Simulation Based Modeling of Optical Burst Switching Networks

A Thesis Submitted to The University

of Kent

For the Degree Of Doctor of Philosophy

In Electronic Engineering

by

Alexios Louridas

June 2008

Acknowledgments

I will like to thank my supervisor Dr. Nathan Gomes who I have enjoyed working with for his extremely large patience that he showed me. I will also to thank him for the knowledge that he provided me on the subject of optical communications and optical networks. I also want to thank the support that I received from the support group from OPNET Technologies on the software used OPNET Modeler.

Many thanks go to my friends Dr Salavat Magazov for his knowledge on optical devices, Dr. Kamiran Haider and Dr. Indran Silvaraja for their support and the large conversations. Last but not least I would like to thank Joel for the good time we had in pubs and that he never refused any dispute on any philosophical or political discussion. Also I would like to thank all my friends here in Canterbury, which I do not name so I will not have any inconveniencies if I forget anyone, for the good time we had during my studies.

Special thanks go to my family for the continuous support both in financial and sentimental way. Also a special thanks to my partner in Canterbury, Louiza, for all the time that we stayed up at night to study and for her wonderful cooking when I was exhausted. Also I would like to thanks her for her continuous belief and support on me.

Abstract

Optical Burst Switching (OBS) has been suggested as an intermediate step towards full Optical Packet Switching (OPS). There are considerations over the algorithms/protocols to be used. For example, different algorithms have been proposed for assembling bursts, for selecting the wavelength channel to transmit the burst on (scheduling) and for Class of Service (CoS) provision.

The first aim of the research discussed was to develop a robust simulation model for OBS which could be used in the future for developing and investigating different algorithms involved in OBS networks. We succeeded in developing a unique OBS model in OPNET Modeler that includes all stages of an OBS network. These stages include burst assembly, Class of Service and scheduling algorithms. The model is targeted at networks that may have near-term implementation potential (without wavelength conversion and optical buffering, although the model is capable of modelling such processes). The way the model was developed has made it possible for future researchers to add any burst assembly, Class of Service and scheduling algorithms and use them to investigate the performance of burst dropping probability in an OBS network using the Just Enough Time reservation technique.

The second aim of the research was to develop a new optimization algorithm for the simple scheduling schemes proposed for wavelength allocation in optical burst switching (OBS) networks. The operation of this algorithm is compared to standard first-fit and random scheduling schemes with different combinations of burst assembly algorithms. Lower burst dropping probability in comparison to other scheduling schemes is shown for our new optimized scheduling scheme.

The final aim of the research was to demonstrate for the first time the effect on burst dropping probability when combining different algorithms of all stages. It has been proved that the performance of an OBS network has a more complex interdependence between the choice of assembly algorithms and scheduling algorithms, suggesting that the choice of scheduling and assembly algorithms must be considered together.

Table of Contents

CHAPTER 1
INTRODUCTION

1.1 Communication Networks

Communication has been in existence since the beginning of mankind. It is human nature to exchange ideas between each individual in the fastest way possible, therefore communication became the pivot for technological advancement in human societies. Communication systems either as simple as smoke signals in the air or as advanced as optical pulses in glass fibre, always need a medium to transmit the information from one site to another. These sites are usually called nodes and multiple nodes connected together via a medium become what are known to the world as networks.

This chapter will start with the aims of the thesis. Then it will follow with a short historical reference to optical technology and optical data networks. The discussion will then be focused on existing optical networks. The chapter will move on to describe the reasons for the need for all-optical networks. Finally it will conclude with an overview of the thesis that will briefly describe the contents of each of the following chapters.

1.2 Aim of Thesis

Network traffic has been increasing rapidly in the backbone networks as the usage of the internet has become more popular and its usage will keep increasing as the number of households connected to the internet goes up. A large demand has been seen for applications that will increase the traffic even further, such as voice over IP, entertainment, video on demand and gaming [1]. All-optical networks offer a solution to the large demand on traffic throughput and provide higher speeds compared to the networks available today, features which are needed for transferring the information in the future. The most likely candidate for all-optical networks has been optical packet switching (OPS). The main problem in developing OPS is that the technology of optical components has not yet reached a mature enough state that will make OPS an affordable option. Thus, optical burst switching (OBS) has been seen as a technologically viable option or a transitional option between the already existing backbone networks and OPS. OBS can be more easily implemented than OPS with the available optical technology and components and it can be a lot more affordable today and more adaptable to the existing backbone networks. There is a concern though in the literature

that OBS performance especially in burst dropping and end to end delay is not attaining the required level. The current research in OBS networks is focused on reducing the performance in burst dropping and end to end delay. In the research done up to today for OBS, it has been presented that different algorithms can be adopted in different stages of the OBS networks that could improve significantly these performances.

The research on OBS leads us to concentrate our work on two areas:

- Developing or improving algorithms in OBS
- Researching the performance of an OBS network for different combinations of algorithms

Before the aims are explained in more detail, it is necessary to review the basics of how OBS networks operate. Much more detailed explanation will be presented in the following chapters.

An OBS network can be separated into four stages. The first stage, called reservation, is the process of sending data bursts and their headers through the network. There are many reservation techniques that can be used in OBS networks and that have been researched in the literature. More detail on the different reservation techniques that have been developed will be presented in Chapter 2. The second stage, called burst assembly, is the process of assembling packets into bursts. Again there are many algorithms that describe different methods of burst assembly discussed in Chapter 3. The third stage, quality of service (QoS), is where the bursts are assigned a service class or priority and the manners in which the bursts are sent in order to distinguish the different classes of service. The different algorithms for QoS are described in Chapter 3. The fourth and final stage is called scheduling and is the process of actual transmission of bursts on different wavelength channels. Many algorithms have been proposed in the literature for the manner in which the channel is selected and they are explained in Chapters 3 and 6.

The first aim and the main focus of the research was to improve the performance of an OBS network relying only on simple, low cost technology which is feasible at present, without using technology that would overcomplicate the nodes.. After the extended research of the literature on OBS networks it was concluded that the most significant improvements in OBS can be achieved at the scheduling stage and so it was decided to concentrate the research effort on the scheduling stage.

At the same time the new algorithm was being developed, the second aim of the thesis was being investigated after some preliminary results showed an interesting effect caused by interaction between algorithms. Thus the final aim was to concentrate on researching the effect of a number of combinations of different stage algorithms in an OBS network. In the literature, algorithms of each stage were always compared to each other without ever, as far as the author is aware, investigating the effects that the algorithm has on other stage algorithms working in combination with it. In order to fill this gap in the research, a model had to be designed to investigate multiple algorithm combinations, including the new algorithms on scheduling that were being developed. Overall, three algorithms for the burst assembly stage were to be implemented, two for quality of service and four for scheduling including our own proposal.

In order to accomplish both aims, the model has to accommodate all stages of an OBS network and each stage must be able to use any of the algorithms. So a simulation model was needed for overcoming the complexity of the investigation. OPNET Modeler offers a solution to the implementation of such a model that would be a sufficiently close approximation of a real OBS network. In order to maintain the validity of the model, it had to be comprised of two different types of nodes: an edge node and a core node. An edge node is the source/destination of the burst and is where the bursts are converted from electronic to optical domain and back. The core node is a simple building block of a switching fabric that is capable of receiving optical bursts and routing them to the appropriate output until they reach their destination - another edge node.

1.3 Optical network fundamentals

1.3.1 History of Optical Networks

The first concept of guiding light waves through glass fibre was introduced by Charles Vernon Boys in 1887. In the 1970's, the first low loss optical fiber was created. In the 1980's, computers were starting to take on a more personal nature, meaning that computers now started to appear in households rather than only in businesses premises. Until then optical digital asynchronous networks were developed without using a general standard, having in effect incompatibilities between different component vendors. In the middle of the 1980's, the Synchronous Optical Network (SONET) standards [2-5] were formulated by the Exchange Carriers Standard Association

(ECSA) for the American National Standards Institute (ANSI). At the same time the equivalent European standard appeared, called Synchronous Digital Hierarchy (SDH) [5-7], which was formulated by the International Telecommunication Union (ITU). The introduction of the World Wide Web (WWW) to households caused an exponential increase in traffic. Thus the scientific community started investigating all-optical networks as a way to accommodate the projected increases in traffic in the future.

1.3.2 Optical network components

Optical networks need to be capable of transmitting data, voice and video quickly, efficiently and cost effectively. Fibre optic technology is at the core of today's high speed networks. Data is transmitted at the speed of light inside fibres and can reach speeds up to several tens of Gbps and work is being done to increase the speeds even further up to several Tbps [8, 9].

A communication system in general is separated in three sections: the transmitter, the medium and the receiver [Figure 1-1]. The same principle applies to optical communication systems, and in the following paragraphs individual parts of the optical communication system will be briefly analysed in order to provide an overview of optical network components that will help, primarily, in explaining how OBS networks operate.

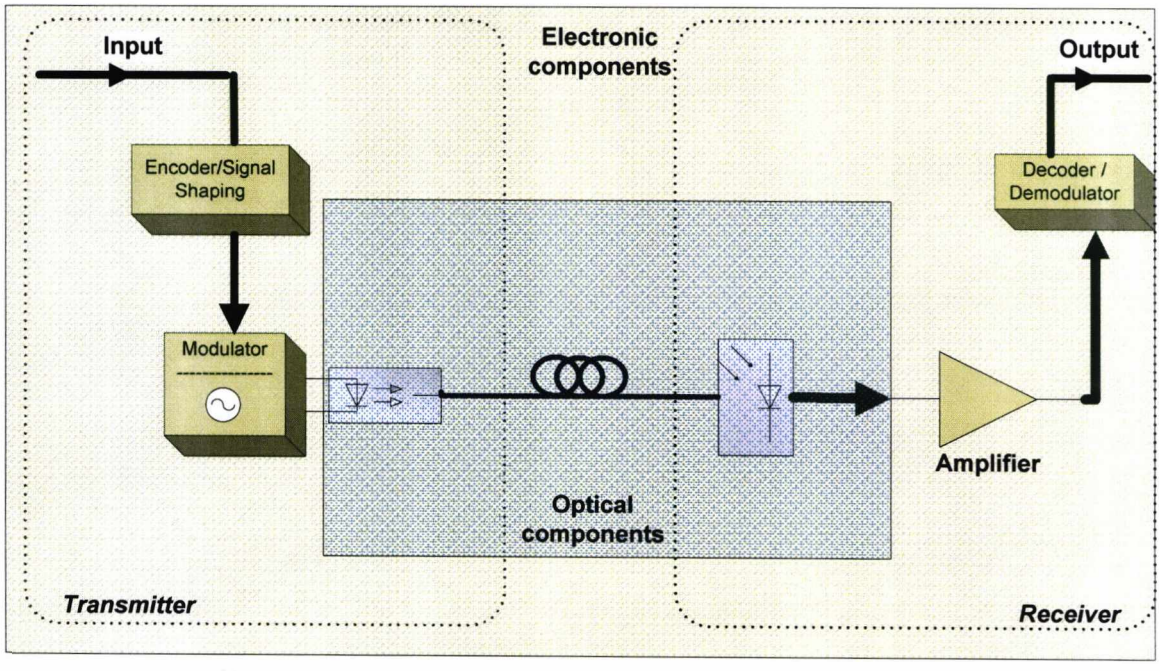


Figure 1- 1 Block diagram of an optical communication system

1.3.3 Optical transmitter

An optical transmitter as seen in Figure 1-1 consists of two electronic parts and one optical part. As a signal comes to the transmitter it enters an encoder for digital transmission or a signal shaping circuit for analogue transmission. The next stage is a modulator for digital transmission or an electronic driver stage for analogue transmission that will drive the optical source, generating light. For analogue systems the light is transmitted at different intensities, whereas for digital systems the two levels (0 and 1) are represented by two levels of intensity. The two most common ways of generating light in optical networks are either a LASER or a light emitting diode (LED). These components essentially transform an electrical signal that represents voice, data or video into optical signal.

1.3.4 Optical receiver

An optical receiver as seen in Figure 1-1 comprises several components. A light sensitive device is used to receive the optical signal by transforming the optical signal into an electrical one. There are many types of light sensitive devices, but the most common are based on semiconductors. These are called photodiodes or phototransistors and different types exist that have different characteristics to accommodate different purposes. The electrical signal is then amplified and filtered to recover as much of the original signal that was distorted in the optical medium as possible. The amplified signal is then demodulated and decoded to obtain the information that was originally transmitted[1].

The optical transmitter is usually referred to as an electronic to optical converter (E-O converter) and similarly the receiver is referred to as an optical to electronic converter (O-E converter).

1.3.5 Optical fibres

Optical fibres provide the transmission medium between nodes in an optical network. Optical fibres are distinguished into two categories depending on the size of the core relative to the wavelength. Fibres with larger cores are referred to as multimode fibres and those with smaller cores single mode fibres. The difference in physical

[1] The questions of error correction, modulation/demodulation and other aspects of the communication path are omitted from the discussion as they are outside the scope of this thesis.

dimensions of the optical fibre core influences the way an optical signal can be transmitted inside the different types of fibres.

In simple terms, in multimode fibre the larger core allows the light to take many different paths through the core. Each of these paths is called a mode, hence the term "multimode". The light travelling through each path has different speed and therefore arrives at the destination at different times. This phenomenon is called modal dispersion. Obviously, the longer the fibre the more pronounced this effect becomes. To mitigate for modal dispersion, the core can be designed in such a way that a faster mode travelling through the centre of the core is slowed down and other modes sped up. As a result, all modes arrive at approximately the same time at the destination and modal dispersion is minimised. Due to this property of multimode fibre it is used for data transmission up to tens of Gbps.

Single mode fibres have only one path through which optical wave can travel, therefore single mode fibres do not suffer from modal dispersion. Single mode fibres can transmit an optical signal up to 15km at data rates of tens of Tbps.

1.3.6 SONET/SDH

In order to understand the need for an all-optical network, a description of today's optical networks needs to be presented. SONET and SDH will be briefly described here.

SONET and SDH were designed to provide a standard in line rates, coding schemes and bit rate hierarchy in optical networks. The standardisation provided a reliable interconnectivity between different vendor components. The main concept of SONET/SDH is to define a technology to carry many signals of different capacities into an optical network. The first step in defining SONET/SDH was to have a base signal. The base signal in SONET is known as synchronous transport signal level 1 (STS-1) and has a bit rate of 51.84 Mbps. Higher bit rate signals are defined as integer multiples of STS-1, creating 4 additional signal levels as shown in Table 1-1. The equivalent name for STS in SDH is the synchronous transport module (STM) with the first level STM-1 operating at 155.52 Mbps. For each of the signal levels an optical domain counterpart exist called optical carrier levels (OC). The relationship between STS, STM and OC is shown in the Table 1-1.

1.3.7 Wavelength Division Multiplexing

In order to increase efficiency several signals can be transmitted via the same fibre by wavelength division multiplexing (WDM). Each individual data signal is coded/pulse shaped and modulated onto a separate wavelength. All wavelengths are then combined in a multiplexer and sent through a single fibre. Each laser carrier can have its own independent bit rate, and the total bit rate is the sum of all bit rates of individual signals. A demultiplexer is used at the destination to recover all the wavelengths individually. For selecting a single wavelength, optical filters are used to filter out the unwanted wavelength depending on the receiver.

WDM is a very important step to realisation of an all-optical network as it provides the necessary speed increase in the networks due to more efficient usage of fibres. Throughout the thesis whenever an optical link is discussed WDM is assumed to be available. More information on WDM technology and components can be found in [10, 11].

Table 1- 1 SONET/SDH Multiplexing Hierarchy

SONET/SDH OC	**SONET/SDH Signal**	**Bit Rate**
OC-1	STS-1	51.48 Mbps
OC-3	STS-3/STM-1	155.52 Mbps
OC-12	STS-12/STM-4	622.08 Mbps
OC-48	STS-48/STM-16	2488.32 Mbps
OC-192	STS-192/STM-64	9953.28 Mbps

From Table 1-1 it can be seen that signals of the order of 10 Gbps can be transported which is rather low compared to the capacity that a fibre can handle and also low compared to the capacity that is needed in today's core networks due to the increase in traffic. Thus an all-optical network would be able to accommodate larger bit rates increasing the bandwidth of the existing networks significantly and would be able to

accommodate the increase in traffic at present and into the future. The problem then is what happens at the switching nodes. This is where all-optical networks are important (WDM already provides transmission link capacity).

1.3.8 Optical Switches

In order to send an optical signal from its source to its destination it has to travel through different paths in the network. Nodes called switches interconnect multiple networks into a larger network. Switches forward traffic based on the destination information contained in the header of data packets or bursts. Switches are divided into two types of switching the "store-and-forward" and the "cut-through" switching. The "store-and-forward" type copies the whole information and stores them in buffer until the destination is obtained from the header. The "cut-through" type of switching reads and finds the destination from the header and starts forwarding without storing the information in the switch. In optical switching today the optical signal has to be converted in an electrical signal in order for switches to read and process the information from the optical signal. Directly processing information from the optical signal is still being developed within research laboratories [12], however the technology is not advanced enough to become commercially viable. Although an optical signal can reach high data throughput speeds, a switch using O-E-O conversion is limited to the speeds of the electronic part. This thesis concentrates on an all-optical network and therefore switching nodes are modelled in such a way as to accommodate for O-E and E-O conversion in optical switches.

1.4 Optical Burst Switching

The all-optical network which is going to be investigated in this thesis is the optical burst switching. The main concepts that are going to be discussed in this thesis are end to end delay, bust loss probability and throughput. What is meant by end to end delay in OBS is the time taken for a burst / packet to be transmitted across a network from source to destination. In this thesis when end to end delay is used it is assumed to be for a burst and not packet except where referred to. Burst loss probability is the number in percentage of bursts dropped in an OBS network. Throughput is defined as being the amount of data transferred successfully from source to destination in a given amount of time.

1.5 Overview of Thesis

The thesis is divided into 7 further chapters. The first two chapters (2 and 3) present a review of some of the literature on OBS that is necessary for the understanding of the main parts of the thesis. In the first part of Chapter 4, an introduction to OPNET Modeller is presented in order to familiarise the reader with the simulation environment and help understanding the simulation model presented in the second part of Chapter 4. Chapter 5 presents the new scheduling algorithm that has been developed and results explaining its operation and performance. Chapter 6 presents the results of the comparative analysis of different algorithms at each stage and how various combinations of algorithms affect the OBS network performance. The contents of each chapter are described in more detail, below:

Chapter 2: In this chapter OBS is described in more detail and compared to OPS and circuit switching. It presents to the reader the reason for OBS being seen as the solution for future networks in the near-future. This is mainly accomplished by referring to some of the technological advances in optical network components.

In the OBS literature, there is a lot of research dedicated to reservation techniques. The chapter presents the most popular techniques for reservation and tries to identify the advantages and disadvantages of each of them. Comparative analysis takes into account end to end delay, probability of burst loss, and bandwidth utilization of each of the reservation techniques.

Chapter 3: This chapter focuses mainly on the operation of the OBS edge node. It identifies in great detail the components that make up an edge node. As was mentioned earlier there are three other stages besides reservation in OBS: scheduling, burst assembly, and QoS assignment; and all these functions are performed in the edge node of an OBS network. The chapter presents the different algorithms found in the literature for burst assembly, scheduling and QoS. The algorithms are presented through a comparison between them. The reasons for the focus of the research on the particular scheduling algorithms studied in this thesis will become clear. The Chapter finishes with queuing theory models that describe the scheduling algorithms.

Chapter 4: In this chapter the operation of OPNET Modeller is briefly described. It will also become clear why OPNET Modeller was chosen as a modelling environment for the project. This chapter also provides all the necessary terminology for the reader to understand the model described in later chapters. The chapter provides also the reasons

behind selecting such a modelling technique and finishes with stating the OBS models objectives.

The second part of this chapter is focused on the modelling of an OBS network, which forms a core part of the work described in this thesis. It provides details on how the theory described in Chapters 2 and 3 is put into the simulation models. It also demonstrates how close the model is to real life optical components by comparing some of the results to the test-beds described in Chapter 3. Initially, the edge node modelling is described and flowcharts are given on the operation of the algorithms employed by the edge node. It moves on to explain the modelling of the core node and how the switching is performed. At every stage the model is compared to real life components described in the reviewed literature. The chapter finishes with a demonstration of correct operation of the individual components in the model as well as correct operation of the model as a whole.

Chapter 5: In this chapter the step-by-step research accomplished for this thesis is presented, which led to the creation of a new, improved scheduling algorithm. The chapter explains different scheduling algorithms performances and discusses methods of improving the performance of these algorithms. The explanation is based on queuing theory models that describe the scheduling algorithms as mentioned on Chapter 3. Results of the accomplishments are presented and explained. In more detail, the improvement of the new scheduling algorithm in terms of burst loss probability compared to other scheduling algorithms is presented. Finally, different network topologies were investigated and it was shown that for all researched topologies the advantages of the new scheduling algorithms are apparent.

Chapter 6: In this chapter different stage algorithms presented in Chapters 3 and 5 are put together for different network topologies and the results, such as, burst end-to-end delay and burst loss probability are presented. The chapter focuses on the last aim of the thesis where it is shown that the combination of different stage algorithms not always produce performance results expected from the literature described in Chapters 3 and 5. The reasons for these unexpected results from the interaction of the algorithms are given and supporting evidence for these reasons is presented in great detail.

Chapter 7: This chapter is a conclusion of the thesis. It once again reviews the focus of the research and explains its contribution to the research of OBS. It presents the conclusions on the performance of an OBS network and finishes with the possible ways of future progress of this research topic on OBS.

1.6 References

[1] T. Freeman, "Get Ready for Growth," in *Fibersystems Europe in Association with LIGHTWAVE Europe*, 2005, p. 3.

[2] K. Lee and T. Aprille, "Sonet Evolution: The Challenges Ahead," in *IEEE GLOBECOM*, Phoenix, USA, 1991, pp. 736-740.

[3] U. Black, *Optical Networks. Third Generation Transport Systems*: Prentice Hall, 2002.

[4] M. Jones, R. Butler and W. Szeto, "Sprint Long Distance Network Survivability: today and tomorrow," *IEEE Communications Magazine*. vol. 37, pp. 58-62, Aug. 1999.

[5] R. Ramaswami, *Optical Networks: A Practical Perspective*: Morgan Kaufmann Publishers Inc, 2001.

[6] D. Cavendish, "Evolution of Optical Transport Technologies: From SONET/SDH to WDM," *IEEE Communications Magazine,* vol. 38, pp. 164-172, June 2000.

[7] G. Held, *High Speed Digital Transmission Networking: Covering T/E-Carrier Multiplexing, SONET and SDH*: John Wiley and Sons Ltd, 1999.

[8] K. Fukuchi, T. Kasamatsu, M. Morie, R. Ohhira, T. Ito, K. Sekiya, D. Ogasahara and T. Ono, "10.92-Tb/S (273 / Spl Times / 40-Gb/S) Triple-Band / Ultra-Dense WDM Optical-Repeatered Transmission Experiment," in *Optical Fiber Communications Conference (OFC)*, Anaheim, USA, 2001.

[9] S. Bigo, Y. Frignac, G Charlet, W. Idler, S. Borne, H. Gross, R. Dischler, W. Poehlmann, P. Tran, C. Simonneau, D. Bayart, G. Veith, A. Jourdan and J. Hamaide, "10.2 Tbit/S (256x42.7 Gbit/S Pdm/Wdm) Transmission over 100 Km Teralight Fiber with 1.28 Bit/S/Hz Spectral Efficiency," in *Optical Fiber Communications Conference (OFC)*, Anaheim, USA, 2001.

[10] A. K. Dutta, *WDM Technologies: Active Optical Components: Active Optical Components*: Academic Press Inc., 2002.

[11] A. K. Dutta, *WDM Technologies: Passive Optical Components: Passive Optical Components*: Academic Press Inc, 2003.

[12] Naoya WADA, "Ultra-Fast Photonic Packet Routing Technology" *Journal of the communications Research Laboratory*, vol. 49, pp. 21-36, 2002.

[13] J. M. Senior, *Optical fiber Communications: Second Edition*: McGraw-Hill, 1992

[14] G. Keiser, *Optical fiber Communications: Third Edition*: McGraw-Hill, 2000

[15] Jue, P. Jason and Vokkarane, Vinod M.; *Optical Burst Switched Networks*, Springer, Optical Networks Series, 2005

CHAPTER 2

THEORY ON OBS

The increasing demands placed by such "all-in-one quadruple-play" [1] model means an increase in the deployment of fibre-optic networks. As WDM is today the main technique in optical networking, as the aggregate bit rate increases, new technologies for switching are needed. For the mentioned network demands, optical packet switching and/or burst switching can provide the necessary solutions.

In the new networks to be developed, data must be manipulated at very high bit rates, so electronic switching is not feasible. Optical switching and manipulation of data are one of the unique abilities of the optical packet and burst switching networks that can provide for this high bit rate manipulation, as no electronics in the switching or processing of the data are involved. Section 2.1 describes the theory behind burst switching, analyses the development of an OBS network and compares it with other known switching techniques. Sections 2.2 and 2.3 analyse a number of architectures for OBS networks and presents a comparative analysis.

2.1 Switching and Optical Switching

2.1.1 Circuit and Packet Switching

The evolution of multiplexing over the years of telecommunications has seen a generalisation of two types of switching – circuit switching and packet switching.

Circuit switching has developed out of synchronous digital techniques and as the name implies is a technique that establishes a dedicated circuit (or channel) between nodes before the users may communicate. Each circuit and its entire bandwidth is dedicated to one connection and, therefore, cannot be used by other users until the circuit is released and a new connection is set up. Even if no actual communication is taking place in a dedicated circuit, that channel still remains unavailable to other users. Channels that are available for new calls to be set up are said to be idle. In certain cases to avoid delays in setting up connections telephone service providers offer a permanent circuit through their systems. These circuits are called dedicated or leased lines and usually can accommodate higher bandwidth than is available in a switched circuit. Types of circuit switched networks include the public switched telephone network (PSTN) and integrated services digital networks (ISDN) [2, 3].

In networks using packet switching there might be routes through different switches that are used for a specific end to end connection. These routes can be either set up during the initialisation of the switches and become Permanent Virtual Circuits (PVCs), or can be created on demand in which case they are called Switched Virtual Circuits (SVCs). Types of packet switched networks include the frame relays (FR), X.25 and the asynchronous transfer mode (ATM) [2, 3].

Further development in technology led to introduction of Digital Subscriber Line (DSL). This technology can reach speeds up to 10 Mbps for asymmetric DSL (ADSL) and even higher for Very high speed DSL (VDSL) up to 52 Mbps. DSL technology utilises bandwidth beyond voice transmission and by using filters it is possible to separate voice and data channels and have them operating independently. The main constrain of xDSL technology is the limitation on the physical distance between the user and the local digital exchange. [4]

In the late 20th century, due to economic and geopolitical changes in the world, computer traffic started to increase exponentially and more users were able to afford telephone lines. This made circuit switching a very expensive way of communication. So for the reason of cost another way of networking had to be introduced to overcome the expenses of circuit switching. One of the alternatives to circuit switching is to send traffic only when it is needed. In circuit switching when a connection is established, data can be send automatically to the recipient, but in a shared network there has to be a system to identify data for switching nodes to know where the destination is. As it is impossible to label each bit individually, bits are gathered together into packets, cells or frames with each having a label that provides information about the data inside them. The switches in packet switching networks must be able to determine from the information provided within the label where to send the packet. This process is called "routing".

The dilemma for future research is whether it is cost effective to push electronics to the limit by creating monstrous equipment to accommodate these speeds. On the other hand it might be more efficient to move to the newly developed all-optical switching that will instantly give networks a new upper limit imposed by fibre optics itself. [5]

2.1.2 Optical packet switching

Optical networks until now have been opaque (non-transparent to transmission protocols); meaning that from a source to a destination there is some interruption at intermediate nodes that have O-E-O conversion, where data experiences change of format and/or modulation scheme. This is a primary cause of limitation of the bit rate of the network, as electronic circuits can handle only a limited amount of bits per second. In addition, recent technology development in optical devices presents us with the opportunity to make all-optical network technologically feasible. All this caused a shift in research efforts towards realisable all-optical networks. To summarise the reasons for utilising the all-optical networks:

- larger bandwidthwith bit rates up to tens of Gbps and even Tbps using more expensive equipment [6,7].
- Bit rates, modulation schemes and packet formats can be different for each transmission and so no unified standard is required. Only the edge nodes (source and destination) need to know the bit rates, modulation and packet formats thus leading to payload transparency [8-10].
- The cost of electronic switching devices, either ATM routers or Ethernet switches, is increasing non-linearly with the data rates of the channels [11].
- The cost of electronic components increases even further as the data rates increase because the power required with equipment increases. As a result the size of the power supplies also increases and becomes an important issue. So in the case of all-optical switches the initial cost is much lower, because the data rate is not proportional to cost and needs far less power for operation. Due to this simplicity service lifetime savings are made by making problem determination and maintenance simpler [8-10].

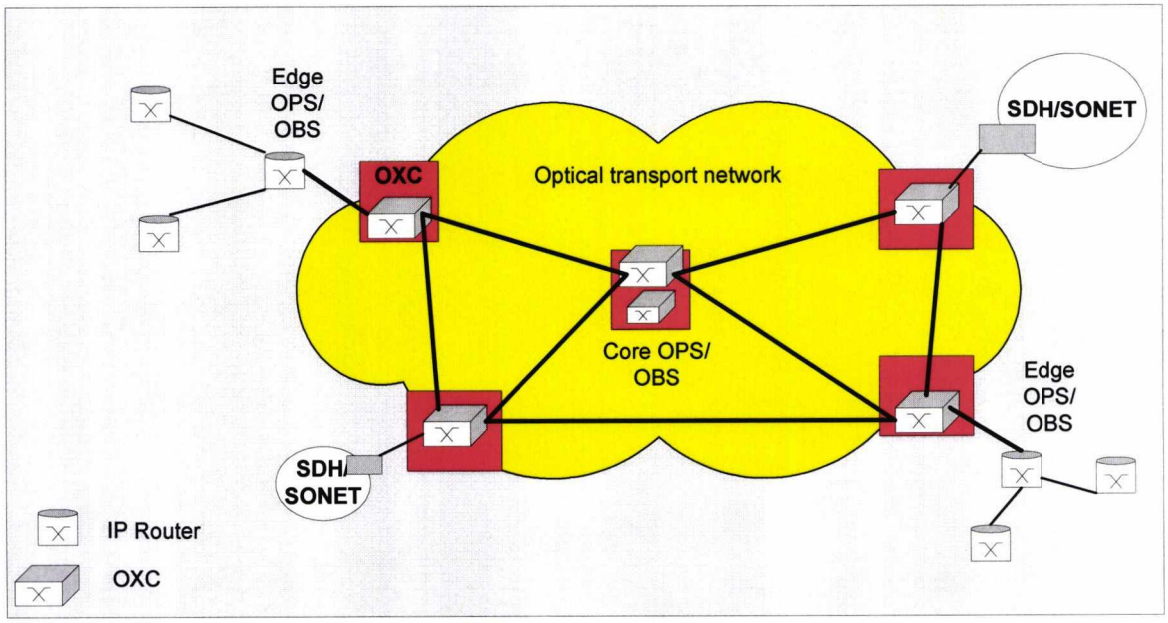


Figure 2-1 Applications of OPS and OBS as core and edge routers

In figure 2-1 we can see an optical network, which uses optical packet switching / OBS at core and edge routers. From the figure we can see that we can connect the optical transport network (OTN) to different kinds of networks, for example to an existing opaque optical network such as SDH/SONET. Two functions for optical packet switching are seen in figure 2-1. Firstly, in a core node where it is necessary to switch packets and perform label swapping inside the optical layer, while transparently transferring the data payload. This makes the number of optical cross-connects (OXC's) smaller and so a better utilization of the total network capacity is being achieved. The second function that OPS performs is in an edge router: here it acts as an interface between the OTN and the service layer by aggregating the traffic from a large number of IP routers and ATM switches and so making the OPS networks independent of data rates and data protocol. This helps the network to perform more efficiently than if an electronic router was used instead.

The main advantage of using OPS in this manner is that it reduces the number of network layers, and therefore simplifies the management of the network (Figure 2-2). The networks that are in operation today use four layers as seen in figure 2-2(a), the IP layer for data transfer in applications and services, ATM for traffic management, SONET/SDH for switching and dense-WDM (DWDM) for multiplexing. In OPS, the network will bypass the ATM and SONET/SDH layers by placing their functions in the IP layer. To accomplish the bypass of the two middle layers, generalised multiprotocol

label switching (GMPLS) is introduced. In networks using MPLS each packet has a label that contains forwarding information only for the next hop. GMPLS is an extension of MPLS where not only packet switching occurs but also wavelength, time and space switching [9-15].

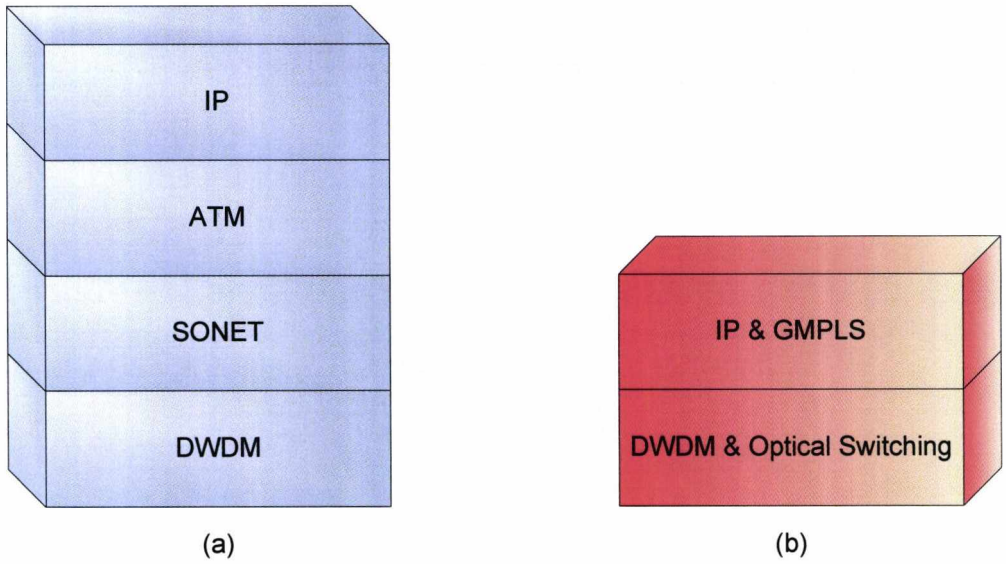


Figure 2- 2 Network Architectures from present day (a) to the future (b)

2.1.3 OPS network analysis

In general, OPS networks are divided into two categories: unslotted networks and slotted networks [10, 16]. The latter have a fixed packet size, where the slots are occupied by the header and payload together with some guard times. Packets from different core nodes may arrive in the same slot time interval in the same core node. The network in the input ports has to align the packets together in time slots before entering any node. To perform the alignment of packets synchronisation is needed to be in phase with the clock of the network. Of course synchronisation is also required for extracting information from the header and the data correctly. Because of both synchronisations mentioned before, the circuits that have to be built are complex and this increases the cost even more.

Unslotted networks on the other hand have variable packet sizes. This makes synchronisation for alignment of packets redundant, and so the node structure is a lot simpler than in a slotted network. Although the node structure is simpler without synchronization, contention resolution is required. Contention is important for both slotted and unslotted networks but in the case of unslotted networks the probability of

contention is much higher. The reason being that packets arrival is unpredictable. The probability of contention in such networks is increased and so the throughput is lower. Now many ways to overcome the problem of contention have been investigated and so the throughput can be increased to desirable levels. Although the whole node structure is simpler than in the slotted networks which require extra units to align packets, the cost is still debatable [17]. This is because, to perform contention resolution in unslotted networks so that desirable levels of throughput are reached, there might be a need for more contention resolution components than needed in a slotted network.

Before synchronization is discussed, it has to be seen where and when delays appear inside the nodes and between them. Between the nodes the only medium that has any effect is the optical fibre itself as we are only interested in optical networks. The delays that appear in fibre are caused by three parameters: chromatic dispersion, fibre length and temperature, where dispersion and temperature variations can also cause effective fibre length variations. Fibre length is the most common factor that is usually considered and produces a delay depending on the distance between two nodes. Dispersion is the effect wherein different components of the transmitted signal travel at different velocities inside the fibre. Chromatic dispersion is the phenomenon by which different spectral components of a pulse travel at different velocities. The propagation speed of a packet is also temperature dependent as the properties of the fibre material change. To minimize all of these delays special fibres can be constructed to achieve stable delays, which then can be compensated for statically inside the nodes [10].

Delays inside the nodes are mostly caused by the switching fabric. So for any OPS networks where contention resolution is required, delays may appear from the contention technique employed.

Now that causes of delays within a network are presented, it is possible to go on to discuss how synchronisation is accomplished in OPS networks. Because in OPS the header defines the beginning of the packet, strict synchronization is needed. Guard times are required between the two ends of the payload to account for any jitter while header insertion or erasure occurs. Although solutions exist to detect and extract packet headers in the electronic and optical domain at speeds in the order of Gbps, there is still a need for headers to be processed fast enough to be reinserted before the packet is transmitted from the node.

2.1.4 Contention resolution for use by OPS and OBS

As in many networks in various communications systems there are moments when switches need to decide which data will be allowed to access limited resources when multiple data instances are received at the same time. This situation is referred to as contention and different networks have different techniques of resolving contention. In OBS and OPS networks three main contention resolution techniques have been suggested in literature. Here they will be discussed in greater detail.

A. Optical Buffering

Buffering in electronic nodes is accomplished by random access memory (RAM). Optical random access memory is still not available. Memory could be achieved by converting the input optical signal into electrical form, storing it, and re-transmitting it, but modern electronics are limited in the access time and processing time to store and read information and must be designed for a specific data rate. Optical memory has partially been achieved with a loop of optical fibre around which the signal circulates. This storage mechanism is transparent to the bit rates. This optical memory loop (OML) must allow data to be switched in, re-circulated with little distortion, and switched out. The main problem with this kind of memory is that it is not possible to extract the information stored in it at any moment; once a packet is launched into an OML it is necessary to wait until it exits after a fixed delay. Also OMLs are very bulky and expensive and packets cannot be stored in OMLs indefinitely.

There is a distinction between two types of optical buffering; single-stage or multistage, however, multistage optical buffering is still rarely proposed for networks and will not be considered in the thesis. There are many techniques for creating single-stage optical buffering, for example keys to optical packet switching (KEOPS) [18-19], Asynchronous Transfer Mode Optical Switching (ATMOS) [20], Shared Memory Optical Packet (SMOP) [21] and so on.

B. Deflection Routing

Although optical buffering is easy to understand it has some major limitations as explained in 2-1(A). In deflection routing however there is a need for only a small amount of buffering or in some special cases no buffering at all.

The way that deflection routing avoids contention is by rerouting the packet from one node to another. Buffering might exist in a node but for limited numbers of packets. There is a special case of deflection routing called *hot-potato* routing. In this special case no buffering is available in any node in the network. Although it has low cost and less hardware, the throughput of the network suffers. The disadvantage of deflection routing is that in high traffic conditions a packet may end up travelling inside a network forever. This is called "deadlock". To avoid deadlock at high traffic a limit to the number of hops where a packet can travel has to be set.

For slotted networks deflection routing works quite well with a good throughput, but for an unslotted network, numbers of copies of packets in the network might arise. This happens as a result of the rise of congestion due to the asynchronous nature of unslotted OPS and OBS networks. As the load in the network increases the number of packets that are circulating is increased due to deflection routing and so the throughput is reduced further. To avoid this we again have to set a specific number of hops a packet can travel. An optimal number in hopping and buffering has to be set in order for the network to work with an acceptable throughput. The optimal number can either be set in the setup of the network or it can be traffic dependent. In order for each node to recognize where the packet has been a time stamp has to be given in the header every time it passes through a node. Of course this means that the network should have a global clock and should be synchronized. Another way to keep track of the number of hops is for packets to have a counter that is decremented as it passes through each hop as is done in IPv6 protocol. Although deflection routing seems complex to design it can be implemented easily and with a low cost, as no or little buffering is needed.

C. Wavelength Conversion

Wavelength conversion can be implemented as a stand-alone scheme or together with one of the two schemes that have already been discussed. By looking at the advantages and disadvantages of the previous two schemes it can be understood how wavelength conversion can improve them. In optical buffering the throughput and performance of the network is very good but implementation is problematic. On the other hand, deflection routing is very simple to implement, but suffers from low throughput and poor performance. By applying wavelength conversion on these schemes their disadvantages can be minimized or even eliminated entirely.

In an optical buffering node the signals are de-multiplexed into wavelengths channels and then wavelength converters are used to locate available wavelengths. The optical buffering takes over and handles each packet accordingly; it can either direct it to an output or into some delay lines. A testbed that used wavelength conversion is the KEOPS project [16] and the wavelength switched packet network (WASPNET) [22]. In both networks mentioned the number of optical buffers is reduced and packet loss is reduced to a minimum. In systems with a number of transmitters and receivers equal to the number of wavelengths, wavelength conversion in conjunction with deflection routing can achieve high throughput. Also wavelength conversion provides a tremendous increase in the performance of a network using hot-potato routing. [23].

Although wavelength conversion works well if the number of wavelength channels is equal to or higher than the number of transmitters and receivers that exists in a switch it does not however have a desirable effect when a small number of wavelengths is used. For these networks the best performance is achieved with optical buffering [10].

2.1.5 Need for optical burst switching networks

One of the main concerns when sending a packet in the optical domain is the extraction of the header. There are many methods for extracting the header either electronically or optically [24, 25]. The problem that is created if electronic extraction is used is that electronic processing is also required. If electronic possessing is involved, the bit rate of the system has to be limited in order for the network processor (NP) to be able to process the header information (typically few ms)[26, 27]. Processing in the optical domain is still at a primitive stage. The extraction can be accomplished by the subcarrier multiplexed (SCM) header [10, 28, 29].

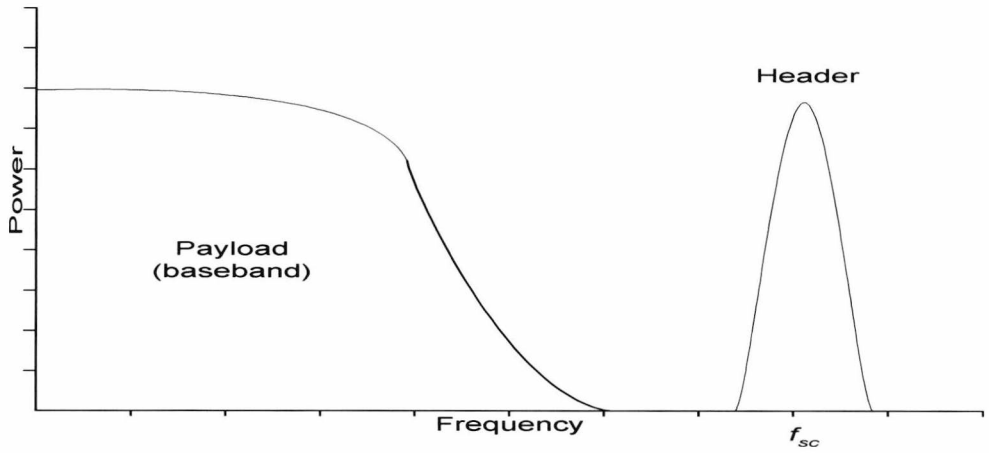


Figure 2- 3 Power spectrum of the payload and the header sub-carrier frequency

The header and payload are multiplexed together in the same wavelength, where the payload is encoded at baseband and the header on a chosen subcarrier frequency at a lower bit rate (figure 2-3). The extraction of the header can then be done easily by a simple photodetector, without any filtering needed. This can be seen in figure 2-4 where the headers are transmitted at different subcarrier frequencies. Also, subcarriers must be as close to each other as possible to avoid using multiple photodetectors with different frequency responses, as this will complicate and increase the cost of the extraction circuit. Finally, it is necessary to keep the header bit rate as low as possible, but not too low as to increase the processing time. The biggest disadvantage of SCM is the limit on the bit rate of the payload that it imposes. An increase in bit rate means spreading of the baseband spectrum, which will eventually overlap with the header subcarrier frequency.

A more simple approach is to send the payload and the header on different wavelengths. The header is extracted by demultiplexing the signal and the header processed electronically.

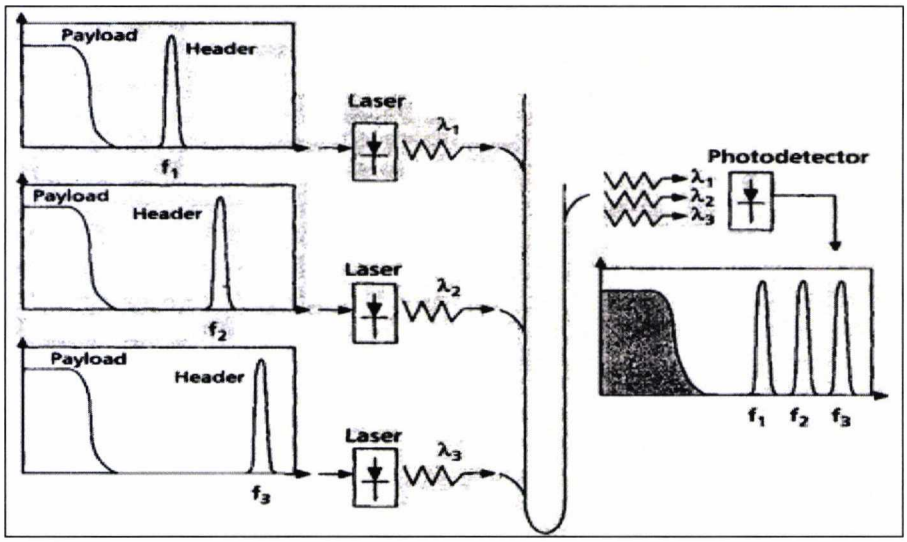


Figure 2- 4 Header retrieval in SCM [10]

Another concern with OPS is the need for contention resolution. All of the techniques in contention resolution are very costly and some even very complex to incorporate in systems as discussed in section 2.1.4. In OPS we have seen that everything is done in the optical domain with few exceptions such as electronic processing of headers. Research has been carried out into photonic processors but these are still not available and would take more than a few years before they appear in the market in the most optimistic prognosis [30].

In general, optical technology lacks optical transistors that would enable the creation of simple memory and switching in the optical domain. Because of limitations in optical technology as well as the high cost of deployment, an alternative technique has been developed which compromises on the cost and the complexity of a system, called burst switching OBS which can be an intermediate technology until problems with OPS outlined above are resolved.

2.2 OBS Mechanisms and Topologies

OBS is a variation of OPS. Packets are aggregated into longer bursts and switched together, and the header is sent on a separate channel, usually in the form of what is referred to as a Burst Header Cell (BHC).

The network is divided into two regions the edge and the core as is shown in figure 2-1. The edge region has nodes called edge nodes that deal with burst assembly/disassembly, QoS and scheduling (the technique to send bursts in different channels). The core region has nodes called core nodes and they are nothing more than

optical switches. In both regions a reservation technique is adopted to define the way the BHC and the data burst are forwarded through the network.

To minimize the effect of the conversion the payload is read and a header is formed and sent before any transmission is made in the optical domain. As shown in figure 2-1, OBS (as OPS) has the ability to send data from any type of network to another. The link between the edge node and the core node has more than two channels, as one is used as a control channel where the BHC is sent. The other channels are used to send the bursts, with each channel using a different wavelength, multiplexed using WDM technique. It has also been proposed that the control channel can be sent in a separate fibre than the data for ease of implementation.

A great deal of investigation and research has gone into scheduling, assembly and quality of service (QoS) techniques in OBS but a final decision on which will be followed still remains open to debate. In Chapter 3 the most important of these techniques will be discussed and examined in detail but before going into detail on the way these techniques operate in nodes in an OBS network a more general investigation will be presented on the reservation architectures that have been proposed in the literature on OBS networks.

As mentioned before, the key feature of OBS is that the switches inside an OBS network are informed of any incoming burst when the corresponding burst header cell is received. Many reservation techniques have been proposed to allow for this feature. The variations in the protocols are mainly due to the differences in burst arrival time relative to the BHC and mechanisms to perform the switching as a burst passes. In the following subsections five reservation techniques are mentioned each with its advantages and disadvantages, which can be used in the architecture of OBS network. In section 2.3 the reservation techniques operation will be described and a comparison will be provided at the end of the chapter.

2.2.1 Tell-And-Wait

In the tell and wait (TAW) [30,32] reservation technique the burst header cell is sent through the network to reserve the channel in each of the intermediate nodes and the source edge node only sends the burst when the entire virtual path has been set up. The virtual path is defined as the concatenation of all the channels/wavelengths reserved

from the source edge node to the destination edge node. In effect the TAW technique resembles the known, old fashioned virtual circuit packet switching method.

In figure 2-5 can be seen the transmission of the BHC through the control channel, as transmitted from the source through intermediate nodes until it finally reaches its destination. The BHC contains information on the wavelengths that are being reserved. Every time a BHC is received by a node, the node automatically sends an acknowledgement message back to the node which has sent the BHC in the first place containing information of the channel reservation. The channel is then reserved indefinitely until a release message is received.

Each node has to be able to identify an available channel and to be able to deal with other incoming bursts. Thus the intelligence of the nodes used in TAW has to be very high and, therefore, more complex nodes have to be used.

