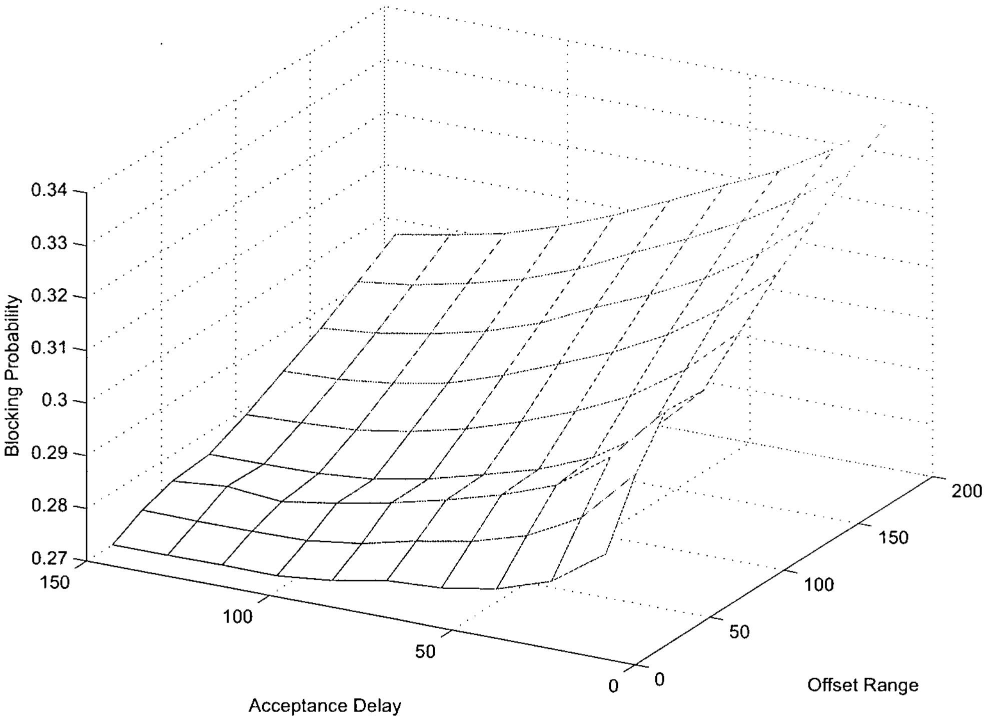


Figure 7.14: Effect of varying the acceptance delay and the offset time range on the blocking probability.

increases the effect the optimization process. Therefore, we conclude from Figure 7.14 that large values of the processing delay are only required when the offset time range has a large value.

7.6 Concluding Remarks

In this chapter, we have introduced a novel class of wavelength scheduling algorithms in OBS networks called batch scheduling algorithms. We have described an optimum batch scheduler that serves as an upper bound for the performance of the batch scheduling algorithms. We have introduced four heuristic batch scheduling algorithms: SLV, MCF, LIF and SSF. The MCF and LIF were shown to be superior in scheduling variable size bursts. The LIF algorithm is simpler than the MCF algorithm, however, the LIF algorithm favors larger bursts. In case of constant size bursts, the SLV and SSF algorithms were shown to perform better than the other algorithms. Moreover, we have studied the effect of the acceptance delay parameter on the performance of the batch scheduling algorithms. We have shown that generally it is possible to obtain a reasonable performance enhancement using the batch scheduling algorithms even when the acceptance delay is small.

Chapter 8

Conclusions and Future Research

We conclude this thesis with a summary of our contributions and suggestions for future work.

8.1 Summary

The main objectives of this thesis were:

- The proposal of new methods for QoS provisioning in OBS networks.
- The development of new wavelength scheduling algorithms to reduce the blocking probability in OBS networks.

The following is a summary of each of the work chapters. In Chapter 3, we presented a survey of all previously proposed wavelength scheduling algorithms in the literature for JET-based OBS network. In addition, we proposed two wavelength scheduling algorithms: Min-AV and Max-NGV. We compared the performance of the newly proposed algorithms to those previously proposed using discrete-event simulation. Simulation results showed that, in general, the best performing algorithm is the Min-AV algorithm.

The advantage of the Min-AV algorithm becomes apparent when the average burst size is small, which shows that the Min-AV algorithm effectively reduces the fragmentation level of wavelength channels. Results in this chapter also appears in [33],[40].

Chapter 4 presented a simple and effective scheme for QoS provisioning in buffer-less OBS networks. The scheme is called PPJET, and it provides service to different traffic classes based on a strict priority order. The PPJET scheme utilizes the JET signaling protocol, in conjunction with a new channel-scheduling algorithm called PLAUC-VF. PPJET minimizes blocking of high priority bursts by preempting reservations for lower priority bursts. Therefore, in PPJET reservation requests for a burst can only be blocked by a reservation request with equal or higher priority, or by a lower priority burst that started service. We evaluate the performance of the PPJET scheme through simulation. Results showed that PPJET can achieve significant performance enhancements over the well known PJET scheme in terms of blocking probability and end-to-end delay. Results of this work have been published in [29],[30].

In Chapter 5 we presented a new analytical model for calculating the blocking probability in JET-based OBS networks. Relationship to the problem of calculating the reservation probability in advance reservation systems was also discussed. The proposed analytical model takes into consideration the effects of the burst offset time and the burst length on the blocking probability. We used a $(M+1)$-state non-homogenous Markov chain to describe the state of an output link carrying M wavelength channels. In addition, we modeled each wavelength channel by a 2-state Markov chain. In the proposed model the offset time is drawn from a specified distribution so that wavelength reservation requests, made before a given time, build up to be a workload whose mean value declines with the reservation starting time. Furthermore, the blocking probability we expressed in terms of first passage time distributions to account for the burst length. To verify its accuracy, the model results were compared with results from a discrete-event simulation model. The proposed model results were found to be in satisfactory agreement with simulation results. This work also appears in [37],[38],[39].

Chapter 6 presented a detailed architecture for providing quantitative quality of service guarantees in LOBS networks. In the proposed architecture, packets are assembled into data bursts based on their respective traffic class at ingress nodes. The burst assembly algorithm employs two parameters to control the burst blocking probability and burst assembly delay. A fair packet queueing algorithm is deployed in each edge node to regulate access to a wavelength scheduler. For LOBS core nodes, we presented a novel approach for scheduling control bursts in the control plane of these nodes to guarantee fair bandwidth allocation. Based on the information provided by the queued control bursts, the core scheduling algorithm creates a virtual queue of data bursts in core nodes, then it selects the eligible control burst to be processed by the wavelength scheduler. We presented analytical results for the delay and the blocking probability in the proposed architecture, and used simulations to demonstrate that the proposed architecture provides accurate and controllable service guarantees in LOBS networks. Results from this work appear in [31],[32].

Chapter 7 proposed a novel class of wavelength scheduling algorithms for OBS networks. The proposed wavelength scheduling algorithms schedule a batch of data bursts together instead of scheduling them one by one. Previously proposed wavelength scheduling algorithms are considered greedy algorithms in the sense that they consider every reservation request individually, and make the choice that looks best at the moment. On the other hand, when the wavelength scheduler becomes free, a batch scheduling algorithm will wait for a small amount of time equal to the acceptance delay parameter, then it will process all the reservations requests in the batch queue together instead of the previously used greedy approach. This acceptance delay enables the batch scheduling algorithms to accept the requests that will maximize the utilization of wavelength channels. We clearly showed a relation between the problem of batch scheduling and the interval graph coloring problem. Based on this relation, we described an optimal batch scheduler that serves as an upper bound on the performance of batch scheduling algorithms. Furthermore, we introduced four heuristic batch scheduling algorithms. The performance of the proposed algorithms was evaluated using a discrete-event simulation model. Simulation results suggested that batch schedulers could decrease the the blocking probability by 25% compared the best previously known wavelength scheduling algorithm. This work has been published in [34],[35],[36].

In conclusion, among the main contributions of this thesis, we proposed:

- Two new methods for QoS provisioning in OBS networks.

- An analytical model for calculating the blocking probability in OBS networks.

- Several wavelength scheduling algorithms that are able to reduce the blocking probability in OBS networks.