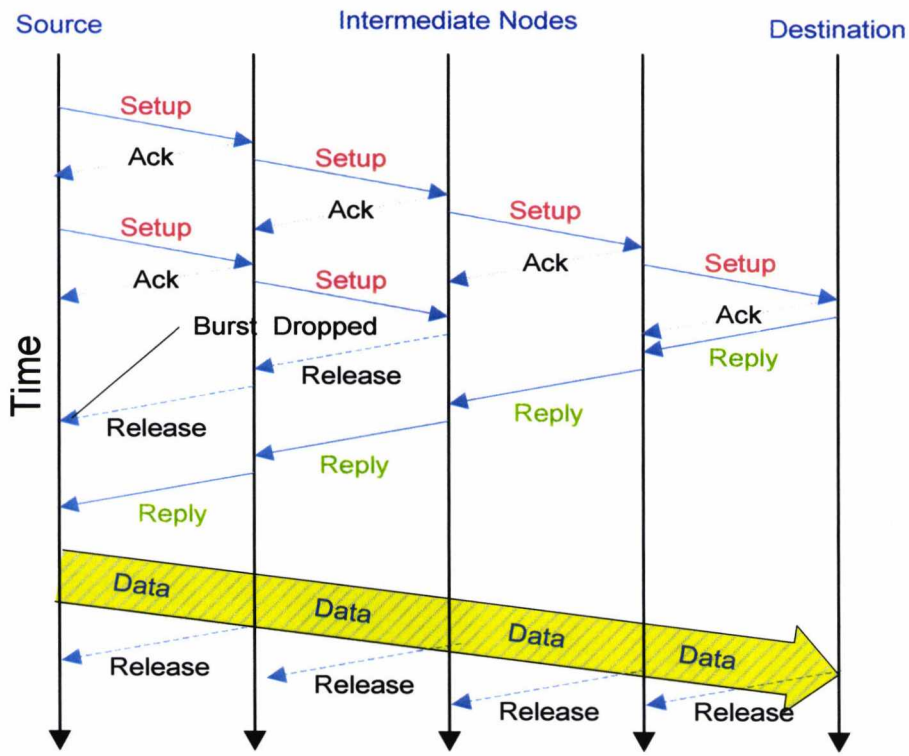


Figure 2- 5 Tell and Wait reservation technique for a burst transmission

The destination node after receiving the BHC and reserving an available channel sends a confirm reply message to the source via the forwarding of the reply through the intermediate nodes that have already reserved a channel for the burst transmission. When the reply message is received at the source node it immediately releases the burst.

The burst is forwarded through the virtual path that has been created. Each intermediate node after transmission of the burst has finished sends a release message to the previous node to free the virtual path between them. Therefore the time a channel is reserved starts from the time at which it has received the setup message until the time a release message has been received.

In some cases a channel might not be able to be reserved at which point a release message is sent back immediately. This can also be seen in figure 2-5. When the release message reaches the source it informs it of the unsuccessful virtual path reservation and the burst is dropped. The burst can be salvaged by higher network layers or from the optical layer if the TAW architecture accommodates for such cases.

2.2.2 Tell And Go

In the tell and go (TAG) reservation technique proposed by Hudek and Muder in 1995 [33] the virtual path is reserved at the same time as the burst is transmitted. If the reservation of a wavelength is not possible the burst is dropped in the intermediate node.

The source will send the setup message and after some time called the guard time the burst is sent to the next node. The guard time includes the time estimated for the next node to discover if a channel is available and also the time for the switch in the next node to be ready to accept the incoming burst. So when an intermediate node receives the setup message it identifies if there is a channel available and if so a new setup message is sent to the next node. In each node optical buffering has to be available in order for the burst to be delayed as soon as it is received by a time equal to the guard time for the switch in the next node to be ready to accept the burst and establish the channel for the following transmission. If a node finds that there is no available channel the burst is dropped by the preceding blocked link. There is no release message from any part of the path nor any confirmation of available channel, as shown in Figure 2-6. In some cases [32] the setup message will continue its path to the destination where the destination will send an acknowledgement or non acknowledgement message back to the source to inform that the burst has arrived or that it has been dropped accordingly. This implies that the source will have to keep a copy of the burst. There are many variations on the TAG technique, the most well known being just in time (JIT) and just enough time (JET), which will be discussed next.

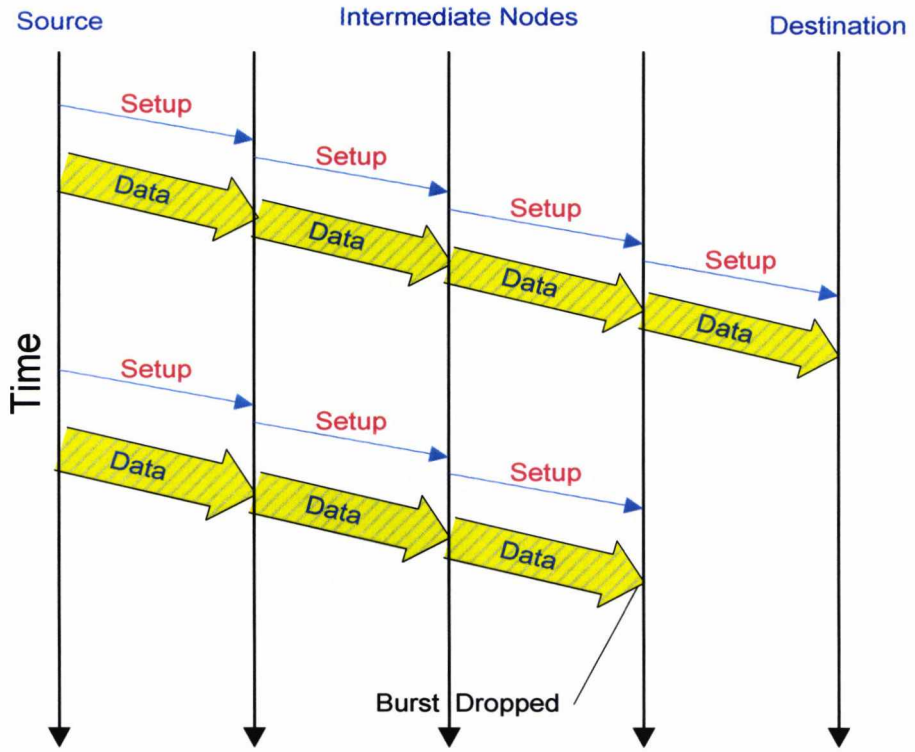


Figure 2- 6 Tell and Go reservation technique for a burst transmission

2.2.3 Just-In-Time

Just in time (JIT) proposed by Wei [34, 35] is very similar to TAG, but slight variations have caused it to be considered separately in this thesis. JIT can be separated into two cases, implicit and explicit, depending on the information contained in the BHC. If no information regarding the length of the burst is sent within the BHC it is called explicit and when information on the burst length is sent via the BHC it is called implicit JIT. In implicit JIT the information on burst length is added in the calculation of the guard time. This of course means that the nodes have to be more intelligent and extra processing is required.

In both cases of JIT, as shown in Figure 2-7, when the BHC arrives at the first node after the source, the node has to send an acknowledgement to inform the source to send the data burst. At each intermediate node, the node has to calculate if the original guard time is sufficient for the burst to arrive to the next node well after the BHC in order for the switch to be setup, and if not then optical buffering has to delay the burst accordingly to accommodate for the following switch configuration delay.

In explicit JIT the burst length information is not transmitted thus an explicit release message has to be sent from the delivering node to the receiving node to release the channel that was reserved. Thus in JIT the channel is reserved for as long as it takes for the burst to go to the next node.

In implicit JIT where information of the data burst length is known, each node can implicitly release the channel by itself without the need of signalling.

Baldine et al. [36] in the ATDnet testbed also mentions a keep alive signal being transmitted through the control channel in the case of a long burst. This will keep the channel reserved until a release message has been delivered to the receiving node. In Figure 2-7 it can also be seen that dropping occurs at intermediate nodes when a path could not be setup.

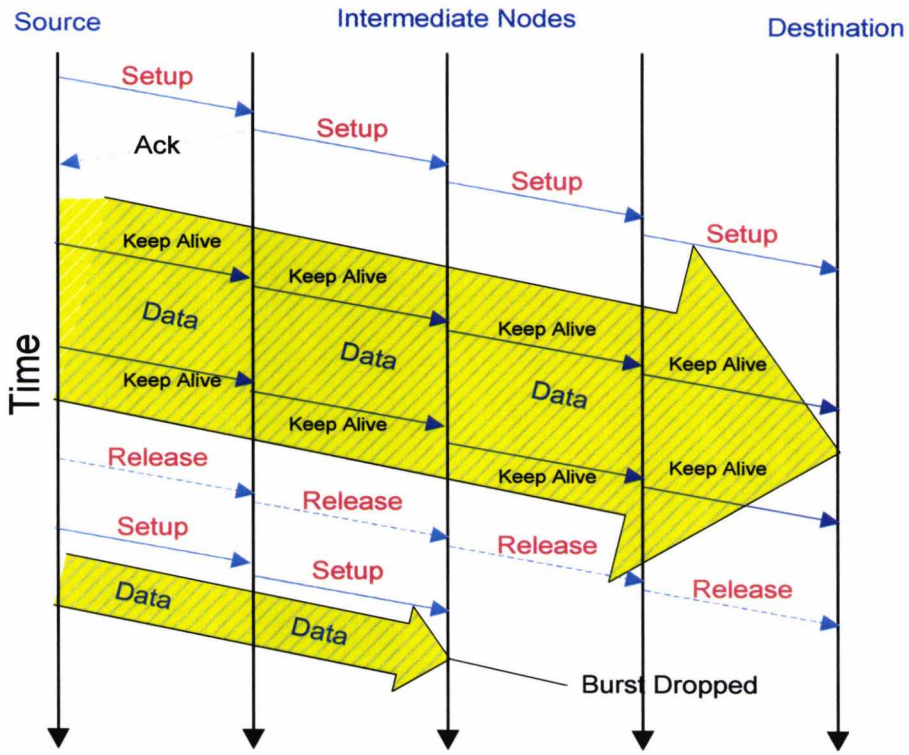


Figure 2- 7 Just In Time reservation technique for burst transmission

2.2.4 Just-Enough-Time

The just enough time (JET) reservation technique originated in the paper by Yoo [37] where it was proposed that an offset time should exist between the control packet and the actual burst. The offset is based on the number of hops, burst length and the distance from the source to the destination.

Figure 2-8 shows the operation of JET in more detail. The source will send the BHC ahead of the burst and it will inform the intermediate nodes about the time the burst will arrive so the switching can take place if possible. This means that the channel is reserved not all the time but only when the burst is actually transmitted. This has as a consequence of an increased complexity of the processing involved in the nodes because when the BHC arrives the node has to calculate precisely if it can handle the burst at that specific time. So it has to calculate the time it takes to setup the switch and the time it will take for the burst to pass through, and also must keep track of all the incoming bursts that it can handle. Although more complexity is needed in the electronic part of the node, the optical part is very simple as no buffering is required in order for the reservation technique to work. When the offset time, as shown in Figure 2-8, has passed the burst is transmitted through the network with the only delay being the propagation delay through the fibres connecting the nodes from the source to the destination.

In case of contention upon arrival of the burst, the node will drop it, as can be seen in the lower part of Figure 2-8. Thus the occurrence of a dropped burst is known to the node and the dropping node does not have to forward the BHC to reach the destination as was done in other reservation schemes; also no release messages are needed.

Another important aspect of JET is the requirement for calculating the offset time for the whole trip of the burst from source to destination. The routing tables in the edge nodes include information on all possible edge node destinations and have information on the number of hops and delays involved through core nodes in order to correctly calculate offset times for the whole trip. This has the advantage that core nodes need to process the BHC only for reserving a channel to the next node and thus optical buffering is not required.

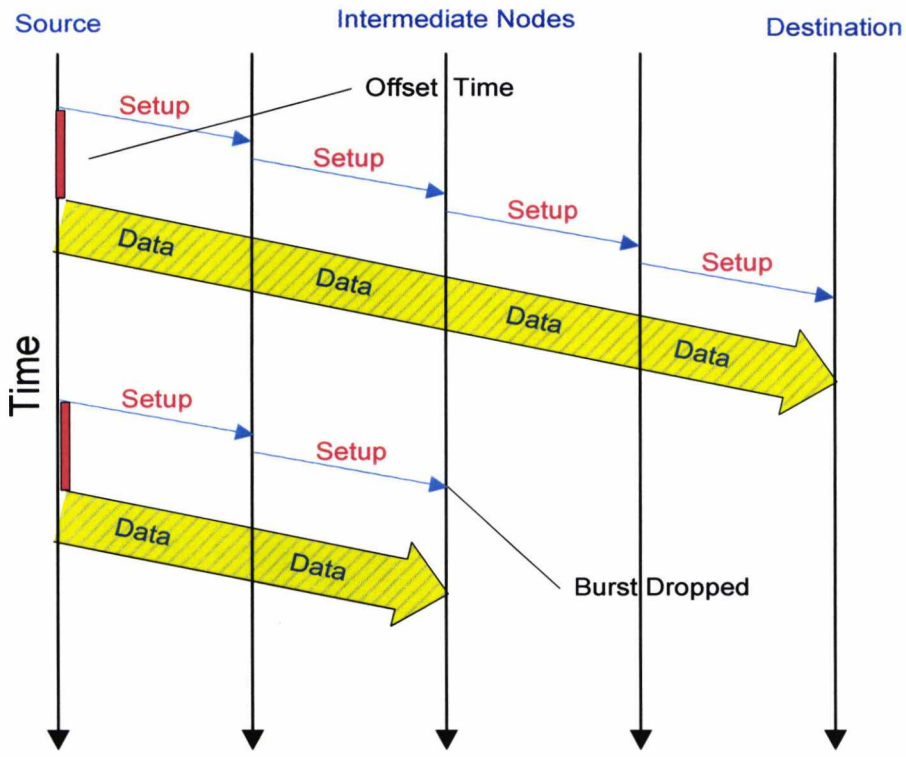


Figure 2- 8 Just Enough Time (JET) reservation technique for a burst transmission

2.2.5 Intermediate Node Initiated

Intermediate node initiated (INI) reservation scheme is a combination of the JET and TAW schemes as discussed by Karanam [38]. An initiated node is established between the source and the destination node that will deal with the reservation of channels in the network. The INI scheme is split in to two parts: the first part is the sending of information from source to the initiated node, and the second part the sending of information from the initiated node to the destination. The first part operates in a similar way as the TAW scheme whereas in the second part the scheme resembles the JET reservation technique.

Figure 2-9 shows the transmission of a burst for the INI reservation scheme. Firstly a BHC is sent to the destination from the source. An intermediate node is selected to be the initiated node and sends a reply back to the source as soon as it has received the BHC. As the reply travels back to the source it reserves a channel at the nodes along the path to the source. When the reply reaches the source the burst is sent at the reserved channel and time. As the reply travels back to the source the initiating node also sends a BHC for channel reservation for the second part of the path to the destination. The reservation is accomplished according to the JET scheme.

Burst dropping can occur either in the first part of the path or the second as shown in Figure 2-9. If a channel cannot be reserved at the second part of transmission the burst is dropped as in JET in the node that failure occurred. In the first part burst dropping will be known while the BHC reply passes through the nodes trying to reserve the channels. At this instance the node at which no reservation could be accomplished, sends a failure packet back through the control channel to the initiated node notifying it of the failure. The initiated node then sends a release message to the nodes that have reserved channels already.

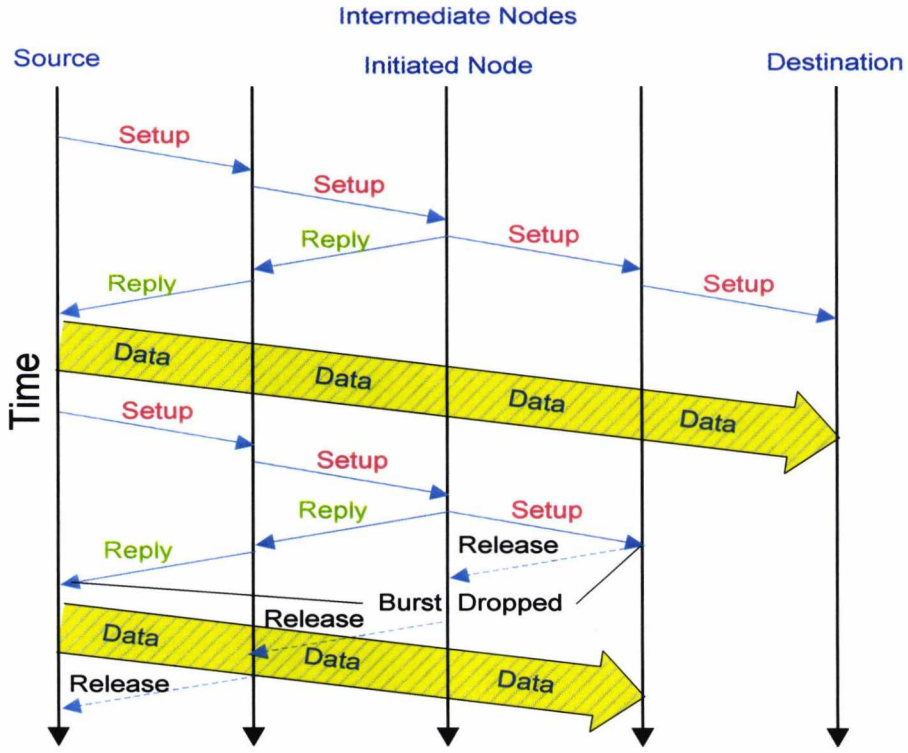


Figure 2- 9 Intermediate Node Initiated (INI) reservation technique for a burst transmission

2.3 Comparison

It is very difficult to ascertain which of the reservation schemes presented up to now is the best choice for use in OBS networks. The comparison that will be given on the schemes will be based on certain important aspects of any network. First, end-to-end delay will be investigated, in each case the delays involved will be given and end-to-end delays calculated. The second important aspect is the bandwidth usage in the networks, followed by a third aspect of the probability of dropping in each scenario. As a final

comparison between the reservation schemes optical buffering, adaptability and complexity of circuitry will be introduced in a general discussion.

It has to be noted that JIT is very similar to TAG in many cases and performance between them is highly dependent on the aspects that we are going to investigate. Hudek [33] investigates JIT and TAG in more detail, and this investigation surpasses the depth of the comparison performed in this thesis.

2.3.1 End to End delay

In order to correctly compare the end-to-end delays for each of the scheduling techniques some assumptions have to be taken into consideration. These assumptions apply for all reservations schemes and are:

- All links between nodes have similar number of wavelengths
- All links have the same length between nodes
- The bit rates on all channels are equal
- The time δ_{cd} for channel discovery is equal in all nodes
- The time δ_{config} for the switch to be configured is equal in all reservation techniques and in all nodes
- The transmission and reading times of the BHC are assumed to be negligible

The end-to-end delay for the TAW technique for a successful burst transmission is the sum of the propagation delay of the burst from the source to the destination ($m \times \delta_{prop}$), of the switch configuration delays and the channel discoveries times ($n \times (\delta_{cd} + \delta_{config})$) and of the end-to-end round trip of the setup message ($m \times (BHC_{setup} + BHC_{reply})$). Where n and m are the number of nodes and the number of links respectively between the source and destination $m = n-1$, and $BHC_{setup} = BHC_{reply}$.

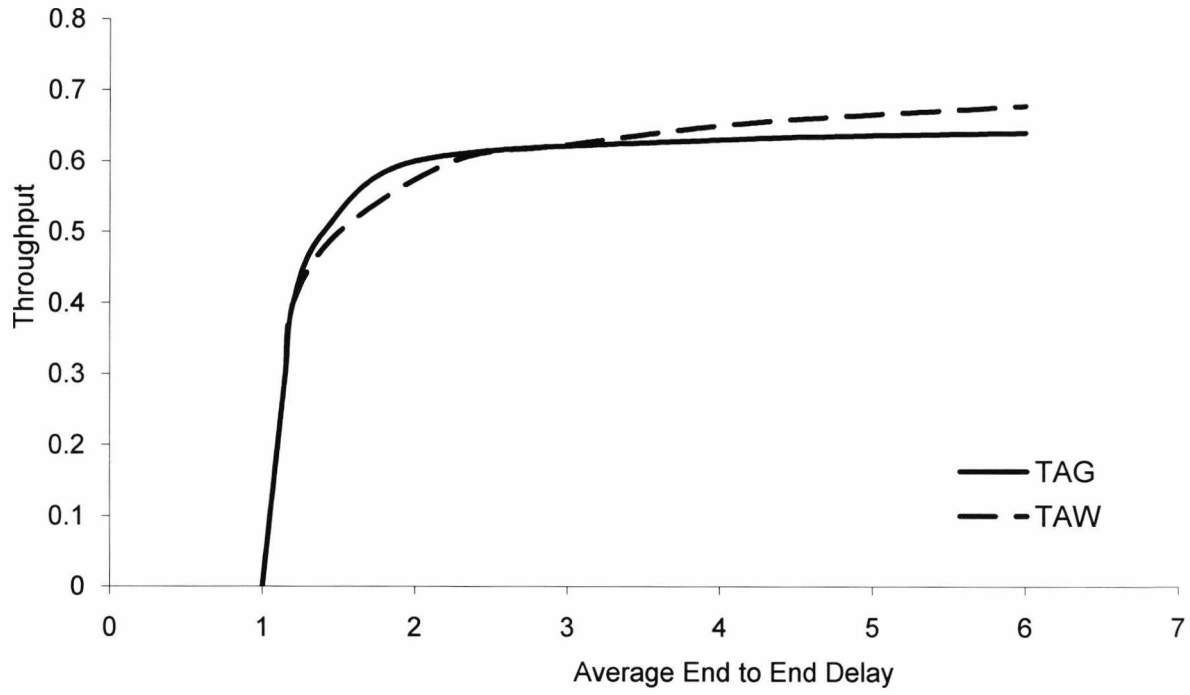
In case of an unsuccessful burst transmission the end to end delay is equal to round trip delay of the header cell from the source to the node at which the unsuccessful reservation of channel occurred. It has to be noted that the end to end delay of an unsuccessful burst transmission is a waste of bandwidth as it cannot be used for any other transmission.

TAG was introduced to decrease the end-to-end delay that existed in TAW systems by reducing the round trip delay, thus the reason for sending the burst nearly simultaneously with the header cell. This in effect has the elimination of the setup and reply propagation delays in the end-to-end delay equation. The equation is then based on the addition of the propagation delays on each link that the burst is passing through and the delays produced for discovering available wavelengths and the switch configurations. The new delays that are added are the delay from the guard time δ_{guard} between the BHC and the burst, and the optical buffering delays δ_{buffer} in each node to produce the guard time. So it can be seen that optical buffering is needed in TAG in each core node to be able to produce the guard time. In the case of TAW the only buffering needed is in the edge nodes to store the burst until the reply BHC arrives. However, the buffering in the edge node can be done in the electronic domain and thus can be stored indefinitely.

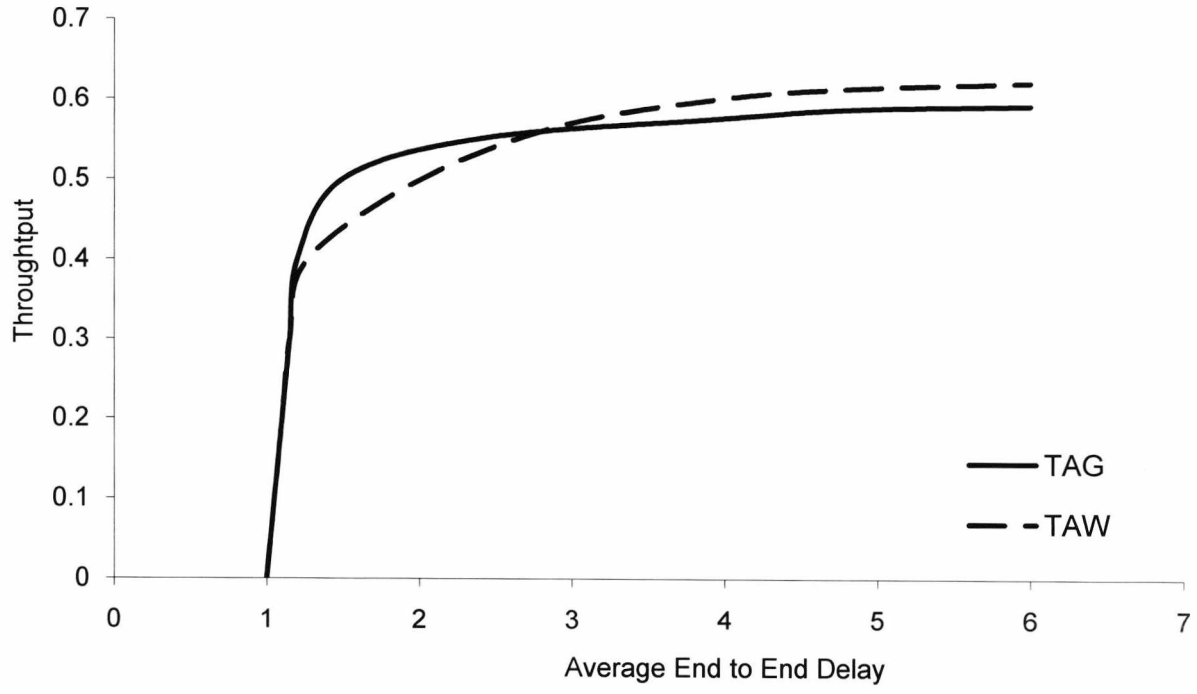
For TAG, when an unsuccessful burst transmission occurs the delay involved has only to do with the propagation of the burst and configuration delays from the source to the node where the dropping occurred. The problem again, as in TAW, is that the channel is occupied and cannot be used for any other transmission of a burst while the dropped burst transmission is occurring.

From the end to end delay equations on TAG and TAW it can be seen that if the propagation delay is small the TAW outperforms TAG as TAG incurs a penalty in bandwidth compared to TAW which is proportional to the burst size. Exactly the opposite occurs when the propagation delay is large.

Figures 2-10 and 2-11 show the average end to end delay D for 3 and 6 hop networks against throughput S for large average burst length (> 5Kbits) and for small average burst length (< 100 bits) respectively, as calculated and presented by Widjaja [32] in his comparison of the two protocols. From Figure 2-10 it can be seen that TAW performs slightly better than TAG for large burst lengths providing higher capacity but suffers higher delay. In the case of small burst lengths as presented in Figure 2-11 TAG performs far better in comparison with TAW in providing higher capacity, as expected, as the time required for the reservation of channels in small bursts situations is much smaller for TAG.

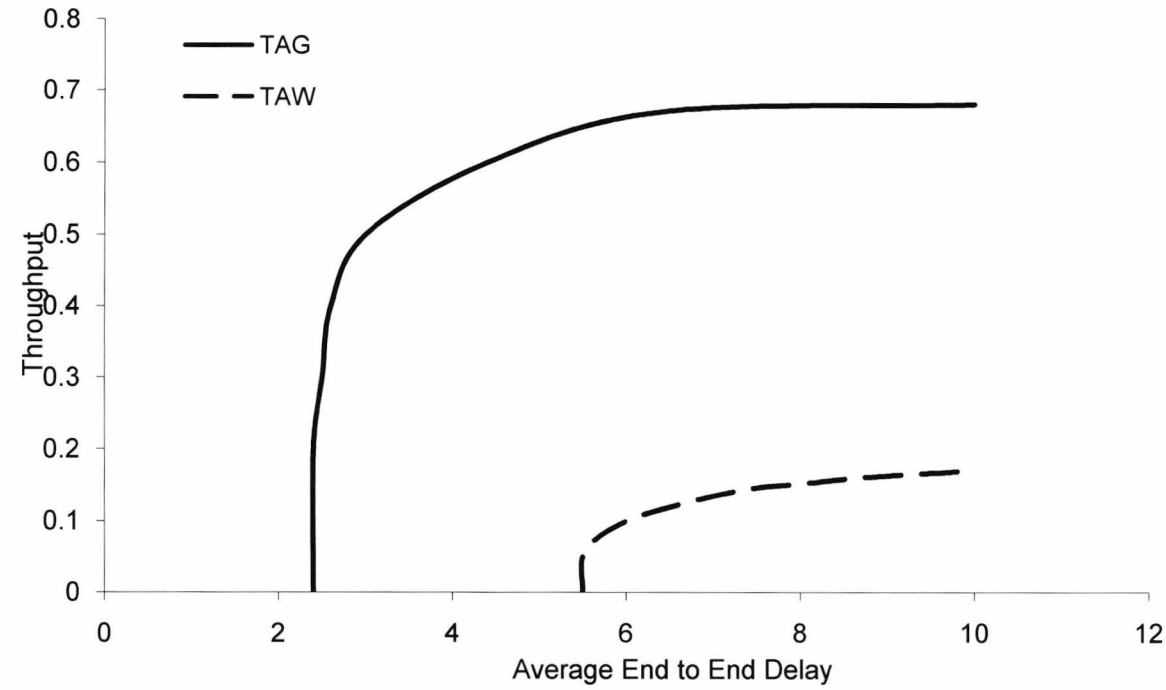


(a)

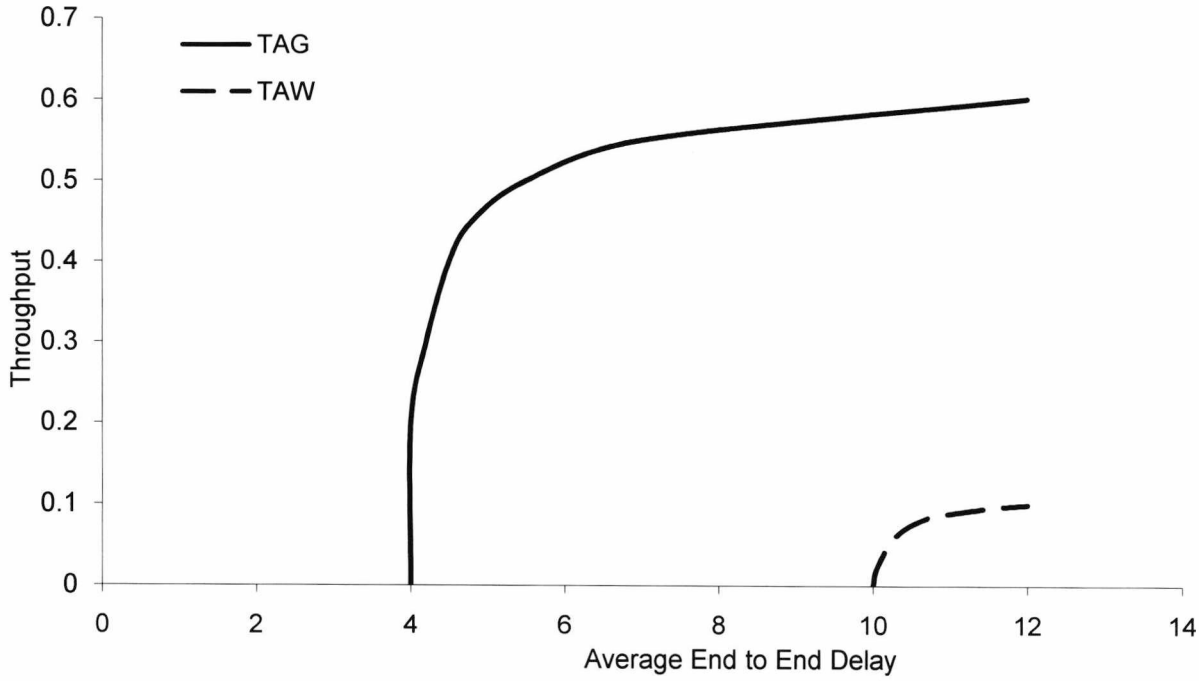


(b)

Figure 2- 10 Average delay against throughput S for large bursts (a) 3-hop network (b) 6-hop network [32]



(a)



(b)

Figure 2- 11 Average delay against throughput S for small burst (a) 3-hop network (b) 6-hop network [32]

The next scheme of reservation is JIT. As was mentioned earlier, JIT is very similar to TAG with the difference being the optical buffering in the nodes, which can be omitted in the JIT situation. The added delay is the acknowledgment propagation delay from the first core node to the source. In both cases of JIT, implicit and explicit, the equation for end to end delay is similar to TAG with the exclusion of the acknowledgement propagation delay and the optical buffering delay. In the special case of explicit, a small delay is added for the release message delay.

The main difference though in the two schemes of JIT and TAG is not only the buffering in the delay but also the channel discovery delay which is different in each case. Channel discovery in general is highly dependent on the information that a scheme has to process, thus the more information the BHC contains the larger the delay becomes in channel discovery. In TAG the BHC contains information of the burst size that is used only in the core nodes to calculate the time of the burst arrival and the guard time needed for the BHC to arrive to the next node before the burst. In implicit JIT where information is also sent about the burst length, the channel discovery time is larger as each core node has to calculate the time of arrival and departure of the burst and store the information until the time comes where the core node will need to release the channel. Finally in the explicit JIT the reservation is made and kept until a release message is received, so the BHC only contains information concerning the destination of the burst, thus minimum delay appears for channel discovery.

As far as delay is concerned the JIT and TAG only differ by some microseconds depending on the equipment used in each node and on how fast they can process information [35, 36].

The JET reservation technique as with JIT has its end to end delay mostly dependent on the propagation delay of the burst, but unlike JIT the offset time between the BHC and the burst plays a more significant role on the final end to end delay. The equation for the end to end delay is similar to the case of JIT except, of course, the delay of the guard time is significantly less in JIT than the delay of the offset time in JET as it is calculated for the whole trip of the burst in the latter.

A very important observation is that the channel discovery delay in a JET reservation solution has a larger value than in any other scheme for the reason of a much greater computational effort that has to be made in each edge node in order to find if a channel is available during the specific period of time that the channel in needed. This, of course, increases the overall end to end delay of a single burst but because the

system only reserves a channel at an exclusive period of time (the time that the burst occupies any given link) the network can handle much more traffic than any other reservation technique. This has effect of reducing the average burst end to end delay in the network as traffic is increased. In Figure 2-12 taken from M.Yoo and C. Qiao [37] the throughput against the latency for a JET scheme compared to a TAG reservation scheme is shown. It can be seen that the end to end delay in the JET scheme is much lower than with TAG and that it has a better throughput performance as well. This of course is expected as JET can accommodate a lot more traffic than other schemes, so the larger delay of an individual burst compared to TAG is smaller on average as traffic increases.

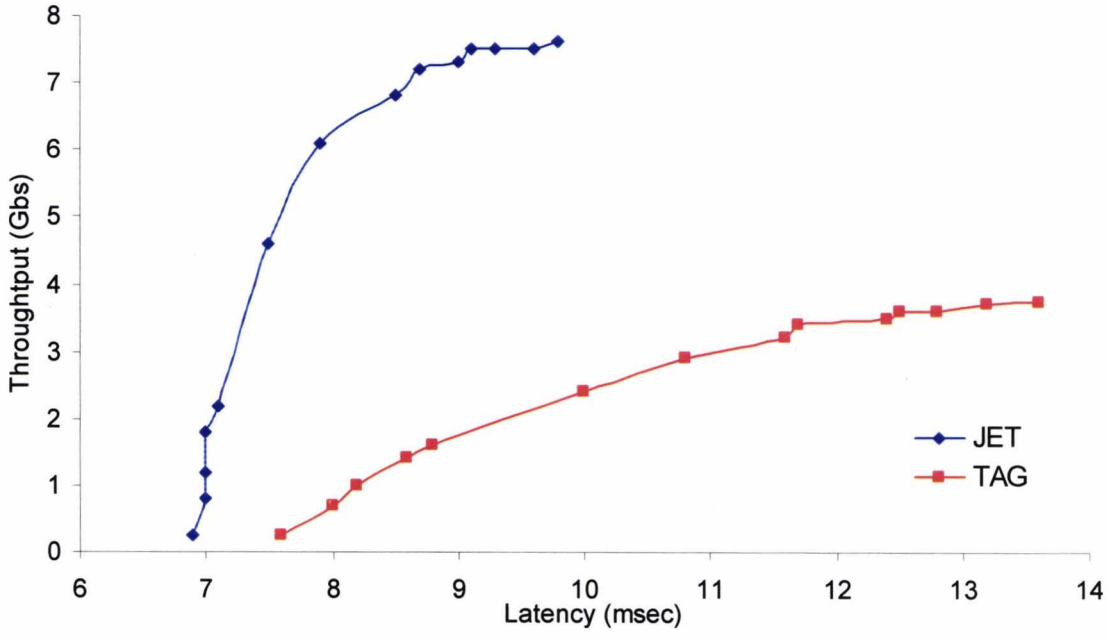


Figure 2- 12 Throughput against average end to end delay

The final scheme that is analysed is the INI reservation technique where the end to end delay is increased compared to JET but reduced compared to TAW. This is done by reducing the round trip delay between source and destination. The equation is similar to the equation for TAW but with an added delay for the offset time. The differences that appear are that the reply BHC would have only to travel the number of nodes from the initiated node to the source and not from destination to source. The offset delay is of course smaller to the normal JET offset delay as the offset time does not need to be

calculated for the whole burst trip but needs to be calculated from the initiated node to the destination.

In Figure 2-13 a comparison is made between the INI, TAW and JET schemes (first presented by Karanam et al. [38]). From Figure 2-13 it can be seen that the INI reservations scheme is simply a compromise in performance between JET and TAW. The INI technique performs better as the number of hops between source and initiated node decreases but never outperforms JET when the average end to end delay is concerned. In contrast the performance in end to end delay is better than TAW as the number of hops between source and initiated node decreases. It has to be mentioned that if the initiated node is at the source then INI performs exactly as JET and if the initiated node is at the destination it resembles TAW.

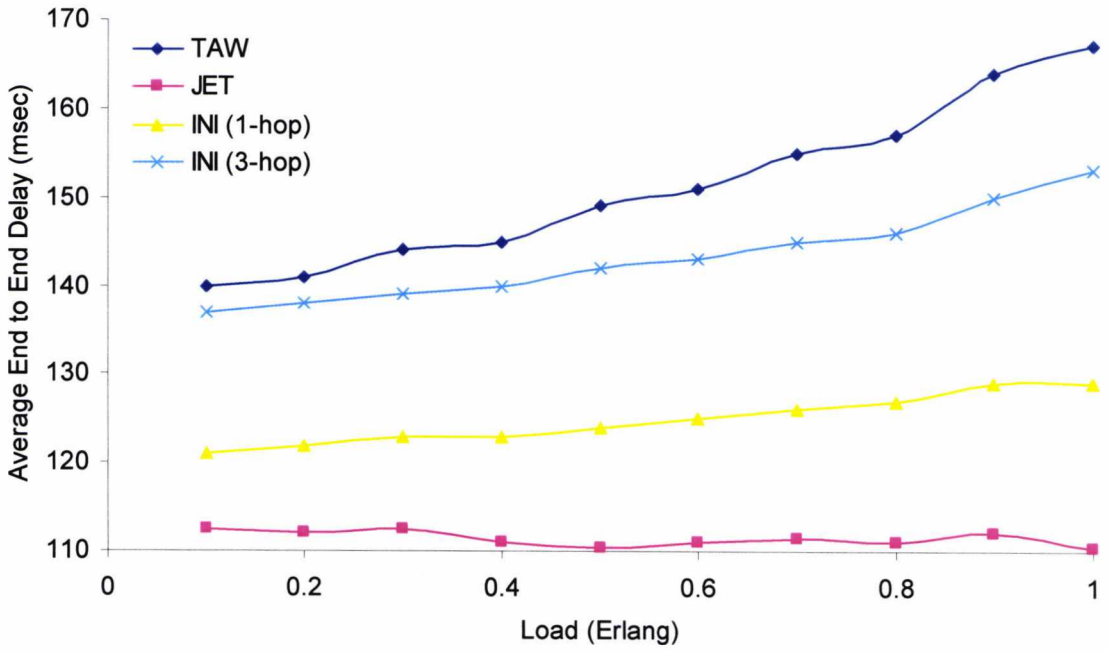


Figure 2- 13 Load against average end to end delay

2.3.2 Burst Loss Probability

The next major metric for quantifying OBS network performance is the burst loss probability. The tradeoff for each reservation scheme for having a better performance in end to end delay is having a reduced performance in burst loss probability. The significant factor that affects both end to end delay and burst dropping probability is the size of the burst as was shown with TAG in the investigation of end to end delays for the different reservation schemes.

The investigation of burst loss probability will start with the comparison of the TAW and TAG reservation techniques. In the results of Figure 2-14 obtained from Widjaja when analysing the performance of the two schemes [32] the maximum throughput of the network against the number of hops in the network is shown. Also, the author in the same paper presents in the case of TAG the propagation delay at which a burst is being transmitted has no effect on the maximum throughput whereas in TAW the throughput decreases as the average burst size decreases. This can be interpreted as follows: as the burst length decreases the burst loss probability in a TAW system increases rapidly whereas burst length has no ill effect on a network with the TAG scheme implemented. Also TAG produces far less burst loss dropping probability for small bursts compared to TAW and in case of large bursts TAW outperforms TAG. Another useful point that has to be made is that the performance of TAW in burst loss probability can be increased significantly if large buffers are installed at the nodes as they will increase the time of storing bursts and thus increasing the number of burst waiting for transmission after a channel has been freed. However increasing the size of the buffers is not as easy as it sounds. With increasing the size of buffers the complexity of the node's construction is increased.

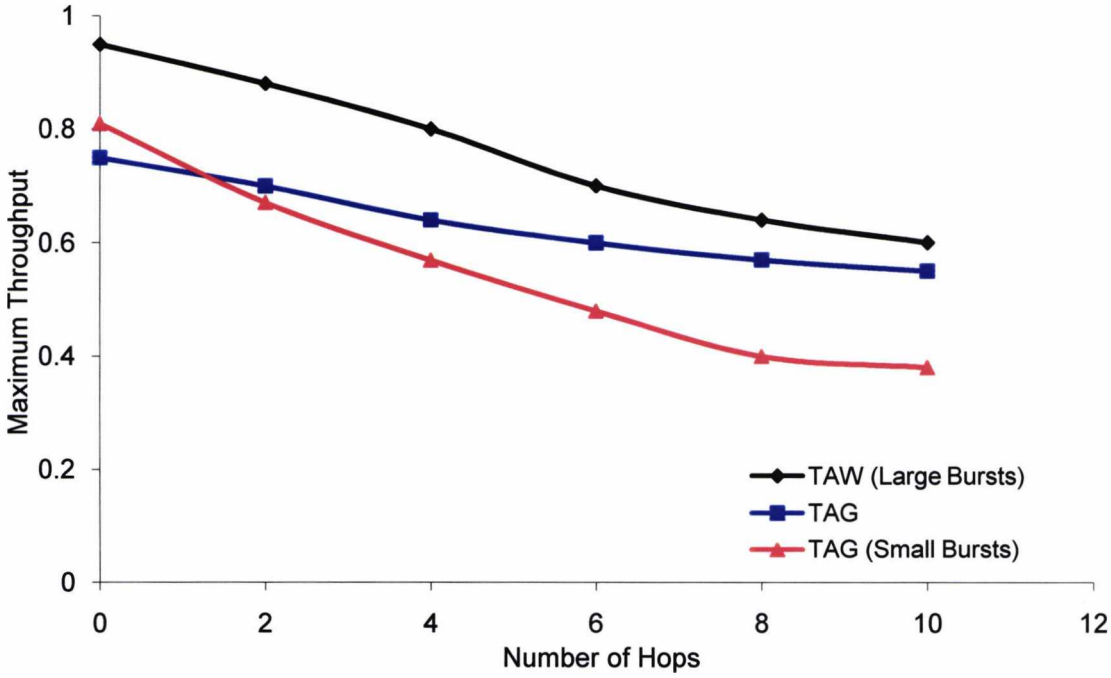


Figure 2- 14 Throughput against Number of hops for a TAG and TAW reservations schemes

Figure 2-15 shows the burst loss probability of INI compared to the TAW and JET reservation schemes. The figure was reproduced from Karanam [34] where the INI signaling scheme is described. It depicts the burst loss probability against load in Erlangs for all three reservation techniques. The figure clearly shows the increase in burst dropping as the offered load is increased and shows the JET scheme as being worse than TAW and INI. This was expected as in TAW the burst loss probability is low since burst dropping mostly occurs at source nodes, whereas in JET and INI burst dropping can occur anywhere in the network. Again, as with end to end delay, it can be seen that INI provides a middle level performance between TAW and JET.

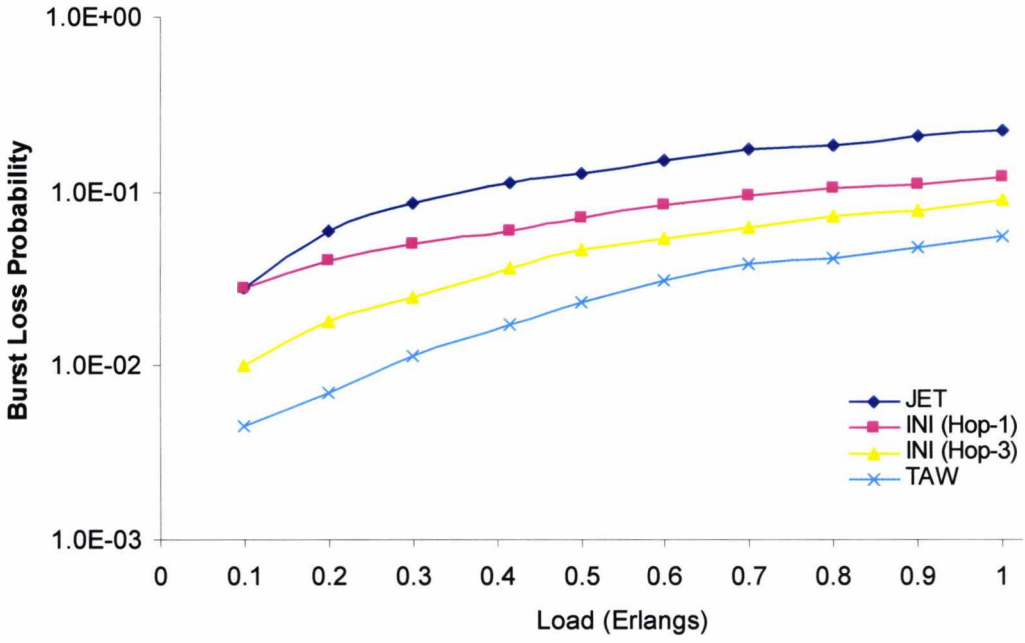


Figure 2- 15 Burst loss probability against load for TAW, INI and JET reservation schemes

2.3.3 Other comparison aspects

The TAW scheme presented seems to be the most cost effective and simple to build. The two main problems, however are the high end to end delay which makes it unattractive and the very low bandwidth utilization. The next candidate is probably TAG but having to have optical buffering increases the complexity of the system and so the cost rises. Although the bandwidth utilization and end to end delay time are superior to TAW it has a very high burst loss probability which makes it undesirable. The reason for this is that the higher layers in the network have to resend information thus still wasting bandwidth. In some cases rescheduling can be done in the optical edge nodes [39-41]. If this optical rescheduling is implemented then complexity increases, thus the cost of implementation become very high.

JIT shares a lot of similarities with TAG but optical buffering is not required so making the cost lower than TAG. But again as with TAG, JIT has a high burst loss probability although end to end delay is far better than TAW and it has better bandwidth utilization than TAW, thus making it a viable solution in some cases. [42, 43].

The next scheme analysed was JET, where the delay is decreased considerably compared to all other schemes and where high bandwidth utilization is provided. Burst loss probability is not as severe as in TAG and JIT but still lacks the performance of

TAW. No buffering is needed in JET but the complexity of the circuitry in nodes is high as each optical node has to accommodate fast arriving BHCs that need to be read and used for bandwidth reservation quickly in order that no bursts are dropped due to queuing of BHCs and thus increasing the burst loss probability with this scheme.

The INI reservation technique provides a middle choice between JET and TAW. It trades off burst loss probability and end to end delay for TAW and JET respectively. In more detail, the INI scheme for burst transmission before the intermediate node operates exactly as the TAW scheme which has a low burst loss probability but high end to end delay and so reduces the overall loss that is high after the intermediate node which the transmission resembles the JET protocol which in turn has a low end to end delay. Thus the INI scheme can be said to "average" the burst loss probability values and the end to end delay values of JET and TAW. Although INI provides a good compromise between TAW and JET which are the most known reservation techniques it still lacks the ability to be implemented in the near future as optical buffering is needed.

Thus JET seems to be the most popular of the schemes and a large amount of research has been spent into reducing the burst dropping probability which is its main disadvantage. Research has shown that burst loss probability can be reduced significantly to acceptable levels, thus the research in this thesis is mostly focused on such a reservation technique.

2.4 Summary

In order to more precisely understand how the reservation schemes compare, table 2-1 provides an overview of all schemes presented so far. Before starting comparing using the table it is needed to understand some typical values for high, low or medium delay and bandwidth. By low delays we assume a typical 0.0004 seconds if a 1Mbps link is used. High delays are assumed to have typical values of 40 seconds. Typical values for bandwidth/utilisation are for high utilisation a 1 Mbps link can support 35 users where as for low bandwidth a similar link can support only 10 users. Having these typical values in mind we can now go on to compare the different reservation schemes [49].

Although circuit switching has provided in the past and in some cases provides today a reliable method of communication, it has certain draw backs such as low bandwidth

utilization (a typical 1Mbps link can support 10 users) and high average delay time (40 seconds for sending a 40Mb). Advancement in technology has increased the amount of information inside networks, and thus circuit switching is not capable of providing an adequate choice. It has to be noted that the amount of traffic will increase even more and so a better solution on network technology is needed that will be capable of handling the information of today and of the future. Such a technology is provided by OPS because it has minimum delays (typical 0.0004 seconds), low burst loss and high bandwidth utilization (a typical 1Mbps link can support 35 users). Its adaptability is very high also that makes it a perfect candidate for future networks. The problem that arises is that technology, although it is advancing at very fast, still lacks the capability of producing optical components that will make OPS feasible and so communication companies are not willing yet to make the step towards OPS. A solution to the problem with OPS is to find a middle ground solution that can provide sufficient speeds and be versatile enough to accommodate different types of networks. This type of networking is OBS where a high amount of research is spent on OBS networks. In the chapter just presented five types of OBS reservation schemes have been presented and analysed. The final verdict can be seen in table 2-1, where the JET seems to be at the moment the leading scheme to be implemented and JIT following very close behind. It is safe to say that JET provides lower delays from all other schemes, which in the newly introduced internet technologies, where video streaming is proving to be popular, is a critical advantage from other OBS schemes. JET provides high bandwidth utilization, higher than any other OBS schemes, which makes it a very good choice for companies investing on optical networks. Both just mentioned advantages of JET are overshadowing, in our opinion, the main disadvantage of higher burst loss probability from other schemes. The second best scheme – JIT provides a mediocre solution for OBS networks as in all comparison criteria it provides small improvements from TAW, TAG and INI without making a significant advantage over other schemes. In the case of using TAW as a reservation scheme for OBS although it provides the lowest burst loss probability, this advantage is overshadowed by the high average delay time and the low bandwidth utilization which are the main disadvantages of circuit switching in the first place. Finally, TAG is the only solution that needs optical buffering and thus an expensive solution. It has also a very high burst loss probability which makes it unsuitable and expensive choice for OBS. The research has been mostly focused on JET

and JIT where some test-beds have been created which are going to be described in more detail in the next chapter.

Table 2- 1 Overview of reservations techniques in OBS.

Reservation Scheme	Average Delay Time	Burst Loss Probability	Optical Buffering	Bandwidth Utilisation	Adaptivity/Complexity
Circuit Switching	High	Low	Not Required	Low	Low
OPS	Low	Low	Required	High	High
TAW	High	Low	Not Required	Low	Low
TAG	Medium	High	Required	Medium	Medium
JIT	Medium	High	Not Required	Medium	Low/Medium
JET	Low	Medium	Not Required	High	Medium
INI	Medium	Medium	Required	Medium	Medium

2.5 References

[1] T. Freeman, "Get Ready for Growth," *Fibersystems Europe in Association with LIGHTWAVE Europe*, 2005, p. 3.

[2] Cisco Systems Inc, *CCNA 3 and 4: Companion Guide* Cisco Press, 2003.

[3] F. Mazda, *Telecommunication Systems and Applications*: Focal Press, 1996.

[4] B. Lee, J. M. Cioffi, S.Jagannathan and M. Mohseni, "Gigabit DSL", *IEEE Trans. On Communications,* Vol. 55, No 9, September 2007.

[5] J. Y. Hui, K.W. Cheung,"Optical Versus Electronic Switching for Broadband Networks", IEEE Network, November-December 1996, pp. 21-25.

[6] K. Fukuchi, T. Kasamatsu, M. Morie, R. Ohhira, T. Ito, K. Sekiya, D. Ogasahara and T. Ono, "10.92-Tb/S (273 / Spl Times / 40-Gb/S) Triple-Band / Ultra-Dense Wdm Optical-Repeatered Transmission Experiment," in *Optical Fiber Communications Conference (OFC)*, Anaheim, USA, 2001.

[7] S. Bigo, Y. Frignac, G Charlet, W. Idler, S. Borne, H. Gross, R. Dischler, W. Poehlmann, P. Tran, C. Simonneau, D. Bayart, G. Veith, A. Jourdan and J. Hamaide, "10.2 Tbit/S (256x42.7 Gbit/S Pdm/Wdm) Transmission over 100 Km Teralight Fiber with 1.28 Bit/S/Hz Spectral Efficiency," *Optical Fiber Communications Conference (OFC)*, Anaheim, USA, 2001.

[8] D. K. Hunter and I. Andonovic, "Approaches to Optical Internet Packet Switching," in *IEEE Communications Magazine*, 2000, pp. 116-122.

[9] G. I. Papadimitriou, C. Papazoglou, and A. S. Pomportsis, "Optical Switching: Switch Fabrics, Techniques, and Architectures," *Journal of Lightwave technology,* vol. 21, February 2003, pp. 384-405.

[10] S. Yao, B. Mukherjee and S. Dixit, "Advances in Photonic Packet Switching: An Overview," *IEEE Communications Magazine*, 2000, pp. 84–94.

[11] A. Bianco, E. Leonardi, M. Munafo, F. Neri, W. Picco, "Design of Optical Packet Switching Networks", *Global Telecommunications Conference(GLOBECOM),* Vol. 3, 17-21 November 2002, pp. 2752 - 2756

[12] G. Armitage, "MPLS: The Magic Behind the Myths," *IEEE Communications Magazine*, 2000, pp. 124–131.

[13] D. O. Awduche, "MPLS and Traffic Engineering in IP Networks," in *IEEE Communications Magazine*, 1999, pp. 42–47.

[14] T. Li, "MPLS and the Evolving Internet Architecture," *IEEE Communications Magazine,* pp. 38-41, December 1999.

[15] R. Doverspike and J. Yates, "Challenges for MPLS in Optical Network Restoration," *IEEE Communications Magazine*, 2001, pp. 89–96.

[16] P. B. Hansen, S. L. Danielsen, and K. E. Stubkjaer, "Optical Packet Switching without Packet Alignment," *24th European Conference in Optical Communications* Madrid.

Spain, 1998.

[17] S. L. Danielsen, P. B. Hansen, and K. E. Stubkjaer, "Wavelength Conversion in Optical Packet Switching," *Journal of Lightwave Technology,* pp. 2095–2108, 1998.

[18] C. Guillemot *et. al.*"Transparent optical packet switching: The European ACTS KEOPS project approach", *Journal of Lightwave Technology*, pp 2117-2134, 1998.

[19] D. K. Hunter, M. C. Chia and I. Andonovic, "Buffering in optical packet switches." *Journal of Lightwave Technology*, pp. 2081–2094, 1998.

[20] G. Rossi, O. Jerphagnon, B.-E. Olsson, and D. J. Blumenthal, "Optical SCM Data Extraction Using a Fiber-Loop Mirror for WDM Network Systems," *IEEE Photonics Technology Letters,* vol. 12, pp. 897-899, July 2000.

[21] G. Bendelli, M. Burzio, P. Gambini, and M. Puleo, "Performance Assessment of a Photonic ATM Switch Based on a Wavelength-Controlled Fiber Loop Buffer," *Optical Fibre Communication Conference (OFC)*, California. USA, 1996, pp. 106-107.

[22] A. Bononi, "Analysis of Hot-Potato Optical Networks with Wavelength Conversion," *Journal of Lightwave Technology,* vol. 17, pp. 525-534, April 1999.

[23] K. Hunter, M. C. Chia, I. Andonovic, K. M. Guild, A. Tzanakaki, M. J. O'Mahony, L. D. Bainbridge, M. F. C. Stephens, R. V. Penty and I. H. White, "Waspnet: A Wavelength Switched Packet Network," *IEEE Communications Magazine*. vol. 37, 1999, pp. 120-129.

[24] I. Glesk, K. I. Kang, and P. R. Prucnal, "Ultrafast Photonic Packet Switching with Optical Control," *Optics Express,* vol. 1, pp. 126–132, 1997.

[25] M. Murata and K. Kitayama, "Ultrafast Photonic Label Switch for Asynchronous Packets of Variable Length," in *IEEE INFOCOM*, New York, USA, 2002, pp. 23–27.

[26] J. Lu, J. Wang, "Analytical Performance Analysis of Network-Processor-Based Application Designs," in *Proceedings of the 15th International Conference on Computer Communications and Networks* Arlington, USA, 2006.

[27] T. Wolf, M. A. Franklin, "Performance Models for Network Processor Design " *IEEE Transactions on Parallel and Distributed Systems,* vol. 17, pp. 548-561, June 2006.

[28] G. Rossi, O. Jerphagnon, B.-E. Olsson, and D. J. Blumenthal, "Optical SCM Data Extraction Using a Fiber-Loop Mirror for WDM Network Systems," *IEEE Photonics Technology Letters,* vol. 12, pp. 897-899, July 2000.

[29] H. J. Lee, V. Hernandez, V. K. Tsui, and S. J. B. Yoo, "Simple, Polarization-Independent, and Dispersion-Insensitive SCM Signal Extraction Technique for Optical Switching Systems Applications," *Electronics Letters,* vol. 37, pp. 1240–1241, September 2001.

[30] G.C. Giakos, S. Sumrain, "Photonic Molecular Processors", *Instrumentation and Measurement Technology Conference, (IMTC),* Ottawa, Ontario, *2005,* pp. 1124-1128.

[31] A. Detti and M. Listanti, "Application of Tell and Go and Tell and Wait Reservation Strageties in an Optical Burst Switching Network: A Perfomance Comparison," in *8th IEEE International conference on telecommunications ICT,* Bucharest, Romania, 2001.

[32] I. Widjaja, "Performance Analysis of Burst Admission-Control Protocols," *IEE Proceedings-Communications,* vol. 142, pp. 7-14, February 1995.

[33] G. C. Hudek, D. J. Muder, "Signaling Analysis for a Multi-Switch All-Optical Network," in *IEEE International Conference on Communications (ICC)*, Seattle, USA, 1995, pp. 1206 - 1210

[34] J. Y. Wei, J. L. Pastor, R. S. Ramamurthy and Y. Tsai, "Just-in-Time Optical Burst Switching for Multi-Wavelength Networks," in *5th International Conference on Broadband Communication (BC)*, Hong Kong, 1999, pp. 339-352.

[35] J. Y. Wei,R. I. McFarland, Jr., "Just-in-Time Signaling for WDM Optical Burst Switching Networks," *Journal of Lightwave Technology,* vol. 18, pp. 2019 - 2037, December 2000.

[36] I. Baldine, M. Cassada, A. Bragg, G. Karmous-Edwards, D. Stevenson, "Just-in-Time Optical Burst Switching Implementation in the ATDnet All-Optical Networking Testbed," *IEEE Global Telecommunications Conference GLOBECOM*, San Francisco, USA, 2003, pp. 2777 - 2781.

[37] M. Yoo, C. Qiao, "Just-Enough-Time (Jet): A High Speed Protocol for Bursty Traffic in Optical Networks," *Digest of the IEEE/LEOS Summer Topical Meetings,* pp. 26 - 27 August 1997.

[38] V. V. R. Karanam, J. Jue, "Intermediate Node Initiated (INI) Signalling: A Hybrid Reservation Technique for Optical Burst-Switched Networks," *Optical Fiber Communications Conference (OFC)*. vol. 1 Atlanta, USA, 2003, pp. 213 - 215.

[39] C. Lan, C. Bauer, "Rescheduling-Based QoS Control Algorithms for Optical Burst Switching," *Proceedings in International Conference on Communication Technology (ICCT)*, Beijing, China, 2003, pp. 617 - 621.

[40] S. K. Tan, G. Mohan, K. C. Chua, "Burst Rescheduling with Wavelength and Last-Hop Fdl Reassignment in WDM Optical Burst Switching Networks," *IEEE International Conference on Communications (ICC)*. vol. 2 Anchorage, ALASKA, 2003, pp. 1448 - 1452

[41] S. K. Tan, G. Mohan, K. C. Chua, "Algorithms for Burst Rescheduling in WDM Optical Burst Switching Networks," *International Journal of Computer and Telecommunications Networking* vol. 41, pp. 41 - 55, January 2003.

[42] GI. Baldine, G. N. Rouskas, H. G. Perros, D. Stevenson, D. Stevenson, "Jumpstart: A Just-in-Time Signaling Architecture for WDM Burst-Switched Networks," *IEEE Communications Magazine*. vol. 40, 2002, pp. 82 - 89.

[43] G. N. Rouskas, "Jumpstart: A Just-in-Time Signaling Architecture for WDM Burst-Switched Networks," *Digest of the IEEE/LEOS Summer Topical Meetings,* pp. TuA1-19 - TuA1-20 July 2002.

[44] C. M. Gauger, "Trends in Optical Burst Switching," *Proceedings of the SPIE*, Orlando, USA, 2003, pp. 115-125

[45] M. Yoo and C. Qiao, "A Novel Switching Paradigm for Buffer-Less WDM Networks," in *Proceedings on Optical Fibre Communication (OFC)*, San Diego, USA, 1999, pp. 177-179.

[46] J. S. Turner, "Terabit Burst Switching," *Journal of High Speed Networks,* vol. 8, pp. 3-16, January 1999.

[47] C. Qiao, "Labeled Optical Burst Switching for IP and WDM Integration," in *IEEE Communications Magazine*. vol. 38, 2000, pp. 104-114.

[48] S. Verma, H. Chaskar, and R. Ravikanth, "Optical Burst Switching: A Viable Solution for Terabit IP Backbone," *IEEE Network,* vol. 14, pp. 48-53, November/December 2000.

[49] A. Johnson, LAN Switching and Wireless, CCNA Exploration Labs and Study Guide, Cisco Press 2nd Revised edition 2008.

CHAPTER 3

OBS ARCHITECTURE

As was discussed in Chapter 2, several algorithms have been investigated in the literature concerning the operation of OBS networks. In Chapter 2 a discussion on reservation techniques in OBS networks was presented, whereas in this chapter algorithms concerned with burst assembly, CoS and scheduling will be discussed. Although many algorithms have been proposed for burst assembly, CoS and scheduling, only those used in the research of this thesis will be discussed in detail, others will only be briefly mentioned to present a clearer view of the algorithms that exist.

In this chapter, a way of explanation for burst loss probability in core nodes for the most frequently used scheduling algorithms are introduced by using queuing theory to partially describe the scheduling algorithms. The explanations will then help us describe, in Chapter 5, the new scheduling algorithms which we introduce in this thesis and compare with the results from our simulation model.

3.1 Reasons for Selection of Algorithms

The literature in OBS algorithms on burst assembly, class of service and scheduling is vast. Although we have investigated many algorithms on all areas concerning OBS networks, this thesis will only present a few. The selection of the algorithms presented was based on the hardware available at the time, the low cost of components and the relevance of our aims. Thus as at the current time wavelength conversion is not yet fully possible and it is not cost effective algorithms that relied on wavelength conversion were not investigated. Optical memory is still not available and the only close technology is optical buffering which is not cost effective, thus algorithms that involved optical memory or optical buffering were not investigated. Finally algorithms used were mostly based on simple solutions in order to introduce a low cost OBS network.

3.2 Burst Assembly Algorithms

In an optical burst-switching network, the first step to forward incoming bursty traffic is to assemble the bursty data in the burst assembler. The assembler has to be able to classify the data according to the class of service, and according to the routing information. We consider two ways to assemble multiple packets into a data burst. The first is the segmented method, and the second the non-segmented. The differences between the two methods can be seen in figure 3-1.

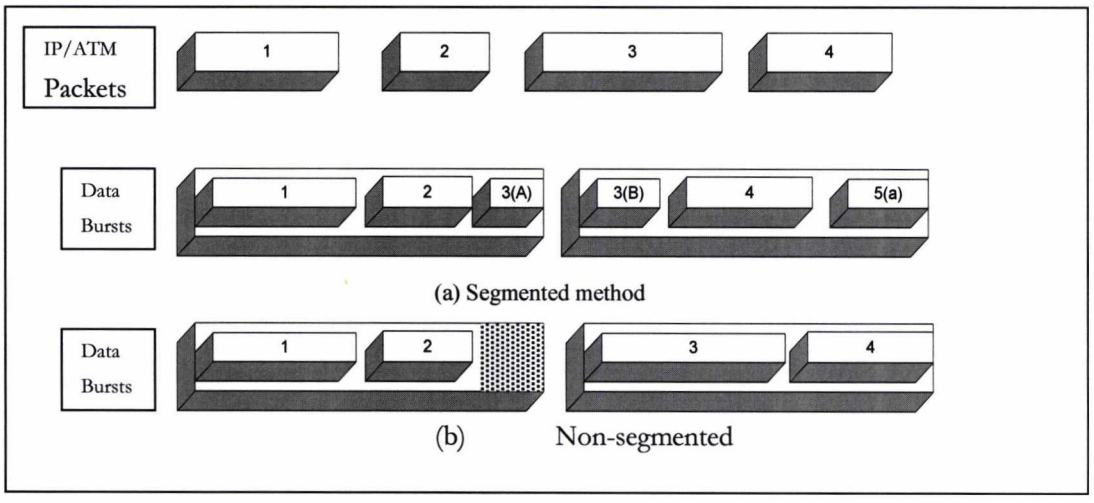


Figure 3- 1 Data burst assembly methods

The segmented method receives packets until the space for the burst is filled and in the process, the incoming packets are segmented if necessary as shown in Figure 3-1(a). The non-segmented method only takes whole packets that can fit inside the burst and uses padding in the unused space (Figure 3-1(b)). The advantage of using the segmented method is the high bandwidth utilization over to that offered by the non-segmented method. The large disadvantage though is the high complexity on the hardware and protocol design. In optical burst-switching networks there is already a high toll to be paid due to processing of the header cell in the electronic domain and so the non-segmented method seems better suited for the assembling of data bursts. A third way which will also give a higher

utilization can be implemented by using variable burst size. This is accomplished by not putting any padding in the non-segmented method, and so sending the burst as it is.

In an OBS network the offset time is usually set according to the average size of the data burst as it has a direct link to the delays that a burst experiences as it is processed through the network as will be discussed in chapter 7. So the larger the data burst the higher the offset time would have to be. For this reason the algorithm to be used for the assembly of the burst should not vary the burst sizes without having some limits set on it and should have high utilization factor. We are going to consider four algorithms:

- Fixed Assembly Time (FAT) algorithm
- Variable Assembly Time (VAT) algorithm
- Minimum Burst Length Fixed Assembly Time (MBLFAT) algorithm
- Dynamic Burst Size Decision (DBSD) algorithm

All these algorithms have a simple design that have a queue for each destination that aggregates the incoming packets to create a burst ready for transmission when a timer expires in each queue that starts with the first incoming packet. Each timer is reset every time a burst is created.

3.2.1 Fixed Assembly Time Algorithm

The algorithm was proposed by An Ge and Franco Callegati [1]. This algorithm uses a timer-counter in order to generate data bursts (i.e. in a specific time interval a burst would be created irrespective of the burst size) and uses a variable burst length. There is also a minimum length for the burst to be created. This is to keep the load on the control channel under control. In turn though this raises the question of how much time should an assembler wait until the burst size reaches the appropriate length. Therefore a limit has to be placed on the waiting time for a burst to be created.

In figure 3-2 the algorithm operation is described via a flowchart.

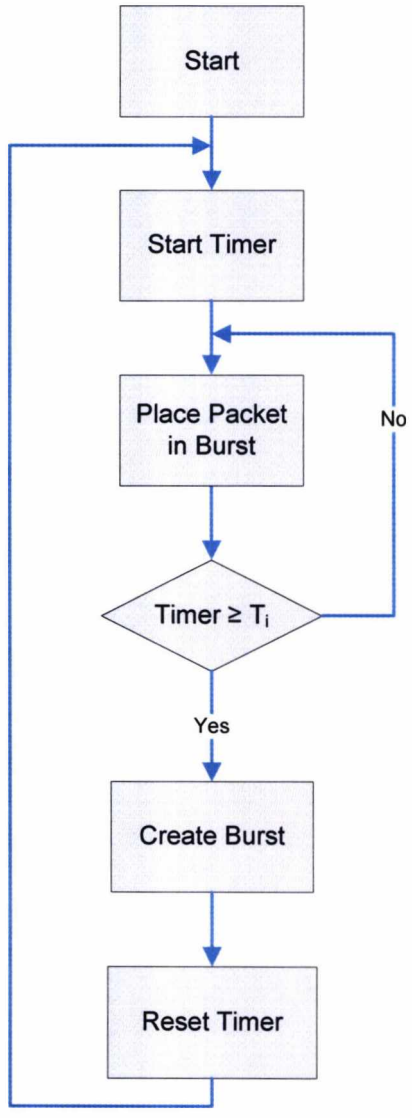


Figure 3- 2 Flow chart for the Fixed Assembly Time algorithm

The way the algorithm works has three steps, firstly a timer starts counting when the first packet arrives. The second step checks if the timer has reached a time of T_i where it creates the burst and is queued for transmission. If time T_i has not been reached the system waits for either another packet to arrive or the timer to reach the value of T_i. If at T_i the burst is less than the minimum length L_i it is padded up to length L_i. The final step is to reset the timer and wait for the next packet arrival, where the process restarts.

Although the algorithm is very simple to implement and would require a minimum amount of hardware the data burst utilization is very low. Also such an algorithm will generate continuous blocking of data burst inside an edge node. Imagine two destination queues transmitting within the same time period. The edge node would have to reserve the bandwidth for the first burst that arrives. The second burst arriving from the second destination queue node would have to be buffered. Because of the use of the same time period the bandwidth would always be reserved for the first destination queue and the data bursts from the second always buffered. The blocking problem just mentioned though can be resolved by increasing the number of wavelengths in the system and by having a good scheduling algorithm. The relationship of scheduling and burst assembly algorithms will be scrutinized in more detail in Chapter 7 of this thesis.

3.2.2 Variable Assembly Time

This algorithm is used to overcome some of the problems with the FAT algorithm mentioned before, as it has an adaptive nature depending on the load [2]. The timer has a predefined initial time that it would wait until the burst is created; this time varies depending on the load at the input of the edge node.

The algorithm assumes the network uses a single path to route each burst flow where in the extreme situation the burst transmission occupies the whole output bandwidth. The average burst length can be calculated according to equation 3-1 below as described in [2],[3],[4] and [5]:

$$Average\ BL = \xi \times (Past\ \ Average\ BL) + \eta \times (Sampled\ BL) \qquad 3\text{-}1$$

where *Average BL* is the average burst length and *Sampled BL* is the most recent average burst length. The weights ξ, η are positive numbers that satisfy ($\xi + \eta = 1$). By defining the average burst length the equation for calculating the assembly time T_i for each queue can also be defined as in [2]:

$$T_i = \alpha \times \frac{Average\ BL}{Bandwidth\ of\ all\ Channels} \qquad 3\text{-}2$$

where $\alpha \geq 1$ is called the assembly factor, and is equal to the number of queues the output link shares (i.e. the number of destinations) as explained in more detail in [2].

For us to understand better how the algorithm operates we present in figure 3-3 the flowchart of the VAT assembly algorithm.

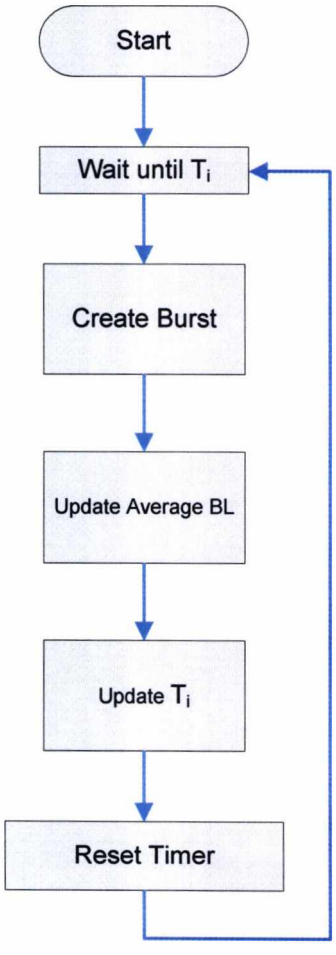


Figure 3- 3 Flow chart for the Variable Assembly Time algorithm

A timer starts counting when the first packet arrives, when the timer reaches a time of T_i it creates the burst which is queued for transmission. After the burst is created the average burst length is calculated using equation 3-1. Having calculated the Average BL the algorithm calculates the new time T_i for the queue timer from equation 3-2 and resets it to the new value.

In general VAT is more flexible than FAT but involves more complexity in changing the timer value during the operation of an OBS edge node.

3.2.3 Minimum Burst Length Fixed Assembly Time

The algorithm is an advancement of the FAT algorithm [2]. It introduces another expiration factor for generating a burst. The systems will create a burst when either a timer expires or a minimum burst length has been achieved. Figure 3-4 show the flowchart of the operation of the MBLFAT algorithm.

A timer starts counting when the first packet arrives and the burst assembly length starts adding the new packets. If the timer reaches a time of T_i it creates the burst, which is queued for transmission and cancels the counting of the burst length, but if the Burst length reaches a minimum burst length of BL_I the burst is created and the timer is cancelled The timer and the burst length are reset to 0 and wait for the next packet arrival where they restart.

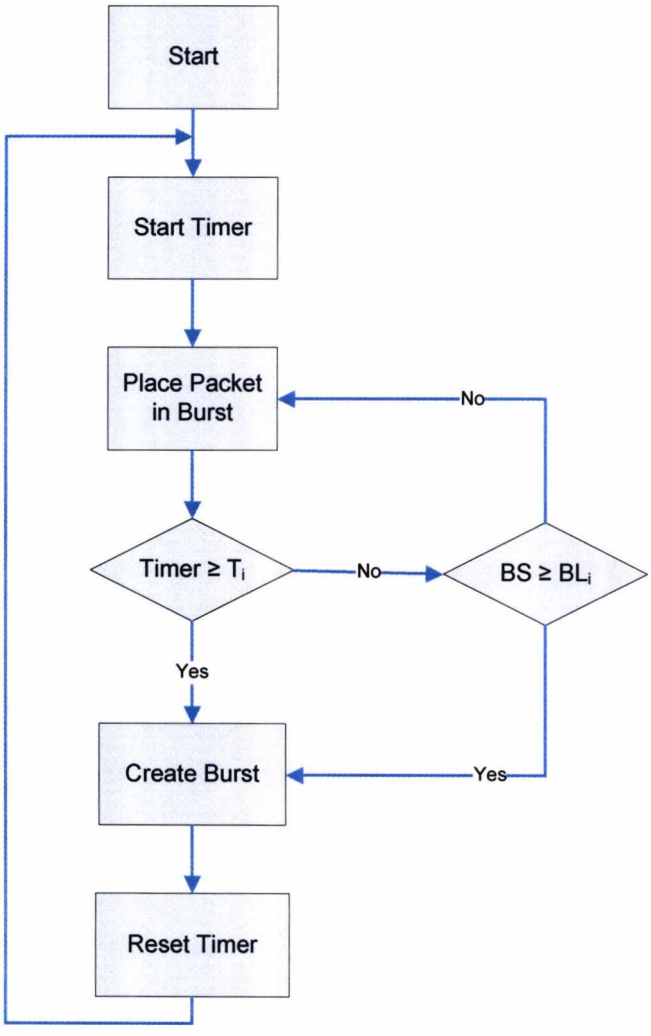


Figure 3- 4 Flow chart for the Minimum Burst Length Fixed Assembly Time algorithm

3.2.4 Dynamic Burst Size Decision (DBSD) algorithm

A better algorithm was implemented by Se-Yoon Oh, Hyun Ha Hong and Mincho Kang [6] where they used hysteresis characteristics in the queuing model of an edge node. In this algorithm three parameters exist. One is a timer (T_I) that if exceeded will send the burst data no matter how large it is and then it resets. A second parameter (N_I) counts the number of times, after a successful completion of a burst; the queue size went over and below certain upper and lower limits, respectively. The third parameter (BS) is there to see if the lower or the upper limit of the burst size has been reached.

The system includes multiple steps of queue sizes. So we have multiple pairs of upper and lower queue size limits. This can be seen more clearly in figure 3-5. There is also another upper limit on the count number (N_i). This is the number of times we crossed over another step. So if the queue size after a burst has been assembled has passed M times into another step then it moves to that step. This means that the burst size will also change. The crossover to other steps only occurs when there is an increase or decrease of the traffic load from the current one.

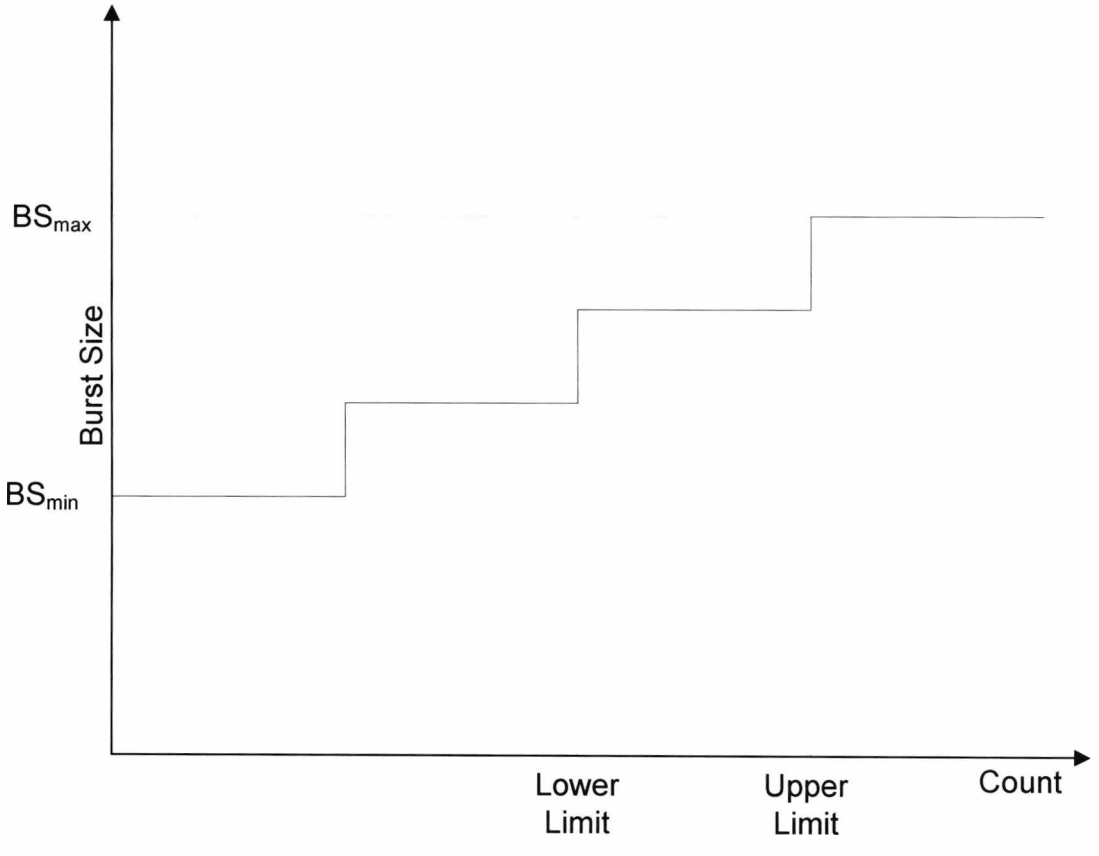


Figure 3- 5 Burst size decision scheme

The purpose of using this model is to avoid any large variation in burst size when using one threshold. Here we have multiple steps depending on the amount of traffic received from the edge node. We can actually determine the burst size either continuously or discretely. In this model the discrete method is used where several stable states exist with each having an upper and lower limit of the count parameter. To understand better how the

algorithm operates figure 3-6 shows the operation of the DBSD algorithm with the aid of a flowchart.

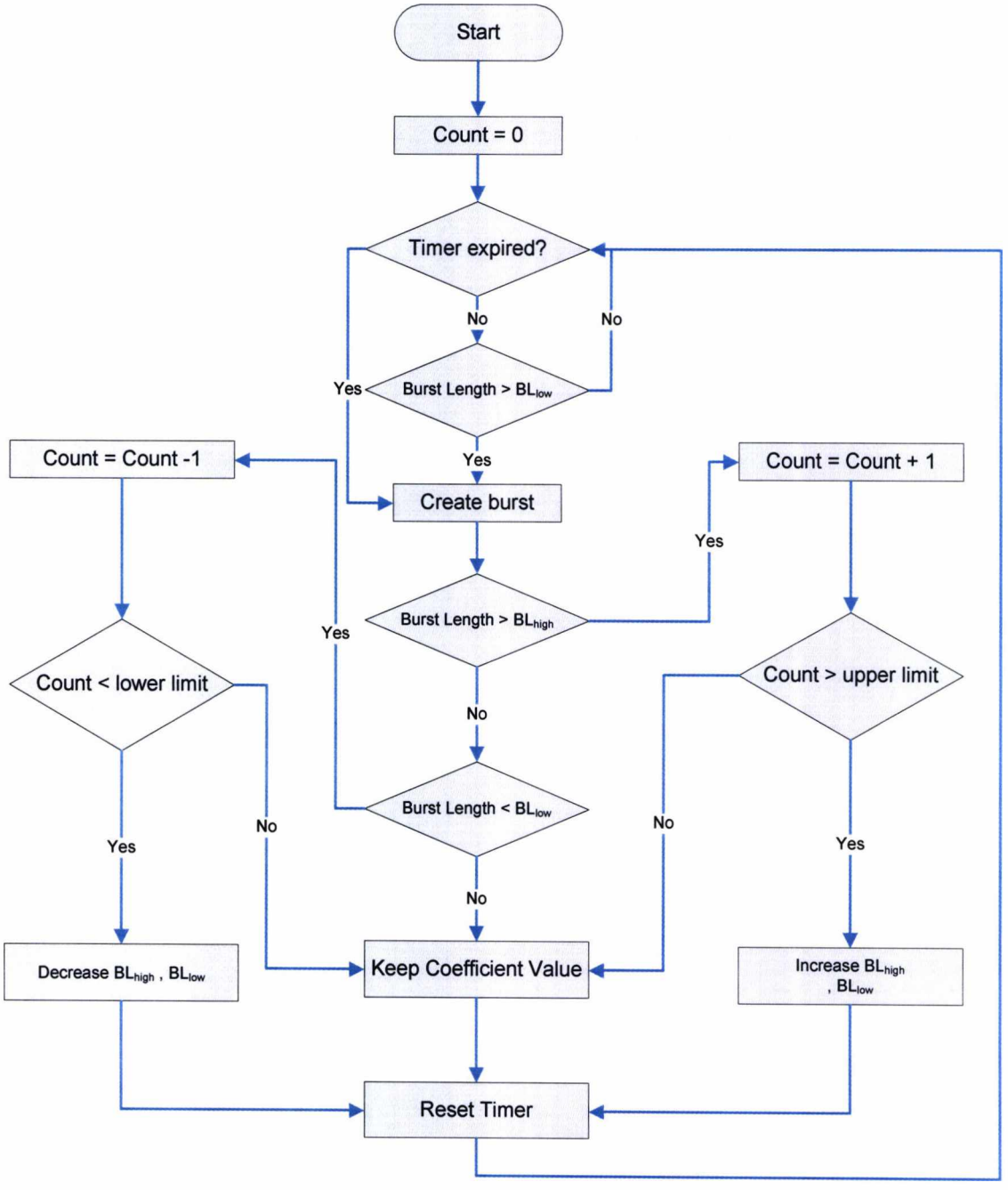


Figure 3- 6 Flow chart for the Dynamic Burst Size Decision algorithm

When a packet first arrives the count is set to zero with a predefined assembly time. After the timer has expired or the minimum burst length has been reached (i.e. BL_{low}) the burst is created, the algorithm then compares the burst length against the upper limit of the burst length (BL_{high}). If the current burst length is larger than BL_{high} the count is increased by one. If now the count limit has been reached then the burst length limits are increased and the algorithm moves the limits to the next step as was shown in figure 3-5. On the other hand if the burst length is not higher than the BL_{high} the burst length is now compared to BL_{low}. If the burst length is lower than BL_{low} the count is decreased by one step and the count value is compared to its lower limit, where if the count value is lower than its lower limit then the algorithm deceases by a step and thus decreases the limits of the burst length (BL_{high} and BL_{low}). The final option is that the burst length resides between the two limits (BL_{high} and BL_{low}), where the timer is simply reset.

3.3 Class Of Service (CoS) Algorithms

There can be two ways to accomplish operation of quality of service in OBS networks. As packets arrive with different CoS, packets with similar destinations and different CoS can be aggregated in single bursts. The other way is to aggregate packet of similar destinations and CoS in the same burst. The main problem that exists in the first method is that CoS algorithms would have to know the CoS of all the packets available in the burst in order to be able to separate the CoS of the actual bursts and also higher intelligence hardware is needed in core nodes for distinguishing the burst's CoS. In the latter method where all the packets with similar destination and CoS are aggregated in a single burst can perform CoS in OBS networks a lot more simply as the burst will have the same CoS as the packets that are contained within it.

3.3.1 Additional offset time

Offset-based schemes offer quality of services by introducing different offset times on different class according to their priority. It heavily relies on the fact that a greater offset

time translates into an earlier reservation and thus a higher probability of successful reservation [7]. The exact way this works can be explained by assuming that we have two classes (0 and 1) and class 0 has higher priority than class 1.

Figure 3-7 shows a single channel having a burst of class 1 arriving before a burst of class 0 but burst of class 1 is being dropped as the reservation for the class 0 burst was done on time $T_0<T_1$, where T_0,T_1 are the times of the control packets of bursts of CoS 0 and 1 respectively.

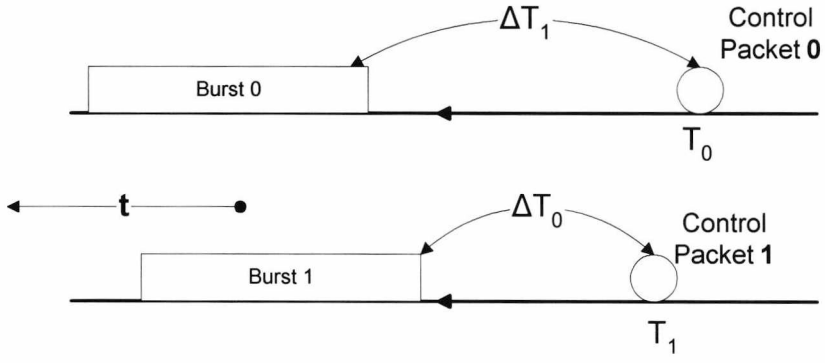


Figure 3-7. Optical Burst Switching with CoS: the additional offset time approach.

This approach work on a Just-Enough-Time(JET) OBS system but will not work on a Tell-And-Go(TAG) system as there is no offset between the control packet and the burst.

This approach has the advantage that it does not have to use any fibre delay lines (FDLs) and so the cost and the complexity are reduced to a minimum, though an extension to this approach exists which does use FDLs [8]. A higher priority burst will have a higher probability of reserving the channel, or if not available, reserving the FDL, earlier than a lower priority burst. This has as an effect to reduce blocking probabilities in high priority classes.

3.3.2 Preemptive reservation

This approach is based on higher priority bursts having higher priorities in reserving wavelengths than lower priority bursts [9].

A switch keeps updating its usage table. When it receives a request for a burst it tries to identify a wavelength available to the request according to its class of service. If the attempt is successful the request is scheduled and the usage table of the switch is updated. If on the other hand it is unsuccessful it uses a verification algorithm that finds a lower priority burst that can be pre-empted and the new burst is scheduled in the place of the pre-empted burst. If even the verification algorithm does not find a burst to be pre-empted then it is simply dropped.

There are two aspects on how to pre-empt a burst.

A burst is pre-empted during transmission: In this case the switch truncates the burst and sends a signal to describe the end of the burst, and thus all the nodes ahead can learn about the pre-empted burst.

A burst is pre-empted prior to transmission: The usage table is updated and the pre-empted burst is simply not sent until a channel is again available according to the approach already discussed.

The main disadvantage of the algorithm is that the usage table has to be always monitoring and so complexity increases and cost as well. The advantages though are that it reduces blocking probability and has higher resource utilisation.

The approach can be improved, reducing the blocking probability, using segmentation techniques [10].

Other approaches exist but are not so well known and have very high blocking probabilities. Some of these approaches are:

Resource reservation: This approach gives each class predefined resources, i.e. they would give access to all channels for high priority classes and less channels for lower priority bursts [11].

Intentional Dropping: Bursts are dropped in order to maintain a certain probability loss. The differentiation between classes is accomplished by actively dropping bursts [12].

The advantage of this technique of placing packets of different CoS in the same burst is by knowing the position of each packet, segmentation of the burst can be done thus improving CoS in a network and

Rescheduling of BHCs: Bursts do not follow a first in first out algorithm but they follow a scheme based on the CoS of each burst. This approach sacrifices delay for low probability loss. Also it involves having complex modelling and thus high cost [13].

3.4 Scheduling Algorithms

The last algorithms that are to be discussed are the scheduling algorithms, also mentioned in the literature as channel scheduling. Scheduling algorithms are used to fit any bursts that have passed the burst assembly section onto the channels available. In this thesis we are only going to present the most well known algorithms with focus on the algorithms that are simple in feasibility as our main focus of the thesis is to show a cost effective technique to create an all optical OBS system.

3.4.1 First Fit (FF)

The first technique that is going to be mentioned is that of FF introduced by Y. Xiong [16] and Zhu in 2000[14]. In the literature it is also sometimes mentioned as a simplified version of the Horizon algorithm described by Turner [15]. The basis of the algorithm is to have a predefined order in which channels are selected for scheduling bursts. The FF algorithm will start checking each channel, in a predifined order, which might be different for each edge node, for availability. If a channel is available then the channel is immediately reserved for burst transmission. If no channel is available it is stored until a channel becomes available.

3.4.2 Random

Another algorithm that is related to the FF algorithm is the Random algorithm [14]. This algorithm is a simplified version of FF. The channel selected for transmission is done by randomly querying for an available channel. When an available channel is found it is selected for transmitting the burst. If no channel is available the burst is stored until an available channel is found.

3.4.3 LAUC

One of the most known algorithms is the LAUC introduced firstly as Horizon by Turner [15] and later reintroduced by Xiong [16]. LAUC keeps track of one value, the unscheduled time of each channel. The concept behind this algorithm is to minimize the voids that appear between bursts in a channel. The orders of arrival of the BHCs do not represent the real order of arrival of bursts due to the offset introduced in edge nodes. The algorithm aims to minimise the voids by selecting a channel that has the latest available unscheduled time. So, as shown in Figure 3-8, if at time t a burst arrives with duration L the scheduling algorithm selects the channel where the smallest gap appears between time t and the end time of the last burst before t. In the case where data channels have not yet scheduled any bursts they are selected for transmission. When a burst cannot find an available channel for selection the burst is delayed until an available channel appears.

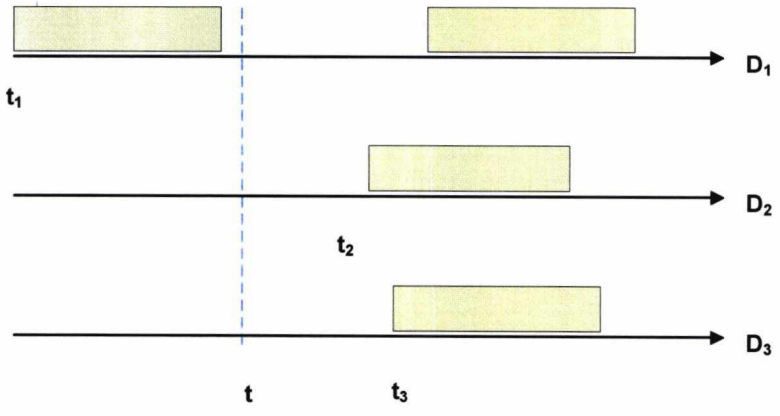


Figure 3- 8 Operation of LAUC scheduling algorithm.

3.4.4 Scheduling Algorithms with Optical Memory

This subsection concentrates on other scheduling algorithms that require some sort of optical buffering. These have not played an important role in the aim of this project but it is imperative to mention some of them.

One known method is taking the already mentioned algorithms and applying void filling (VF). With VF the OBS network has to keep a record of the gaps that appear between the scheduled bursts and their size. When a new burst arrives the scheduler used has to search through the recorded voids and find a suitable insertion point according to the length of the newly arrived burst and the length of a void. The channel selected to fill a potential void that may appear depends on the scheduling algorithm used. So if a FF-VF scheduling algorithm is used the channel selected for searching a void is done according to the order selected by the FF algorithm. Whereas, in LAUC-VF the algorithm finds the channels that are available for the smallest time period between the arriving time and the end of the last scheduled burst.

Some other scheduling algorithms which need optical buffering were presented by Xu [17] called minimum starting void (Min-SV), minimum ending point (Min-EV), maximum starting void (Max-SV) and maximum ending void (Max-EV). In these algorithms the scheduling is performed geometrically. If a void interval I_j exists it is represented as a point with coordinates (s_j, e_j) on a two dimensional plane whose x axes represent the starting points and the y axes the ending points. The scheduling is performed by finding, depending on the algorithm used, the minimum/maximum ending or starting time.

3.5 Theory of Scheduling Schemes

The algorithms that we have been experimenting with are based on the scheduling algorithms of first fit (FF) and latest available unused channel (LAUC)[1].

Both of these algorithms are well established [2] thus making them perfect candidates for introducing an improvement scheme on them. In this section the probability of selecting any particular channel in these algorithms is described using the help of some simple queuing theory.

3.5.1 Common Aspects

Before continuing to examine each of the scheduling algorithms in detail, the origin of some of the queuing problems in an edge node should be considered. In figure 3-9 a simple queuing system

is shown for the edge node. The traffic is received with an arrival rate λ_1 from an IP cloud, in the form of packets, which are placed into the queue that represents the burst assembler. The burst assembler has a service rate of μ_1 that is dependent on the burst assembly algorithm in use and the speed of the hardware used by the burst assembler for shifting the bursts out. In the case of FAT the service time remains constant, whereas in MBLFAT and DBSD this is not true. The arrival rate λ_2 at which the bursts are arriving into the channels is equal to the service rate μ_1 of the burst assembler queuing system. The scheduler then is split into an $n + 1$ parallel queuing system for an OBS edge node that can transmit bursts into n wavelengths. The service rates of all the queues involved in the scheduler are dependent on the burst sizes, because the service rates in each channel are the transmission times of the bursts. The arrival rate of each queue in the scheduler is dependent on the arrival rate of the other queues in the scheduler. So it can be understood that describing the scheduler queuing system and providing us with a mathematical solution to the probability of selecting a channel can be more than a challenge. The mathematical solution is close to impossible without some assumptions for the system, as the arrival and service rates of each queue in the scheduler are a function of the burst assembly queuing system. The assumptions that can be made for making the system simpler will have to be that the arrival rate going into the schedulers system remains constant and that the service times of all queues in the scheduler system also remain constant. In the case where all channels are busy the system has to be assumed to drop the bursts rather than storing them in a queue for later transmission. The reason of this assumption is to avoid any addition to the arrival rate which will appear if a queue is placed to store the bursts that cannot be send due to the simultaneous usage of all channels, as shown in figure 3-9.

Although the system was simplified with the assumptions taken, the solution to describe the probability of sending to channels is still very tedious to be obtained and the solution will be very inaccurate because of these assumptions. As a result investigating the scheduling of an edge node using a mathematical solution is not suggested, especially in the modern day where computer simulations can be done quickly and can provide us with a more accurate result. Thus in the following sections (3.4.2 - 3.4.4) the probability of selecting a channel for different scheduling algorithms will be investigated using some simple queuing theory to describe the results provided by our OPNET Modeler model for OBS networks and prove the results from the simulation model are accurate enough to a real system.

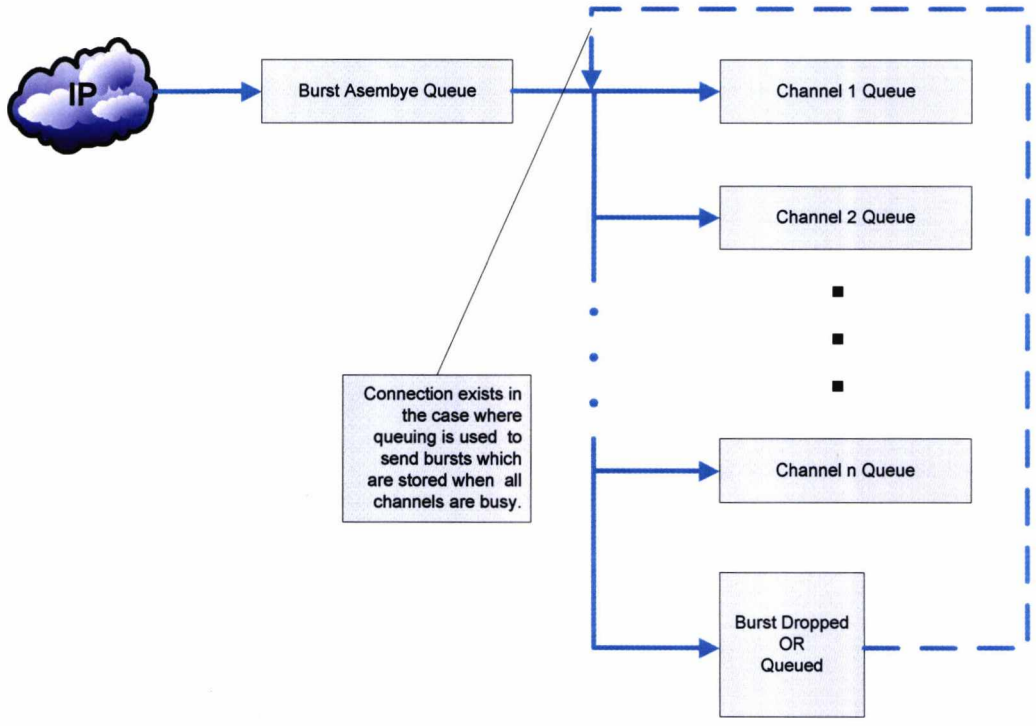


Figure 3- 9 Simple queuing model for an OBS edge node

3.5.2 First Fit Algorithm Theory

The way a FF scheduling algorithm operation can be represented is shown in figure 3-10(a). Figure 3-10(b) is a representation of a multistage queuing system with each stage representing a queue for each channel of the equivalent scheduling algorithm shown in 3-10(a). The channel selection order is what distinguishes FF from other scheduling algorithms. The selection of a channel, or channel assignment as it is most frequently known, is obtained through inspecting each channel in a predefined order until an available channel is found, this channel being assigned for transmission of a burst. In figure 3-10(b) it can be seen that the order remains the same at all times while every time a departure of a burst occurs in any of the queues(channels) the system returns to the original queue to discover if it is available.