8.2 Future Work

We present below a list of research problems that can be investigated as possible extensions for research work reported in this thesis.

1. The intuition behind the work in Chapter 7 was that it may be advisable to reject an initially unblocked request if later requests will make better use of wavelength channels. Our approach to the problem was delaying the acceptance or rejection of a request until we see more requests. A different approach to the problem is possible. If we assume that the arrival process is known, for example non-homogenous Poisson process as in the case of Chapter 5, then we could develop a selective burst acceptance mechanism, with the objective of maximizing the utilization of wavelength channels, while eliminating the need for the acceptance delay. In this case we only know the expectation of the future arrivals. A model that describes the system evolution will need to be developed. In addition, a set of decision rules will need to be devised.

2. A number of new differentiated service mechanisms in OBS networks can be devised based on algorithms discussed in Chapter 7. In particular, in Chapter 7 we have considered the weight of a burst to be equal to its size, however, preferential treat-

ment of bursts could be achieved by assigning to the burst a weight that expresses its class of service. Accordingly, maximizing the weight of the accepted bursts would translate to accepting bursts from higher classes of service. The most important components of this work would be selecting the weight assignment scheme and the selection of the optimization algorithm.

3. The Transmission Control Protocol (TCP) is one of the dominant transport protocols today. The TCP protocol has unique features, e.g. congestion control and retransmission, that will certainly be affected by the OBS network characteristics. Therefore, a fruitful area of future research would be to study the influence of burst assembly algorithms, offset-time, OBS signaling protocols on TCP's performance. It would be particularly interesting to study the effect of mechanisms proposed in this thesis on TCP.

4. Work done in this thesis has been evaluated using a highly detailed discrete-event simulation model. Therefore, network topologies used were limited in dimension. It would be interesting to simulate the proposed mechanisms in a network with large number of nodes, large number of wavelengths and a sophisticated topology. This is indeed a challenging task as the number of events in such network would be very large to the degree that the simulation running time would be prohibitively long. One solution would be to redesign the simulation model to run on parallel architectures.

Bibliography

[1] M. Arkin and E. Silverberg. Scheduling jobs with fixed start and end times. *Discrete Applied Mathematics*, 18(1):1–8, November 1987.

[2] H. Cankaya, S. Charcranoon, and T.S. El-Bawab. A preemptive scheduling technique for OBS networks with service differentiation. In *Proceedings of IEEE Global Telecommunications Conference- Globecom'03*, pages 2704–2708, San Francisco, December 2003.

[3] M. C. Carlisle and E. L. Lloyd. On the K-coloring of intervals. *Discrete Applied Mathematics*, 59(3):225–235, May 1995.

[4] M. Casoni, E. Luppi, and M.L. Merani. Impact of assembly algorithms on end-to-end performance in optical burst switched networks with different QoS classes. In *3rd International Workshop on Optical Burst Switching*, San Jos, December 2004.

[5] Y. Chen, M. Hamdi, and D. Tsang. Proportional QoS over OBS networks. In *Proceedings of IEEE Global Telecommunications Conference- Globecom'01*, pages 1510–1514, San Antonio, November 2001.

[6] I. Chlamtac, A. Ganz, and G. Karmi. Purely optical networks for terabit communication. In *Proceedings of IEEE INFOCOM 1989*, volume 3, pages 887–896, Washignton, DC, April 1989.

[7] R. L. Cruz. A calculus for network delay, Part I: Network elements in isolation. *IEEE Transactions on Information Theory*, 37(1):114–131, January 1991.

[8] A. Demers, S. keshav, and S. Shenkar. Analysis and simulation of a fair queueing algorithm. In *Proceedings of ACM SIGCOMM'89*, Austin, September 1989.

[9] P. De Dobbelaere, K. Falta, L. Fan, S. Gloeckner, and S. Patra. Digital MEMS for optical switching. *IEEE Communications Magazine*, 40(3):88–95, March 2002.

[10] K. Dolzer, C. Gauger, J. Spth, and S. Bodamer. Evaluation of reservation mechanisms for optical burst switching. *AE International Journal of Electronics and Communications*, 55(1):18–25, January 2001.