The FF scheduling algorithm introduced by Y. Xiong [16] and Zhu in 2000[14] is a special and simpler case of the LAUC introduced by the same author. To find the different equations for the system the procedure of stochastic balance is used.

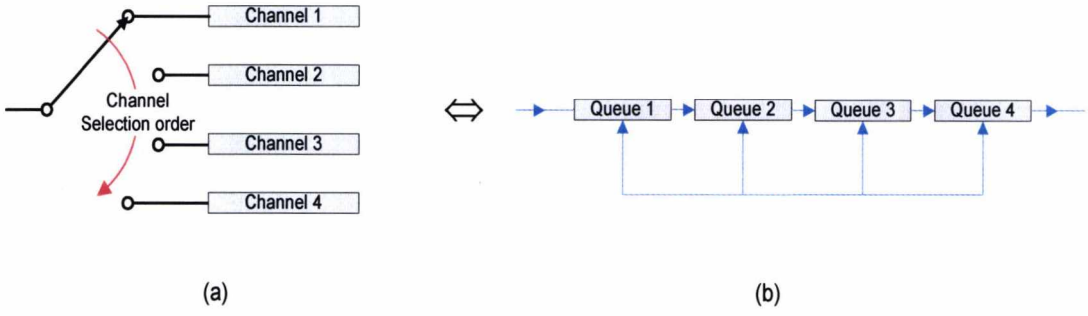


Figure 3-10 (a) First Fit channel selection, (b) Equivalent multistage queuing system

The best model to describe the FF scheduling algorithm, assuming the arrival and service times are exponential, is a M/M/1 model. Figure 3-11 shows the state transition diagram for the M/M/1 model. Arrivals on the channels are assumed to be "births" to the system, since if the system is in state *1* (where each state represents the number of channels that are busy) and an arrival occurs, the state is changed to *2*. On the other hand if a departure occurs while in state *1* the system goes to state *0*, this transition being referred to as a "death" process.

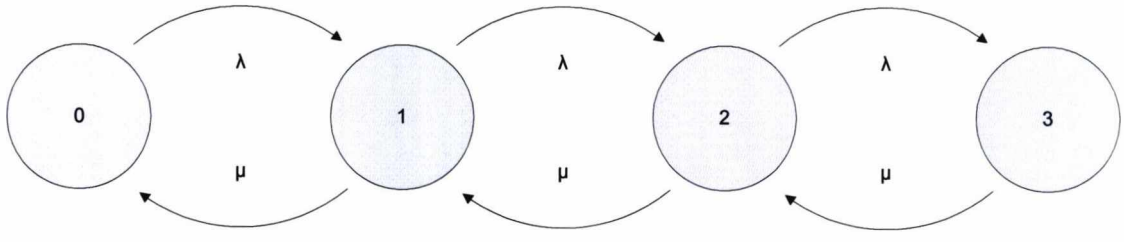


Figure 3- 11 State transition diagram for an M/M/1 system

The system described in figure 3-11 assumes that the order of channels is dependent on the number of channels being busy. This means that each state represents the number of channels that are occupied in that same order, so for example when a "birth" process occurs and the system is in the 0 state, where no channel is occupied, the system transits to state n meaning that the first channel in the order will now be used for the arrival of the burst, if another "birth" occurs then the system moves to state 2 meaning the first channel is busy and the second channel in the order is

now used for transmitting the new arrival and so on. When though a "death" process occurs, the channel that has been freed, it is assumed to be the first in the order that was occupied. Thus as an example, if state *2* represents that the two first channels in the order are occupied and a "death" occurs (the system moves to state 1), the channel that is now free is the first channel in the order and only the second channel is occupied.. This of course is not strictly the case with FF, but it can be thought to be a very close approximation as it will be shown in the simulations later in this section.

Finally, solving the M/M/1 system [20] provides us with the steady state solution:

$$p_n = \left(\frac{\lambda}{\mu}\right)^n p_0 \tag{3-1}$$

where

$$p_0 = \frac{1}{\sum_{n=0}^{\infty} \rho^n} \tag{3-2}$$

and

$$\rho = \frac{\lambda}{\mu}, |\rho| < 1 \tag{3-3}$$

It can clearly be seen from equation 5-1 that the probability of only one channel being occupied is higher than that of two being occupied as:

$$p_1 > p_2 > p_3 > p_4 \tag{3-4}$$

$$\text{for } \frac{\lambda}{\mu} < 1 \tag{3-5}$$

Equation 3-5 states that λ has to be always smaller than μ, which makes sense as if the arrival rate was larger than the service time at some point the system would have all channels busy and so another arrival would result in the burst being dropped. Thus equation 3-1 would no longer be applicable, as the system is not stable anymore which contradicts the way of the solution of an M/M/1 system of a stochastic balance.

In order to prove the above thinking, results were obtained through simulations of a single edge node transmitting 19,057 successful bursts with 3 different edge nodes as destinations through one core node. The source was generating bursts of exponentially distributed size with an exponential mean inter-arrival rate.

In figure 3-12 the channel selection is shown against the normalized probability of each channel being selected for a successful transmission using the OBS simulation model and the mathematical solution of M/M/1. The average arrival rate of a burst was found from the simulation to be equal to approximately 0.0001 seconds and the average service rate to 0.0002 seconds, which was used for obtaining the theoretical values. The results were taken by using the order of channels 1,2,3,4 which of course this will change from edge node to edge node. Thus it can be clearly seen that channel 1 is preferred to channel 2 and channel 2 is preferred to channel 3 and so on. The discrepancies between the theoretical and the simulation values can be associated due to the assumptions that were made for using the M/M/1 model, especially the assumption that no bursts are stored if all the channels are occupied. It can be safe now to assume that when the FF scheduling algorithm is being used the first channel of the order by which the selection is done is more probable to be selected than any other channel.

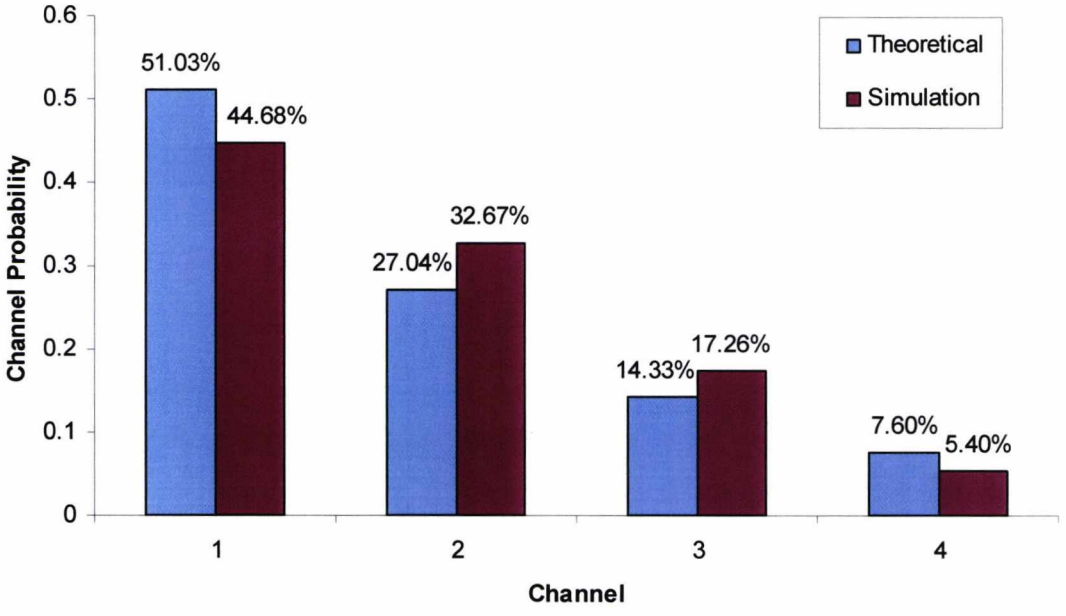


Figure 3-12 Channel selection probability using a theoretical and simulation method for an FF scheduling algorithm

The next simulation involved increasing the traffic to the edge node concerned. Figure 3-13 shows the normalized probability of channel selection with different traffic applied to the sources of an edge node. It can clearly be seen that the probability of selecting the first channel of the fixed order is more probable and the probability is reduced as we move to the second and even a further reduction is observed in the third and fourth channel. The next observation that can be made from Figure 3-13 is that as the traffic is increased the probability of favouring a channel over another is decreased and as the load approaches 100% there reaches a situation where there is an equal probability of sending on any channel and the order of scheduling in FF has absolutely no effect. This of course is starting to happen at a load of approximately 52% as shown in Figure 3-13 and it will be mentioned further on as the settling down time. This can be explained by using equation 3-1 and understanding what is happening to the service rate as the traffic is increased. As the traffic is increased the arrival rate does not change in the scheduler, because of the burst assembly algorithm which was assumed to have a fixed service rate. Although the arrival rate in the scheduler remains constant, the burst sizes are increasing and thus the service time of the bursts is also increasing. This effect can be seen for $\rho \rightarrow 0$ (ρ *tends to zero)* which for a four channel system it means the

probability of selecting a channel becomes: $p_n \approx 1/4$. (i.e. 25% probability for transmitting a burst in each channel)

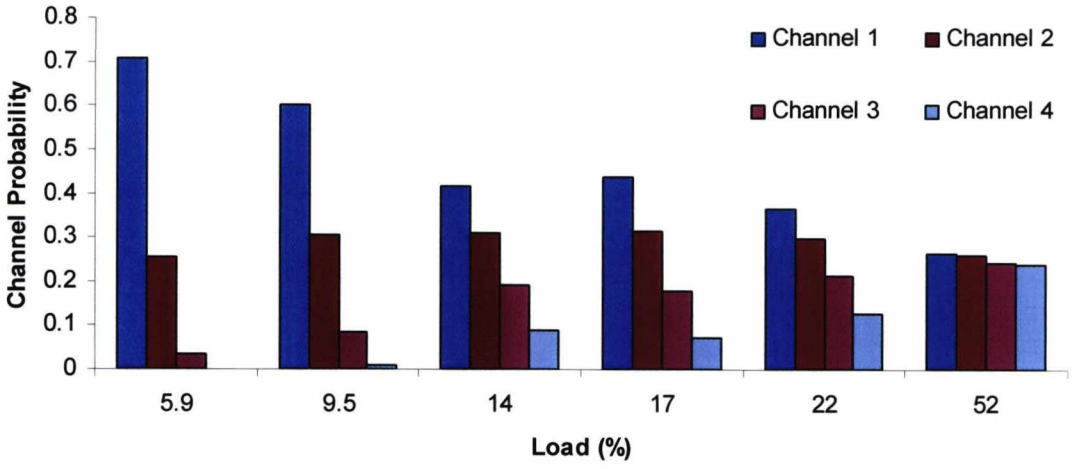


Figure 3- 13 Channel selection probability for different loads for an FF scheduling algorithm

3.5.3 Random Algorithm Theory

The next model that will be described is the random scheduling algorithm where a channel selected for scheduling is found completely randomly when a burst is ready for transmission. The queuing model for random scheduling algorithm can be thought of as an M/M/C system with c servers representing the number of channels. The state diagram for an M/M/C is shown in Figure 3-14 for c channels and n number of bursts.

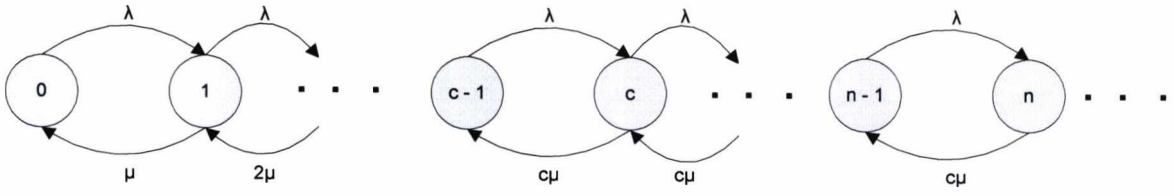


Figure 3-14 State transition diagram for an M/M/c system

The solution to the queuing model of the M/M/C system described by the state diagram in Figure 3-14 finds the state equations [20]:

$$p_n = \begin{cases} \frac{\lambda^n}{n!\mu^n} p_0 \ , \ (1 \le n \le c) \\ \frac{\lambda^n}{c^{n-c} c!\mu^n} p_0 \ , \ (n \ge c) \end{cases} \qquad (3\text{-}6)$$

where c is the number of parallel servers, i.e. in our case channels.

So it can be assumed that all channels are equally probable to service a burst at any given time making the probability of selecting any channel equal. The simulation proof of this can be seen in figure 3-15.

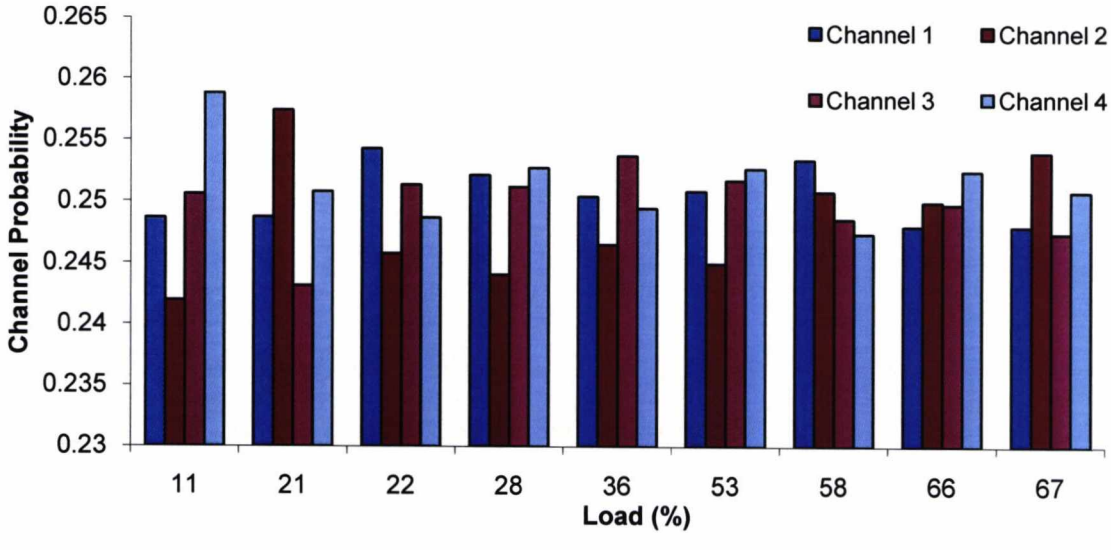


Figure 3-15 Channel selection probability for different loads for a random scheduling algorithm

Figure 4-15 reveals the randomness of the channel selection probability. This can be observed at all traffic loads as the differences in probability of selecting a channel are minor thus showing the probability of selecting any channel to be close to the value of 25% that is expected. A final point arrived from the above graph is that when using the random scheduling algorithm, the variation in

probability of selecting a channel reduces with higher load thus the statistical variation is smoothed out.

3.5.4 Latest Available Unused Channel Algorithm Theory

The scheduling in LAUC depends highly on the last scheduling that was observed. This is because the channel that last transmitted a burst has priority over any other. LAUC will be examined in two instances in order to understand later the effect of the improvement made by our improvement of the algorithms.

In both the FF and Random scheduling algorithms our investigation was simplified by either making assumptions on the priority system of scheduling channels like in FF or either on assuming the service and arrival rates to have markovian, or usually called memoryless properties, that can be investigated analytically. In LAUC, steady-state results as were obtained for the FF and the Random scheduling algorithms, cannot be obtained as the scheduling priorities are changed through time. It can be understood that it is difficult to describe the algorithm of LAUC as the scheduling becomes too complicated for an analytical model. The simplest way is to describe it by a simple logical analysis and then to move on to proving the analysis through simulation results as was done for the other two algorithms.

The first instance of describing LAUC is when the traffic is low. As it will be mentioned in Chapter 4, the traffic that is going to be investigated in an OBS network has a bursty nature [21], thus there are periods, which shall be called "on" periods, where packets arrive frequently and periods, which shall be called "off", where low or no traffic is observed. In the beginning of an "on" period or in an "off" period, LAUC schedules bursts between a number of channels, that are less than the number of channels supported by the edge node. So when a burst arrives the algorithm will try to find the last scheduled channel but when all of the channels have not scheduled a burst for a long period of time then the channel selection is made by a fixed order as in FF. When, eventually all channels have been used at least once, LAUC is working normally and moves to the second phase, in which we are interested, where it schedules a burst to the channel that was last scheduled as long as it is not busy. If it is busy then it selects the second last and so on.

LAUC can be thought of as a FF algorithm but with the channel selection order changing as the traffic is increased. With very low traffic LAUC will probably never go to the second phase of

operation as described previously. This of course, means that it works precisely as a FF algorithm but with interchanging order. The algorithm will start sending the data on the first channel of the order and will only move to the second when the first one is busy, at that point LAUC takes over and changes the order of the channels making the second channel of the initial order first and the first channel from the initial order second. This order changes at low loads, and the interchange keeps occurring between the first two channels in the original order until at a higher load the third channel will start appearing, and become involved in the order interchange. As the traffic load is increased, LAUC moves to the second phase of operation where all channels are being scheduled. As traffic load increases even further the channels will be scheduled with interchanging order between all channels at this instance.

In figure 3-16 the simulation results are shown for the same network used previously for the proof of operation of the other algorithms. The figure shows LAUC at low loads scheduling burst only in the first two channels with occasionally use of channel 3. Channel 4 comes into the picture when the traffic starts to increase and after traffic increases further, scheduling in any channel is approximately equally probable. The reason that only few channels from the ones available are scheduled for transmission at low traffic loads can be explained from the first phase explanation given earlier, that LAUC interchanges the order at which it selects channels between a portion of the number of channels available. Also it can be observed that channel 4 comes into operation with less traffic than that of FF, which implies that the algorithm will reach the point where all channels are equally selected with less traffic than FF as the scheduling at high traffic loads interchanges the order between all channels. Also a smaller difference can be seen between the probabilities of selecting channels which is observable in the 22% load of the graph. It can be seen that bursts, with LAUC in operation, have quickly obtained equal probability of them being sent in any channel.

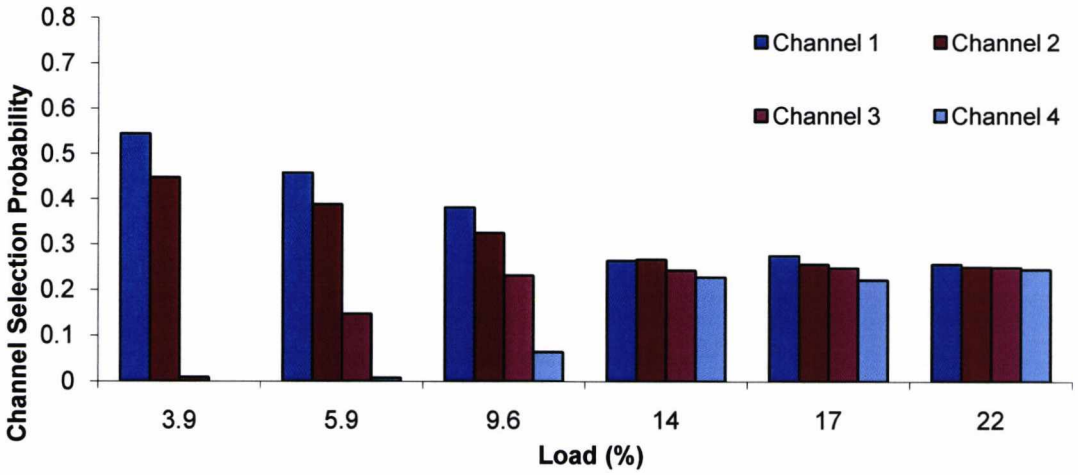


Figure 3-16 Channel selection probability for different loads for an LAUC scheduling algorithm

3.6 Summary

In this Chapter we presented algorithms for burst assembly, CoS and scheduling found in literature. Algorithms are all the time being published but in the thesis we considered some of the most important ones. Many algorithms have been presented in literature for scheduling but we presented only the ones that can be possibly be used without the usage of FDLs and wavelength conversion. We presented the details of operation of these algorithms in order to help us understand how they were modeled in Chapter 4.

3.7 References

[1] An Ge and Franco Callegati, "On Optical Burst Switching and Self-Similar Traffic", IEEE Comm. Vol. 4, no. 3, March 2000.

[2] Xiaojun Cao, Jikai Li, Yang Chen, Chunming Qiao "Assembling TCP/IP Packets in Optical Burst Switched Networks", IEEE Global Telecommunications Conference (Globecom). 2002, Taiwan.

[3] V. Jacobson Congestion Avoidance and Control, SIGCOMM Symposium on Communications Architectures and Protocols, pages 314-329, 1988.

[4] V. Jacobson, Modified TCP congestion avoidance algorithm. Note sent to end2end-interest mailing list, 1990.

[5] D.D. Clark and J. Hoe, Start-up Dynamics of TCP's Congestion Control and Avoidance Schemes, Technical report, Jun.1995. Presentation to the Internet End-to-End Research Group.

[6] Se-yoon Oh, Hyun Ha Hong and Minho Kang, "*A Data Burst Assembly Algorithm in Optical Burst Switching Networks*", ETRI Journal, Volume 24, Number 4, August 2002.

[7] Myungsik Yoo and Chunming Qiao, "Supporting multiple classes of services in IP over WDM networks, Global Telecommunications Conference 1999.

[8] Myungsik Yoo and Chunming Qiao, "QoS performance in IP over WDM networks, IEEE Journal of Selected Areas in Communications 18, pp. 2062-2071, October 2000.

[9] Chi-Hong Loi, Wanjium Liao and De-Nian Yang, "Service differentiation in optical burst switched networks", IEEE Globalcom, pp. 2313-2317, November 2002.

[10] Vinod M. Vokkarane and Jason P. Jue, "Prioritized routing and burst segmentation for QoS in optical Burst-Switched Networks", Optical fiber communications conference (OFC), March 2002.

[11] G. Hu, K. Dolzer and C. M. Gauger, "Does burst assembly really reduce self-similar traffic?", Optical Fiber Communications Conference (OFC), March 2003

[13] Jonathan S. Turner, "*Terabit Burst Switching*", Journal of High Speed Networks, 1999, 8(1), 3-16.

[14] Y. Zhu, G. N. Rouskas and H. G. Perros, "A comparison of allocation policies in wavelength routing networks", Photonic Network Communications, 2(3):265-293, Aug. 2000.

[15] Jonathan S. Turner, "*Terabit Burst Switching*", Journal of High Speed Networks, 1999, 8(1), 3-16.

[16] Y. Xiong, M. Vandenhoute, and H. C. Cankaya, "*Control Architecture in Optical Burst-Switched WDM Networks*", IEEE J. on Sel. Areas in Comm., vol. 18, no. 10, pp. 1838-1851, Oct. 2000

[17] J. Xu, C. Qiao, J. Li, and G.Xu, *"Efficient Channel Scheduling Algorithms in Optical Burst Switched Networks,* Infocom 2003.

[18] L. Dubois, G. Mounie, and D. Trystram Analysis of Scheduling Algorithms with Reservations *Proceedings of the 21st IEEE International Parallel and Distributed Processing Symposium*, Long Beach, CA, USA, March 26, 2007.

[19] C. Qiao and M. Yoo "Optical burst switching (OBS) – a new paradigm for an optical Internet", *Journal of High Speed Net.*, vol. 8, pp. 69-84, Jan. 1999

[20] Donald Gross, Carl M. Harris, "*Fundamentals of Queuing Theory*", John Wiley & Sons Inc, 1998.

[21] M. Dueser and P. Bayvel, "Bandwidth utilization and wavelength re-use in WDM optical burst-switched packet networks", *Proc. IFIP 5th Working-Conference on Opt. Net. Design and Model. (ONDM 2001),* vol. 1, pp. 23-24, Vienna, Austria, Feb. 2001.

CHAPTER 4

DESIGN OF OBS MODEL IN OPNET MODELER

Before embarking on a detailed explanation of the designs in Chapter 4, it is important to understand some basics of the simulation program (OPNET Modeler) that was used. This Chapter is a very brief introduction for understanding the models built in OPNET. OPNET provides us the program Modeler that has a hierarchical method of designing models for network simulation. The hierarchy begins with the project editor, where we either use standard library models or models designed by the user in other editors that can be used in the network model. The Project editor is where all the simulations are performed and results are obtained using a results editor. Each model in the project editor can be described in more detail by another model that can be viewed in the node editor. Similarly the models in the node editor can be viewed in more detail in the process editor where in turn the design can be viewed through in the actual coding behind them in C/C++.

In next part of the Chapter we will introduce the model that was designed and built in OPNET Modeler in order to simulate optical burst switching as accurately as possible. The analysis of the design will involve three main areas. The first area is the modeling of the edge node. The analysis will involve a detailed description of how the model was built and why, focusing mostly in the node editor domain. Here, there will be an explanation of the attributes assigned in the model for its proper operation. The description of the model moves on to the modeling of the algorithms introduced in Chapter 3; a detailed description will be presented of how these were modeled in the process editor. Finally, this first area will be concluded by including proof of the operation of the model.

The second area involves the core node design and it will follow the same pattern of description as with the edge node mentioned before. For the work that involves the design of a whole OBS network in the network domain. The description of the links, packet formats and configuration issues are all covered in the Appendix together with a final proof of operation of the OBS network model built in OPNET Modeler.

4.1 Network Domain

The creation of a network is done in a workspace called the "project editor". The network domain defines the topology of a communications network. The communication objects that are contained inside it are called nodes and their specific capabilities are defined inside their node models which are developed in the node editor, discussed later in this section. Many nodes contained within a single network model can be based on the same node model. Each of the nodes in the same network model is referred to as a node instance to distinguish it from the group of nodes sharing the same node model. Usually when referring to a node in the network domain the node instance is assumed rather than the node model. Figure 4-1 shows an example in OPNET of a network model. The three offices are described by one node model, the two plants by another node model and the corporate office by the last node model, whereas the whole network model consists of six node instances.

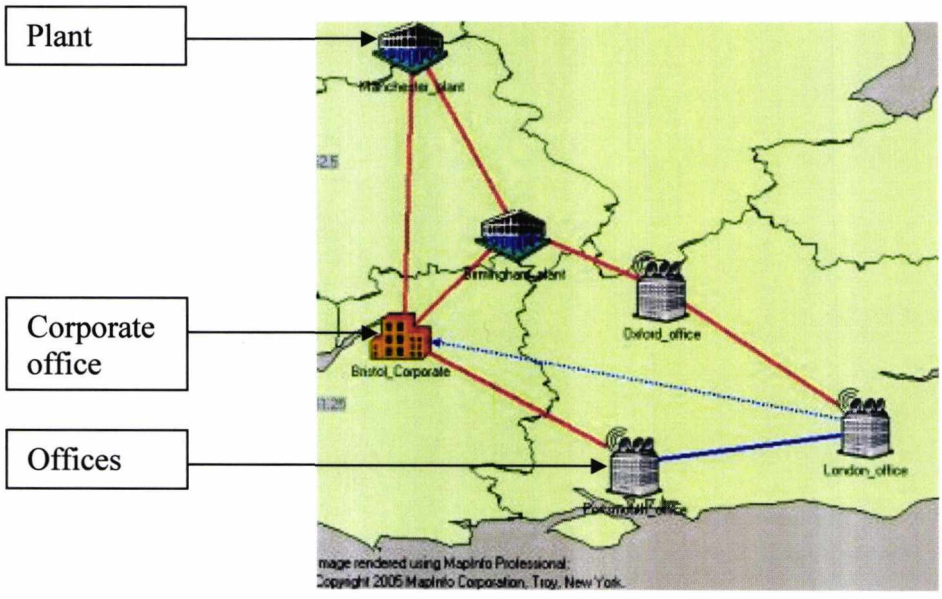


Figure 4-1 Example of a network model

Nodes used in the models created for the completion of any project require the ability to communicate to other nodes via links referred as duplex point-to-point links, which can be represent any kind of link from an optical fibre to a coaxial cable or twisted pairs. Other types of links are available as bus links, single point-to-point links

but were not needed for the aim of the research described in this thesis. These links are necessary for the nodes to operate in the network domain.

The project editor can have a geographic context, such as maps of countries or of the world, for the network model development. The placement of the nodes in the network editor are just for visual effect for understanding the distances involved between the nodes and thus the delays involved. The map itself does not provide a direct relation of distance/delay to the node instances; instead the user has to provide in each link between node instances the correct distance/delay.

4.2 Node Editor

The node editor is where the implementation of the node models occurs. The models are designed by using smaller building blocks called modules. Some of the modules have predefined sets of capabilities that can be configured via set built-in parameters. Other modules are programmable and their function can be set in the process model that will be described later in this chapter. The modules and their function in the node editor can be described in table 4-1.

Table 4-1 List of Module Descriptions

Module Name	**Description**
Processor	Is the core of the model that we create and there has to be at least one in each design. It is here that all the operation of the node is designed and implemented.
Queue or Buffer	It can be used for buffering or even for producing delays in the network element.
Transmitters and Receivers	We can either have point to point, bus or radio transmitters and receivers.
Wires	Their function is to interconnect the modules together. They can convey packets or statistical values.

A node model can consist of multiple modules of different types. The connection between the modules is accomplished via "wires". The statistic wires convey statistical information used to transport needed values from one module to the other. The packet streams are wires that convey only packets. The transmitters and receivers together with the wires have a fixed function capability that contain built-in parameters such as the data rate, the packet format that they can handle plus other set parameters. In Figure 4-2 an example of a node model is given. The example node model consists of different types of wires, processors and has one pair of transmitter/receiver.

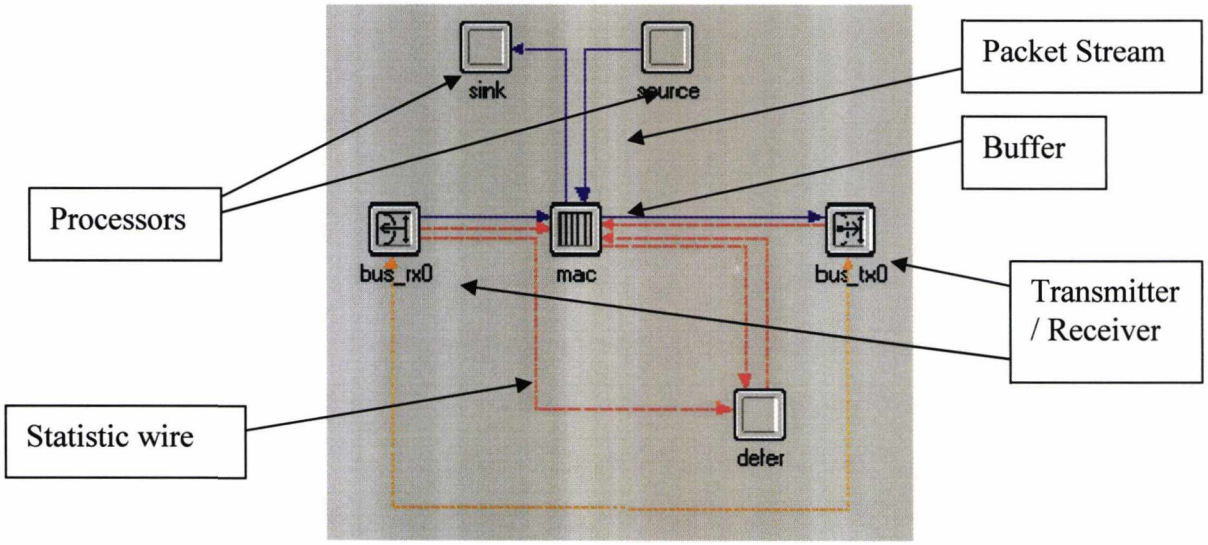


Figure 4-2 Example of a node model

4.3 Process Editor

The design of the processor and queue modules is done in the "*process model editor*". This is the editor where the true algorithm of the processor is implemented. The algorithm is accomplished by designing a finite state machine, where each state is programmed using C or C++ code. For a state to start executing the commands in it, an interrupt has to occur. Interrupts are events that are directed at a process and that may require it to take some action. They may be generated by sources external to a process (for example other modules), or by a process for itself. Interrupts usually correspond to events such as messages arriving, timers expiring, resources being released or state

changes in other modules. In Figure 4-3 an example of a process model is presented. It represents a simple process for a switching node. The transition between states takes place after an interrupt has occurred. As shown in Figure 4-3 an interrupt can occur from different sources. A stream interrupt in the node editor drives a transition in the "*idle*" state which occurs from a packet arrival in the node. A process interrupt occurs from the "*idle*" state when the packet arrived has its header obtained and rewritten, that has as a consequence another occurrence of a transition that will forwarded the packet into the transmitter in the node editor.

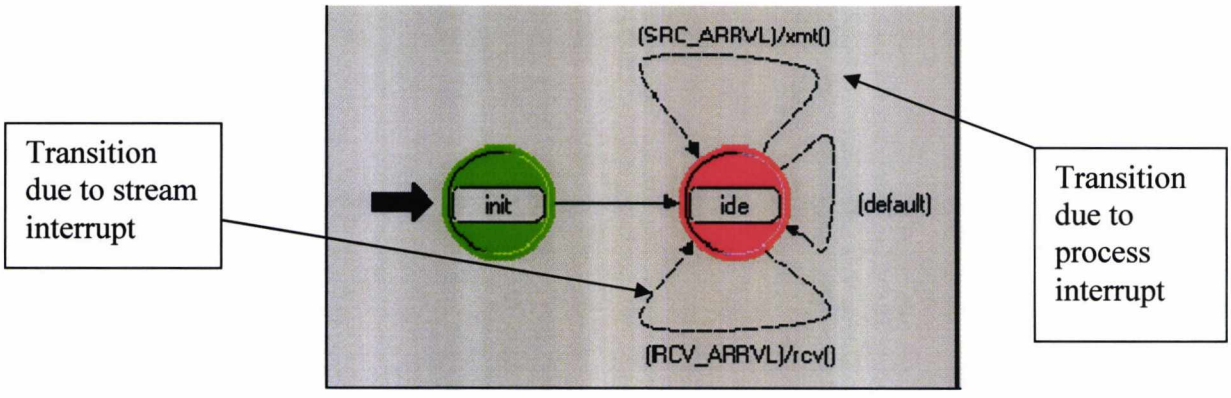


Figure 4-3 Example of a process model

There are two stages where the code is written in each state. These stages are known as "*Enter Executives"* and the "E*xit* Executives"and are illustrated in Figure 4-4. As shown in Figure 4-4, a state can be either forced or unforced. When a state is unforced after executing the enter executives, the process model becomes idle. What is meant by idle is that the control is sent to the simulation kernel. The next time the process model is invoked by an interrupt, the execution begins again from the "*exit executives*" of the state. When on the other hand a state is forced then after the enter execution it proceeds to the exit stage and proceeds to the next state.

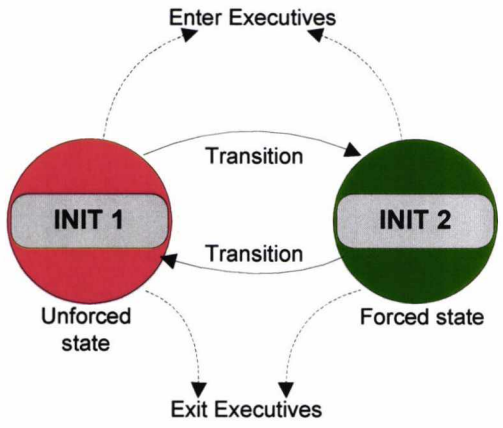


Figure 4-4 Forced and unforced states in the process model editor.

The changeover between states is called a transition and can be of two types. It can be conditional where a condition must be satisfied in order to make the transition and the line connecting two states becomes dashed. It can also be unconditional where the transition will take place after all the executions in the source state have finished.

4.4 Packet/Link Editor

In the project editor all nodes are connected via links that can be generated through the link editor. In the link editor attributes can be assigned such as the physical distance of these links, the data rate of the transmission, and also the packet formats that can be used in the links. By setting the attributes mentioned before the link models created provide the transmission and propagation delays in the network being simulated.

The packet formats that are supported by the links can also be generated in OPNET using the packet editor. Packet formats define the internal structure of packets as a set of fields. For each field, the packet format specifies a unique name, a data type, a default value and a size in bits. A packet format may contain one or more fields, represented graphically as a set of colored rectangular boxes. The size of the boxes is proportional to the number of bits specified in their size attribute.

4.5 Edge Node Design

In Figure 4-5 a block diagram of an edge node is shown which explains the functions of the node and provides an understanding of the mechanisms used within it. Packets arrive from a network outside the OBS network, where first some of their features are extracted in the classifier block. The features extracted are the size, the destination and the class of service of each packet. After all such information is known a packet is stored and added to a burst according to its destination and its class of service. The burst assembly block includes a class of service algorithm and a burst assembly algorithm. After several packets have been received and according to the burst assembly algorithm, the burst is created and is forwarded to the switch. The switch creates a burst header cell, where all of the information about the burst is included. Both the created burst header cell and the actual burst are then forwarded into the network and a small delay is introduced between the BHC and the actual burst. On the other hand, when a BHC arrives, the switch records the time the burst is expected and deletes the BHC after it has recorded any statistics required. When the burst arrives the switch will check if it was expecting the burst and, if everything is as expected, will forward it to the disassembler block; if the burst was not expected it is deleted. In the disassembler, the packets are extracted from the burst which in turn are forwarded to their destination.

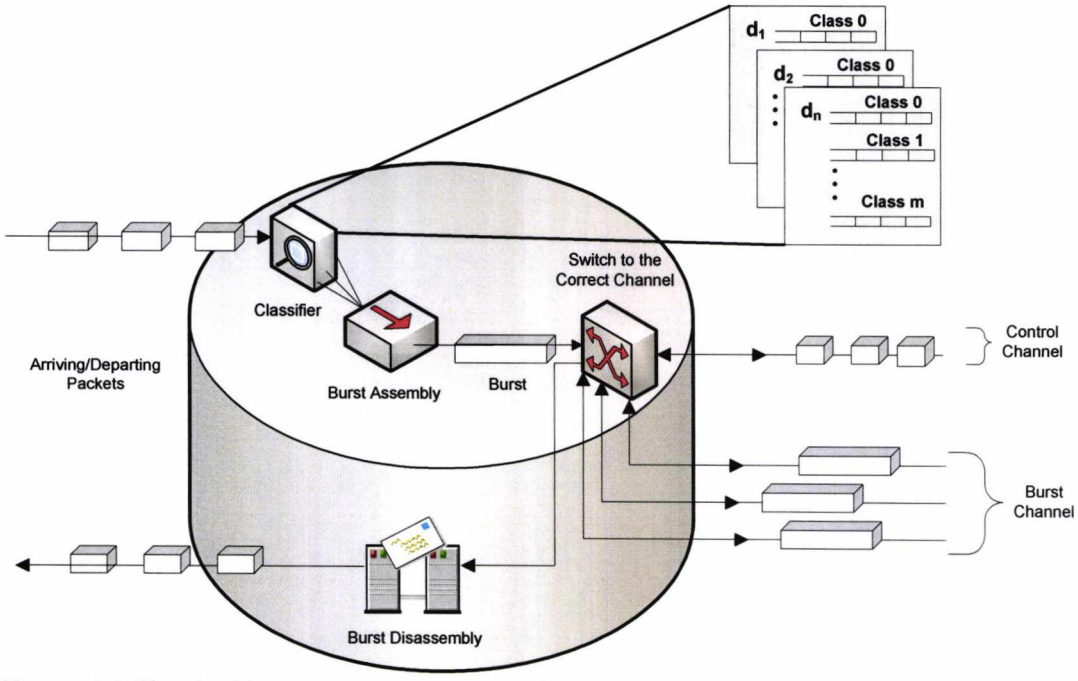


Figure 4-5 Simple Edge Node block diagram for an OBS Network

4.5.1 Structure of an Edge node

This section describes the implementation of an edge node using the node editor. Figure 4-6 presents the design for the edge node in OPNET Modeler. The design can be segmented into four parts. Each part will be described here by following the processes undertaken for a single packet from being received from a higher layer in the network to being transmitted to back to a higher layer at the destination node.

The first part that a packet will cross entering an edge node is the source block. The source block is where the packet's destination and the class of service are read and then forwarded accordingly to the correct stream for creation of the burst. When a packet arrives in the source block, from a packet source, it has two fields. The first field is used for storing the destination of the packet; according to this field the "destination" process forwards the packet to the next block, the classifier block. The classifier block is divided into ten identical groups of processes. Each of the groups represents a different destination address, so it can be seen that in the network domain an OBS network can be designed with a maximum of ten edge nodes. Each of these destination groups have a process called "decision" connected to four queues each representing a different class of service. The packets second field is read in order for the model to decide at which "decision" process they should enter.

When a packet enters the classifier block it enters the "decision" process. The process contains all the burst assembly algorithms described in Chapter 3 and via its attributes the user can select any of the available algorithms. The process is connected via a statistical wire and a packet stream to each of the four queues. The purpose of the statistical wire is to signal to the queue when the burst assembly algorithm has decided that a burst should be created. Then the queue will assemble the packets that are contained within it and generate a burst. The burst now will have a destination address and a class of service assigned to it. The burst follows the path to the next part of the edge node, the transmission block.

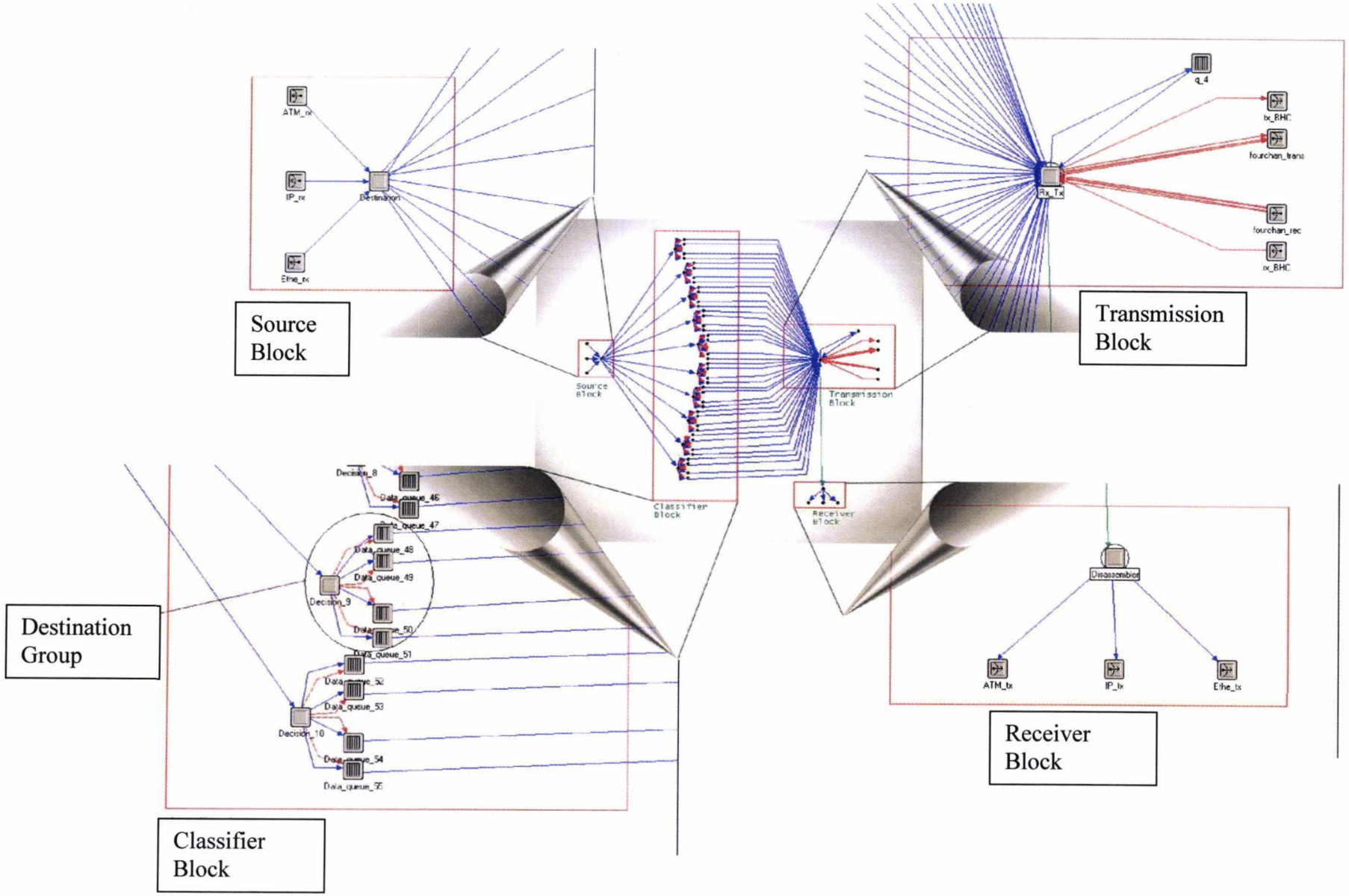


Figure 4-6 Node Model for an Edge Node of an OBS Network

In the transmission block the burst enters the process "Rx_Tx" which is used for all transmission and reception of any burst in the edge node. The process is connected to all channels available; this makes it ideal for the process to run the algorithms of class of service and of the scheduling algorithm. So in its attributes one can select any of its available algorithms for both class of service and on scheduling, making different statistics available for selection for investigation by the user of the performance of the OBS network. After executing the algorithms "Rx_Tx" process is ready to send the burst, but before that should execute the process to generate the BHC. The "Rx_Tx" process transmits the BHC through the appropriate channel and then after a short period of simulation time the process transmits the data burst.

The next step in understanding the model, is investigating what happens if a burst is received in an edge node. Before any data burst arrives in the edge node, a BHC is received. The "Rx_Tx" process extracts the information within the BHC and initialises the functions inside the "Rx_Tx" process for recording the statistics ready for the burst arrival. When the burst arrives all the statistics are recorded, such as delay of the burst, and the burst is forwarded at the receiver block. The receiver block has only one process, the "disassembler" where the packets are extracted from the burst and according to their destination they are forwarded to the appropriate stream, after recording their arrival time for their (packet) end-to-end delay to be calculated and recorded.

The design of an edge node in the node editor creates the illusion of delays between each of the processes. Delays do not exist inside the node editor except if specified by the process models. The delays that were included in the models are read and write access time of any memory used, which for an average range typical memory is approximately 5.4 ns for 8 bytes, so for example a read and write operation of a large burst size 150Kbytes the write and read time would be 0.2 ms [2]. The way that this delay was included in the model was by connecting a queue to the "Rx_Tx" process. The queue represents an electronic memory for bursts to be stored in case all of the data channels are occupied when the burst is ready for transmission. It has to be noted that the data is still in electronic format and only converted to optical when a free channel is ready for transmission of the actual burst.

4.5.2 Modeling of Algorithms and Modules

Most of the processes that have been described above included the algorithms mentioned in Chapter 3. In the following section a detailed analysis of the way the processes work is given, together with the way the algorithms were implemented within the network nodes.

Destination Process

The first process that is going to be described is the "destination" process. Figure 4-7 shows the finite state machine (FSM) of the process module and table 4-2 the events and interrupts of the module. The first state entered is the "Init" state where all the initialization for variables and statistics is performed. After the initialization of the process, the module, moves to the "idle" state where it waits for an interrupt to occur. The only interrupt that will then enable a change of state is a packet arrival, which causes a stream interrupt and transition to the "RCV_PCK" state. In this state, the packet is checked for a valid destination, and the class of service is read. The CoS field in the packet is retained and the packet is forwarded to the appropriate packet stream according to its destination address and CoS. Then the module returns to the "idle" state where it waits for another packet interrupt.

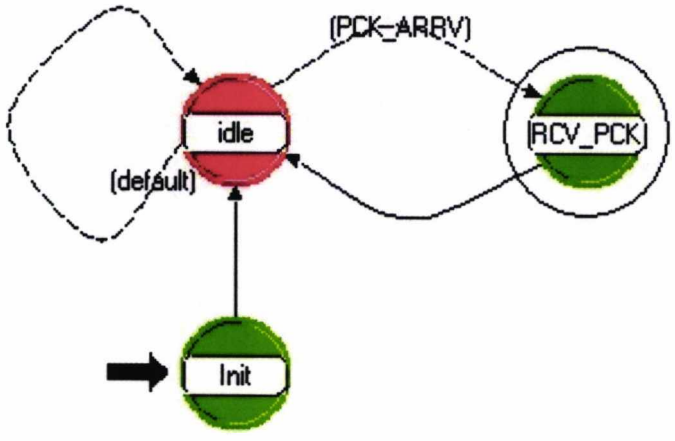


Figure 4-7 Finite State Machine of Destination Process Module

Table 4-2 State-Interrupt Table for the Destination Process Module

States	Interrupts	Description
"Init"	≡ "idle"	Initialization of variables and statistics
"Idle"	Packet ≡ "RCV_PCK"	Waiting until a packet is received
"RCV_PCK"	≡ "idle"	Checks destination, creates class of service for packet and forwards packet to appropriate stream

Decision Process

The next process that a packet is forwarded to, as mentioned before, is the "Decision" process. Algorithms for burst assembly are modeled in this module. The algorithms involved are FAT, MBLFAT, VAT and DBSD. Figure 4-8 and table 4-3 show the design of the process model FSM and its state interrupts, respectively. First, initialization of variables is performed, followed by an "idle" state. The initialization also obtains information of the type of burst assembly algorithm the user wants to utilise for the simulation. This is accomplished using a value in the edge node model to select the type of burst assembly algorithm. Other values are the timer size and the maximum burst size. The latter is only valid when using the MBLFAT or DBSD burst assembly algorithms. For MBLFAT and FAT these values remain constant throughout the whole simulation whereas for VAT and DBSD the values are varied according to the traffic received.

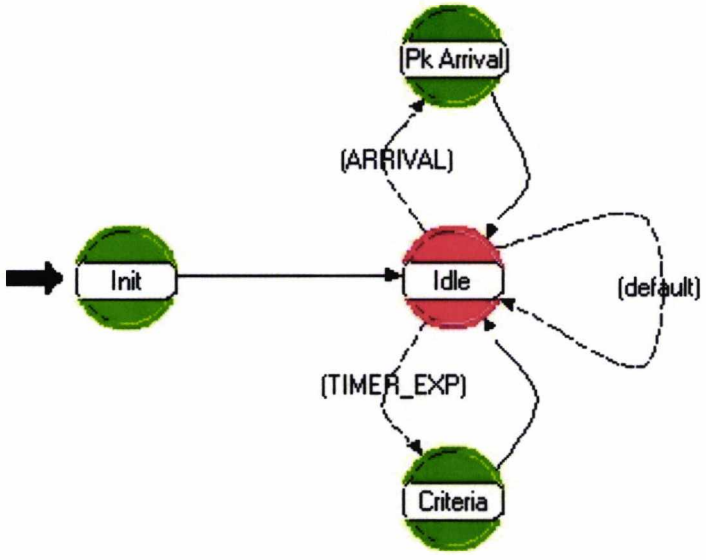


Figure 4-8 Finite State Machine of Decision Process Module

When a packet arrives the process is directed to the "Pk_Arrival" state. Assuming the first packet of a burst is being received, the "Pk_Arrival" state starts the timer of the burst assembly algorithm currently used. The process will then set a self interrupt to occur at the timer expiration time. When subsequent packets arrive, the process again moves to the "Pk_Arrival" state, but this time the process recognizes the timer has already been set and so the process just forwards the packet to the next module and adds the packet's length to the burst size calculation. The packet is forwarded to the corresponding queue according to its CoS. Before the process transits back to the "idle" state it stores the packets length and thus calculates the burst size at that moment. If the burst has passed the designated maximum burst size attribute and if the burst assembly algorithm used is either the MBLFAT or the DBSD a statistic interrupt is sent immediately to the statistic wire connected to the module and the queue concerned (i.e. CoS). At the same time the self interrupt for the timer is cancelled and then the timer attribute and the burst length attribute are reinitialized with the previous or new values according to the burst assembly algorithm used.

When the timer expires the process moves from the "idle" state to the "Criteria" state. In this state a statistical interrupt is sent to the relevant queue, with the timer and the maximum burst size attribute being initialized again.

Table 4-3 State-Interrupt Table for the Decision Process Module

States	Interrupts	Description
"Init"	≡ "idle"	Initialization of variables and statistics, and recognition of type of burst assembly algorithm
"Idle"	Packet ≡ "Pk_Arrival" Self ≡ "Criteria"	Waiting until a packet is received or the timer expires
"Pk_Arrival"	≡ "idle"	Forward packet to queue and perform calculations for the burst assembly algorithms. Send signal for burst creation to queue if maximum burst size criteria has been reached
"Criteria"	≡ "idle"	Timer criteria has been reached, send signal for burst creation to queue

Data Queue Process

The process model that follows in the path of a data packet is the "Data_queue" module where the data burst is generated. The FSM can be seen in Figure 4-9 and the state-interrupt table in table 4-4. The number of Data queues is equal to the number of classes of service existing in the network. For our model four classes of services were implemented, each queue representing one of them. No delay has been associated with these queues as the delay produced in writing the packet into memory is incorporated in the burst assembly algorithm mentioned before. It has to be mentioned that the FSM design of the model is from an already existing OPNET model queue the "acb_fifo" that has been changed in parts of its coding to suit the operation required here. The original design from OPNET was a simple first in first out queue for a number of packets.

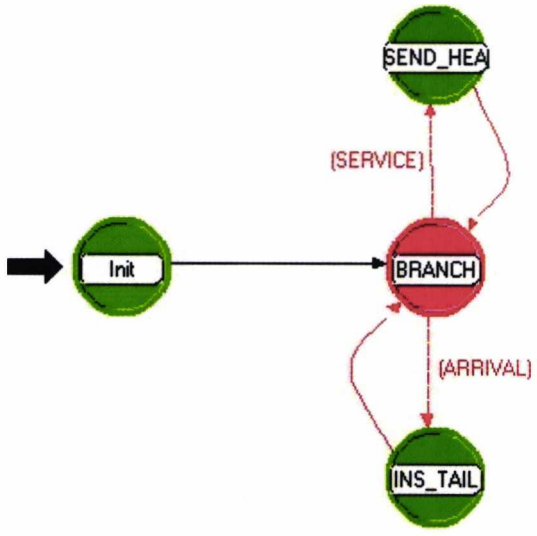


Figure 4-9 Finite State Machine of Data Queue Process Module

The first state is again the "Init" state where the initialization takes place. The next state is the "BRANCH" state where a packet arrival or a statistical interrupt is awaited. In the case of a packet arrival the process moves on to the "INS_TAIL" state. This state is where the burst is created and kept until ready to be sent. If the queue is empty, meaning the first packet to be included in a burst arrives, the state creates a new burst which can encapsulate other packets. Any packet that arrives is added to the burst that is being created until a statistical interrupt is received. When a statistical interrupt occurs the program moves to the "SEND_HEAD" state. In this state, the burst is forwarded to the next process and the queue is emptied.

Table 4-4 State-Interrupt Table for the Data Queue Process Module

States	Interrupts	Description
"Init"	≡ "BRANCH"	Initialization of variables and statistics
"BRANCH"	Packet ≡ "INS_TAIL" Statistical ≡ "SEND_HEAD"	Waiting until a packet is received or when a signal is received signifying the creation of a burst
"INS-TAIL"	≡ "BRANCH"	Creates a burst, and encapsulates any arriving packet in the burst
"SEND_HEAD"	≡ "BRANCH"	Forward burst to the next process model

Rx Tx Process

The most important process that is now going to be described is the "Rx_Tx". This is central to the edge node operation; it is where most of the scheduling algorithms appear, where the transmission and reception of bursts occurs and where the BHCs are created and destroyed. Figure 4-10 shows the finite state machine of the "Rx_Tx" process model.

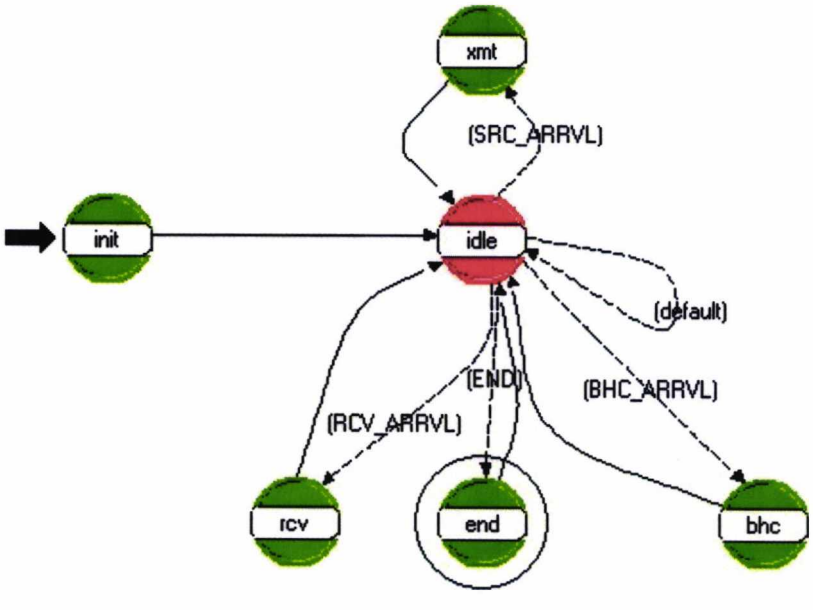


Figure 4-10 Finite State Machine of Rx_Tx Process Module

The "Rx-Tx" process model starts from an initial state where it waits for a begin simulation interrupt from the kernel, and where the process module is initialised and obtains information about the algorithms to be used in the edge node. After initialisation, the process model transit to the "idle" state and waits for a burst or a BHC to arrive or an end simulation interrupt to occur. If a burst arrives from within the edge node the path followed in the FSM will be to move to the "xmt" state. The purpose of this state is to generate and assign for each burst a BHC; it then forwards both the BHC and the burst to the corresponding packet stream, each representing a channel. The "xmt" state is also where the offset QoS algorithm is executed and all scheduling algorithms are performed except the LAUC.

The steps that are involved in the "xmt" state will now be analyzed. Assume a burst arrives in the "Rx-Tx" process module such that its FSM transits from the "idle" state transit to the enter executives of the "xmt" state; the first action is running the offset class of service algorithm (if this was selected in the attributes of the edge node) and setting the appropriate offset delay between the BHC and the burst, according to the algorithm. Secondly, the scheduling algorithm selected will be executed, followed by the creation of the BHC. A BHC is only generated if the scheduling algorithm was able to find a free channel for transmission. In case no free channel was found the burst is sent to memory to wait for the scheduled channel to become available. When the channels have become available the BHC is then generated. The BHC will then be sent to the control channel, and the burst to the data channel designated from the scheduling algorithm.

When a BHC arrives from the OBS network the "Rx-Tx" process module enters the "bhc" state where the information contained within the BHC is read; the arrival of the data burst is then awaited at a time specified by the BHC. The BHC is destroyed and the information contained is stored briefly until the burst corresponding to the BHC arrives in the edge node. When the burst arrives from a receiver the process goes to the "rcv" state where the burst is compared with the data stored previously from the BHC, if a match in the ID can be found the burst is forwarded to the disassembling part of the edge node, after some statistics have been collected about the burst and the information stored from the BHC is deleted. In the rare case, that the burst does not coincide with any information from any BHC, it is destroyed and the user is informed by means of a message in the simulation run window. This

rare case was included first for error checking in our code and second for future improvement of the model which will might include bit error rates.

From Figure 4-10 it can be seen that another state exists in the "Rx_Tx" process module called "end". This state is entered only when an end simulation interrupt occurs. In this state before the simulation actually finishes statistics are gathered that can only be obtained at the end of the simulation. For example, such statistics involve the total number of bursts and packets sent successfully, the number of bursts and packets destroyed/dropped and finally statistics of the probability of burst dropping. Table 4-5 shows the state interrupt table for this crucial process for a more clear understanding of the interrupts and the operation within the states involved.

Table 4-5 State-Interrupt Table for the Rx_Tx Process Module

States	Interrupts	Description
"Init"	≡ "idle"	Initialization of variables and statistics, and recognition of type of scheduling and CoS algorithm
"idle"	Packet (burst within node) ≡ "xmt" Packet (BHC) ≡ "bhc" Packet (burst) ≡ "rcv" End simulation≡ "end"	Waiting until a packet is received or when a signal is received signifying the end of simulation
"xmt"	≡ "idle"	Executes offset CoS algorithm and scheduling algorithms. Creates BHC and transmits BHC and data burst
"bhc"	≡ "idle"	Write statistics and information for burst arrival
"rcv"	≡ "idle"	Forward burst to the disassembler block
"end"	≡ end of simulation	Collect statistics at the end of simulation

Memory Process

The next process module in the edge node, to be explained is the queue connected to the "Rx_Tx" process. As mentioned before this works as a memory module. The design of its FSM can be seen in Figure 4-11 and its interrupt and state table in table 4-6. The process of this module is one of OPNET Modeler's standard queue processes named "acp_fifo" process. This process has been altered in some of its coding in order to act as a memory rather than as a queue. The purpose of using this queue was that the original method of operation of the queue was to obtain packets and store them for a particular amount of simulation time and then forward them to the output stream. The alteration made was for it to only transmit a particular packet when asked and adding a delay equal to that of writing and reading the length of the burst into memory.

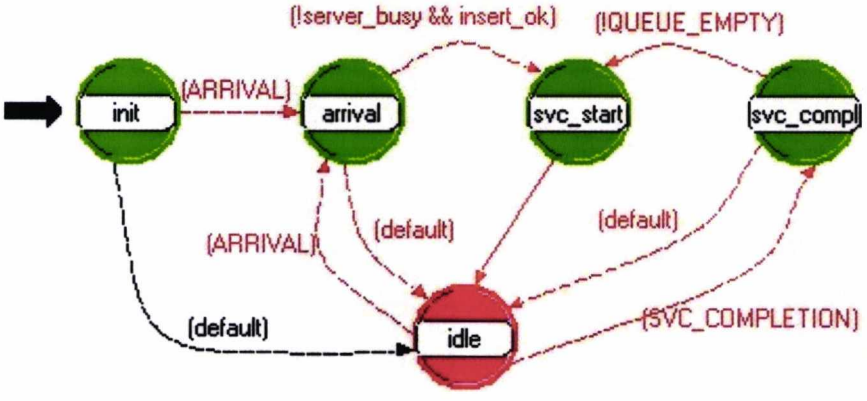


Figure 4-11 Finite State Machine of memory Process Module

The model will start only if a burst arrives. If no packet arrives during the simulation, the model will not be executed thus saving simulation memory and time. Assuming that a first burst arrives, the model is initialized in the "Init" state and then transits to the "arrival" state. The initialization involves obtaining two attributes from the user which are the size of the queue and the time of read/write operation to memory. The size of the queue is usually set to a large value that will not affect the simulation results by having bursts dropped in the edge node. It has to be noted that in reality this memory is in the electrical domain and thus size and cost are significantly reduced in the creation of a node compared to optical buffering. The "arrival" state checks if the queue is not full and will give the burst permission to be stored by

following the transition to the "svc_start" state which sends the burst into the queue. If the queue is full the burst is dropped and the process then moves to the "idle" state. At the "svc_start" state the burst is stored after a delay of the write time attribute collected before. After completion of the "svc_start" the memory process module goes to the "idle" state and waits for either another burst arrival, in which case the same state transitions are followed again, or the occurrence of a remote interrupt, in which case the model moves to the "svc_compl" state. In the case a remote interrupt is received while in any other state the model will ignore it until it has reached the "idle" state. The purpose of this is to avoid any extraction of a burst from the queue while in the write process. When in the "svc_compl" state, the burst that was first inserted in the queue is forwarded to the output and deleted from memory; the process model then returns to the idle state.

Although all the states and the operation of the modified model have been examined there is a small adaptation that was added when using the void filling algorithm at the edge node. The "!QUEUE_EMPTY" transition in the original OPNET model library was taken when the queue was not empty and so prompted the model to run the code in the "svc_start" which was to create another self interrupt for the next packet in queue. The transition in our modified model when using void filling is used every time the code is executed in the "svc_compl" state so the model moves to the "svc_start" state. In the "svc_start" state the code not only places the burst in to the queue but also records its size; when a burst is already forwarded it is in this state that the burst is deleted from the queue. The recorded information about the burst stored is used in the "svc_compl" state to get the burst from the queue according to the requirements of burst size needed by the scheduling algorithm.

Table 4-6 State-Interrupt Table for the memory Process Model

States	Interrupts	Description
"Init"	≡ "arrival"	Initialization of variables and attribute collection
"idle"	Packet ≡ "arrival" Remote ≡ "end"	Waiting until a packet is received or when a remote interrupt occurs is signifying the start of reading of a burst from memory
"arrival"	≡ "svc_start"	Checks if a burst can be written to memory
"svc_start"	≡ "idle"	Write burst to memory
"svc_compl"	(no void filling) ≡ "idle" (with void filling) ≡ "svc_start"	Read burst from memory

Disassembler Process

The last process in the edge node is the disassembler module. The finite state machine for the process can be seen in Figure 4-12 and the state-interrupt table in table 4-7. The FSM is very simple; the model starts from an "idle" state where the model waits for a burst to arrive in order to move to the "RCV_BUR" state. For the first time we can see that no initialization state exists as the model has a very simple function to perform. After receiving a burst and thus entering the "RCV_BUR" state, the model extracts all the packets from within the burst and forwards them to their proper destinations.

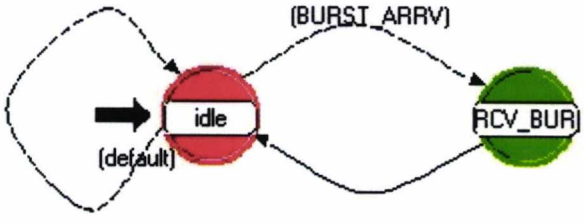


Figure 4-12 Finite State Machine of the disassembler Process Module

Table 4-7 State-Interrupt Table for the disassembler Process Model

States	Interrupts	Description
"idle"	Packet ≡ "RCV_BUR"	Waiting until a burst is received.
"RCV_BUR"	≡ "idle"	Extracts the encapsulated packets from within the burst and forwards them to the appropriate destination

4.5.3 Results of Operation

In order to test the coding written to perform all the functions described in the previous subsection some testing procedures were taken when writing the code. A function was written to create a text file when running the simulation. The function writes a line of information in the text file every time an event is performed. Each event was catalogued together with the exact simulation time at which that event occured, and also with important data such as the size of the packet concerned or the number of fields in a packet according to the part of code being investigated and many other depending on the event happening. The method was proven reliable as it could be used to find and identify any errors in the operation of the model. Each of the modules in the edge node was thoroughly investigated by itself and then in conjunction with its neighboring processes.

Although the tests carried out identified the correct procedure of events within the processes in the edge node, it only tested the source and the receiver block in the node editor and partially all other blocks. In order to test the blocks fully, other tests had to be carried out in order to view if the algorithms used within the processes also worked according to the manner of operation expected.

In order to perform a test a small network was designed in the network domain. A source was placed to transmit in the edge node packets to a certain destination and record statistics for the packets being transmitted from the edge node. Having this simple network design we were able to investigate the operation of the algorithms by looking at different parameters in the edge node according to the algorithm used.

To test the burst assembly algorithms five parameters had to be seen at all stages within the edge node. The five parameters were the timer for the generation of a burst, the number of bits in each packet and burst and finally the traffic received and transmitted. By controlling the packets generated it was easy to predict exactly the outcome of our test and compare it to the actual results received.

The tests that were involved for assuring the operation of the burst assembly algorithms modified the node design, simplifying it by not including the transmission block and replacing it with a sink. The operation of the sink was to gather statistics concerning the five parameters needed in order to test the bursts assembly algorithms. The sink also forwards the burst to the receiver block where we can disassemble the bursts in to the original packets. Having the packet statistics from the source, the bursts statistics from the sink and finally the packet statistics from the receiver block, it is easy to compare the results between them and prove that the burst assembly algorithms are operating correctly. Each of the packets has a unique ID given by OPNET Modeler; by storing at the source the IDs, the time and the size at which they were forwarded into the burst assembly algorithms it was possible to compare the time it took for generating the burst and the size of the burst created in the sink with the values stored at the source. Then at the disassembler we could see if all the packets sent by the source were received correctly by comparing the IDs of the packets.

The next test involved the testing for scheduling algorithms with the class of service algorithm and if bursts were delayed for more than a reasonable amount of time. The test was carried out by including coding in the "Rx_Tx" process that recorded any dropped bursts in the sending process while transmitting on purpose extra bursts in the same channel at the same time with any active scheduling algorithm. Dropping in edge nodes is not permitted as bursts can be stored electronically until they can be sent, only at that time would they be transferred into the optical domain. So by checking during simulation for burst dropping in edge nodes we could establish if the scheduling was performed correctly and if bursts were being sent to the delay process. This by itself only checks that a burst was delayed in the queue and partially tests that a scheduling algorithm works, it does not provide us with proof that the delay provided to the burst was actually correct. To provide a proof of the correct operation of any scheduling algorithm it was necessary to check

the time of arrival and the size of each burst arriving, thus calculating the transmission time and identifying when a channel is occupied. Looking carefully through the text file generated we could see if any violation had occurred in the scheduling algorithms. Viewing the departure of all bursts it was also easy to see the class of service of each burst and so to test if the class of service algorithm used was implemented successfully. The delay was tested by looking at the delay statistic from the queue representing the delay inside the edge node.

4.6 Core Node Design

The core node architecture is very simple to understand, the main feature being the switching of bursts from one edge or core node to the another, as can be seen in Figure 4-13. In each connection there is one control channel and multiple data channels, each on a different wavelength.

The BHCs arrive from any of the control channels and are processed in the BHC process block. The BHC process block checks from where and when the incoming data burst is arriving and to where it must depart. It tries to find if the path through a given wavelength channel can be used at the specified time, if available and sets the switch matrix ready to receive and forward the burst in the specified time. If the wavelength on which the burst is arriving is occupied by the transmission of another burst at that time then there are two options that can be followed, depending on whether the core node has a contention resolution scheme operating within it. Then, if the wavelength is not available, the contention resolution scheme is put into operation to find an available time and/or a free wavelength depending on the scheme used and sending the appropriate information to set up the switch. If the contention resolution scheme also fails to provide an available wavelength, then the burst is dropped. Of course, if no contention resolution exists the burst is dropped immediately.

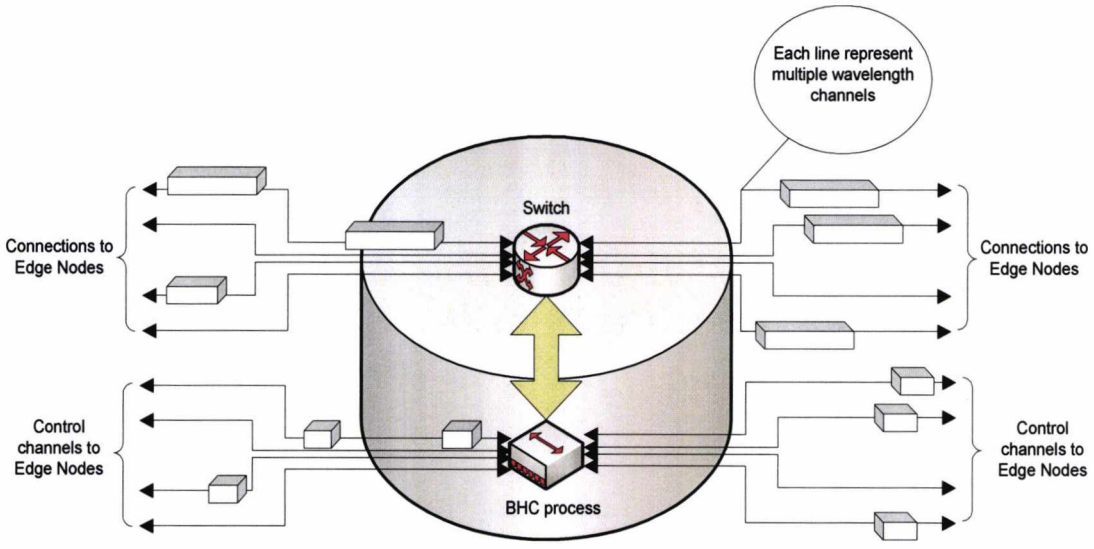


Figure 4-13 Simple Core Node model of an OBS Network

4.6.1 Modeling of Algorithms

In the edge node modeling it was decided to have ten destination options. For this reason a core node had to be designed to be able to accommodate this maximum number of edge nodes if a single core node existed in a network.

As shown in Figure 4-14 there are 20 point-to-point transmitters and receivers, which each represents a separate connection. All of them are connected to the processor block "control_box". The ten that represent the burst channel connections have four channels coming out and in of the transmitters and receivers respectively each representing a single wavelength. The "control_box" block was programmed to be able to switch received data bursts from one channel to another depending on the BHC information. Also it has to re-write the BHC and update it accordingly. The node also has attributes for the user to be able to select the contention resolution scheme if any is needed. The schemes designed were buffering and wavelength conversion. Buffering had to be designed in such a way as to represent a realistic optical memory.

Until now optical memory has partially been achieved with a loop of optical fiber around which the signal circulates. This storage mechanism is transparent to the bit rate so it can exceed 100 Gb/s. This optical memory loop (OML) must allow data to be switched in, re-circulated with little distortion, and switched out. The main problem with this kind of memory is that it is not possible to extract the information

stored in it at any instance that it may need to be used; instead we have to wait until it exits at a certain time.

For the above reason the way the buffering had to be modelled was by including various delays of specific time to resemble several OML. Two attributes were included for specifying OMLs, the number of OMLs and the delay of the smallest OML. If more than one OML was simulated, their delays would be multiples of the smallest OML. It has to be noted that no queues were used in the node editor to represent these delays and instead software was written in the process block. This was done in order to be able to have an interchangeable number of OMLs without changing the design of the core node model.

Other attributes included in the core node are used for routing algorithms and for some of the class of service and scheduling algorithms that can only be set in the core node. Such algorithms are the void filling from scheduling algorithms and the preemptive reservation from QoS algorithms. In both the above algorithms the buffering option has to be switched on as both need some type of memory in order for the algorithms to operate correctly.

Statistics gathered in this node include burst and packet dropping for individual core nodes as well as for the total number of core nodes involved in the network domain.

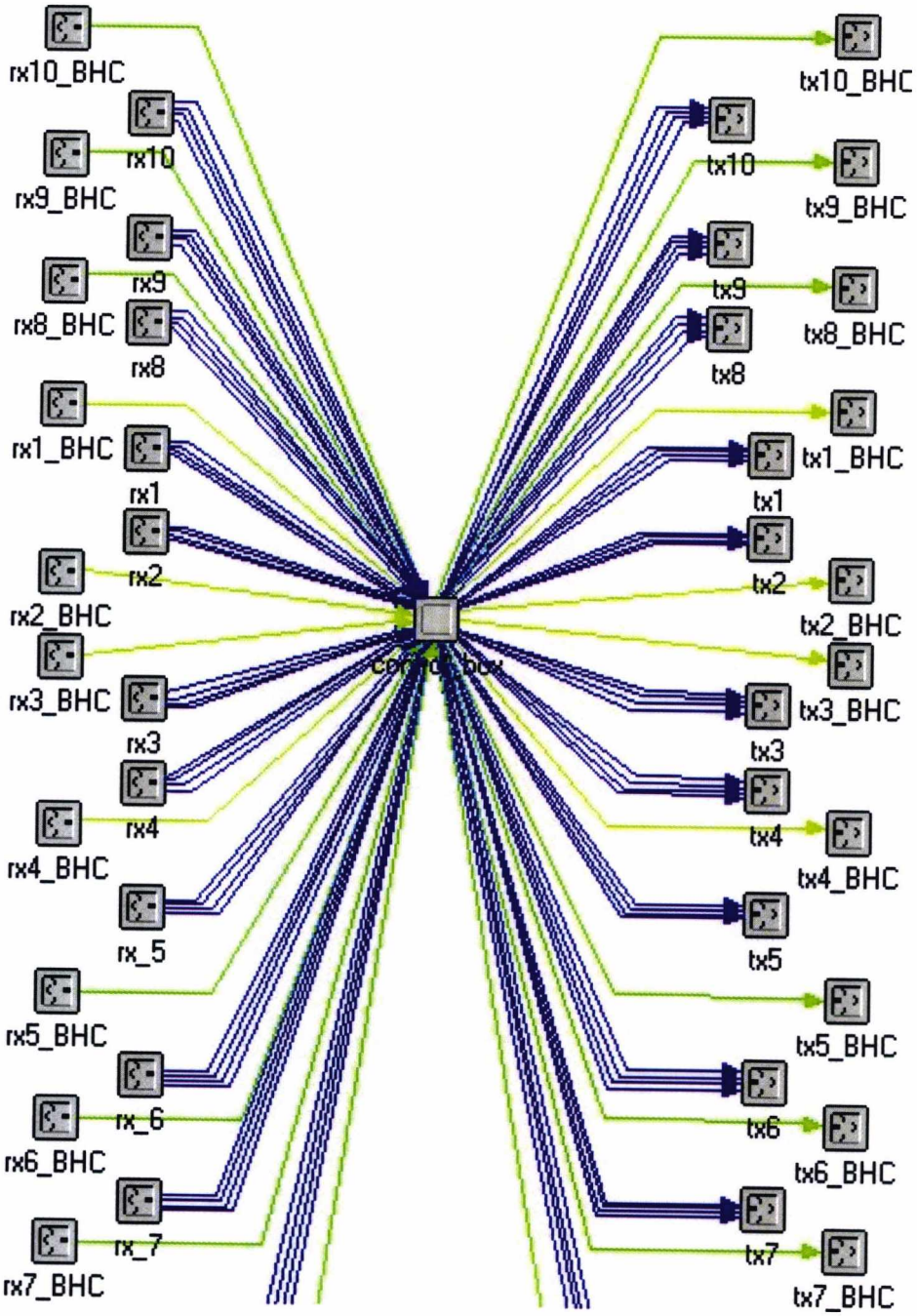


Figure 4-14 Node Model for a Core Node of an OBS Network

4.6.2 Modeling of Algorithms and Modules

The core node has only one process as the only operation required is to switch the incoming traffic to the correct output. In Figure 4-15 the finite state machine of the "control_box" process is shown and in Table 4-8 the state-interrupt table of the process. The process starts with an initialization state that configures all the necessary variables and statistics and gathers the attributes inserted in the network domain from the user. It then moves on to the "idle" state where it awaits a stream interrupt event to happen. If a stream interrupt occurs from one of the BHC channels it moves to the "check_1" state. In this state it recognizes from which node the burst was received and via this the burst moves to the correct state. A "BHC" state exists for all possible nodes connected to the core node, with each number representing the number of the BHC channel in the node editor. So for example the "BHC1" state is only entered if a stream interrupt occurred at the "rx1_BHC" channel in the receiver block at the node editor. All the states called "BHC" have the same coding except for some variables which are allocated different values according to the node being represented.

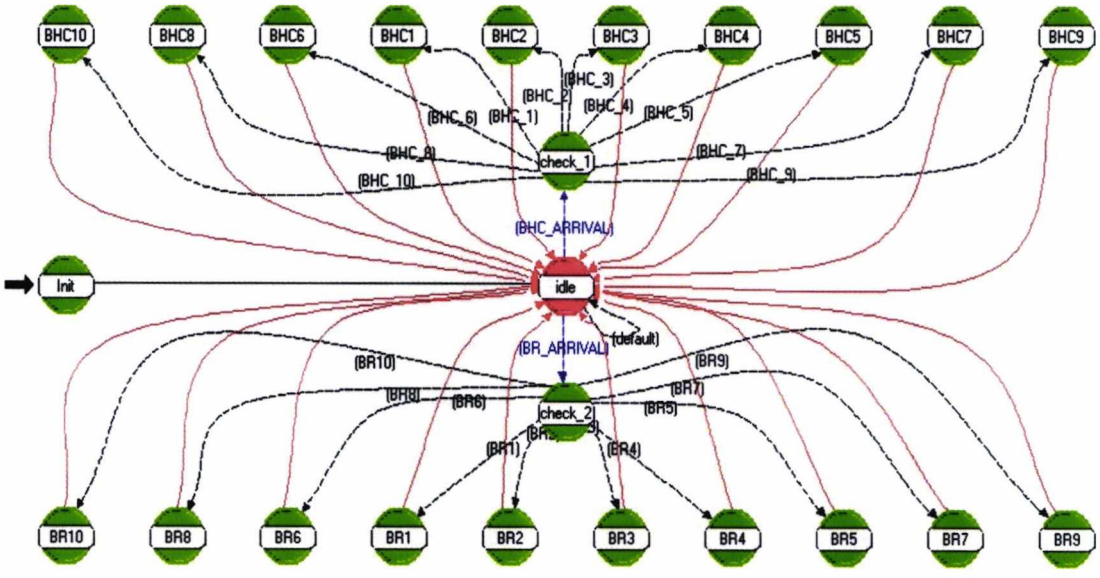


Figure 4-15 Finite State Machine of control box Process Module

While in any of the "BHC" states, the first function that is performed is the reading of the information contained in the BHC. After storing such information, it has to be decided if the channel on which the burst will arrive will be available at the time of the burst arrival. The state checks if a channel is available through three

scenarios. These can be seen in Figure 4-16; each of the scenarios considers all cases of how a burst can overlap with another so causing the channel to be unavailable. In the time axis T_{a1} and T_{a2} are the arrival times of two scheduled bursts and T_{d1} and T_{d2} the departure times, so that ($T_{d1} - T_{a1}$,) and ($T_{d2} - T_{a2}$,) are the burst lengths of the scheduled bursts. Then BT_a and BT_d are the arrival and departure time respectively for the incoming burst. The first scenario as shown in Figure 4-16(a), shows the departure time being in between T_{a1} and T_{d1}, denoting the possibility of overlapping when the departure time is between a scheduled burst length. The second scenario (Figure 5-16(b)) shows the arriving time BT_a of the incoming burst placed inside the burst length of the second burst. Finally, in the last scenario, as shown in Figure 5-12(c), the incoming burst arrival time BT_a and departure time BT_d can be placed in such a position that they are not overlapping inside a scheduled burst length but the scheduled burst length can be in between the burst length of the incoming burst. Some of the scenarios describe are of course dependant on the scheduling algorithm used.

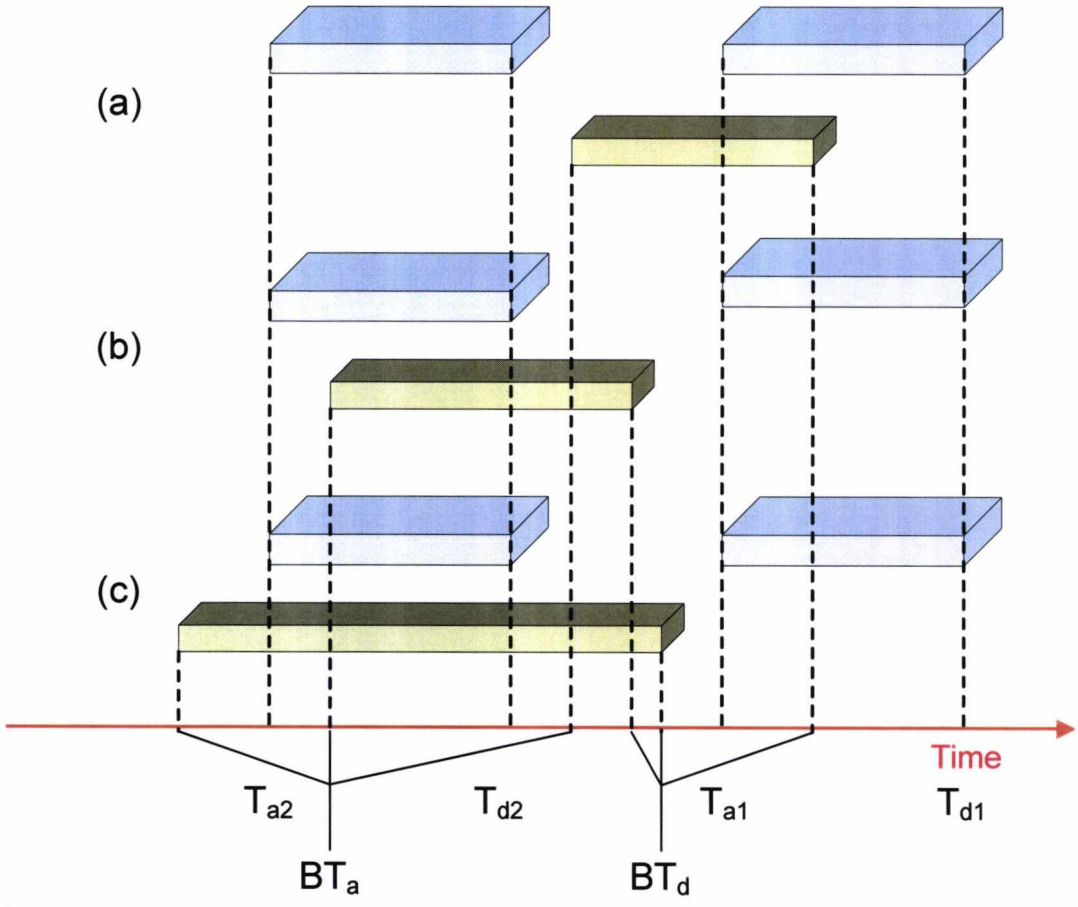


Figure 4-16 Scenarios for overlapping bursts in a channel. (a) $T_d > BT_d > T_a$ (b) $T_d > BT_a > T_a$ (c) $BT_d > T_d - T_a > BT_a$

In order to anticipate for these scenarios the "BHC" state tests these conditions by calculating and storing the burst length and the arrival time of a scheduled burst. When an incoming burst is anticipated the burst arrival and departure times are checked against all stored bursts and the new burst information is stored together with a flag indicating if the burst should be dropped or not.

At this point if the channel that is checked for availability is not available the "BHC" state would finish and the process would transit back to "idle" state. This of course depends if the any of the contention resolution algorithms is turned on. If they are, then the algorithm turned on will be activated and the overlapping scenarios would be checked again for any other available channels in case of wavelength conversion or if the channel occupied is free at other time instances if buffering is in operation. After this has been done the process again moves to the "idle" state.

Table 4-8 State-Interrupt Table for the control box Process Model

States	Interrupts	Description
"init"	≡ "idle"	Initialization of variables and attribute collection
"idle"	Packet & BHC channel ≡ "check_1" Packet & burst channel ≡ "check_2"	Checks if the packet is a BHC or a Burst
"Check_1"	Packet & Stream ID ≡ "BHC * "	Checks the arriving node
"Check_2"	Packet & Stream ID ≡ "BR * "	Checks the arriving node
"BHC * "	≡ "idle"	Schedules the burst transmission or a burst drop
"BR * "	≡ "idle"	Burst is forward to the appropriate node through the correct channel or it is dropped and statistics are collected

When a burst arrives at the core node a stream interrupt in any of the channels will force the "control_box" process to transit to the "check_2" state. At this state, as in "check_1" it looks from which node the interrupt came from and accordingly the process moves to the appropriate "BR" state. After entering the "BR" state, the process checks the stored information collected previously from the "BHC" states and finds which information corresponds to the incoming burst. It looks if the burst has to be dropped, delayed or sent. If delayed it depends on the information collected previously to see the amount of time the burst has to wait before transmitting; if it is to be sent, the wavelength to be used (and whether wavelength conversion is required) is checked. After handling the burst, the "BR" states will delete the information stored for the particular burst and will transit back to the "idle" state waiting for the next interrupt.

The operations described above will lead to finite but very short delays in the core node when describing an OBS core node in the "real" world. The delay a burst should experience is near zero as the switch is passive and it just guides the burst into the correct channel. Although a lot of information is stored in the processing, the delay produced by this is already calculated for in the edge node and is thus accommodated in the time difference between the BHC and the data burst arrival. The

only additional delay appearing is the time the switch has to remain in the required position for the burst to pass through the switch. Due to the reservation technique used in the model it has been assumed that this time is equal to the transmission time of a burst of length L_b at a bit rate equal to the channel bit rate in the optical links. The only other delay appears if buffering is activated and this depends on the values the user has inserted for the delay lines used and the number of buffers in use.

4.6.3 Results of Operation

The same techniques for testing applied in the edge node were used to check the correct operation of the core node. As the edge node had already been tested and verified the setup for testing the operation of the core node could include the edge nodes. The setup for testing the core node involved a core node connected to ten edge nodes in the network domain. A source was also connected to each of the edge nodes to send out packets according to different test scenarios as described below.

The first test performed was to view if the switching was performing correctly. A single source was activated to send packets for a certain period of time and then cease. The destination was set to a specific edge node and the core node was set with no contention resolution activated. The result of this test can be seen below in Figure 4-17. The Figure shows the throughput of the channels being used as a burst is received and after when the burst is transmitted from the core node. It can be clearly seen that the desired switching has taken place as the source (node 7) transmitting the data bursts had the destination set to node 3, where they have been received.

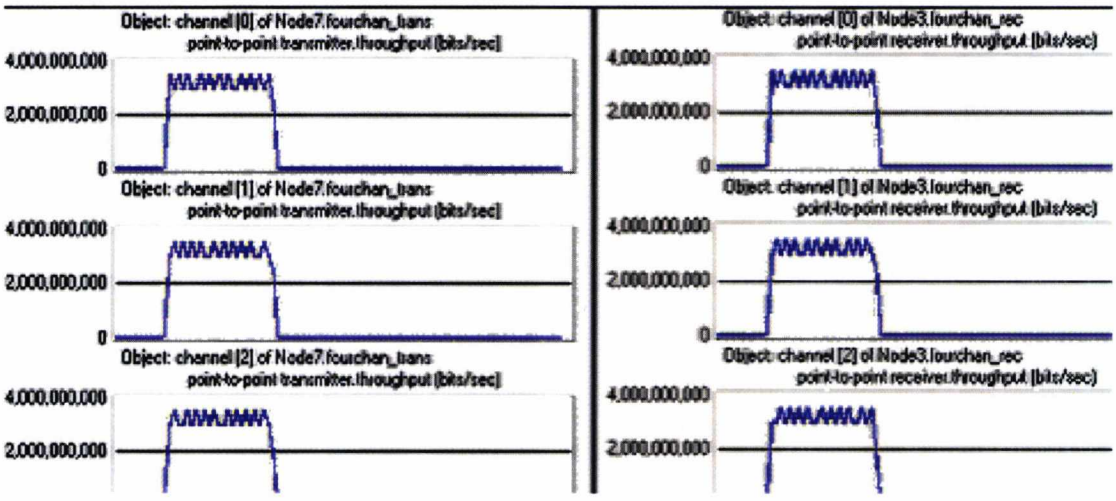


Figure 4-17 Results for testing the switching capabilities of a core node.

After the switching was tested and verified the next step in the testing procedure was to verify the dropping of bursts. This involved the viewing of the text file generated every time a BHC arrived in the core node. The text file that was generated included information of all the stored bursts in the core node, including the arrival and departure time of the bursts scheduled and their corresponding burst length. At the bottom of the stored information the current burst arrival and departure time as well as burst length were written. Finally the result of the process of deciding whether the burst could be scheduled or not was written. Looking at the text file generated it was possible to check manually the scenarios of overlapping bursts and verify that the process was operating correctly.

The next test performed was the verification of all the contention resolution algorithms. To perform this test it was necessary to produce intentional overlapping of bursts. This was done by storing false data of bursts in the process of the core node and then sending a burst to the core node, with the wavelength conversion activated or the buffering activated or even both. This was performed several times to verify that the algorithms of contention resolution were working according to the correct method. Every time the test was performed the text file generated was checked until a level of confidence was built to make us certain of the correct operation of the algorithms.

The last test performed was the checking of the algorithms for burst assembly and the CoS used by the core node. The test had the same setup as the original setup

used for testing the switching operation of the core node as mentioned at the beginning of this subsection.

Finally another test was performed to view the correct operation of the optical burst switching network. Now that the two node types were fully tested and verified individually it was necessary to see if the delay given between the BHC and the burst in the edge node was correctly being calculated. If the delay given was not enough the result would be to see some bursts arriving earlier than their BHC counterpart. To test this aspect of OBS the burst was given a unique ID number similar to an ID number given to the BHC counterpart. Every time a burst was received in a core node its ID was checked to the IDs already stored by the "BHC" state. If a match could not be found we terminated the simulation immediately and an error message was displayed in the simulation window. This was checked with the maximum number of edge nodes connected to the core node and with varying inter-arrival time into the edge nodes to simulate different weight of traffic load in the network. After running the simulation for several seconds and with no error message being displayed we were satisfied that the delay given between the BHC and the burst in the edge node was correct.

The next step in the model testing was to design a network topology presented in the literature in order to fully test the algorithms and to prove our results. In order to verify the models credibility a model was created to resemble the test-bed described by Sun et. al.[3]. The topology was just a single core node connected to four edge nodes via four wavelength channels, each of 4.8 Gbps and a separate wavelength at 600 Mbps acting as the control channel. The edge nodes were connected to multiple IP sources used to generate traffic for the edge nodes; the sources used varied between ON and OFF states as proposed in [3]. The distributions used in these sources have a purely random interarrival time, with a constant packet length of a maximum IP packet size.

In Figure 4-18 some preliminary test results can be seen where burst loss is calculated against the offered load in Mb/s. The results obtained are for a network without any contention resolution method applied to it and with four channels each having a bit-rate of 1.57 Gb/s, in order to correspond with the test bed then models were created for multi-core node networks.

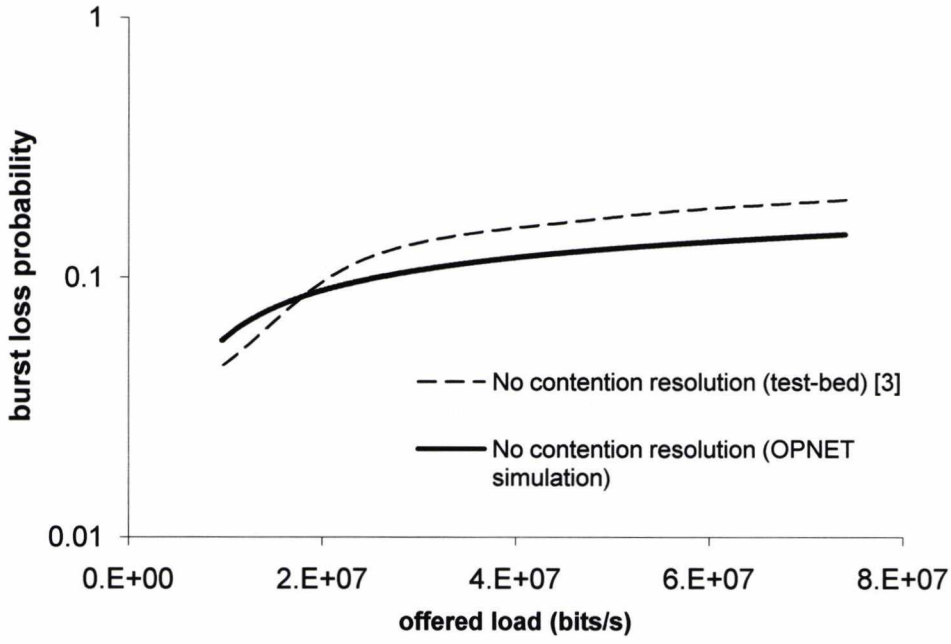


Figure 4- 18 Simulation of test-bed [4] using simulation model in OPNET

4.7 Summary

In this chapter, OPNET Modeler was introduced for the reader to be able to follow the chapters involving the design of the models used for this project. The description of the links, packet formats and configuration issues are all covered in the Appendix together with a final proof of operation of the OBS network model built in OPNET Modeler. The chapter as a first step focused mostly on the terms that are going to be used when explaining the later OPNET Modeler models and also with the architecture of OPNET Modeler. A quick summary of each editor in OPNET can be seen in table 4-12.

Table 4-12 Summary of editors in OPNET Modeler

Editor Name	**Operation**
Project Editor	Specify network topology and configure nodes and links. Choose results, run simulations and view results.
Node Editor	Create models of nodes by specifying internal structure and capabilities.

Process Editor	Develop models of decision-making processes representing protocols, algorithms, etc.
Link Editor	Create, edit and view link models.
Packet Editor	Specify packet format, data type and size of fields contained within a packet.

In the next part of the chapter we introduced the model that was designed and built in OPNET Modeler in order to simulate optical burst switching as accurately as possible. The analysis of the design involved three main areas. The first area is the modeling of the edge node. The analysis involved a detailed description of how the model was built and why, focusing mostly in the node editor domain. Here, there was an explanation of the attributes assigned in the model for its proper operation. The description of the model moves on to the modeling of the algorithms introduced in Chapter 3; a detailed description was presented of how these were modeled in the process editor. Finally, this first area was concluded by including proof of the operation of the model.

The second area involves the core node design and it will follow the same pattern of description as with the edge node mentioned before. The final area covered in this chapter involves the design of a whole OBS network in the network domain. The description of the links, packet formats and configuration issues are all covered in this final area together with a final proof of operation of the OBS network model built in OPNET Modeler.

In this chapter we introduced the model created for representing a close to reality OBS network. The first part was the description of the edge node and the modules that were contained within to be able to perform as real life edge node. The flowchart in Figure 4-26 shows the summary of operation of an edge node.