[11] J. Edmonds and R. M. Karp. Theoretical improvements in algorithmic efficiency for network flow problems. *Journal of the ACM*, 19(2):248–264, April 1972.

[12] T. S. El-Bawab and J. D. Shin. Optical packet switching in core networks: between vision and reality. *IEEE Communications Magazine*, 40(9):60–65, September 2002.

[13] J. M. Elmirghani and H. T. Mouftah. Technologies and architectures for scalable dynamic dense WDM networks. *IEEE Communications Magazine*, 38(2):58–66, February 2000.

[14] L. Berger et al. Generalized MPLS (GMPLS) signaling functional description. *RFC 3471*, January 2003.

[15] R. Braden et al. Integrated services in the Internet architecture: an overview. *RFC 1633*, June 1994.

[16] S. Blake et al. An architecture for differentiated services. *RFC 2475*, December 1998.

[17] A. Ge, F. Callegati, and L. Tamil. On optical burst switching and self-similar traffic. *IEEE Communications Letters*, 4(3):98–100, March 2000.

[18] N. Ghani, S. Dixit, and T. S. Wang. On IP over wdm integration. *IEEE Communications Magazine*, 38(3):72–83, March 2000.

[19] S. Golestani. A self-clocked fair queueing scheme for broadband applications. In *Proceedings of IEEE INFOCOM'94*, pages 636–646, Toronto, April 1994.

[20] M. C. Golumbic. *Algorithmic Graph Theory and Perfect Graphs.* Academic Press, 1980.

[21] P. Goyal, H.M. Vin, and H. Cheng. Start-time fair queueing: A scheduling algorithm for integrated services packet switching networks. *IEEE/ACM Transactions on Networking*, pages 690–704, October 1997.

[22] J. J. Harms and J. W. Wong. Performance evaluation of scheduling algorithms for bandwidth reservation. In *Teletraffic and Datatraffic in a Period of Change, ITC 13*, Denmark, June 1991.

[23] B. Hirosaki, K. Emura, S. Hayano, and H. Tsutsumi. Next generation optical networks as value-creation platform. *IEEE Communications Magazine*, 41(9):65–71, September 2003.

[24] G. Hu, K. Dolzer, and C.M. Gauger. Does burst assembly really reduce the self-similarity. In *Proceedings of Optical Fiber Communication Conference-OFC'03*, volume 86, pages 124–126, Atlanta, March 2003.

[25] G. Hudek and D. Muder. Signaling analysis for a multi-switch all-optical network. In *Proceedings of IEEE International Conference on Communications-ICC'95*, Seattle, June 1995.

[26] D. K. Hunter and I. Andronovic. Approaches to optical Internet packet switching. *IEEE Communications Magazine*, 38(9):116–122, September 2000.

[27] 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 *Proceedings of IEEE Global Telecommunications Conference- Globecom'02*, Taipei, November 2002.

[28] M. Izal and J. Aracil. On the influence of self similarity on optical burst switching traffic. In *Proceedings of IEEE Global Telecommunications Conference- Globecom'02*, pages 2320–2324, Taipei, November 2002.

[29] A. Kaheel and H. Alnuweiri. A strict priority scheme for quality-of-service provisioning in optical burst switching networks. In *Proceedings of IEEE Symposium on Computers and Communications-ISCC'03*, volume 1, pages 16–21, Turkey, June 2003.

[30] A. Kaheel and H. Alnuweiri. Priority scheme for supporting quality of service in optical burst switching networks. *OSA Journal of Optical Networking*, 3(9):707–719, September 2004.

[31] A. Kaheel and H. Alnuweiri. Quantitative QoS guarantees in labeled optical burst switching networks. *Submitted to IEEE/ACM Transactions on Networking*, September 2004.

[32] A. Kaheel and H. Alnuweiri. Quantitative QoS guarantees in labeled optical burst switching networks. In *Proceedings of Global Telecommunications Conference-GLOBECOM'04*, volume 3, pages 1747–1753, Texas, 29 November-3 December 2004.

[33] A. Kaheel and H. Alnuweiri. Wavelength scheduling algorithms in buffer-less optical burst switching networks. In *Proceedings of IASTED International Multi-Conference on Wireless and Optical Communications-WOC'04*, volume 1, pages 744–748, Canada, July 2004.