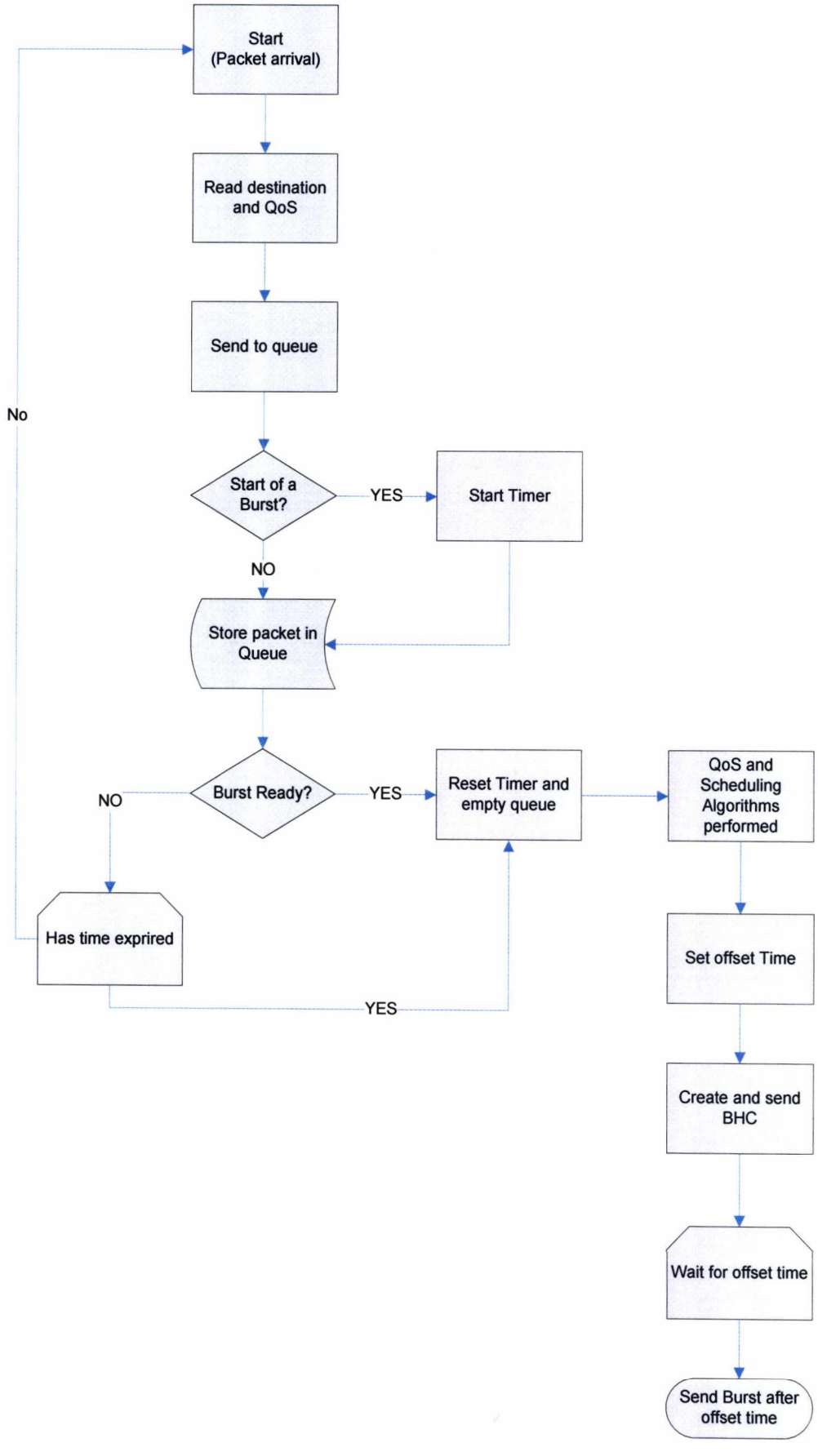


Figure 4- 26 Flowchart describing the operation of an edge node in an OPNET model.

When a packet arrives in the edge node the first operation is to read the destination and the CoS that is stored in the packets header. According to this information it is sent to a queue that stores packets with the same destination and the CoS. The edge node has to now decide if a burst with the same destination and CoS exists or not. In the case that a burst has not been created the timer starts. The edge node stores the packet in a queue waiting for the timer to expire or the upper limit of the burst size to be reached. The edge node keeps checking if the burst is ready. When the timer expires or the burst limit size has been reached the queue is emptied and forwards the burst to the next process. The next process involves performing the algorithms for CoS and scheduling. After a channel has been selected from the scheduling algorithm the edge node will create an offset time that is calculated from the reservation technique and from the CoS algorithm. The next process of the edge node is to create the BHC and forward it to the core network. After the offset time has expired the burst is also sent.

The second part of this chapter was the description of the modeling of a core node in OPNET Modeler. The equivalent flowchart for its operation can be seen in Figure 4-27. The flowchart starts with any arrival in the core node. The first process is to establish if a burst or a BHC has arrived. If a BHC arrives the information contained within concerning burst size channel of the burst arriving, the time of arrival and the destination is extracted. With these information, the core node searches for an available channel at the time the burst arrives. If the search for an available channel is unsuccessful the BHC is destroyed after the burst information is recorded. On the other hand if a channel is available the information is stored and the BHC is updated and sent to the next node.

In the case of a burst arriving, the core node searches the recorded corresponding BHC information to see if there is an available channel. If there is no available channel the burst is dropped and the statistics on burst dropping is updated. If though there is a channel available the burst is forwarded immediately to the next node towards the destination.

The final part that was described in this chapter was the verification of the correct operation of the OPNET model created for an OBS network. Different testing procedures were described to prove to the reader that extensive testing was carried

out to secure the proper operation of the OBS model. The testing results involved individual testing on node models, with finally comparing similar network designs used within known testbeds in the literature.

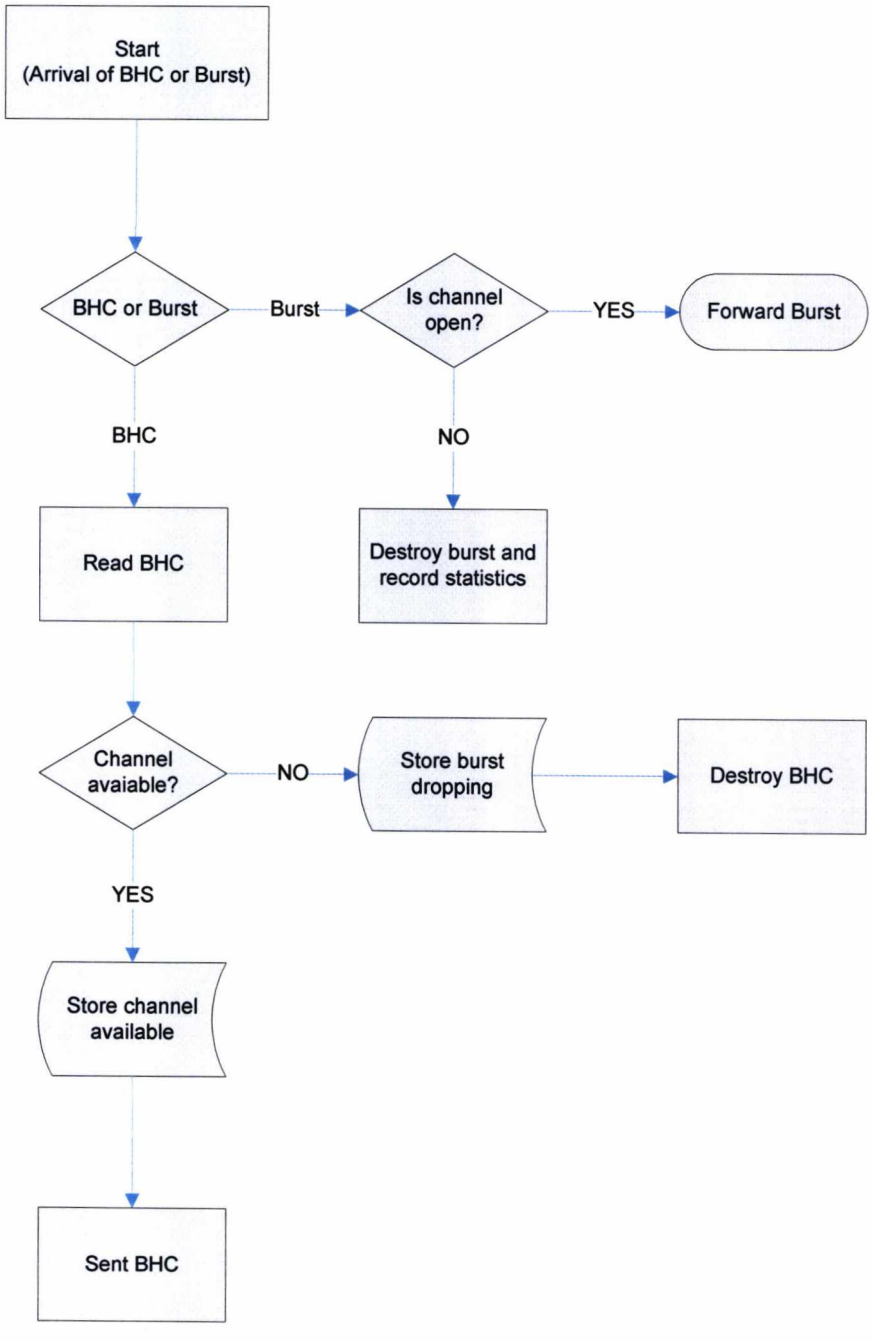


Figure 4- 27 Flowchart describing the operation of a core node of an OPNET model

4.8 References

[1] OPNET Technologies Inc., *OPNET Modeler: Product Documentation,* Release 10.5.

[2] ITAUCOM, "256/512MB ECC Registered Modules based on 32Mx8 SDRAMs" *Data Sheet for memory modules by ITAUCOM,* 2004.

[3] M. Dueser and P. Bayvel, "Bandwidth utilization and wavelength re-use in WDM optical burst-switched packet networks", *Proc. IFIP 5th Working-Conference on Opt. Net. Design and Model. (ONDM 2001),* vol. 1, pp. 23-24, Vienna, Austria, Feb. 2001.

[4] Y. Sun, T. Hashiguchi, V. Q. Minh, X. Wang, H. Morikawa, and T. Aoyama, "Design and implementation of an optical burst-switched network testbed", *IEEE Commun. Mag.*, vol. 43, pp. S48-S55, Nov. 2005

CHAPTER 5

WAVELENGTH CHANNEL ASSIGNMENT ALGORITHM

In Chapter 1 it was discussed that the research presented in this thesis has concentrated on achieving a lower burst drop probability for an OBS network. In order to obtain a lower burst loss probability, first a well devised queuing model was created to fully understand the way the loss probability increases in the core nodes dependent on different scheduling algorithms. Then new methods for the scheduling algorithms were devised and experimentation carried out in order to reduce the loss probability.

In this chapter, a way of explanation for burst loss probability in core nodes for the most frequently used scheduling algorithms are introduced by using queuing theory to partially describe the scheduling algorithms. After examining the reasons affecting burst loss probability, new algorithms are explained and the improvements they have on the limitations of the previously presented scheduling algorithms are described. The chapter will explain in detail the way of operation on our model of the new algorithms that we have devised to work in conjunction with other scheduling algorithms to reduce the burst loss probability. The chapter concludes by presenting simulation results that show the reduction in burst loss probability between the known scheduling algorithms and the modified algorithms proposed.

5.1 Simulation of Scheduling Algorithm in OBS Networks

The theory of the probability of channel usage for scheduling bursts has been discussed thoroughly in section 3.5 of Chapter 3. In this section the advantages and disadvantages of these scheduling algorithms in an OBS network will be shown in order to understand how blocking occurs, and what effects the scheduling algorithms have on the system.

As it can be seen in Figure 5-1, it will be assumed that the OBS network is one with multiple sources that aggregate the traffic received from other networks, all interconnected via the core network. The core network is an assembly of interconnected core nodes. These core nodes are assumed to act as switches without having any capabilities of scheduling as described in Chapter 4 or any method of contention resolution. In order to explain the effects of the scheduling algorithms on the burst dropping in OBS networks it can first be assumed that the core network can be represented via a single core node connected to multiple edge nodes as shown in Figure 5-1. All scheduling is done within the edge nodes and the core node will switch bursts between nodes, it will not perform any wavelength conversion.

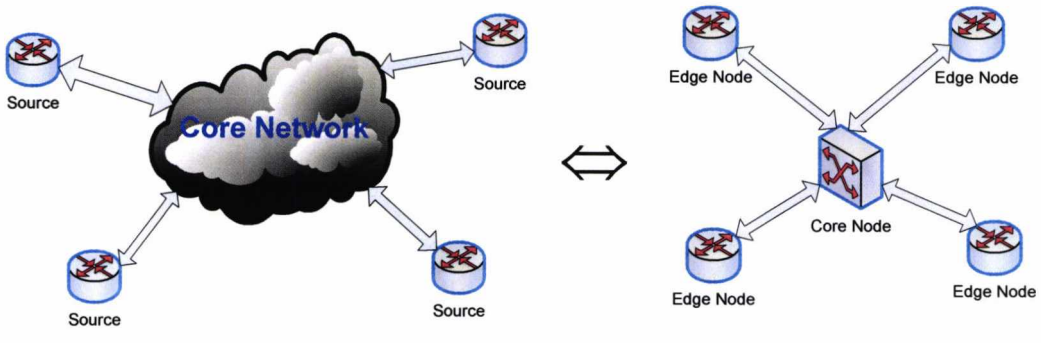


Figure 5- 1 Optical Burst Switching equivalent network

When the FF scheduling algorithm is used all edge nodes will start sending bursts in a fixed order. The order may differ from edge node to edge node but no care is taken in selecting an order. Whatever the fixed order is, the edge node will start sending burst mostly on the first channel of the order, as was described previously that means that dropping occurs mainly in the first channel and in the second channel of the fixed orders. If the traffic is increased it has been shown (3.4.1) that the probability of scheduling to any channel is approximately equal. This has as an effect that the probability of dropping in the core network becomes approximately the same in all channels, and thus the burst dropping

probability saturates as the load increases. This will become clearer in the next subsection, 5.1.1.

In random scheduling there is not a single channel that will appear to have a larger probability of dropping than any other. It has been shown in section 3.4.2 that the probability of sending a burst on any channel also exists in this algorithm when the traffic is increased. Thus burst dropping probability will remain constant after the stabilization in burst drop probability has appeared in all subsequent traffic loads (5.1.1).

Finally in LAUC, as shown in section 3.4.3, the dropping also occurs primarily in the first channels of the order they start with but when scheduling appears in all channels it appears when a smaller traffic load is applied on the network using the LAUC scheduling algorithm than the FF and thus LAUC is expected to reach saturation on burst dropping probability faster than FF as it will be shown in section 5.1.1.

5.1.1 FF, Random and LAUC

For the following simulations one core node was used, and connected to several edge nodes. The assembly algorithm used was FAT and no quality of service was included. Figures 5-1 to 5-5, show the burst loss probability versus the load in a single core node network. For each curve in the figures a series of nine simulations were made, each for a different load on the network shown as the points on the curves. The results presented will show the effect of adding more edges nodes to the single core node and the difference in performance will be explained according to the investigation of the scheduling algorithms in section 3.4.

In all results presented in this section, in each simulation the model calculates the burst loss probability and the load on the network. The calculation of the load was calculated according to the formula below:

$$Load = \frac{\text{Normalised bits sent}}{n \times (\text{Bandwidth of 4 channels})} \tag{5-7}$$

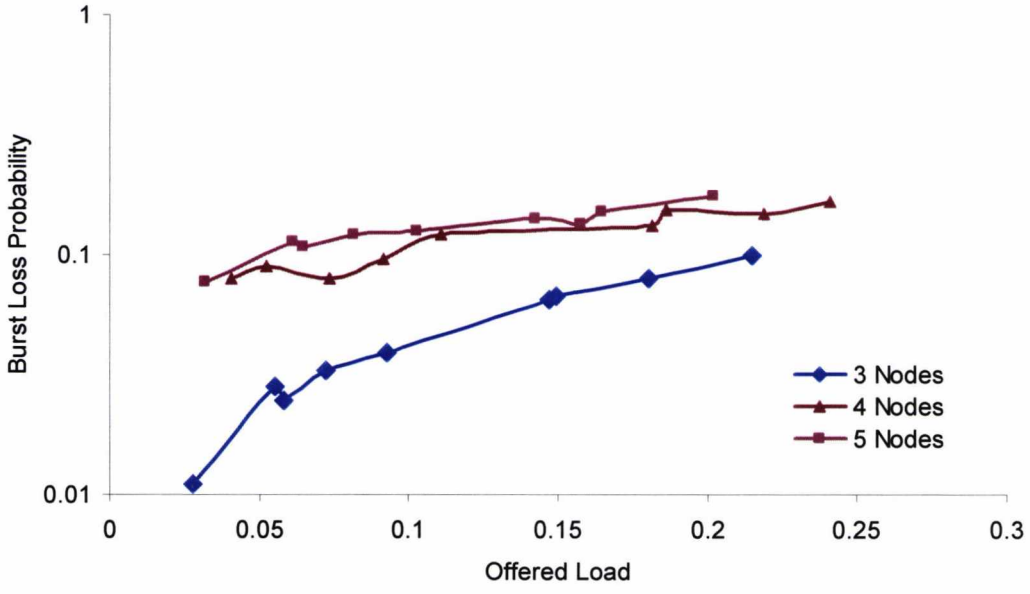


Figure 5-1 Burst loss probability for different number of edge nodes using FF (Single core node)

Figure 5-1 shows the single core node results while using FF as a scheduling algorithm. When using a small number of edge nodes the probability of dropping a burst as shown in Figure 5-1 is lower than when using a larger number. More specifically, the ratio between the burst loss probabilities when using four and five edge nodes is lower than between three and four edge nodes. This can be explained by the number of channels available. This large difference in ratio is expected because when three nodes are being used only three bursts can be sent simultaneously to the scheduling algorithm and thus by having four channels in operation the fourth channel is rarely used because of the manner of operation of FF. So, dropping occurs mostly in channel 1 and 2 and some in channel 3 but far less dropping is present in channel 4. When, however, four or a higher number of edge nodes is used, all 4 channels are used more regularly and the probability of dropping in channel 4 increases rapidly and thus the decrease in the ratio of burst loss probability curves between three, four and five edge nodes.

Another important aspect that has to be noticed is increasing the number of channels. By increasing the number of channels the ratio between the burst loss probabilities when using four and five nodes should increase and the ration between three and four node

should decrease. This experiment was done by increasing the number of channel to 8 and looking at the same graph. The results can be seen in the Figure below. In the graph we now also included three more nodes thus reaching a maximum of 8 nodes, equal to the number of channels. As it can be seen from Figure 5-2 the burst loss probabilities have dropped when using four and five nodes and we see the increased burst loss probabilities when using seven and eight nodes.

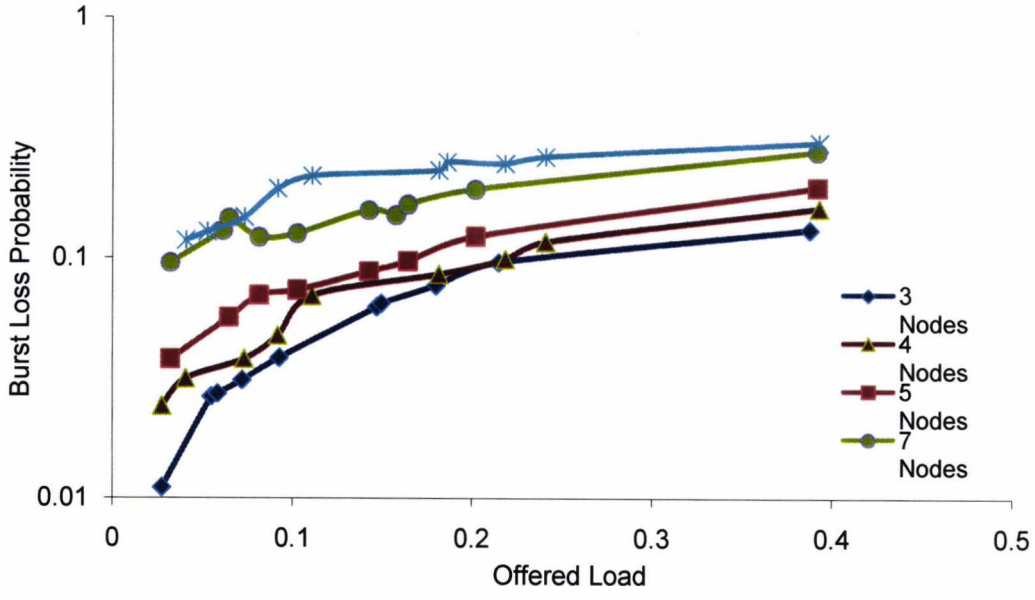


Figure 5-2 Burst loss probability for different number of edge nodes using FF (Single core node)

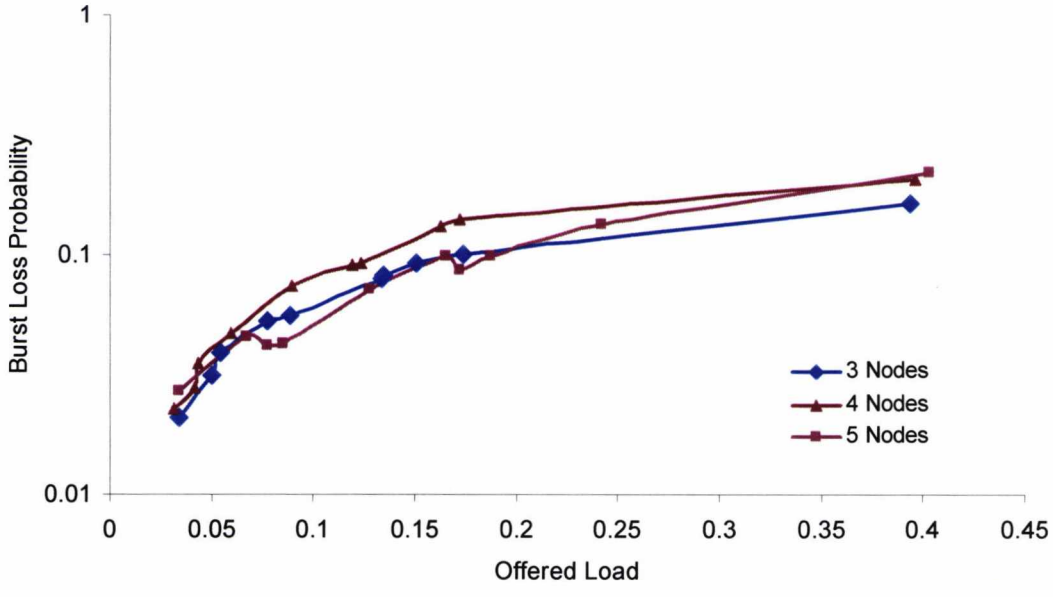


Figure 5- 3 Burst loss probability for different number of edge nodes using random channel selection (Single core node)

Figure 5-3 shows the single core node results while using a random scheduling algorithm. In this type of algorithm the number of edge nodes has a small effect on the burst loss probability. The ratios of burst loss probabilities between different numbers of edge nodes being used are small in all of the range of traffic loads applied.

As was shown in figure 3-16 channel selection at low loads can vary randomly with significant variation in channel selection, thus creating a false effect shown in figure 5-3 which shows that when five edge nodes are used the performance in burst loss probability seems to be better in lower loads compared to when four edge nodes are in operation. It can also be seen that at some low traffic loads using five edge nodes can be better even than when three edge nodes are in operation. In higher loads the variation in probability of selecting a channel is reduced as was shown in Figure 3-16 and so the main difference can only be seen at higher loads as it can be observed in figure 5-3.

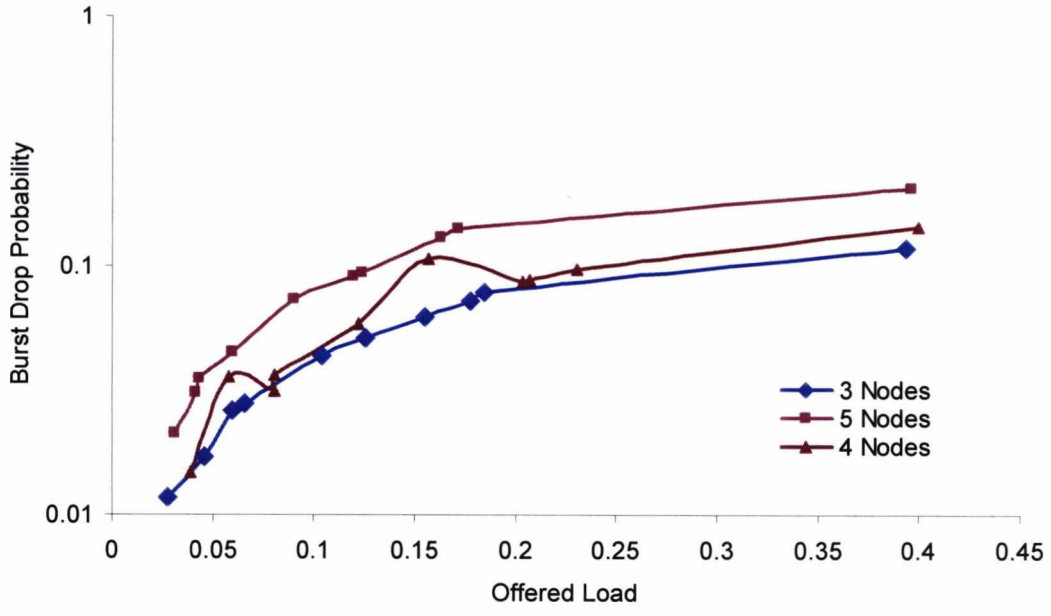


Figure 5-1 Burst blocking probability for different number of edge nodes using LAUC (Single core node)

Figure 5-4 shows the single core node results while using LAUC as a scheduling algorithm. This algorithm as mentioned before has the tendency to lower the ratio between burst selection probabilities of channels. This effect makes the ratios of burst loss probabilities at particular loads between different numbers of edge nodes used to be approximately similar. Figure 5-4 also shows an increase in burst loss probability as the number of edge nodes in operation is increased as was observed in FF.

The next step in the discussion of scheduling algorithms is a comparison between the three algorithms that have been presented until now. This is done using Figures 5-5 to 5-7, that show the scheduling algorithms when using three, four and five edge nodes.

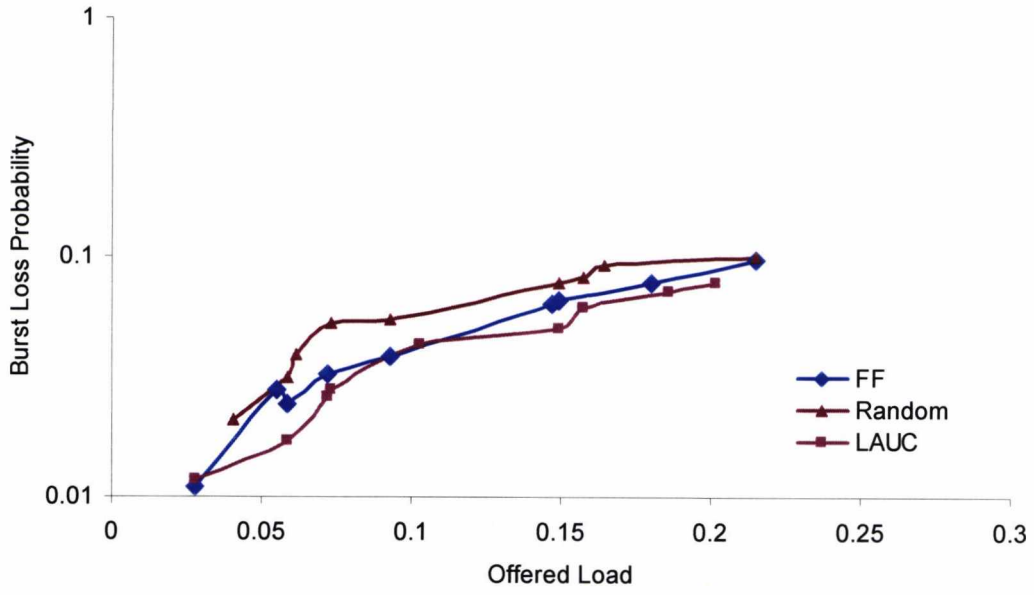


Figure 5-2 Burst blocking probability using 3 edge nodes for different scheduling algorithms (FF, Random, LAUC)

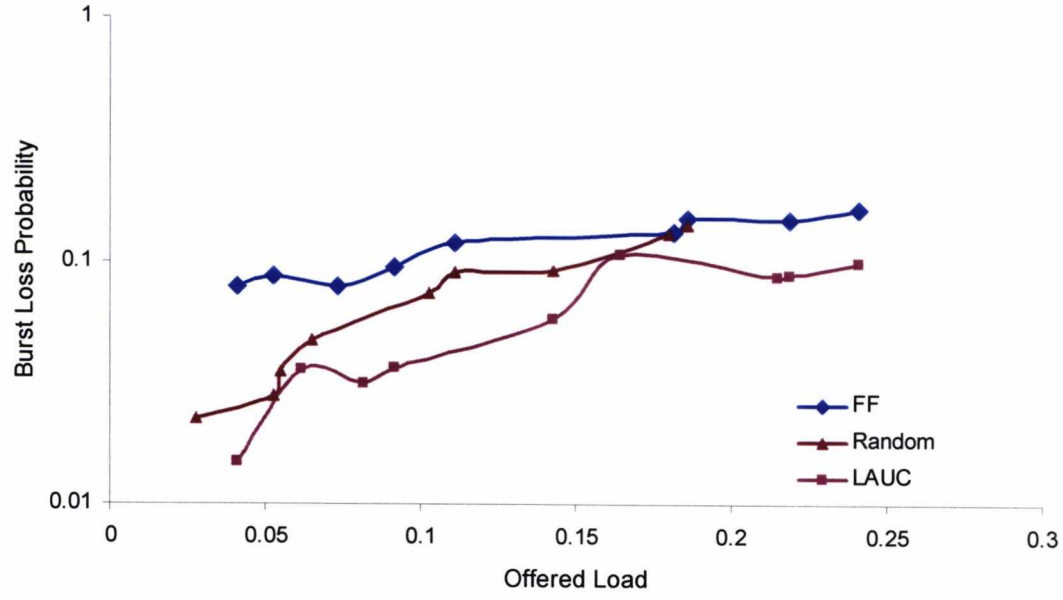


Figure 5- 6 Burst blocking probability using 4 edge nodes for different scheduling algorithms (FF, Random, LAUC)

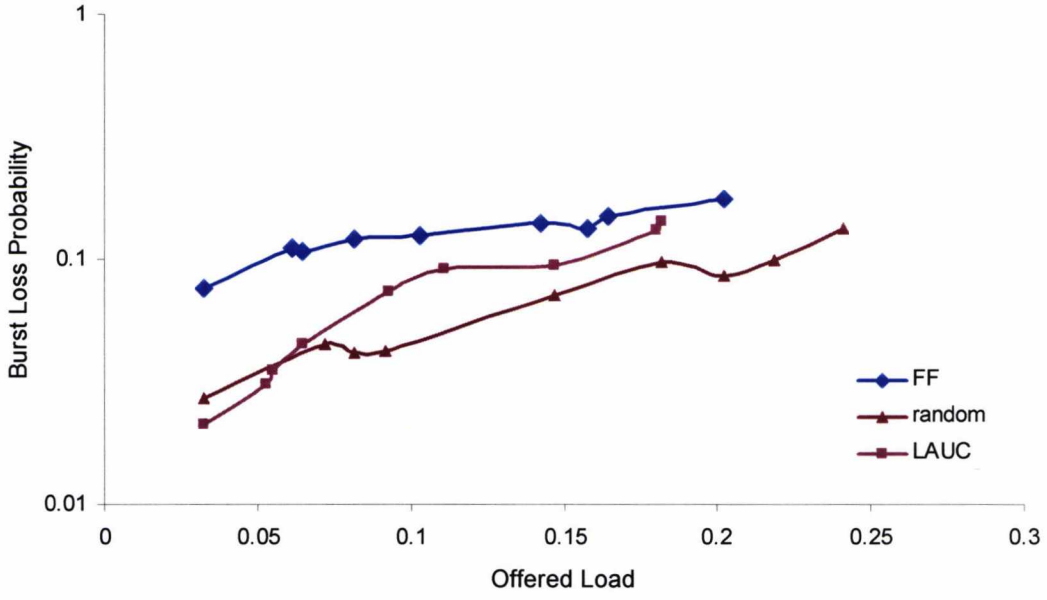


Figure 5- 7 Burst blocking probability using 5 edge nodes for different scheduling algorithms (FF, Random, LAUC)

The difference between FF scheduling and both LAUC and random scheduling for burst loss probability is small when high traffic load is applied in the single core network. At high loads the improvement of LAUC and random scheduling over FF is very small thus proving also what has been found by much other research [1], [2].

The main difference appears in lower traffic loads on the network (Figure 5-5). When the number of channels is larger than the number of edge nodes LAUC shows a slightly better performance than the other two algorithms. The FF scheduling also performs better than random scheduling. The average ratio of burst loss probability in low traffic between FF and LAUC is 1.2 and between random and FF is 1.2. On the other hand the average ratio between random and LAUC is 1.6.

In the case, as shown in Figure 5-6, where the number of channels is equal to the number of edge nodes FF performs a lot worse than the other two scheduling algorithms. The average ratio of burst loss probability between FF and LAUC is doubled to 2.808 and between FF and random is increased to 1.8. In this case LAUC still performs better than random but the average ratio between them is decreased slightly to 1.6.

In the Figure 5-7, where the number of edge nodes is larger than the number of channels, the random scheduling seems to be the best candidate from all three of the scheduling algorithms being investigated. Although at traffic loads lower than 0.1 LAUC seems to perform better than random scheduling the difference is very small. The average ratio of burst loss probability between FF and LAUC is slightly increased to 2.9 and between FF and random a larger increase to 2.6. The average ratio of burst loss probability between LAUC and random is decreased considerably to 0.8.

5.1.2 Optimisation of Scheduling Algorithms

In subsection 3.4.1 it was shown that for both FF and LAUC, most burst dropping occurs in the first two channels. If the order of the channels is different for each edge node then the burst dropping probability should also drop significantly. The order that each edge node will have has to be optimised accordingly. This optimization and the way the orders are created has been termed wavelength channel assignment (WCA). The order of channels (i.e. wavelengths) is now called a code and the assignment of the codes in each edge node is optimised accordingly. The following sections of this chapter are dedicated to explaining how this is accomplished and how a significant reduction in the dropping probability of an OBS network when using this new optimisation algorithm (WCA) on FF and LAUC, is possible.

5.2 Model Implementation

In this section the description of the modeling of the new optimisation of scheduling algorithms will be described. The first subsection, 5.2.1, describes the modeling of WCA in an OBS network and the second subsection 5.2.2, the method of generation of the codes needed for wavelength channel assignment. Subsection 5.2.3 shows the automatic code generation algorithm for creating the codes for each node.

5.2.1 Operation of WCA in OBS

In the use of WCA scheduling, each edge node receives a unique sequence of wavelengths, the code, transmitted via the control channel from a master core node. A master core node has the same functions as all other core nodes in the OBS network except that it has the unique capability within the network to create and send such WCA codes. In WCA, each edge node uses a scheduling scheme of FF or LAUC with the difference being that each edge node has an optimised set of codes with one code per destination. Naturally there is a limit on how many different sets of codes can exist, thus making the network performance dependant on the number of edge nodes and wavelength channels. Such a network using WCA can be seen in figure 5-8.

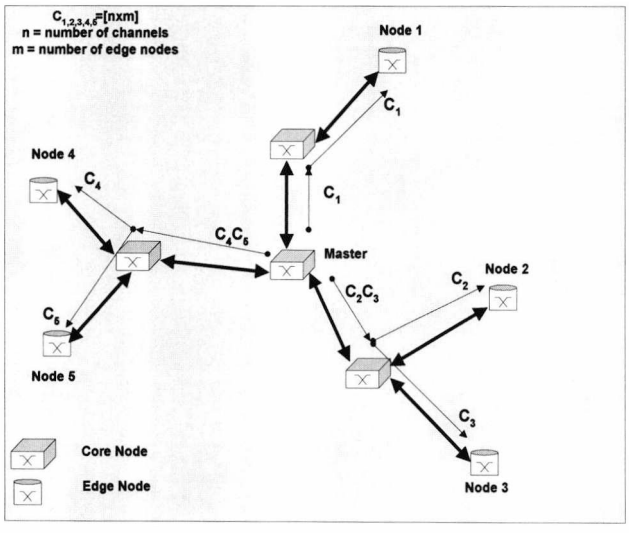


Figure 5- 8 Operation of Wavelength Channel Assignment

5.2.2 Fundamentals of WCA

It is assumed each edge node uses a scheduling scheme of FF or LAUC but that each will receive a different set of codes with one code per destination. Naturally there is a limit on how many different sets of codes can exist, thus making the network performance dependant on the number of edge nodes and wavelength channels. The number of codes that can exist is equal to $n!$, where n is the number of wavelengths used. The number of codes (C) needed for a network with m edge nodes is $C = (m-1) \cdot m$. Thus, if C becomes greater than $n!$, the WCA technique will no longer be able to behave in its desired manner. In general, WCA makes the possibility of an edge node transmitting to another via the same wavelength less probable. However, as the load on a node increases the probability of sending a burst on the same wavelength increases until eventually there reaches a point where WCA has no better performance than other scheduling techniques. This of course is expected with any type of scheduling technique, as at heavy traffic all channels will become fully utilized with no means to avoid some dropping.

In order for the WCA scheme to produce good performance the codes sent to all of the nodes must be optimized according to their "distance" and the network topology. The distance between two codes is simply the Hamming distance when the codes are written in binary form (thus, this distance is simply the number of entries in binary form at which two codes differ).

Therefore, each channel is represented by a binary word and these are combined together to give the final code. The number of bits in the final code is $n\log_2 n$ where n is the number of channels. Thus, in a 4 channel system, each channel is given a different two bit word and the final code is comprised by joining all words together producing a final 8 bit code. A rule must be followed in the way in which the words for each channel are assigned. The rule is that a sequence of channels must have the greatest distance in binary form with the opposite of that sequence. Careful selection of the words for each channel has to be made. So, for example, a channel sequence **1 2 3 4** (each of these numbers is assigned to a specific wavelength for the whole network and is unique for these wavelengths) has an

opposite sequence of **4 3 2 1,** and the distance between them should be designed to be a maximum (=8, for a 4 channel system).

One way to obtain this is to set channels **1→00, 2 →01, 3→10, 4→11.**

5.2.3 Automatic Code Generation

Each network produces a table according to a generation code as shown in table 5-1. The table contains for each edge node a code for each destination. In order to understand how the table is completed there must be an understanding on how the selection influences the core node's probability of dropping. It is easy to understand why in the vertical columns the codes have to have a maximum distance between them. This is of course to reduce the probability of two edge nodes sending simultaneously in the same channel to a third node. The codes in the horizontal rows must also have a distance between them. The reason behind this is that if they had the same codes the probability of sending on the last two channels of the code would increase dramatically, reducing the effectiveness of the vertical code selection. Although there must be a distance between the codes in the horizontal rows the distance does not have to be a maximum as this will make the optimisation in the generation of the matrices very difficult, if not impossible. So the algorithm that has been proposed is to generate codes in vertical columns with as high a distance as possible, and in the horizontal with a medium distance. After numerous simulation runs the optimisation algorithm was found to be effective and generated optimised matrices. The way of checking the optimisation was by calculating the distances between the codes in all the vertical columns and in the horizontal rows for different number of edge nodes. After checking thoroughly the results of the distances between the codes, multiple simulations were performed to view the results of the OBS networks for each of the optimised matrices created in burst loss probability compared to results of the networks with matrices generated randomly (i.e. not optimised). All the models used for our WCA simulations had optimised codes. The results in all cases provided better performance through a reduction in burst loss probability. In more detail, 3 OBS networks

were investigated with three different numbers of edge nodes. More on the results will be given on section 5.3.

Table 5-1 shows the matrix of the codes for the simple case of 3 edge nodes according to the rules described previously. The way the automatic code generation operates is by selecting a random code depending on the number of channels available and then selecting according to the number of nodes *n-1* codes with a medium distance of the first code selected previously, where *n* is the number of edge nodes in the network. This action will fill the first row of the matrix, the next step is then to select the maximum distance from available codes in each column based on the code in the first row of the column concerned. Every time a code is selected with maximum distance another check is done to make certain that the distances between the other codes on the matrix do not become too low, thus trying to make certain that rows have medium to minimum distances between them, while there are medium to maximum distances in columns. It has to be noted that the algorithm will try and use each code only once, if possible.

Table 5-1 Code matrix generation table

Destination / **Source**	Node 1	Node 2	Node 3
Node 1		1234	3412
Node 2	3142		2143
Node 3	2413	4321	

Although the algorithm for producing the matrix can work well in a very small network with few nodes, it becomes very difficult for large networks because codes will start overlapping and low distance codes will occur more frequently. The way to overcome this problem is to establish the number of hops between two edge nodes. The number of hops between them in the network can be found very easily by knowing the number of core

nodes that exist between them. This number of hops can help us, in a large network to produce better optimized codes by making certain that if overlapping codes of low distance are needed, these codes will only be assigned to edge nodes that have a large number of hops between them.

5.2.4 WCA with FF and LAUC

Matrix generation for WCA-FF and WCA-LAUC is different. Although in WCA the table has dimensions of NxN, where N is the number of edge nodes, WCA-LAUC will only use one of the codes in the vertical column depending on which is the first destination used, thus reducing the dimension of the table to 1xN. In WCA-LAUC after all channels are used at least once, the scheduling is carried out by selecting the last used channel. This means that initially when no data has been sent the channel selection is done in the same way as in FF. After all of the channels have been used at least once the effect of the WCA code (this effect will now be called weighting for simplicity) is abolished. So only an initial weighting is needed for the WCA-LAUC and thus a one dimensional table. This in turn has as an effect that in a WCA-LAUC algorithm the number of edge nodes that can be used with different codes is increased. This will be seen more clearly in the next section (5.3) where some of the results concerning the performances of these new algorithms will be shown.

Another way to improve even further the performance of WCA-LAUC is to keep the weighting of the coded channels. WCA-LAUC although it weights the channels in each edge node separately when a simulation starts the effect of this coding plays no more part as the load increases. A way to keep using the coded channels is by weighting each channel as the simulation goes on. This is done by reducing the unscheduled time of each channel by a weighted factor, more for the first channel than the second one and more for the second one from the third and so on, thus creating a weighting on each channel as the load increases. Multiple simulations have been performed to find the best weighting factor. This improved version of WCA-LAUC will now, in the remainder of the thesis, be referred to as WCAW(eighted)-LAUC.

5.3 Simulations and Results For WCA

In this section results will be presented to verify the improvements made on the already existing algorithms using the wavelength channel assignment algorithm. The results are separated into two subsections, the first for a network with multiple edge nodes connected to a single core node as described in subsection 5.1.2 and the second for a number of networks with multiple core nodes and edge nodes with different network topology.

5.3.1 Single Core Node Analysis

This section is dedicated to the results obtained while using a single core node OBS network as described in section 5.2. All the parameters remain the same and the improvement in burst loss probability performance between the WCA scheduling technique and FF / LAUC will be presented.

5.3.2 WCA-FF

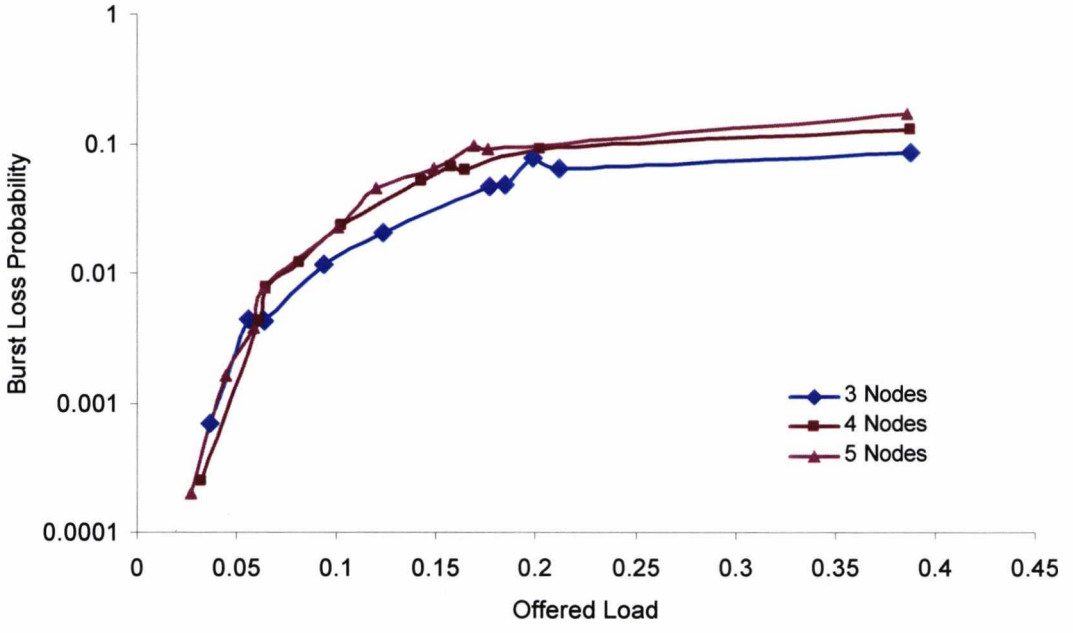


Figure 5-9 Burst blocking probability for different number of edge nodes using WCA-FF (Single core node)

The results presented in figure 5-9 show the single core node results while using the new concept of WCA-FF. It can be seen that by increasing the number of edge nodes from 3 to 4 a degradation in performance is visible. As the number of edge nodes is increased to 5 the performance is degraded even further but this time the degradation in going from 4 to 5 edge nodes is smaller than in going from 3 to 4 edge nodes. This happens as the codes used, although different for each node, may have a distance between them equal to 2 (smallest case distance). This is expected as the number of codes needed is higher than the number of possible different codes as described earlier.

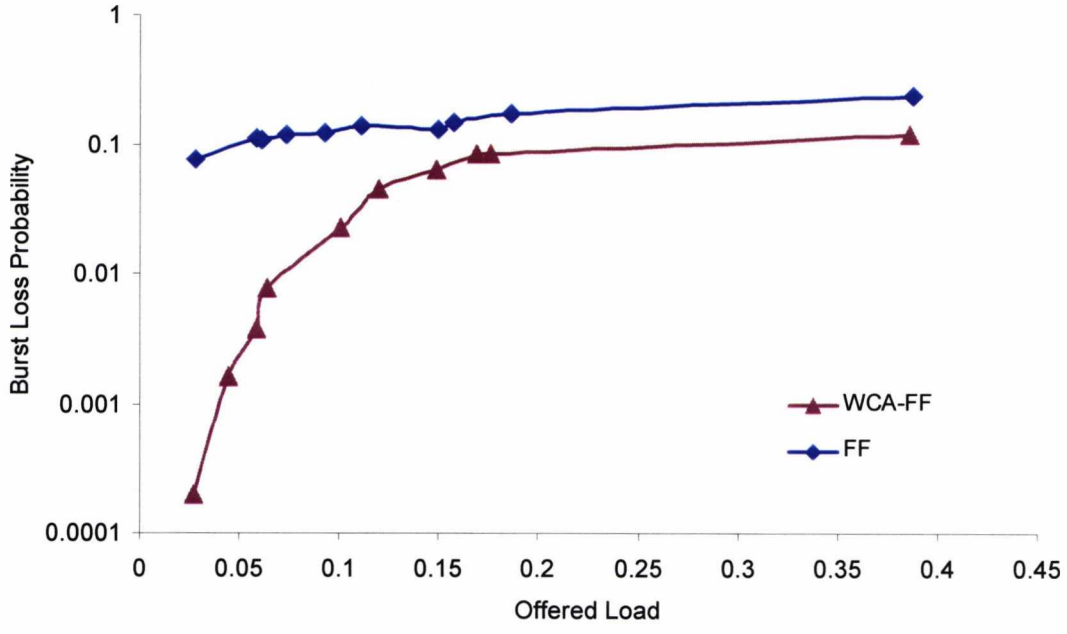


Figure 5-3 Burst blocking probability for FF and the improvement by WCA

The results in figure 5-10 shows the burst loss probability versus the load for the system with one core node and five edge nodes for FF and WCA-FF scheduling algorithms. At low loads the difference between the scheduling algorithms is profound and the performance improvement gained by using WCA is apparent. At higher loads the burst loss probability saturates for both schemes and there is only a small (although noticeable) difference between them. The improvement using WCA noted is dependent on the number of edge nodes operating. As the number of edge nodes increases there is still an improvement but the level of improvement reduces. This is expected of course as the codes have smaller distance between them. So it can be concluded that improvement is highly dependent on the number of channels and number of edge nodes operating.

5.3.3 WCA-LAUC and W-WCA-LAUC

The results presented in figure 5-11 show the single core node results while using the new concept of the WCA-LAUC. It can be seen that the ratios between burst loss probability between number of edge nodes in operation remains similar to those in the case of LAUC. Also shown is the increase of burst loss as the number of edge nodes increases but the ratio between the burst loss probabilities is larger than noted in WCA-FF.

Figure 5-12 shows the single core node results while using the weighted WCAW-LAUC. In this case the ratios between burst loss probability and number of edge nodes in operation do not remain similar to those observed in the case of LAUC and WCA-LAUC. This can be explained from the small difference in the selection probability of different channels in LAUC compared to the much larger difference in FF, as shown previously in figure 5-7 where for FF in low loads most of the scheduling is done in channels 1 and 2 and nearly nothing in channels 3 and 4, where as in case of FF all channels have scheduled bursts but with channel 1 shown a much larger preference over the 2^{nd}, 3^{rd} and 4^{th} channel. Thus LAUC shows a better balance between channel selections than FF which makes the increase in number of edge nodes a more profound difference in burst loss probability. In the case where the WCA-LAUC is weighed the difference stops being large as the selection probability between channels is larger, and thus closer to the way in which FF operates.