[34] A. Kaheel and H. Alnuweiri. Batch scheduling algorithms: a class of wavelength schedulers in optical burst switching networks. In *Proceedings of IEEE International Conference on Communications-ICC'05*, Korea, June 2005.

[35] A. Kaheel and H. Alnuweiri. Batch scheduling algorithms for optical burst switching networks. In *IFIP-TC6 International Conference on Networking-Networking'05*, Ontario, May 2005.

[36] A. Kaheel and H. Alnuweiri. Wavelength scheduling algorithms with delayed acceptance for optical burst switching networks. In *Canadian Conference on Electrical and Computer Engineering-CCECE'05*, Saskatchewan, May 2005.

[37] A. Kaheel, H. Alnuweiri, and F. Gebali. A new analytical model for computing blocking probability in optical burst switching networks. *Submitted to IEEE Journal on Selected Areas in Communications*, September 2003.

[38] A. Kaheel, H. Alnuweiri, and F. Gebali. Analytical evaluation of blocking probability in optical burst switching networks. In *Proceedings of IEEE International Conference on Communications-ICC'04*, volume 3, pages 1548–1553, France, June 2004.

[39] A. Kaheel, H. Alnuweiri, and F. Gebali. A new analytical model for computing blocking probability in optical burst switching networks. In *Proceedings of IEEE International Symposium on Computers and Communications-ISCC'04*, volume 1, pages 264–269, Egypt, 28 June-1 July 2004.

[40] A. Kaheel, T. Khattab, A. Mohamed, and H. Alnuweiri. Quality-of-service mechanisms in IP-over-WDM networks. *IEEE Communications Magazine*, 40(12):38–44, December 2002.

[41] T. Khattab, A. Mohamed, A. Kaheel, and H. Alnuweiri. Optical packet switching with packet aggregation. In *Proceedings of International Conference on Software, Telecommunications and Computer Networks-SoftCOM'02*, volume 1, pages 741–746, Italy, October 2002.

[42] M. Kijima. *Markov Processes For Stochastic Modeling.* Chapman and Hall, 1997.

[43] E.L. Lawler. *Combinatorial Optimization: Networks and Matroids.* Holt, Rinehart and Winston, 1976.

[44] W. E. Leland, M. S. Taqqu, W. Willinger, and D. V. Wilson. On the self-similar nature of ethernet traffic (extended version). *IEEE/ACM Transactions on Networking*, 2(1):1–15, February 1994.

[45] Y. Liang, K. Liao, J. W. Roberts, and A. Simonian. Queueing models for reserved set up telecommunications services. In *Teletraffic Science for New Cost-Effective Systems, Networks and Services, ITC 12*, Italy, June 1988.

[46] H. Luss. A model for reservations for systems for large scale conferencing services. *The Journal of the Operational Research Society*, 31(3):239–245, March 1980.

[47] D. W. Matula. Smallest-last ordering and clustering and graph coloring algorithms. *Journal of the ACM*, 30(3):217–427, July 1983.

[48] D. W. Matula, G. Marble, and J. D. Isaacson. *Graph Theory and Computing*, chapter Graph coloring algorithms, pages 109–122. Academic Press, 1972.

[49] P. Mitra and S. Jason. Nonlinear limits to the information capacity of optical fibre communications. *Nature*, 411:1027–1030, June 2001.

[50] A. Mohamed, A. Kaheel, T. Khattab, and H. Alnuweiri. Evaluation of optical packet switch as edge device using opnet modeler. available online: "http://www.opnet.com/opnetwork2002", August 2002.

[51] C. Murthy and M. Gurusamy. *WDM Optical Networks: Concepts, Design, and Algorithms*. Prentice Hall, 2002.

[52] S. Olariu. An optimal greedy heuristic to color interval graphs. *Information Processing Letters*, 37(1):21–25, January 1991.

[53] 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*, 39(3):128–135, March 2001.

[54] OPNET technologies inc. Web page: `http://www.opnet.com/`, August 2004.

[55] C. H. Papadimitriou and K. Steiglitz. *Combinatorial Optimization : Algorithms and Complexity*. Dover Publications, 1998.

[56] G.I. Papadimitriou, C. Papazoglou, and A.S. Pomportsis. Optical switching: Switch fabrics, techniques, and architectures. *Journal of Lightwave Technology*, 21(2):384–405, February 2003.

[57] Vern Paxson and Sally Floyd. Wide area traffic: The failure of Poisson modeling. *IEEE/ACM Transactions on Networking*, 3(3):226–244, June 1995.

[58] C. Qiao. Labeled optical burst switching for IP-over-WDM integration. *IEEE Communications Magazine*, pages 104–114, December 2000.

[59] C. Qiao and M. Yoo. Optical burst switching (OBS) - a new paradigm for an optical internet. *Journal of High Speed Networks*, 8(1):69–84, January 1999.

[60] C. Qiao and M. Yoo. Choices, features, and issues in optical burst switching. *Optical Network Magazine*, 1(2):36–44, April 2000.

[61] R. Ramaswami and K. N. Sivarajan. *Optical Networks: A Practical Prespective, second edition.* Morgan Kaufmann Publications, 2002.

[62] M. Renaud, F. Masetti, C. Guillemot, and B. Bostica. Network and system concepts for optical packet switching. *IEEE Communications Magazine*, 35(4):96–102, April 1997.

[63] Insight Research. DWDM, SONET, and photonics: The emerging all-optical network 2001-2006. Technical report, http://www.insight-corp.com/, May 2001.

[64] E. Rosen, A. Viswanathan, and R. Callon. Multiprotocol label switching architecture. *RFC 3031*, January 2001.

[65] D. Stephens, J. Bennett, and H. Zhang. Implementing scheduling algorithms in high speed networks. *IEEE Journal on Selected Areas in Communications*, 17(6):1145–1158, June 1999.

[66] J. Turner. Terabit burst switching. *Journal of High Speed Networks*, 8(1):3–16, March 1999.

[67] S. Verma, H Chaskar, and R. Ravikanth. Optical burst switching: A viable solution for terabit ip backbone. *IEEE Network Magazine*, 14(6):48–53, November 2000.

[68] J. T. Virtamo. A model of reservation systems. *IEEE Transactions on Communications*, 40(1):109–118, January 1992.

[69] V. Vokkarane and J. Jue. Prioritized burst segmentation and composite burst assembly techniques for QoS support in optical burst-switched networks. *IEEE Journal of Selected Areas of Communications (JSAC)*, 21(7):1198–1209, September 2003.

[70] V. M. Vokkarane and J.P. Jue. Prioritized routing and burst segmentation for QoS in optical burst switched networks. In *Proceedings of Optical Fiber Communication Conference- OFC'02*, pages 221–222, Anaheim, March 2002.

[71] H. L. Vu and M. Zukerman. Blocking probability for priority classes in optical burst switching networks. *IEEE Communications Letters*, 6(5):214–216, May 2002.

[72] J. Y. Wei and R. I. McFarland. Just-in-time signaling for WDM optical burst switching networks. *Journal of Lightwave Technology*, 18:2019–2037, December 2000.

[73] W. Willinger, M. S. Taqqu, R. Sherman, and D. V. Wilson. Self-similarity through high-variability: Statistical analysis of Ethernet LAN traffic at the source level. *IEEE/ACM Transactions on Networking*, 5(1):71–86, April 1997.

[74] Y. Xiong, M. Vandenhoute, and C. Cankaya. Control architecture in optical burst-switched WDM networks. *IEEE Journal on Selected Areas in Communications*, 18:1838–1851, October 2000.

[75] J. Xu, C. Qiao, J. Li, and G. Xu. Efficient channel scheduling algorithms in optical burst switched networks. In *Proceedings of IEEE INFOCOM 2003*, San Francisco, March 2003.

[76] L. Yang, Y. Jiang, and S. Jiang. A probabilistic preemptive scheme for providing service differentiation in OBS networks. In *Proceedings of IEEE Global Telecommunications Conference- Globecom'03*, pages 2689–2693, San Francisco, December 2003.