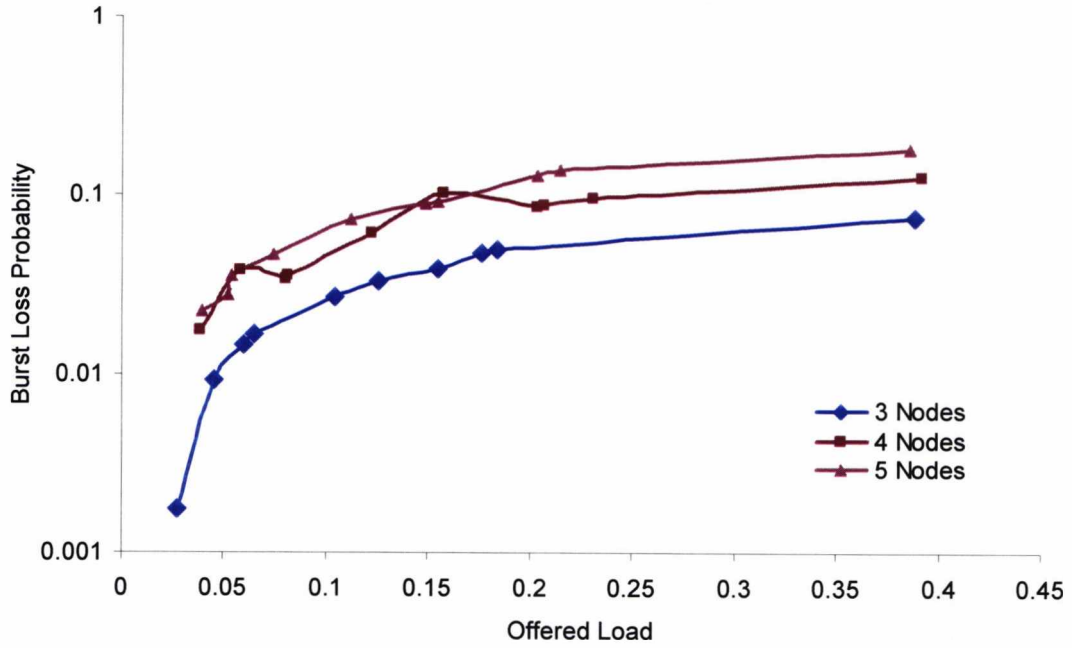


Figure 5-11 Burst blocking probability for different number of edge nodes using WCA-LAUC (Single core node)

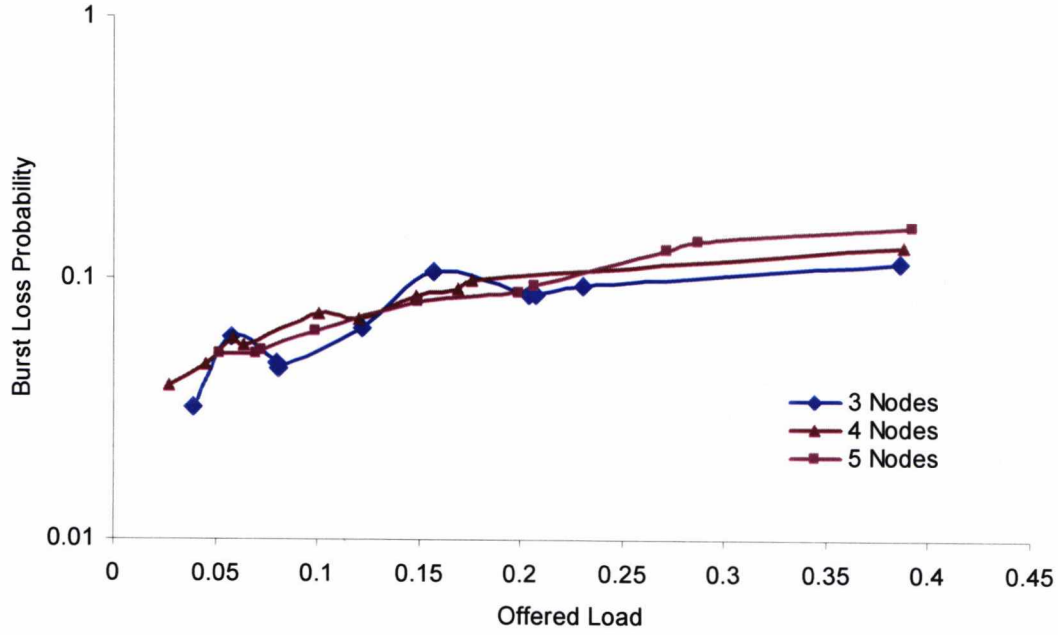


Figure 5-12 Burst blocking probability for different number of edge nodes using WCAW-LAUC (Single core node)

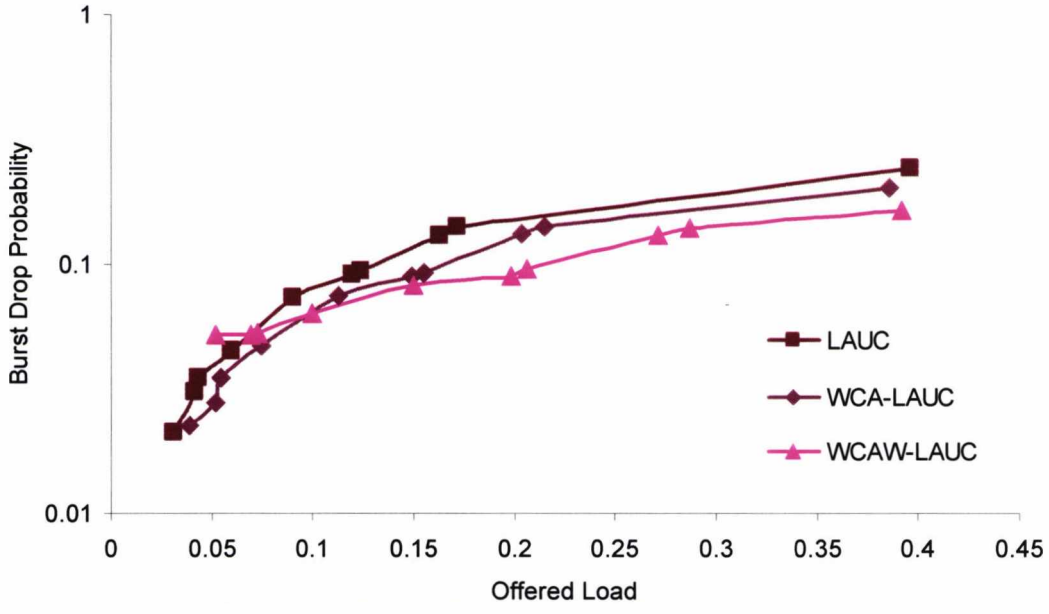


Figure 5-13 Burst blocking probability for LAUC and the improvement by WCA and the Weighted-WCA (Single core node)

The results in figure 5-13 show the burst loss probability versus the load for the system with one core node and five edge nodes for the LAUC, WCA-LAUC and Weighted-WCA-LAUC scheduling algorithms. From the figure it can be seen that an improvement between WCA-LAUC and LAUC at low loads exists although not as large as the one that was observed between FF and WCA-FF. Looking at the case of the WCAW-LAUC, at very low loads the performance is lower than LAUC and WCA-LAUC but with a slightly increased traffic it outperforms both LAUC and WCA-LAUC.

5.3.4 Multi Core Node Analysis

The final results that are going to be presented are for the investigation of the WCA scheme when functioning for multi-core node networks. In order to make certain that results were accurate to real time systems different topologies were tested. Figure 5-19 and 5-21 shows the topologies used for testing our results. Four topologies were used to provide

significant proof of the correct operation of WCA, in each topology routing was set to find the destination according to the least number of hops. The distances between an edge node and a core node remain constant throughout the networks (equal to 120 km), the same as that between core nodes, and the distances of sources to the edge nodes are assumed negligibly small.

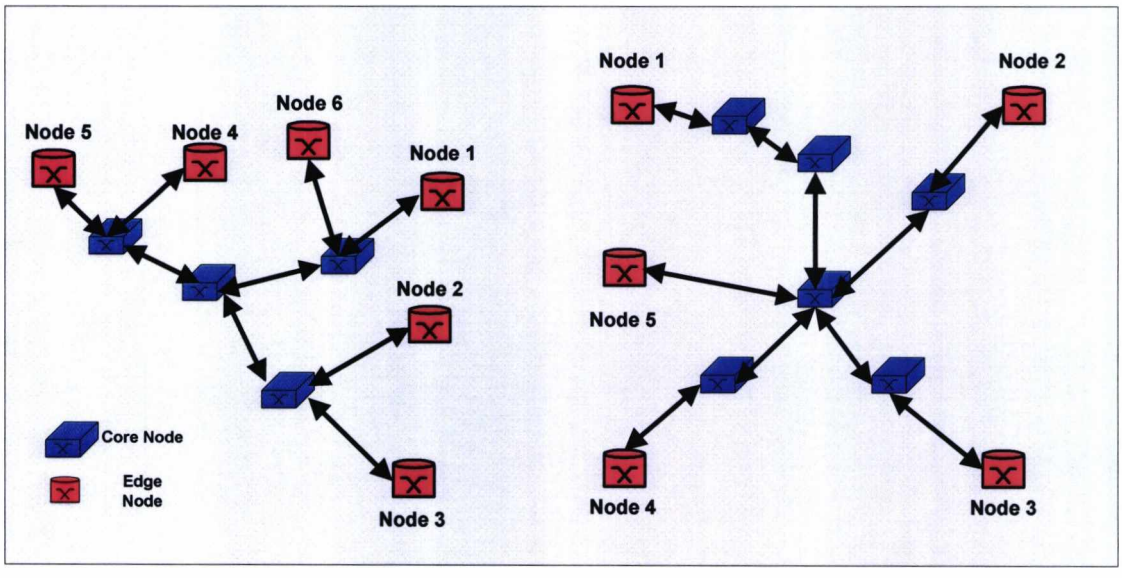


Figure 5-14 OBS network for different topologies in OPNET for investigation of WCA

Results were obtained from both networks shown in figure 5-14 for all scheduling algorithms described in this chapter. Both networks produced very similar results where the results of the second network can be seen in figure 5-15. At low loads FF seems to be by far the worst case scenario followed by all other scheduling techniques. Second worst case is LAUC with a difference in burst loss probability of about 6% at 7% traffic load. The difference between FF and LAUC drops gradually until about 18% of traffic load where FF starts performing better than LAUC and reaches saturation soon after. The interesting part of this phenomenon is that it occurred in both network results of figure 5-15 but it did not occur during the single core network results. The explanation can be either because of the scheduling algorithm reacting differently in multi core networks or due to the topology schemes used thus far.

The next three scheduling algorithms that we are concerned with are the random, WCA-LAUC and WWCA-LAUC scheduling algorithms. All three schemes seem to have

very similar results with random scheduling being fractional worse than the other two at lower loads but fractionally better at lower loads. WCA-LAUC and WWCA-LAUC seem to be extremely similar with WWCA-LAUC performing better in burst loss probability. Although in a single core node network a larger difference in burst loss probability was observed between the WCA-LAUC and WCAW-LAUC schemes there seems to be a very small improvement in multi core networks. Although this is the case, both schemes perform much better than LAUC and FF and marginally better than random scheduling.

WCA-FF can clearly be seen offering a far better performance in both low and high traffic loads. Thus it can be safely concluded that WCA has provided a clear improvement on burst loss probability when used in conjunction with FF or LAUC.

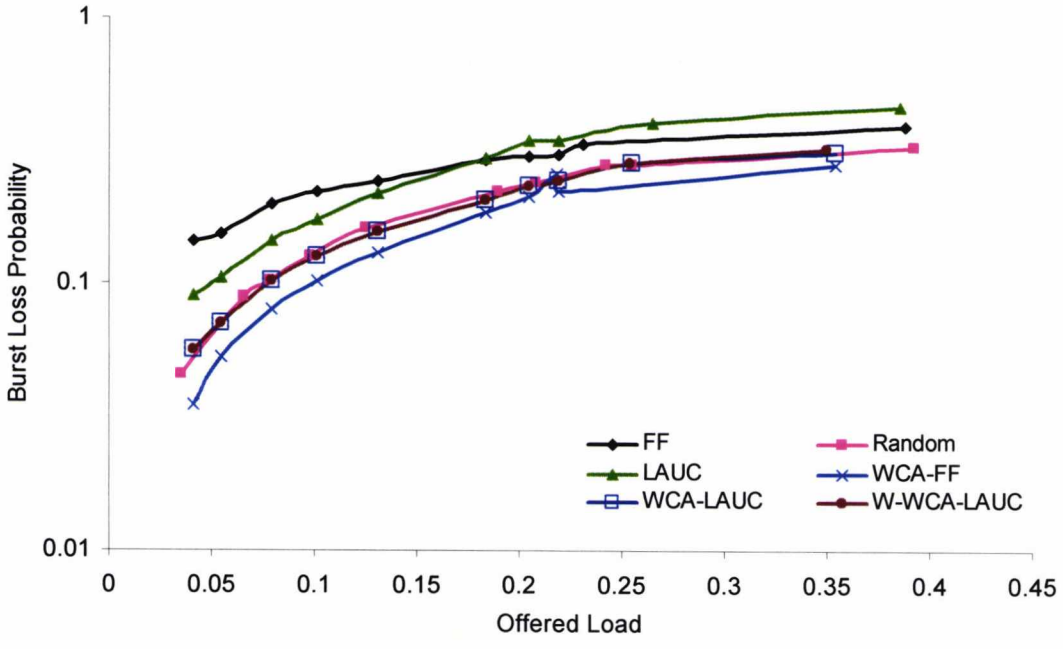


Figure 5-15 Burst blocking probability for different scheduling algorithms using multi core network

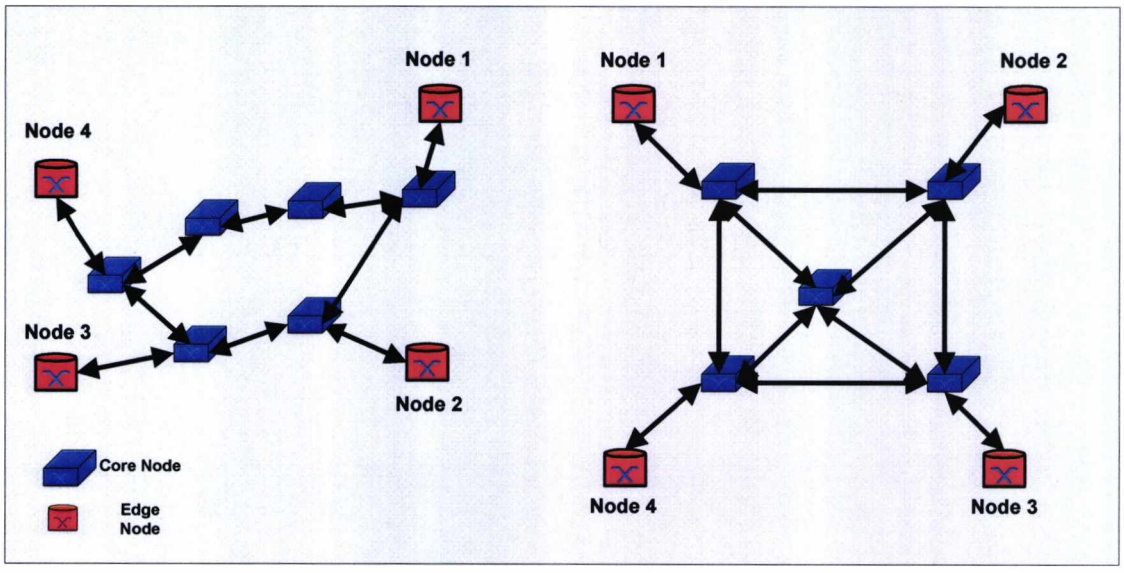


Figure 5-16 OBS design for different topologies in OPNET for investigation of WCA

Figure 5-16 shows another two network topologies used to investigate the performance of the WCA scheme in multi core node networks. The networks have four edge nodes and so an improvement in burst loss probability will be seen as WCA will have better optimised codes distributed in the edge nodes as only 4 channels are available. Again results were obtained for both networks and in figure 5-17 one set of these results is shown.

At low loads FF is again shown to be the worst scheduling algorithm from the investigated algorithms with LAUC being second worst closer this time than when 5 and 6 nodes were in operation. LAUC performs by a small fraction in burst loss probability worse than FF in higher loads as was also observed when 5 and 6 edge nodes were used. This provides us proof that the reasoning for this occurrence is due to the multi core node network rather than the topology in investigation. This can most likely be explained in that LAUC operates only in edge nodes rather than in core nodes ultimately leading to higher burst loss as bursts propagate through multiple core nodes. It has to be noted that if wavelength conversion was available LAUC could also be applied in core nodes which would lead to a better performance than FF at higher loads, although still not better according to [1].

Random scheduling seems to perform similarly as before at lower loads but seems to result in an increase in burst loss at higher loads with saturation levels reaching similar values to those with FF.

When the WCA scheme is used on LAUC the burst loss is reduced in comparison with FF and LAUC and by a small factor compared to random scheduling. The same can be said for WWCA-LAUC as it follows the same improvement with the non weighted WCA with only small signs of further improvement. It can be concluded that in multi core node network scenarios both weighted and simple WCA schemes when applied in LAUC perform similar and thus weighting should only be used as an alternative if wavelength conversion is in operation, where the weighting could be applied in core nodes to improve the performance in burst loss.

In the case of FF the operation of WCA presents a significant improvement in low loads comparable to the improvements obtained in single core node networks. This can be explained by the fact that with 4 edge nodes (for which code optimisation was done) the WCA-FF scheme clearly provides a reduction in burst loss due to the minimization of the probability of sending bursts on the same wavelength channels.

In summary, topology seems to have little bearing on the performance of the WCA scheduling algorithm that an OBS network could be using.

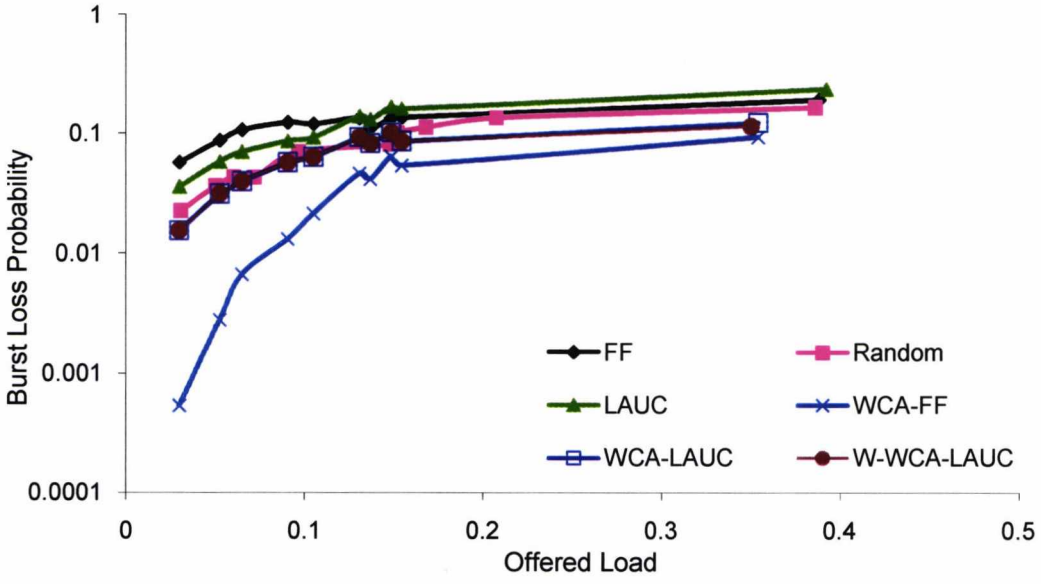


Figure 5-17 Burst blocking probability for different scheduling algorithms using multi core network

5.4 Summary

Results have been obtained for the channel selection probability for FF, LAUC and random scheduling. The results were examined thoroughly and the weaknesses that accompanied them explained with burst loss being the main factor in the investigation. It was shown that the manner in which channels are selected is the main factor affecting the performance in burst loss probability for the scheduling algorithms. This has led to the design of a scheme that could be applied in conjunction with FF and LAUC in order to improve the burst loss performance of the original schemes. The scheme designed has been called wavelength channel assignment as its purpose is to assign according to the number of destinations an equal number of codes each representing the order in which channels are supposed to be selected in each edge node.

The next step was investigating the performance of the newly developed scheme compared to the previous scheduling algorithms. The investigation was separated into two parts: the single core node and the multiple core node investigations. The WCA

optimisation scheme was investigated in a network that has no wavelength conversion or any type of buffering, and showed remarkably improved performance compared to FF, LAUC and random scheduling algorithms in both parts of the investigation.

To be specific, results were obtained for a network with a larger number of active edge nodes than number of channels. In fact, it is interesting to take this case, where there are insufficient codes for WCA to perform optimally, first. With a greater number of edge nodes than channels available the performance of WCA starts becoming very similar to the original scheduling algorithms as the algorithms will generally select the same channels on all edge nodes and so provide similar burst dropping performances. Random selection has an advantage for low/moderate loads compared to FF whereas it has higher burst loss at higher loads as it distributes the load uniformly in each channel, making the probability of transmitting on the same wavelength higher than the other schemes. Finally, we can see that with the same or a lower number of edge nodes than channels (for which code optimisation can be done) the WCA scheme clearly provides a reduction in burst loss due to the minimization of the probability of sending bursts on the same wavelength channels. We must re-iterate that no wavelength conversion has been applied and no buffering resolution has been adopted in any of the results presented, which suggests that the WCA algorithm will be appealing for near term implementation. Another issue that has to be taken into account is that WCA-LAUC and WCAW-LAUC might provide further reduction in burst loss if applied in core nodes when wavelength conversion is available.

Although, the values of burst dropping probability were different for the different network topologies tested, the ratios between the values for the different scheduling algorithms remained almost constant, demonstrating that performance was generally independent of the network topology and that the results can be assumed typical.

5.5 References

[1] Y. Xiong, M. Vandenhoute, H. C. Cankaya, "Control architecture in optical burst-switched WDM networks", *IEEE J. Sel. Areas Commun.*, vol. 18, pp. 1838-1851, 2000.

[2] C. Qiao and M. Yoo "Optical burst switching (OBS) – a new paradigm for an optical Internet", *Journal of High Speed Net.*, vol. 8, pp. 69-84, Jan. 1999

[3] Donald Gross, Carl M. Harris, "*Fundamentals of Queuing Theory*", John Wiley & Sons Inc, 1998.

[4] M. Dueser and P. Bayvel, "Bandwidth utilization and wavelength re-use in WDM optical burst-switched packet networks", *Proc. IFIP 5th Working-Conference on Opt. Net. Design and Model. (ONDM 2001),* vol. 1, pp. 23-24, Vienna, Austria, Feb. 2001.

CHAPTER 6

COMPARISON OF DIFFERENT NOVEL COMBINATION OF ALGORITHMS IN OBS

This Chapter is dedicated to viewing the performance of OBS networks with different combinations of algorithms for burst assembly, QoS and scheduling. The performance will be measured by viewing the results of the different algorithm combinations in average end to end delay of the bursts and in the burst loss probability.

The chapter is divided into three sections; the first section (6.1) explains the simulation environment in detail such as the type of network, the number of the nodes and the type of the traffic that was used to send traffic from the source to the destination, also in this section, we present the evaluation parameters that are used and measured for the different algorithms. In Section 6.2 the simulation results of all the combinations of algorithms will be presented and the results compared to some results from the literature; explanations are provided to why certain results differ from those previously reported. Finally, in section 6.3 a discussion follows to summarise final thoughts on the results obtained and to explain the advantages and disadvantages of each of the algorithms.

6.1 Simulation Environment

Extensive simulations were conducted to evaluate the performance of OBS interconnect ability between burst assembly, scheduling and CoS algorithms. A comparison was made based on burst loss probability using all combinations of the algorithms that were implemented and discussed in Chapter 4 and 5. The simulations were implemented using OPNET 8.00, 10.0 and 10.5. The nodes that were used in the simulations were nodes based on models described in Chapter 4 with different types of topology as described at the end of Chapter 5. Each link between the nodes was assumed to be 120 km. The traffic type was defined by an "On/Off" source which represents a bursty source. The source and destination nodes were randomly selected. Each source transmitted data packets into a link with data rate of 20Gbs with different inter-arrival times depending on the load needed for each simulation.

The simulations were performed on four different network topologies as described in Chapter 5. The maximum link distance between source and destination was 5x120km and the smallest was 3x120km. The simulation running time was set to 7 seconds with all sources stopping transmission at 5 seconds, thus leaving 2 seconds for the network to handle all packets sent. The simulation time used was found after trial and error in order to correctly understand and explain the results obtained. Table 6-1 presents all the information about the simulation environment.

Table 6-1 Simulation Environment Information

Number of the topologies	4
Total simulation time (seconds)	7 Seconds
Packet size (byte)	65535 bytes (Maximum IPv4 packet)
Link distance	120 km
Single Channel Data Rate	4.8 Gbps
Reservation Technique	JET
Burst Assembly Algorithms	FAT/VAT/MBLFAT
CoS Algorithms	AOF/ Preemptive
Scheduling Algorithms	FF/Random/LAUC/WCA-FF/WCA-LAUC/ W-WCA-LAUC

The reservation technique used was the Just Enough Time (JET) (as seen in Chapter 2). The burst assembly algorithms used were the Fixed Assembly Time (FAT), the Variable Assembly Time (VAT) and the Minimum Burst Length Fixed Assembly Time (MBLFAT) (as seen in Chapter 3). For class of service two algorithms were used the Additional Offset time (AOF) and the Preemptive (as seen in Chapter 3).

6.2 Simulations and Results

Approximately 500 simulation runs were performed. The current subsection is divided up according to the burst assembly algorithm used. The reason behind this division is that for each burst assembly it was necessary to perform simulation runs for multiple values of

their parameters, such as the maximum burst size and the assembly time. The values for parameters of each assembly algorithm were changed every time a new scheduling algorithm was used until we found the best performed parameter of each algortihm for bursts dropping probability. These simulation results will not be presented in this thesis as the burst dropping probability variations are similar to what has been described in the literature [1-5]. After each set of simulation runs the best performing parameter is used and the results on burst dropping probability are recorded and saved. In the next subsection we will try to present the most interesting results that were recorded, and discuss these results in more detail.

We will now show the effect of different combinations of scheduling and assembly algorithms and CoS on the performance of an OBS system. Of course, many different combinations were investigated, but we restrict ourselves to some key findings in this thesis.

6.2.1 FF with AOF

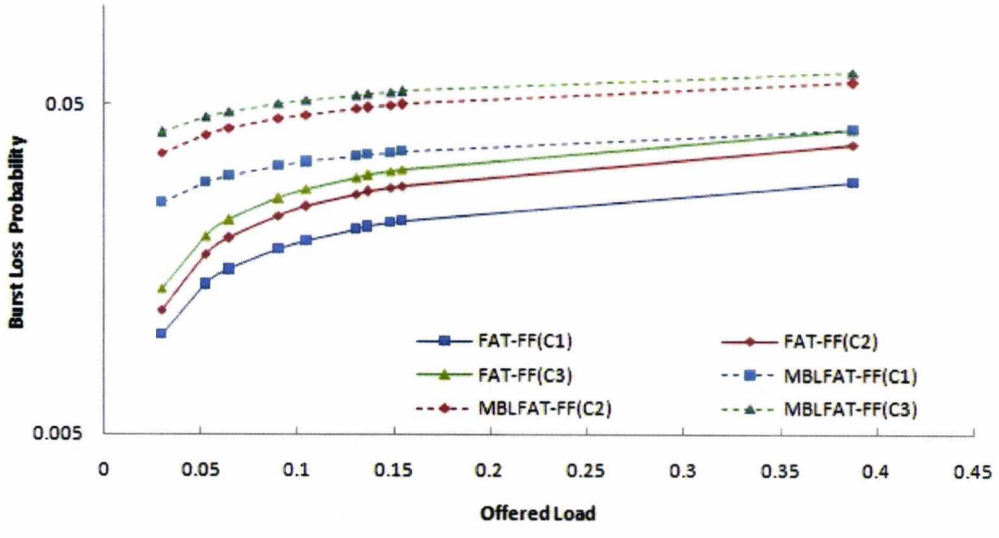


Figure 6- 1 Burst Loss Probability comparison of FAT and MBLFAT for the FF scheduling algorithm.

Figure 6-1 shows the results for the First Fit (FF) scheduling algorithm for burst assembly schemes of FAT and MBLFAT (for a network of 4 edge nodes). MBLFAT, which is purportedly an improvement on FAT, results in higher burst loss for each class of service.

This can be explained by looking at the sizes and the number of bursts in any period of time. The burst sizes in FAT are not controlled; thus, fewer bursts are created in any period of time while the bursts will be longer than those generated in MBLFAT. In the FF algorithm the order in which channels are checked for each edge node remains the same, so the more bursts that are created, the higher the dropping probability will be. It should be noted that the burst loss ratio for the AOF CoS does not change for the two compared assembly algorithms; hence the offset time can be said to have no effect on these combinations.

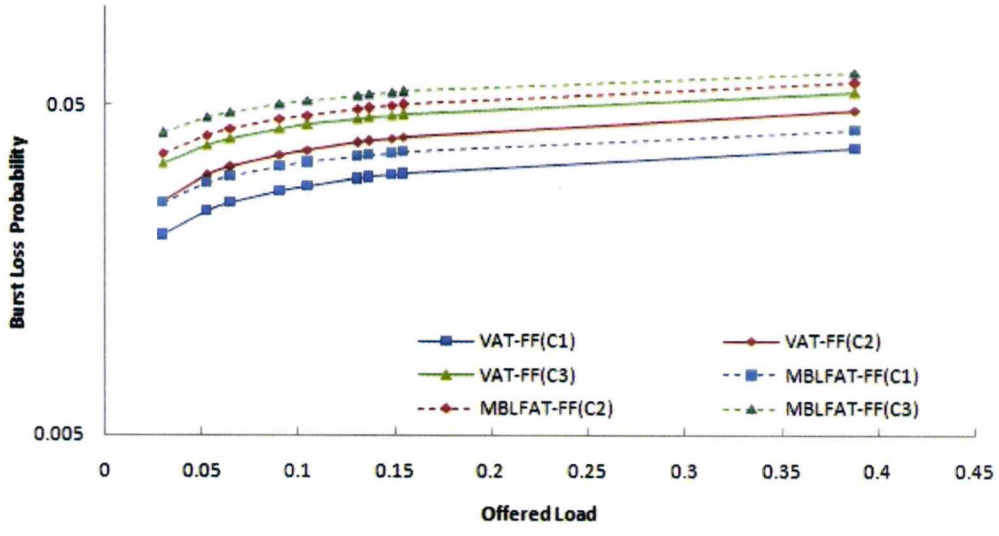


Figure 6- 2 Burst Loss Probability comparison of VAT and MBLFAT for the FF scheduling algorithm.

In Figure 6-2, VAT and MBLFAT are compared. VAT produces better burst dropping probability results in all CoS but not that significant as between MBLFAT and FAT. With the VAT algorithm it can be observed that the difference in burst dropping probability between the CoS ratios still remain similar. VAT keeps changing the assembly time depending on the average burst length calculated, thus the burst sizes are always kept between certain limits, and so the burst will never become too small or too large in order to affect the networks burst loss probability as FAT and MBLFAT.

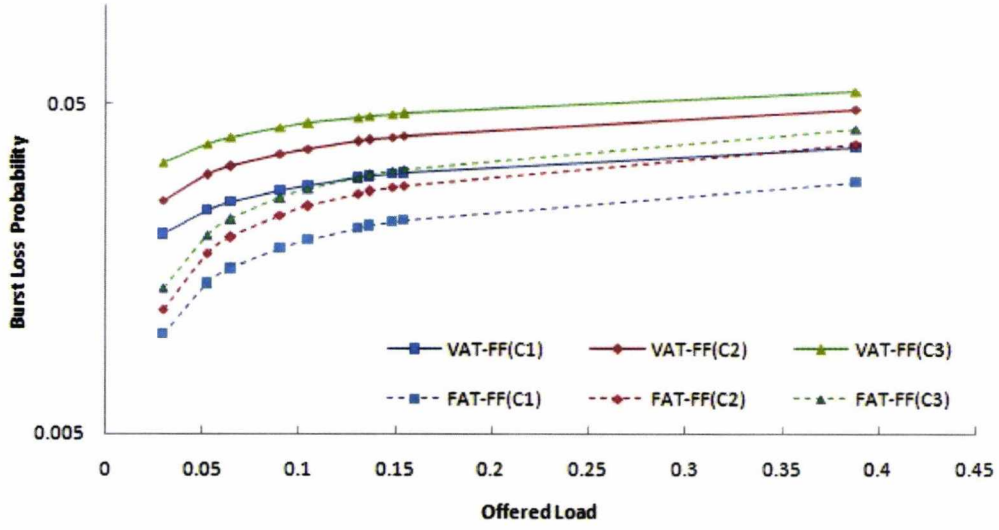


Figure 6- 3 Burst Loss Probability comparison of VAT and FAT for the FF scheduling algorithm.

In Figure 6-3, VAT and FAT are compared. FAT produces lower burst dropping probability for all CoS. VAT produces a variable burst size by changing the assembler time according to the available bandwidth and the average burst length. This in theory should make the burst drop probability better in VAT but this is not observed. As we mentioned before when comparing the MBLFAT with FAT the problem arises due to the way that FF operates.

From Figure 6-1 to 6-3 it can be concluded that the performance of an OBS network, while using the FF scheduling algorithm, in burst drop probability is dependent on the choice of the burst assembler algorithm. The FF scheduling algorithm operates better when the FAT algorithm is in operation compared to MBLFAT and VAT due to the sizes of the burst and the number of bursts. The sizes of the burst in VAT and MBLFAT are limited and thus both algorithms generally will create lower size burst from FAT at least at higher loads. FAT in large traffic loads will generate large bursts where MBLFAT will generate many bursts but smaller in size than FAT. In the case of VAT the number of bursts and the size of the burst are controlled to certain limits depending on the parameters used for setting up the algorithm.

6.2.3 WCA-FF with AOF

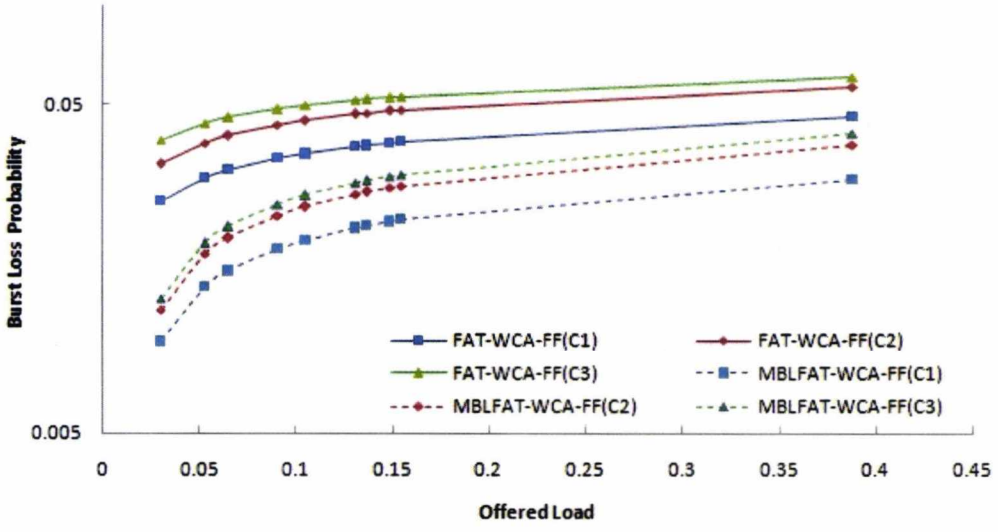


Figure 6- 4 Burst Loss Probability comparison of FAT and MBLFAT for the WCA-FF scheduling algorithm.

In Figure 6-4 results when using WCA-FF with the two burst assembly algorithms are presented. For MBLFAT compared to FAT the performance has improved dramatically for all CoS. In this type of scheduling algorithm the controlled size of a burst plays a very important role, as the scheduling is now also controlled. In FAT only a timer is used to set the size of the burst which has the effect of creating large bursts occupying large time periods thus the probability of dropping for each channel should increase. However, in MBLFAT the sizes are controlled up to an optimized maximum for the purpose of reducing the dropping probability. When FF was used we show that although MBLFAT should have decreased the burst dropping probability this was not the case. The reason given in subsection 7.2.1 was that FF produces lower burst dropping probability when dealing with large size bursts and fewer number of bursts. When WCA-FF is in operation though this is not the case, now the arrival of lower size and higher number of bursts are better selected on which channel they are transmitted.

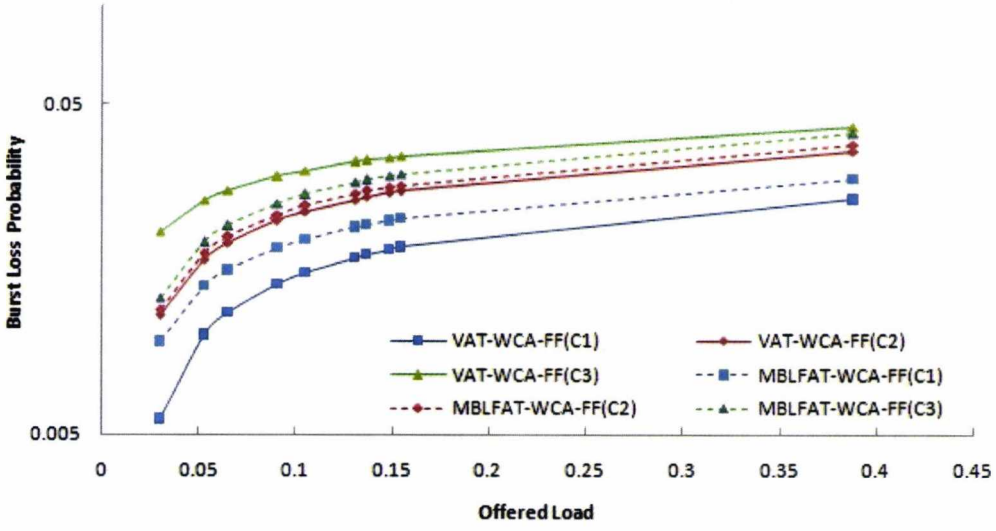


Figure 6- 5 Burst Loss Probability comparison of FAT and MBLFAT for the WCA-FF scheduling algorithm.

Figure 6-5 shows the results on burst loss probability when using the VAT and MBLFAT assembly algorithms and the WCA-FF scheduling algorithm. It can be seen that VAT performs better than MBLFAT in all CoS. In this situation the burst size is controlled by both assembly algorithms. The difference between the assembly algorithms appears because the MBLFAT algorithm has no control over the size of the burst in relation to the bandwidth available. The VAT algorithm, on the other hand, keeps a limit to the burst size which is dependant to the available bandwidth through controlling the assembly time. VAT and WCA-FF together control the burst sizes, the number of bursts and the selection of channel, thus creating a balance between all of these aspects and as a result producing a lower burst loss probability.

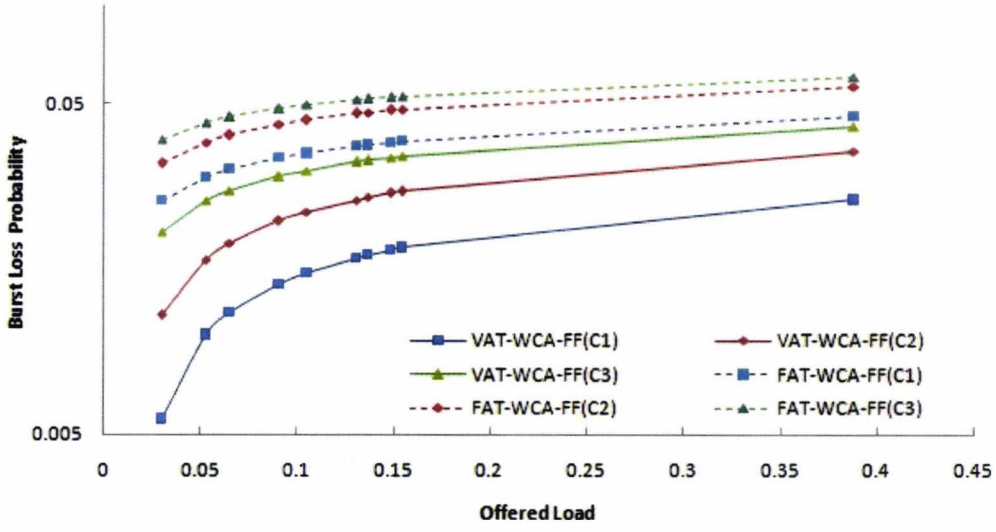


Figure 6- 6 Burst Loss Probability comparison of VAT and MBLFAT for the WCA-FF scheduling algorithm.

Figure 6-6 shows the results on burst loss probability when using the VAT and FAT assembly algorithms and the WCA-FF scheduling algorithm. VAT outperforms FAT in all CoS. As it was stated before FAT has no control over the burst sizes and thus VAT clearly operates as stated in literature.

To conclude, when using WCA-FF algorithm the burst size plays an integrate role in the performance on burst loss probability. Thus VAT and MBLFAT which can control the sizes of the bursts perform better than FAT. On the other hand VAT outperforms MBLFAT because the control of the burst size is not only dependant on the incoming traffic, but also, on the available bandwidth.

6.2.4 LAUC with AOF

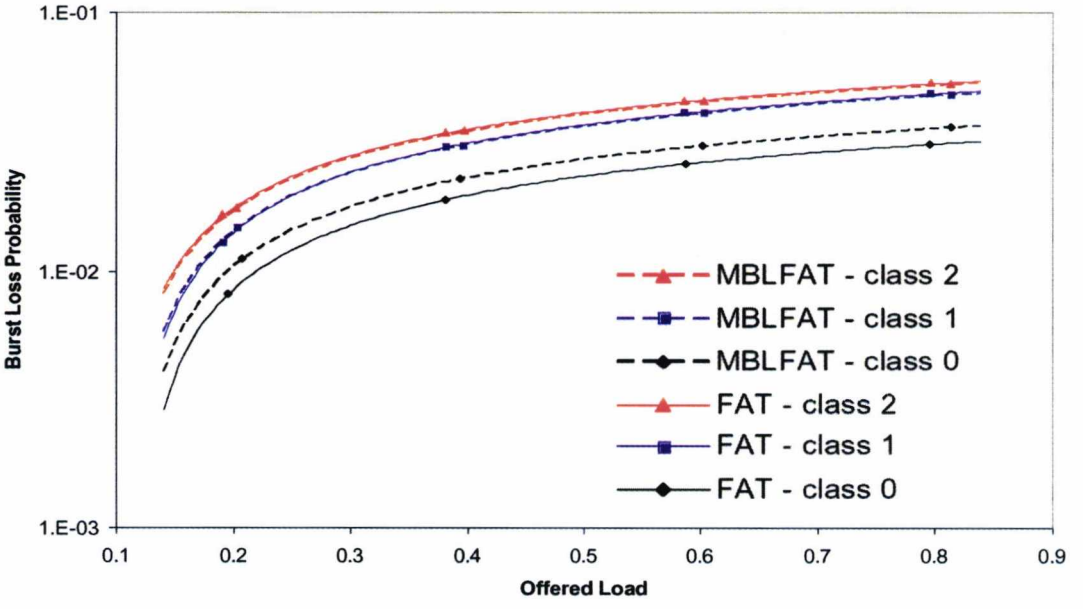


Figure 6- 7 Burst Loss Probability comparison of FAT and MBLFAT for random scheduling algorithm.

When using a random scheduling scheme, the three burst assembly algorithms produce similar results between them. The VAT algorithm outperforms MBLFAT and MBLFAT outperforms FAT, but with a very small difference in burst loss probability in each CoS. Figure 6-7 shows the comparison between the FAT and MBLFAT for three CoS.

6.2.5 LAUC with AOF

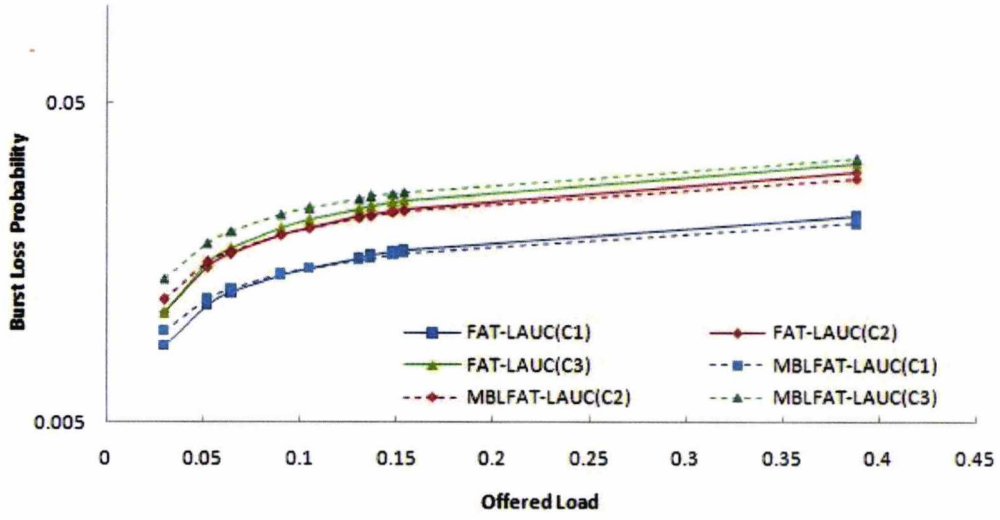


Figure 6- 8 Burst Loss Probability comparison of FAT and MBLFAT for the LAUC scheduling algorithm.

In Figure 6-8 LAUC scheduling algorithm is presented for comparing the FAT and the MBLFAT assembly algorithms. It can be observed that MBLFAT performs very similarly to FAT. In the LAUC algorithm the order in which channels are checked for each edge node remains the same only when the traffic is low, so the more bursts that are created, the higher the dropping probability will be at lower traffic load. Thus as in the FF analysis we can see that MBLFAT still performs worse than FAT. As traffic is increased the probability of using all channels increases rapidly having in effect an improvement in performance on burst loss probability for MBLFAT. Another important result that is observed is the effect of AOF. For lower priority bursts MBLFAT is shown to perform worse than FAT. This effect can be explained by the fact that an extra offset increases the burst loss probability due to the increasing the number of bursts being sent.

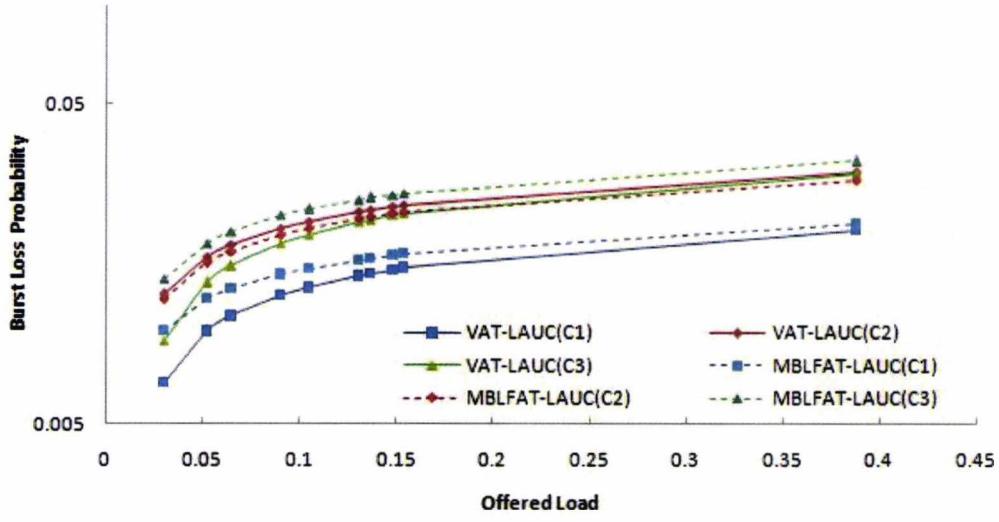


Figure 6- 9 Burst Loss Probability comparison of FAT and MBLFAT for the LAUC scheduling algorithm.

In Figure 6-9 VAT is compared to MBLFAT when using LAUC. VAT outperforms MBLFAT in all CoS. It can also be seen that at low traffic loads VAT performs significantly better. The ratio between the two burst assembly algorithms on burst dropping probability becomes smaller as the load increases, especially for high priority classes.

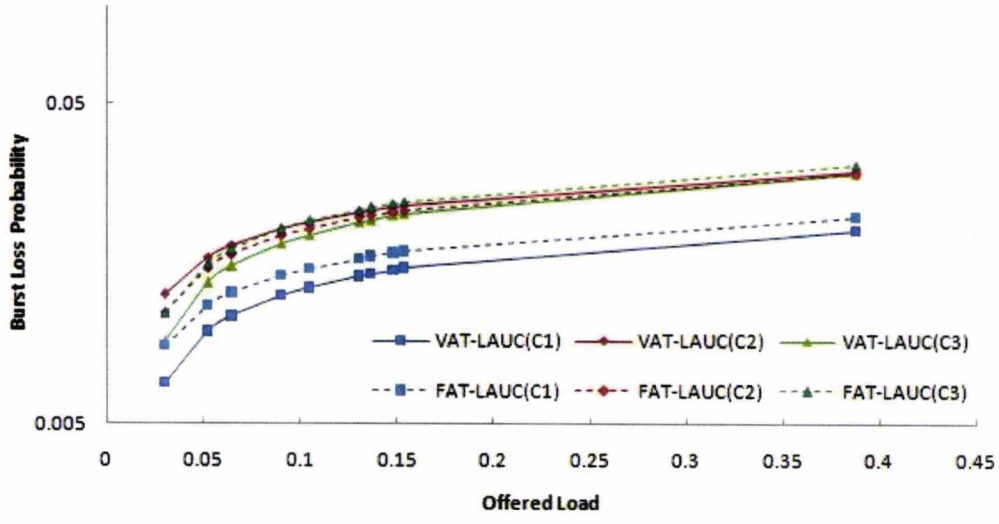


Figure 6- 10 Burst Loss Probability comparison of FAT and VAT for the LAUC scheduling algorithm.

The last result for LAUC can be seen in Figure 6-10, where the FAT and VAT algorithms are compared. VAT outperforms FAT for the two higher priority classes, but at the lowest priority class FAT surpasses VAT in performance for all loads. As the traffic load increases the VAT algorithms starts performing better than FAT in all CoS.

When using LAUC as the scheduling algorithm for an OBS network, the performance in burst loss probability is dependent mostly on the traffic load. The size of the bursts unlike the FF algorithm has more of an impact on the burst loss performance of the network. VAT due to its great adaptability on varying the burst assembly time according to the available bandwidth performs better than the other two algorithms.

6.2.6 WLAUC with AOF

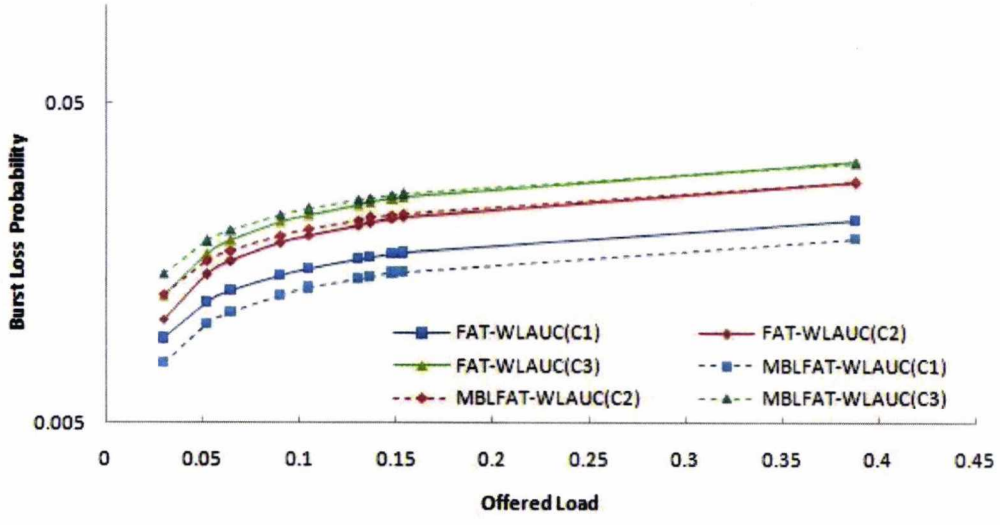


Figure 6- 11 Burst Loss Probability comparison of FAT and MBLFAT for the WCA-FF scheduling algorithm.

The last scheduling algorithm that shows interesting results when comparing for different burst assembly algorithms is the WLAUC. The main ability of WLAUC, as stated in Chapter 5, is to bring a balance at lower traffic loads on the selection of channels. This has as an effect to improve the performance of FAT and MBLFAT on burst loss probability. The main improvement compared to the simple LAUC can be seen on MBLFAT due to the fact that bursts are now transmitted more uniformly into all channels even at lower loads.

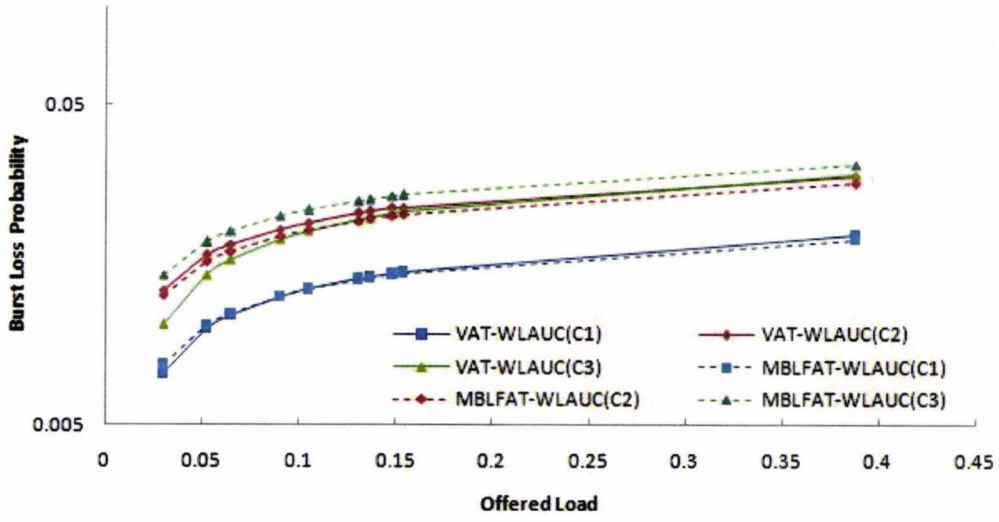


Figure 6- 12 Burst Loss Probability comparison of VAT and MBLFAT for the WLAUC scheduling algorithm.

Figure 6-12 presents results on VAT against MBLFAT when WLAUC is applied. The results presented to us show the improvement of MBLFAT results as was shown in Figure 6-11. VAT on the other hand shows a small withdrawal at low traffic but improves at higher loads. This occurs because using WLAUC at low traffic balances the traffic at all channels thus dropping bursts in all channels increases.

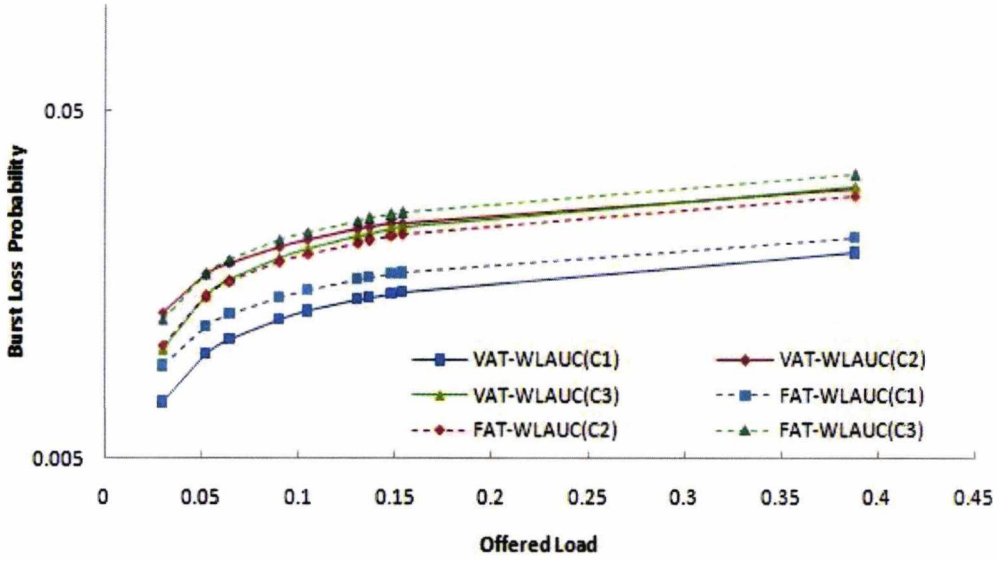


Figure 6- 13 Burst Loss Probability comparison of VAT and MBLFAT for the WLAUC scheduling algorithm.

The last graph presented for WLAUC shows the comparison between VAT and FAT. Due to the weighting on each channel by the use of WLAUC at low traffics for FAT it is observed an improvement is observed but still not substantial enough to surpass VAT for high priority traffic.

In general it can be seen that looking at both scheduling algorithms the improvement of WLAUC is small compared to LAUC, with the exception being at lower traffic loads.

6.2.7 Scheduling Algorithms with Preemptive Reservation

The next series of simulations involved the comparison of scheduling and burst assembly algorithms when preemptive reservation technique for CoS is used. Although extensive simulations were performed similar results were found as when using AOF. Some of these results appear in Figure 6-14 to 6-16. In Figure 6-14 a comparison is presented when using WCA-FF and MBLFAT with preemptive reservation. Figure 6-15 and 6-16 show similar results for FAT and VAT accordingly. In all these figures it can be seen that AOF and preemptive reservation algorithms have little effect on the burst dropping probability for lower priority classes. The main gain appears at higher priority classes where a small improvement can be seen in all of the graphs.

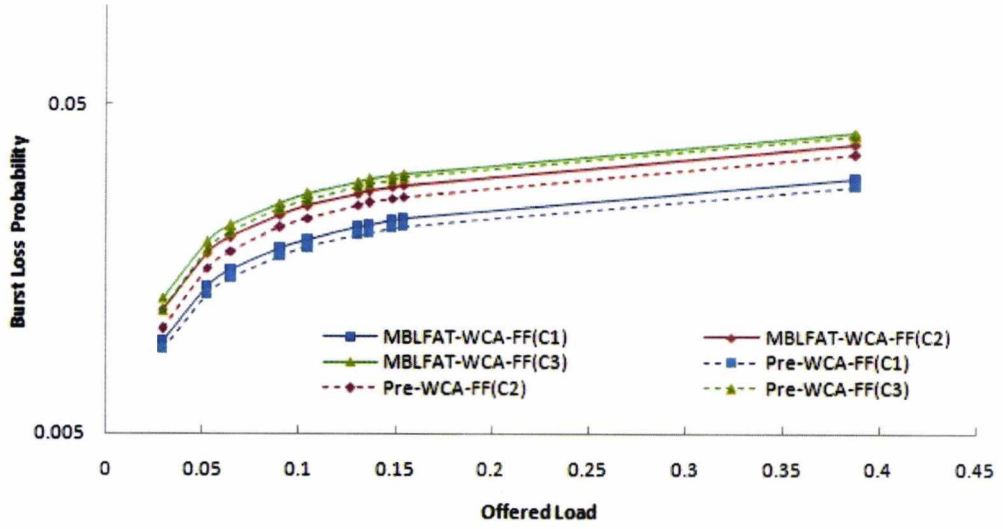


Figure 6- 14 Burst Loss Probability comparison of MBLFAT for the Preemptive reservation using the WCA-FF scheduling algorithm.

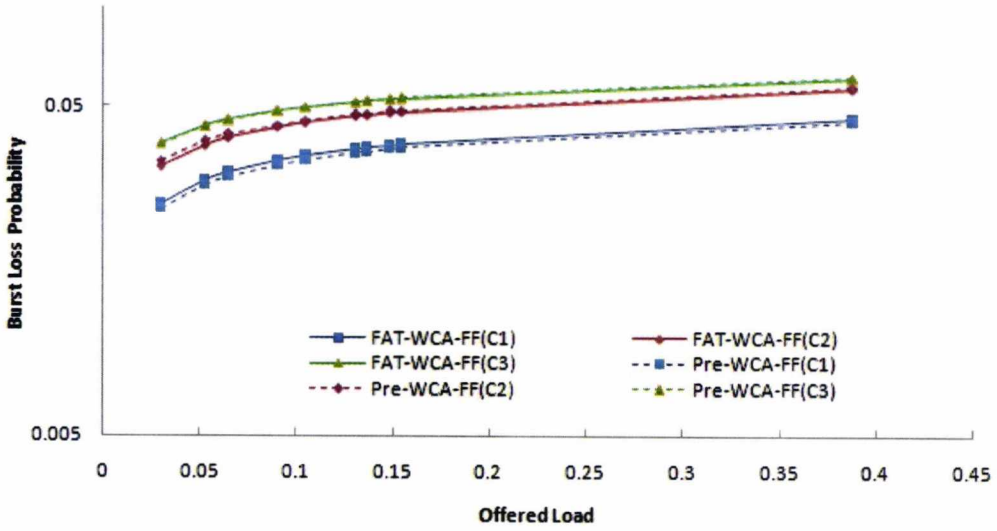


Figure 6- 15 Burst Loss Probability comparison of FAT for the Preemptive reservation using the WCA-FF scheduling algorithm.

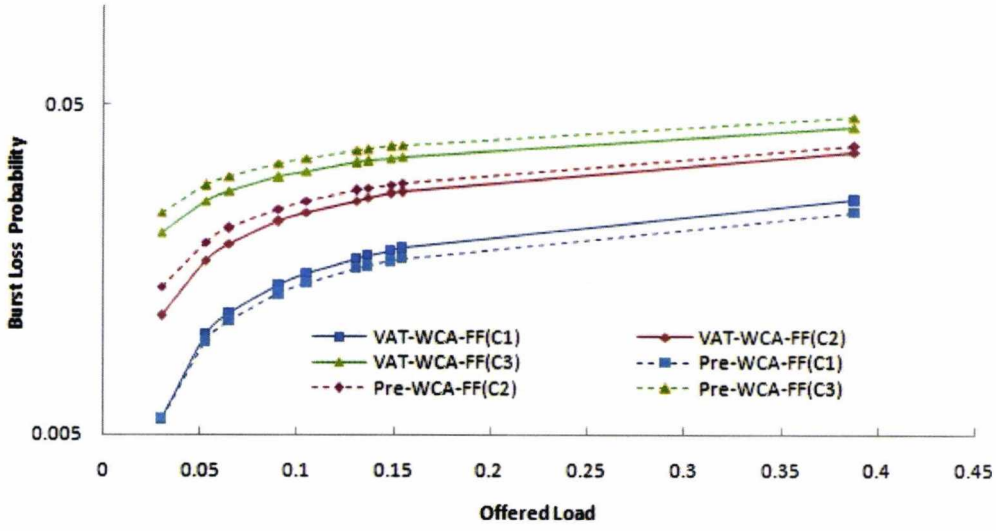


Figure 6- 16 Burst Loss Probability comparison of VAT for the Preemptive reservation using the WCA-FF scheduling algorithm.

6.3 Discussion

As we have seen from section 6.2 each combination has given us something to consider and has introduced a sense of complexity that has not yet been shown until now. In this thesis we are trying to identify some of the complexities and show that great consideration needs to be taken into account when deciding to make an OBS network feasible.

In the next three Figures 6-17 to 6-19 a comparison is made on the scheduling algorithms investigated in this thesis for the highest priority CoS using different assembly algorithms.

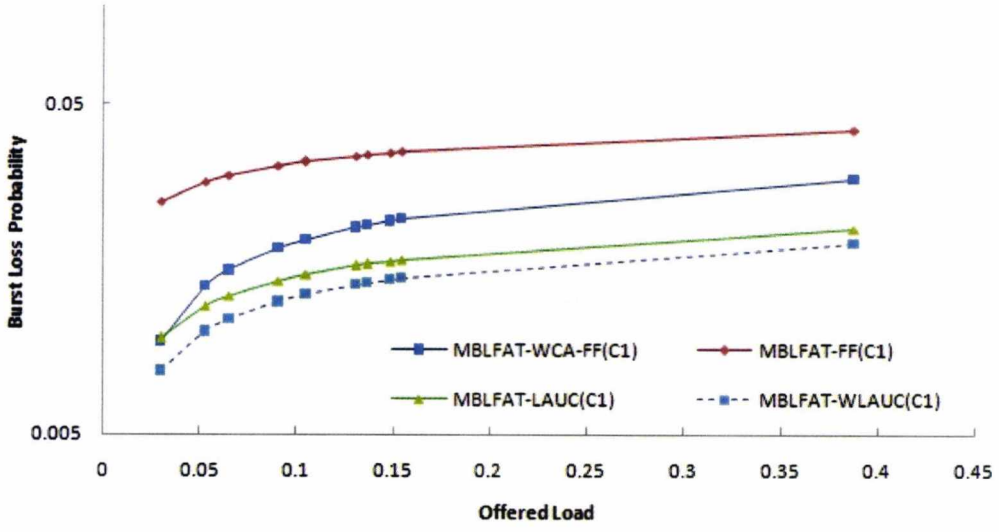


Figure 6- 17 Burst Loss Probability comparison for Class 1 when using MBLFAT for different scheduling algorithms.

In figure 6-17 we see that WLAUC outperforms all other scheduling algorithms when using MBLFAT. The second best is the simple LAUC algorithm with WCA-FF third and last the FF. The results do not of course resemble the results presented in Chapter 5 when burst assembly and CoS algorithms were in use. This shows us that when investigating scheduling algorithms the burst assembly algorithm plays a very important role on the end results on burst loss probability. The next important information that can be deducted from this Chapter of the thesis is that an OBS network reacts differently at low traffic than high traffic. This can be more clearly seen on the next Figure 6-18. The results on Figure 6-18 show the results on burst loss probability on high priority class for VAT, a highly adaptive algorithm.

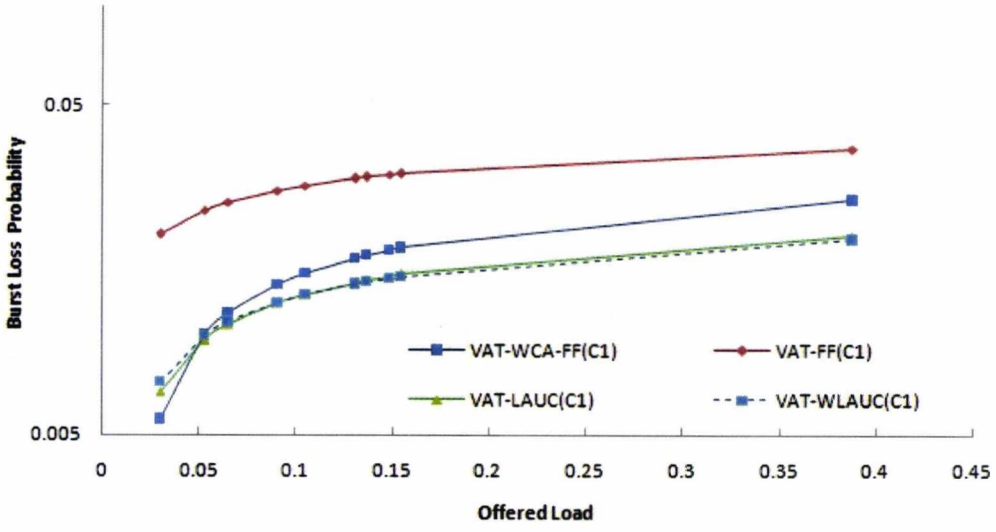


Figure 6- 18 Burst Loss Probability comparison for Class 1 when using VAT for different scheduling algorithms.

In Figure 6-18 it can be seen that WCA-FF has even lower burst loss probability especially at very low traffic. The simple FF algorithm though seems to produce even worse results than when using MBLFAT. The other two scheduling algorithms appear to produce similar results between them and at higher loads the even surpass WCA-FF. It can be said that by applying control on the burst lengths and the number of bursts transmitted in an OBS network the WCA-FF will produce a good performance. It has to be said at this point that FF and WCA-FF are a very simple algorithm that can be applied to an OBS network without too much intelligence hardware and thus with a reduced cost. In Figure 6-19 we can see that WCA-FF performance degrades compared in burst loss probability performance but still provides reasonably good burst loss probability. WLAUC provides the best results from all other algorithms but the results are very similar to the simple LAUC algorithm.

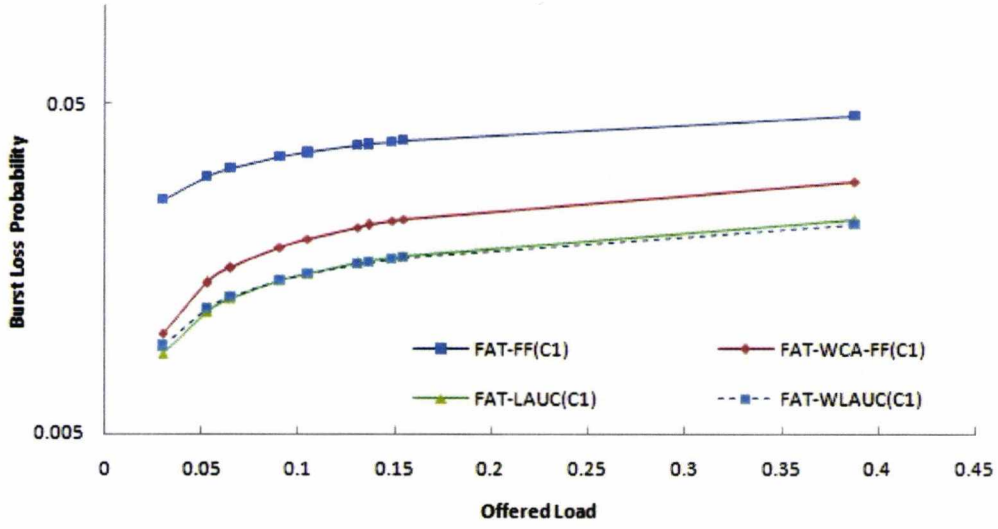


Figure 6- 19 Burst Loss Probability comparison for Class 1 when using FAT for different scheduling algorithms.

6.4 Summary

This Chapter has focused on a comparison between the algorithms of different stages of an OBS network. The results showed the difficulty in finding a single scheduling algorithm or a single burst assembly algorithm that actually surpasses all others. Another important conclusion was that for the CoS algorithms investigated in this thesis there seem to be no significant improvements on burst loss probability, except in high priority classes. Also the effect of the CoS algorithms on the other stage algorithms was not significant except for the AOF as it involves actual manipulation of the burst transmission due to the extra offset added.

For an OBS network to become feasible great deal of consideration has to be taken into account when selecting algorithms because the performance is highly inter-dependant on the combination of burst assembly and scheduling algorithms. The parameters selected in each algorithm have to be considered while all stages of an OBS network design are being considered and not by comparing results of each stage without the consideration of the other stage algorithms. For an OBS network to become feasible many factors have to come under extreme scrutiny and a decision on which algorithms to use for each stage should be dependent on the topology, the present traffic loads and the future expansion of the network.

6.5 References

[1] An Ge and Franco Callegati, "On Optical Burst Switching and Self-Similar Traffic", IEEE Comm. Vol. 4, no. 3, March 2000.

[2] D.D. Clark and J. Hoe, Start-up Dynamics of TCP's Congestion Control and Avoidance Schemes, Technical report, Jun.1995. Presentation to the Internet End-to-End Research Group.

[3] V. Jacobson Congestion Avoidance and Control, SIGCOMM Symposium on Communications Architectures and Protocols, pages 314-329, 1988.

[4] V. Jacobson, Modified TCP congestion avoidance algorithm. Note sent to End-to-End interest mailing list, 1990.

[5] Xiaojun Cao, Jikai Li, Yang Chen, Chunming Qiao "Assembling TCP/IP Packets in Optical Burst Switched Networks", IEEE Global Telecommunications Conference (Globecom). 2002, Taiwan.

CHAPTER 7

CONCLUSION AND FUTURE WORK

OBS networks will connect increasingly higher-speed networks either as part of the long-haul backbone or connect with the huge bandwidth pipes of the existing long-haul backbone. Almost all current metropolitan networks use SONET/SDH technology which was originally intended to operate and handle voice and limited internet traffic. The ever increasing amount of bursty internet data traffic though is now being inefficiently handled by the SONET/SDH networks. More specifically, todays SONET/SDH based networks suffer from limited capacity scalability, poor bandwidth utilization, high provisioning times, and large system complexity. To overcome these problems OPS and OBS have been suggested. The main problem that still exists in these types of networks is the limited commercial technology on optical buffering and switching. In this thesis we investigated a solution for OBS technology without optical buffering and wavelength conversion. Although both these technologies exist they are still in the early stages of development, and are currently limited in speed, scalability and price.

7.1 Thesis Summary

The first step in our research was to create the OBS network simulation model. It was decided to use OPNET Modeler because of the ease of implementation of data network models. The use of OPNET Modeler has the ability to create new models that can be used together with other existing models. The existing models themselves can also be modified to suit any need for the creation of any network model.

7.1.1 Reservation technique

In order to develop an accurate model for an OBS network without the use of optical buffering or wavelength conversion the most plausible reservation technique had to be selected. The analyses of most of the reservation technique were researched and a comparison was made based on the burst loss probability and end to end delay as was

shown in Chapter 2. The JET reservation technique was selected for use for the development of our OBS network model.

7.1.2 Burst Assembly Algorithms

After the reservation technique was selected the next research step was concerned with the aggregation of bursts. Many algorithms appear in the literature but the most investigated were modeled and included in the model for the OBS network simulation. The three burst assembly techniques selected were the FAT, MBLFAT and VAT which were explained in Chapter 3.

7.1.3 CoS Algorithms

Many of the CoS algorithms in the literature are dependent on the reservation technique used. The algorithms that could be easily implemented in the OBS model that was based on the JET reservation technique were the AOF and the Preemptive reservation algorithm.

7.1.4 Scheduling Algorithms

Much of the research carried out was focused on the scheduling algorithms. Many algorithms were investigated, some with low hardware complexity, others with significant hardware complexity. It was decided to focus on the low complexity algorithms as they did not involve optical buffering and wavelength switching. The models that were created were for the FF, LAUC and random scheduling algorithms. No void filling techniques were investigated as thet involved optical buffering and wavelength conversion in the core nodes.

7.1.5 Investigating Scheduling Algorithms

The research presented in the thesis has investigated all scheduling algorithms that were modeled and tried to identify the sources of the increased burst dropping probability in the OBS network. This led to the development of an algorithm (WCA) that works in

conjunction with the algorithms investigated. The next step in the research presented was the improvement offered by the use of the WCA algorithm compared to the existing scheduling algorithms. Results using the WCA algorithm with different types of network topologies were presented and the reasons behind the improved performance in terms of burst dropping probability in an OBS network were explained.

7.1.6 Investigating Different Combinations of OBS Algorithms

The research has concluded with the investigation of burst dropping probability in an OBS network while combining different algorithms for each stage of the network. The stages included were burst assembly, CoS and burst scheduling. The results present a unique view of performance in OBS networks when using different algorithms for each stage. This unique view was made possible due to the unique simulation model designed that included all stages. The results show us the interdependence of the algorithms between different stages.

7.2 Main Original Achievements

7.2.1 Development of an Accurate Complete OBS Model

The first main achievement was developing a unique simulation model that includes all stages of an OBS network. The model was developed in OPNET Modeler, a fast growing network analysis software tool. The model was built in such a way that including new algorithms for burst assembly, CoS and scheduling becomes simple. To include an algorithm a user has to just write and design a function in C++ that can be added to the model function block of the edge node or core node and the algorithm is ready for use. The simulation can analyse any network topology up to 10 core nodes currently, but extensible in principle and include any type of traffic. The model is based on the JET reservation technique and all of the delays that were included in the model were based on those found in real hardware equipment. Another important achievement produced with the design of the model is that a researcher can easily simulate real life network scenarios, such as investigating the network performance by failing any network component or even

investigating video traffic from source to destination by looking at the individual packets sent and received.

7.2.2 Development of Scheduling Algorithms

After investigating the scheduling algorithms into how the burst dropping probability appears in each channel a new algorithm WCA has been developed to act as an addition to the existing algorithms. The main function of WCA is to provide an assignment to channels depending on the position of the edge nodes in the network. Each edge node receives a code that identifies the order at which channels should be selected. A significant reduction in burst dropping probability using the WCA-FF and WCA-LAUC compared to FF and LAUC algorithms has been successfully proven.

7.2.3 The Effect of Interconnection of Different Algorithms in an OBS Network

The last main contribution of the research presented in the thesis, was concluding work on the investigation of the effect on burst dropping probability in OBS networks while combining different algorithms for burst assembly, CoS and scheduling of bursts. We provided the reader with enough results to prove that each algorithm can have a significant effect on the others. Although in the literature many algorithms provide proof that burst dropping probability is reduced compared to other, we have found that this highly depends on the other algorithms that are used. For example MBLFAT should provide a better probability dropping than FAT, but we have proved that this is not the case in all algorithm combinations. Also depending on the scheduling algorithms the burst dropping probability for each class of service is also dependable by the burst assembly algorithms used. We can safely conclude from our results that when investigating an algorithm, in order to have a full picture of the results, we need to examine the effect on the network when used in conjunction with the other types of algorithms involved in OBS.

7.3 Future Work

There is a significant amount of research to be done in OBS before it becomes a feasible option for future high speed optical networks. In the research presented in this thesis, we developed a complicated but robust simulation model that has the ability to analyse a full OBS system. The simulation model incorporates the basic building blocks for an analysis of burst dropping probability for different network loads. The model allows the user to select between different algorithms for burst assembly, CoS and scheduling. As in all areas of technological advancement, extensive research and testing is required before deployment of a system. Using simulations presented in this thesis, further work in increasing the number of algorithms for all areas of OBS networks can lead to a deeper understanding of OBS networks. The core of the model, which includes the reservation technique, can also be replaced to test the CoS, burst assembly and scheduling algorithms by comparing them using other reservation techniques. The model can be developed further by including higher level layers and analysing the performance of these layers when connected to an OBS network.

The analysis in this thesis was mostly based on burst dropping probabilities. Further analysis can be included to investigate real network traffic by importing traffic data from existing networks that might be candidates for replacements by an OBS network in the future. Analysis can also include delays, and new algorithms can be developed to reduce the possible high delays when using certain algorithms.

Another important aspect that can be included in the model is segmentation of bursts and the analyses of burst dropping probability when segmentation is used.

Further work would have to include the study of the financial cost of deploying an OBS network. OPNET Modeler has the capability to include some cost values on all the nodes and links of the model and calculate the financial cost of sending information on the OBS network.

OBS is one of the most active research fields in optical networks. In the last years, research has been focused also in ring based OBS network [1-3]. These types of OBS networks include nodes that are interconnected between by creating a ring based network. These type of nodes though will also have to include solutions for collision detection and

optical buffering. Furthering the existing model to include such types of network will be an even further step to make OBS networks feasible.

7.4 References

[1] Lisong Xu et al., "A simulation Study of Optical Burst Switching Access Protocols for WDM Ring Networks", *Computer Networks*, 41(2), January 2003

[2] H. Wang; G. Wen; Y. Ji, "Implementation of Token-based Optical Burst Switching Ring Network Node and Testbed Using Fixed Transmitter and Tuneable Receiver", *Optical Fiber Communication & Optoelectronic Exposition & Conference*, AOE October 2006, pp: 1 - 3

APPENDIX

DESIGN OF OBS IN OPNET MODELER

In the appendix we are going to cover the design of an optical burst switching network using the nodes described previously. First the source nodes used are to be described up to the process level. Then a description of the routing technique used when using multiple core nodes will be given and finally the section concludes with showing the way in which an OBS network is set up in the network domain. The last part will include results of operation of the OBS network and a comparison to some of the theoretical results found in the literature.

Source Node

The source node can be seen in figure 1. We can see its simplicity as it only contains a "source" process connected to a transmitter and a receiver connected to a "sink" process. In the Figure the process model of the source can also be seen.

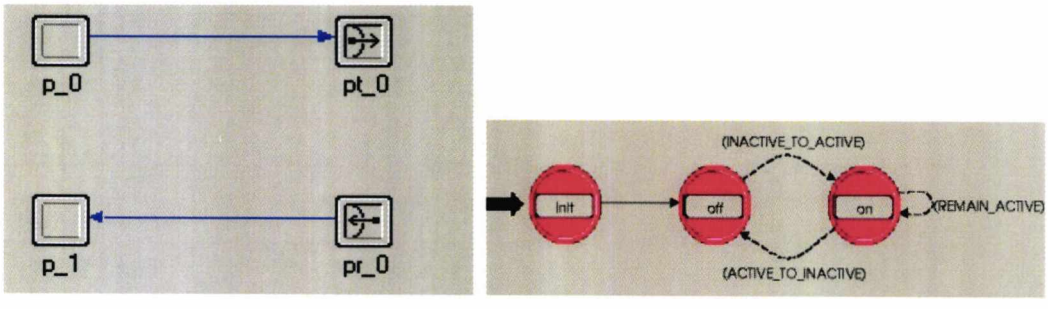


Figure 1. Node Model for a source node and the finite state machine of an OBS Network

The source is a bursty traffic generator which means that it can generate traffic for a certain amount of time and then ceases to generate anything for another period of time. To accomplish the on-off effect, the finite state machine in the process model uses two states to interchange between the on and the off period. Firstly it initializes by getting the inter-arrival time between the packets when at the "on" state and also getting the period of time it has to spend in both the "on" and "off" states.

After initialization, the process moves on to the "off" state and creates a self interrupt according to the time specified in the attributes. When the period in the "off" state ends the self interrupt event occurs and the process transits to the "on" state. At this stage another self interrupt is created for a future time according to the period of time it has to remain in the "on" state stated by the user. While in the "on" period the state will start generating packets according to the inter-arrival time entered by the user and transmitting them through the transmitter in the node editor. Once the time for the self interrupt occurs the process moves to the "off" state where the whole process is repeated until the end of simulation or a time specified by the user in the node attributes.

The "sink" process has only two states, an initialization state for initializing variables and statistics and a "discard" state. At the "discard" state any packet received is destroyed after all valuable information from it has been extracted and stored in the statistical data. Such data will include end to end delay of packets, number of packets destroyed and thus received, and also the size of each packet.

Routing techniques

In order to simulate and implement a more realistic OBS network, multiple core nodes have to be interconnected. The transmitters and receivers in the core nodes can be connected either to core or edge nodes.

When one core node exists it is very simple to calculate the required time difference between the BHC and the burst as there is always a fixed distance between two edge nodes. When more core nodes are included in the network design the distance between edge nodes varies according to the different paths that exist between them. As the distance of the nodes is an essential parameter for the estimation of the offset time for the BHC, an edge node would require to known the distance between each of its destination. This will enable the correct estimation of the offset time. In addition it will enable a choice of an optimized path for the burst when more than one path would lead to the same destination. Finally, making each edge node to be aware of the number of hops (core nodes) between itself and its destination would help changing the routing path if necessary.

The problem of routing was solved by simply creating a text file for each core and edge node, which would declare the distance between all the edge nodes. The creation of each text file is created manually by the user before the beginning of the simulation depending on the architecture of the network being simulated. For the core node the text file contains three columns, the first column states all of the possible destination nodes for the entire network while the second column states the distance of each destination edge node to the specific core node. Finally, the third column declares the next core node to which the BHC and the burst should be transmitted. The core node when running the routing protocol will only read the third column to find the next hop according to the destination in the first column. The second column would only be used if the routing protocol dictates a change in path.

The edge node routing text file on the other hand only needs two columns to accomplish the routing. The first being the destination nodes and the second being the number of hops it will find in its path to the respective destination. By knowing the number of hops it can easily calculate the offset time between the BHC and the burst.

In the case where there is more than one possible path for a burst to reach its destination then the shortest distance is chosen from the edge nodes. In this case for the edge node text file the destination is declared twice with one for each path and the corresponding number of hops. If both paths have the same number of hops then the destination is only declared once. Correspondingly for the core node files the destination is declared twice. The only difference for the core node files is that the destination is mentioned twice even if the distance is the same since the core node to be transmitted to would be different. If this is the case the path is chosen according to the information collected from the BHC or by the intervention of the routing protocol.

Setting up a OBS network

The first task when setting up an OBS network is to place all the edge and core nodes necessary for the network and, then interconnect them with the correct links: the "burst_link" for all the channel connections and the "BHC_link" for all BHC channels.

Each Edge node has four data burst channels and one control channel. It can only be connected to one core node. Table 1 shows the attributes of an edge node and their corresponding use. From the assembly and scheduling attributes only one of the algorithms can be enabled at a time. Also the user must use one of the assembly algorithms else the simulation terminates and displays an error in the simulation window. The scheduling algorithms can be disabled as long as the user enables the LAUC or LAUC-VF scheduling algorithm in the core node. The CoS algorithm can also be disabled or enabled, although the user has to disable or enable the CoS algorithm in the core node in a similar way.

Table 1 Attributes and explanations for an OBS edge node

Attribute	**Explanation**
Assembly	Used to set the burst assembly algorithm.
Scheduling	Used to set the scheduling algorithm.
CoS	Used to set the CoS algorithm.
Factor X (Seconds):	Used to set the difference in offset times for different quality of services.
Address	This is the address of the node. They should be different for each edge/core node. They should start from 1 and incrementing every time by 1.
Num_node	This is the numbers of edge nodes in the whole network. It is used to calculate destinations.
Maximum burst size	This is the maximum burst size for the assembly algorithm MBLFAT only. This has no effect when using the FAT algorithm
Timer	This is the timer for the assembly algorithm. (FAT and MBLFAT)
Hop_Table	This is pointing to a text file that has information on number of hops and routes for each destination.

Typical values for all the attributes can be seen in Figure 2. The lower limit of a data queue is set to 130000 bytes as it is assumed that a burst can have a minimum of two

maximum size IP packets. The timer should be carefully selected according to the arrival time of the source mentioned earlier.

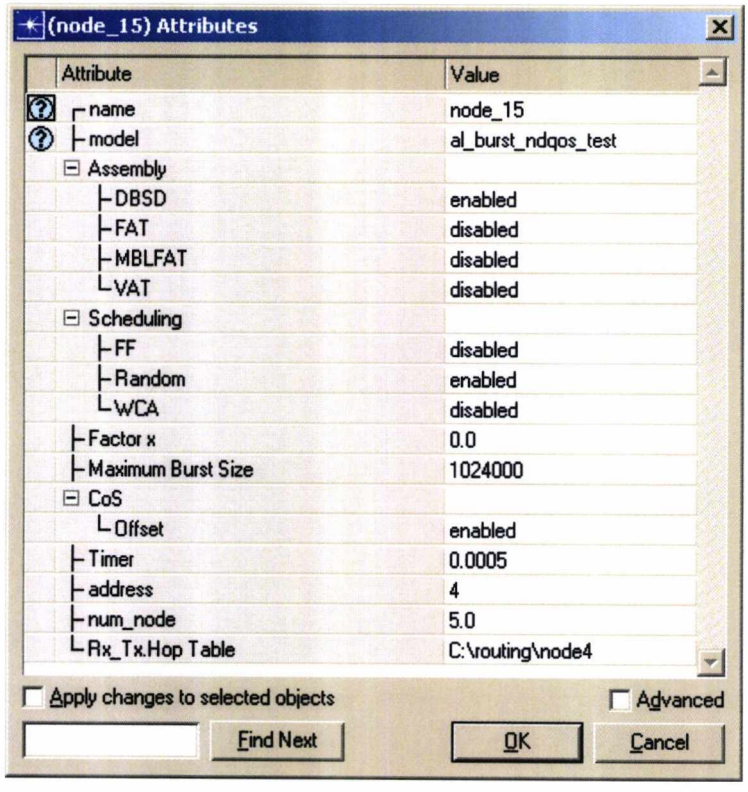
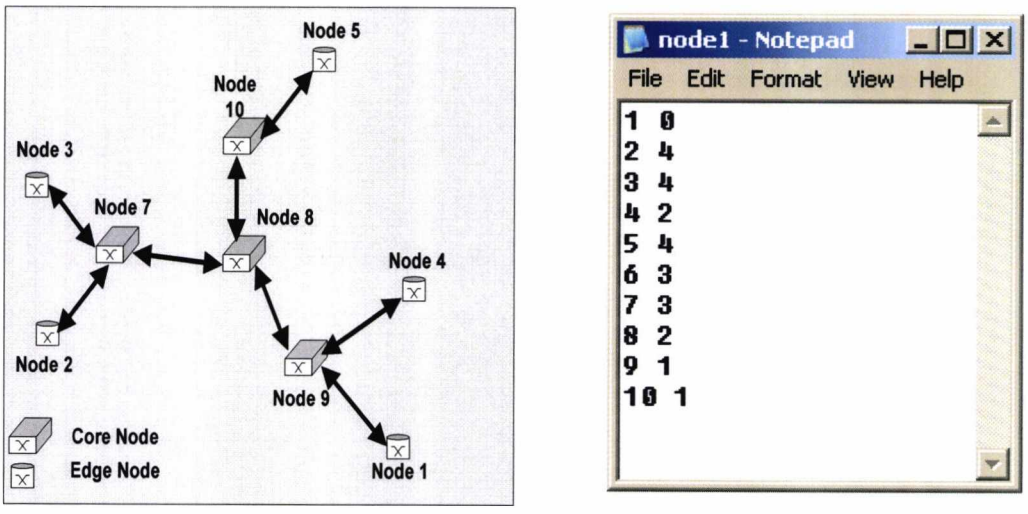


Figure 2 Typical attribute for an edge node in OPNET

The hop table for an edge node with address 1 for the network shown in Figure 3 (a) can be seen in Figure 3(b). The first column has the addresses of all the edge nodes in the network. The second column represents the number of links it has to pass to reach the destination (address) from the source. The purpose is for the network to be able to obtain the number of hops a burst has to make and so calculate the appropriate offset time.



(a) (b)

Figure 3 Network topology (a) and text file for edge node 1(b)

The core node can be connected to a maximum of 10 nodes. But in the model developed a network as a whole can only have 10 nodes (edge/core) in total.

In table 2 the attribute in a core node can be seen together with some explanation. If using the LAUC or the LAUC-VF and/or the preemptive CoS algorithms then wavelength conversion and buffering have to be enabled as all of the algorithms in the core node have to have memory and be able to change the wavelength channel of an incoming burst.

Table 2 Attributes and explanations for an OBS edge node

Attribute	Explanation
Algorithms	Used to set the remaining scheduling and CoS algorithms. Also wavelength conversion is included.
Core ID	This is pointing to a text file that has information on number of hops and routes for each destination.
FDL (seconds)	This is the minimum delay that the first delay line would have. The other FDLs if any would have multiples of this delay.
No_FDL	This is the number of FDL that exist in the core node.

Typical attribute values for a core node can be seen in Figure 4. For the core node with address 9 from the previous network topology the text file has three columns. The first is again the destination node, the second is the number of hops and the third is the core node to which the destination must be connected on demand. The text file can be seen in Figure 5.

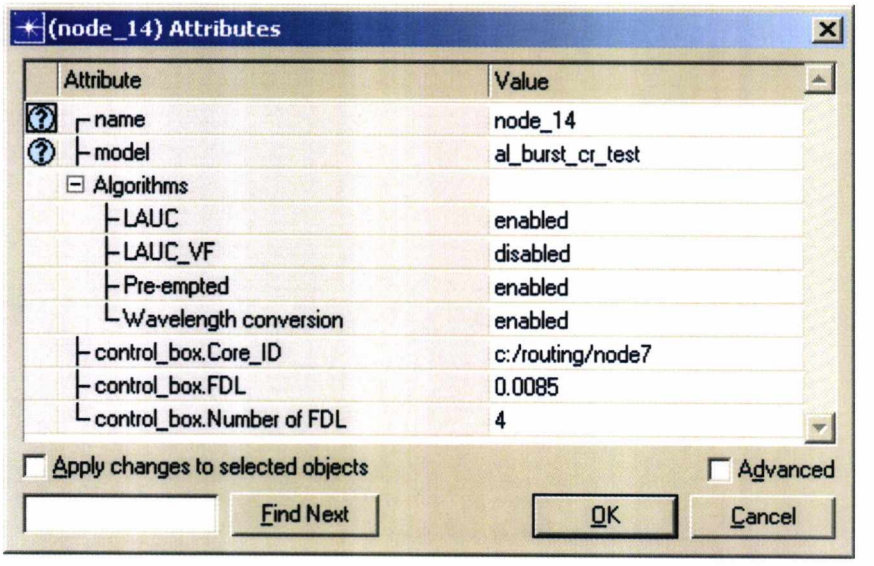


Figure 4 Typical attribute for an edge node in OPNET

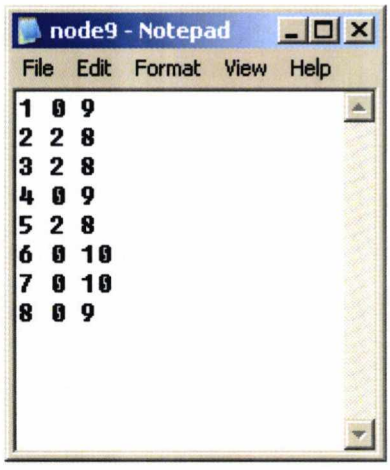


Figure 5 Text file for core node with address 9

The next step for setting up a network is to insert the sources. The attributes in the sources with a small explanation can be seen in table 3.

Table 3 Attributes and explanations for an IP source node in an OBS network

Attribute	**Explanation**
Start Time (Seconds)	This is the simulation time where the Source node will start operating.
ON state time (Seconds)	This is the amount of time at which the source generates packets.
OFF state time (Seconds)	This is the amount of time at which the source node operates but does not send any packets.
Packet Generation argument	These include arguments for: • Inter-arrival time (seconds) • Packet Size (bytes) • Segmentation Size (bytes)
Stop Time (Seconds)	This is the simulation time where the Source node will stop operating.

In the Figure 6 below are some of the typical parameters used.

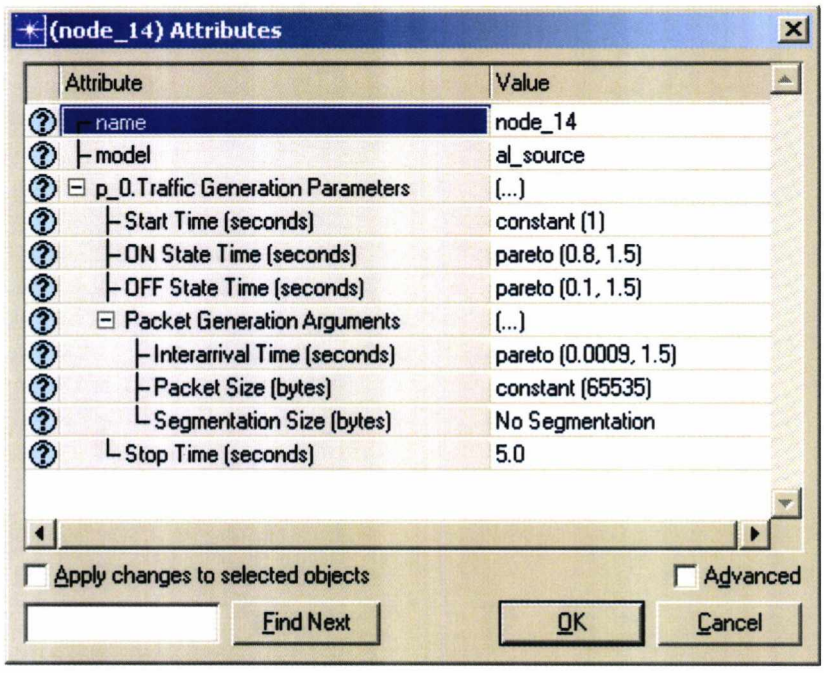


Figure 6 Typical values for an IP source node

The simulation will take up to several seconds. In Figure 7 the simulation time was assumed 7 seconds and the stop time is 5 seconds in order to give the network time to transmit and receive all the reserved bursts and have accurate calculations of the burst dropping. Also we will usually have different start times for different sources as if there was not there would be a peak of reservation at the start time and the dropping probability will be great at that moment which does not correspond to a true network operation. The reason is that OPNET uses a single model for a source node and when calculating the values from the distributions it finds similar values for each of the sources making them having the same simulation times on the ON and OFF states. The distribution used for the ON - OFF states and the inter-arrival time is Pareto as it resembles a bursty traffic. The only parameter changing is the location parameter (first) as the shape parameter would not

have a big effect on our simulations. For the ON state we usually the location parameter varies for each source from 0.5 to 0.9 whereas for the off state a lower value from 0.1 to 0.4. The inter-arrival time remains the same for all source nodes as it will make simpler the calculation of the load at the end of each simulation. We decrease it in order to produce more traffic in the network. It was found that the inter-arrival time should not exceed the value of $7x10^{-6}$, which corresponds to a load of 70% as due memory problems with computer performing the simulation. We are not so much concerned at loads more than 70% as we can firstly easily predict what is going to happen with simple fitting of curves on our results and also in a network they usually calculate the performance for traffic load up 70%. The packet size used is the maximum size of an IP packet. It can be made constant in order to simulate worst scenario or variable to resemble more to a realistic scenario.

All of the above explain the attributes for each node, but the connection of which transmitter goes with which receiver when connecting core nodes together is a little bit more complicated. First we present the attributes that the links must have in Figure 7 according to the network topology in Figure 3 (a). The first two figures show the link connections for node 10 and node 8 and for node 8 to node 7. So care must be taken to connect the number of the transmitter/receiver of a node to the number of the address of the node connected too and vice versa. Although this is not necessary for the correct operation of the system it is highly advisable as if no care is taken the text files for the corresponding nodes would become very complicated with increasing probability of mistakes.

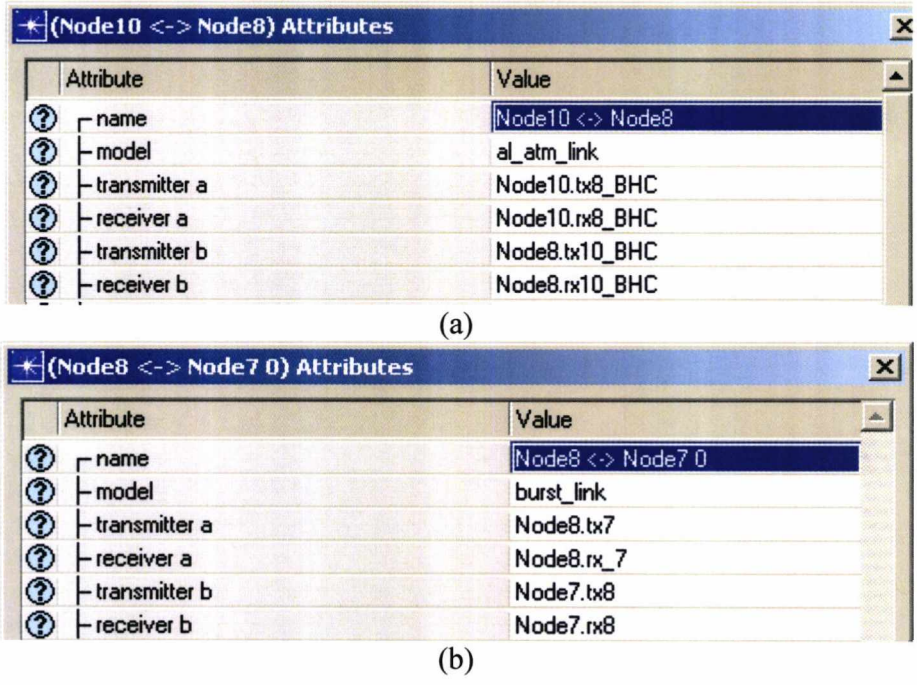


Figure 7 Link attributes for data burst channel

What should be noticed in these attributes are the packet formats which are unformatted for the data burst channels and formatted to "al_at_header" for the control channel. Also the "txdel" model should be NONE for both links. The "propdel" should be "dpt_propdel" and there should be a delay of 0.0004 for both links. Also it is necessary to have the channel count equal to 4 for the data burst link.

For the link between source and edge node everything is completely different as there is no "propdel" and we have "txdel". The data rate is quite high in order to produce enough traffic for four channels, and finally the packets sent are unformatted.