[77] M. Yang, S. Q. Zheng, and D. Verchere. A QoS supporting scheduling algorithm for optical burst switching dwdm networks. In *Proceedings of IEEE Global Telecommunications Conference, Globecom'01*, San Antonio, November 2001.

[78] M. Yannakakis. The maximum K-colorable subgraph problem for chordal graphs. *Information Processing Letters*, 24:133–137, January 1987.

[79] M. Yoo, M. Jeong, and C. Qiao. A high-speed protocol for bursty traffic in optical networks. *In SPIE Proceedings, All Optical Networking: Architecture, Control and Network Issues*, 3230:79–90, November 1997.

[80] M. Yoo and C. Qiao. A new optical burst switching protocol for supporting quality of service. *In SPIE Proceedings, All Optical Networking: Architecture, Control and Management Issue*, 3531:396–405, November 1998.

[81] M. Yoo, C. Qiao, and S. Dixit. Optical burst switching for service differentiation in the next generation optical Internet. *IEEE Communications Magazine*, 39(2):98–104, February 2001.

[82] X. Yu, Y. Chen, and C. Qiao. Performance evaluation of optical burst switching with assembled burst traffic input. In *Proceedings of IEEE Global Telecommunications Conference- Globecom'02*, pages 2330–2334, Taipei, November 2002.

Appendix A

Algorithm for Finding Maximal Cliques in Interval Graphs

In this Appendix we describe an algorithm that we developed for finding maximal cliques in interval graphs. The algorithm is called *MAXIMAL_CLIQUES*, and it is formally described in Algorithm A.1.

The MAXIMAL_CLIQUES algorithm works as follows. The end points of the presented intervals are sorted in ascending order. The algorithms run through the list of end points and inspect point by point. If the inspected point (e) is a left point, then interval v corresponding to point e is added to a set c. If the point is a right point, and the previous inspected point is a left point, then this means that intervals contained in c form a maximal clique. In this case c is added to C, which is the list of maximal cliques, and v is removed from c because the interval has ended. Finally, if point e is a right point, and the previous inspected point was a right point, then point v is simply removed from c because the interval has ended.

The MAXIMAL_CLIQUES algorithm required $O(n)$ time if the end points are presorted, and $O(n \log n)$ if the end points are not sorted.

```
input  : List of intervals $v_1, v_2, \ldots, v_n$
output: List of maximal cliques $C$
initialize  $e \leftarrow -1,\ c \leftarrow \phi,\ C \leftarrow \phi$

Let $\{e_1, e_2, \ldots, e_{2n}\}$ be a list of end points in ascending order;

for $i \leftarrow 1$ to $2n$ do
    $e_{prev} \leftarrow e$;
    $e \leftarrow e_i$;
    if $e$ is a left point then
        Let $v$ = interval corresponding to $e$; $c \leftarrow c \cup v$;
    end
    if ($e$ is a right point) and ($e_{prev}$ is a left point) then
        Add $c$ to $C$;
        Let $v$ = interval corresponding to $e$; $c \leftarrow c \setminus v$;
    end
    if ($e$ is a right point) and ($e_{prev}$ is a right point) then
        Let $v$ = interval corresponding to $e$; $c \leftarrow c \setminus v$;
    end
end

Report $C$;
```

Algorithm A.1: MAXIMAL_CLIQUES algorithm

Appendix B

Offline Optimal Scheduling Algorithm

Authors in [1] gave an algorithm for finding an optimal feasible schedule of intervals while maximizing the total weight of the selected intervals. The scheduling problem is formulated as instance of *minimum-cost flow* problem [43].

1. Represent the problem in the form of an interval graph.
2. Identify all maximal cliques $q_1, \ldots, q_r$ of the interval graph. Order the maximal cliques according to time.
3. Construct a directed graph G as follows:
   (a) Create nodes $v_0, \ldots, v_r$ and arcs (v_i, v_{i-1}) for $i = 1, \ldots, r$. The costs on these arcs equal 0 and their capacities are infinite (each arc represents a clique).
   (b) For each interval, if I_i is in cliques $q_j, \ldots, q_{j+1}$, add an arc (v_{j-1}, v_{j+1}) of cost w_i and capacity 1.
   (c) For each clique j that is not of maximum size, we add an arc (v_{j-1}, v_j) of cost

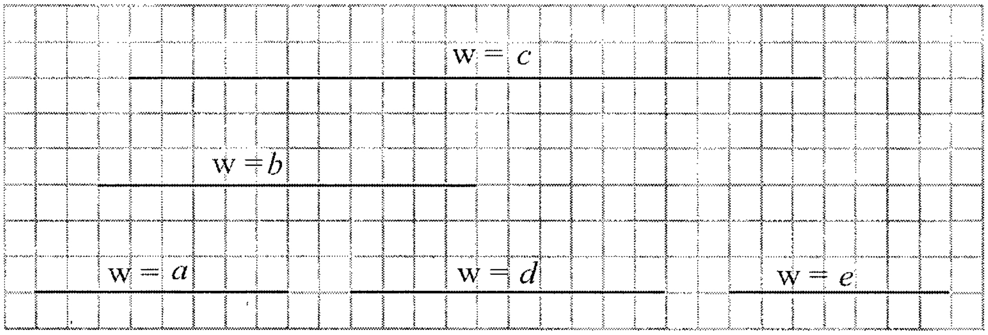
0 and capacity equal to (maximum-clique-size - size-of-q_j).

(d) Consider node v_0 to be a source S, and v_r to be a sink T.

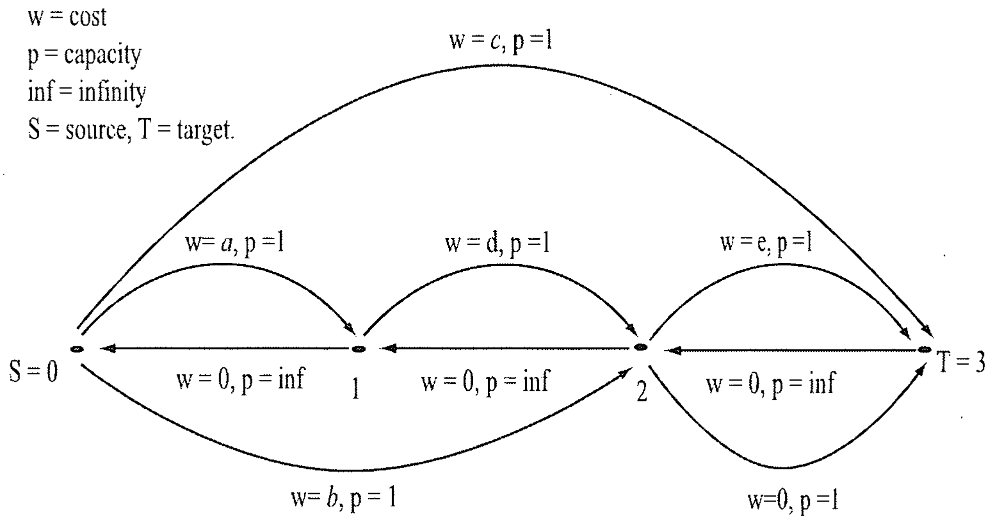
4. We require a flow from S to T of (maximum-clique-size - k), where k is the number of available wavelengths.

An optimal solution to this minimum cost flow problem is in fact an optimal solution to our original problem. If an arc corresponding to interval I_i has a non-zero flow on it in the min-cost flow solution, then we do not process this interval. Thus, we do not process a subset of intervals such that all remaining cliques are of size k or less. This subset is of smallest possible value since it corresponds to a min-cost flow solution. To find the min-cost flow we used the algorithm described in [11]. An example for this construction is shown in Figure B.1.

The maximal cliques in the graph can be found using the algorithm described in Appendix A in $O(n \log n)$, where n is the number of intervals. The complexity of the minimum cost flow algorithm described in [11] is equal to (required flow × complexity of the shortest path algorithm). In our case the required flow is $O(n)$, and the complexity of the shortest path algorithm is $O(n \log n)$. Therefore, the total complexity is $O(n^2 \log n)$.



a. The intervals.



b. The corresponding graph.

Figure B.1: Example for minimum cost flow construction