ABSTRACT

XU, LISONG Performance Analysis of Optical Burst Switched Networks. (Under the direction of Dr. Harry G. Perros and Dr. George N. Rouskas.)

In this dissertation, we studied the performance of Optical Burst Switching (OBS). OBS is a promising new solution for the next-generation optical Internet.

In the first part of the dissertation, we studied a novel WDM ring network with OBS. The ring consists of N nodes, and each node owns a home wavelength on which it transmits its bursts. The ring operates under the fixed transmitter tunable receiver (FTTR) scheme. Control information is transmitted on a separate control channel. We proposed five different burst switching access protocols. We also studied the performance of these access protocols in terms of throughput, packet delay, throughput fairness, and delay fairness under different network parameters: average packet arrival rate, maximum burst size, and minimum burst size. Finally, we proposed a new offset calculation method, which can significantly simplify the access protocol design, and reduce the packet delay for all access protocols.

In the second part of the dissertation, we analyzed an edge node of a WDM OBS mesh network using a new burst arrival process, which is more realistic than the well-known Poisson process. The edge node is modeled as a closed non-product-form queueing network, consisting of special nodes with orbiting customers. Despite the rich literature in queueing network analysis, this particular queueing network with orbiting customers has not been analyzed before. We developed algorithms for both the single-class and multi-class queueing networks. The single-class queueing network is solved using Marie's method. In the case of no converters, we obtained a closed-form expression of the conditional throughput of the special node with orbiting customers. The multi-class queueing network is analyzed by decomposition. Specifically, a multiple-class queueing network is decomposed into a set of two-class queueing networks, and each of them is then solved by Neuse and Chandy's Heuristic Aggregation Method. We also developed a much faster approximation algorithm for the analysis of an edge OBS node with a large number of wavelengths. Comparisons against simulation data suggest that our algorithms have a good accuracy.

Performance Analysis of Optical Burst Switched Networks

by

LISONG XU

A dissertation submitted to the Graduate Faculty of
North Carolina State University
in partial satisfaction of the
requirements for the Degree of
Doctor of Philosophy

Department of Computer Science

Raleigh

2002

Approved By:

To my wife Wei,

and my daughter Crystal (Nannan).

Biography

Lisong Xu was born in Shanxi province, P.R. China in 1973. He received his M.S. and B.S. in Computer Science from University of Science and Technology Beijing in 1997 and 1994, respectively. In 1998, he came to the U.S. to pursue a Ph.D. degree in Computer Science at NC State University.

Acknowledgements

I would like to express my most sincere gratitude to Dr. Harry G. Perros and Dr. George N. Rouskas, my advisors, for their guidance, inspiration and support throughout the last four years. Especially, they helped me on paper writing with the detailed word-by-word editing. Without them, this dissertation would not have happened.

I would like to extend my appreciation to Dr. Douglas S. Reeves and Dr. Michael Devetsikiotis for their valuable discussions and comments regarding my research and dissertation.

Finally, I want to express my utmost appreciation to my wife for her love, support and encouragement throughout my graduate studies.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Optical Switching

Wavelength division multiplexing (WDM) appears to be the solution of choice for providing a faster networking infrastructure that can meet the explosive growth of the Internet. Several different technologies have been developed for the transfer of data over WDM, such as wavelength routing (optical circuit switching) [20], optical packet switching [35, 37], and optical burst switching (OBS) [25]. Wavelength-routed optical networks have already been deployed and, currently, they represent the most promising technology for optical networks. However, wavelength-routed optical networks, which employ circuit switching, may not be the most appropriate technology for the different applications that will use the emerging optical Internet. Optical packet switching is an alternative technology that appears to be the optimum choice. However, at this moment the technology is not mature enough to provide a viable solution. Optical burst switching is a switching technique that occupies the middle of the spectrum between the well-known circuit switching and packet switching paradigms, borrowing ideas from both to deliver a completely new functionality.

1.1.1 Optical Circuit Switching

Even though wavelength-routed optical networks have already been deployed, but it may not be the most appropriate for the emerging optical Internet. For example, it takes at least a round-trip delay to establish a lightpath. This leads to a poor wavelength utilization if the connection holding time is very short. The bursty nature of the data traffic also leads to a poor wavelength utilization. Therefore, in order to fully utilize the wavelength, sophisticated traffic grooming mechanism is needed to support statistical multiplexing of data from different users.

1.1.2 Optical Packet Switching

Optical packet switching [35, 37] is an alternative technology that appears to be the optimum choice. However, at this moment the technology is not mature enough to provide a viable solution.

- One major challenge is the requirement for synchronization [37]. Optical packet switches usually work synchronously. For example, packets arriving at different input ports must be aligned before they enter the switch fabric. However, it is difficult and expensive to implement the synchronization component.

- The second serious issue in the design of optical packet switches is the lack of commercially viable optical buffers. The basic idea of packet switching is the “Store and Forward”, which means that packets are first stored in the packet switches, and then forwarded to the next switch. This is necessary due to output port contention. However, currently there are no random access optical buffers. As an alternative method, optical fibers are used to emulate buffers by delaying packets for a fixed time [35].

- The third difficulty in implementing optical packet switches is the amount of time required to configure the optical switch fabric. Consider an optical packet switch which takes 1 ms to set up a connection from an input port to an output port (an optimistic value given the current state of the art). At a data rate of 2.5 Gbps, typical of today's communication systems, it takes about 5 μs to transmit a 1500-byte packet. Therefore, if a switch operates at the packet level, less than 0.5% of the time is used for switching data, while the rest is wasted in setting up the switch fabric.

1.1.3 Optical Burst Switching

The concept of burst switching first emerged in the context of voice communications in early 1980s [2, 3]. More recently, optical burst switching [25, 27] has received considerable attention as an alternative to optical packet switching. In essence, optical burst switching considers the optical layer merely as a buffer-less transparent media for applications [1]. However, there is no universal definition of optical burst switching. Dolzer *et al.* [12] summarized some widely accepted common characteristics as follows. Note that, not all proposed optical burst switching mechanisms have all those features described below.

- Granularity: the transmission unit size of optical burst switching is between the optical circuit switching and optical packet switching
- Separation of Control and Data: control information is transmitted on a separate wavelength (or channel)
- One-Way Reservation: Resources are allocated using one-way reservation. That is, a source node does not need to wait for the acknowledgement back from the destination node, before it starts transmitting of the burst.
- Variable Burst Length: The size of burst is variable
- No Optical Buffering: The intermediate node in the optical network does not require optical buffers. Bursts go through the intermediate node without any delay.

Specifically, the unit of transmission is a *burst*, which may consist of several IP packets, a stream of ATM cells, HDTV frames, or the raw bit streams from remote data sensors. The transmission of each burst is preceded by the transmission of a control packet, which usually takes place on a separate signaling channel. Unlike circuit switching, a source node does not wait for confirmation that a path with available resources has been set up; instead, it starts transmitting the data burst soon after the transmission of the control packet. We will refer to the interval of time between the transmission by the source node of the first bit of the control packet and the transmission of the first bit of the data burst as the *offset*. The control packet carries information about the burst, including the offset value, the length of the burst, its priority, etc. The purpose of the control packet is to inform intermediate nodes of the upcoming data burst, so that they can make a routing decision and configure their fabric to switch the burst to the appropriate output port. However, in case of congestion or output port conflicts, an intermediate node may drop bursts. Thus, as in connectionless packet switching, there is no guarantee of delivery. Also, consecutive bursts between a given source-destination pair may be routed independently of each other.

Optical burst switching is quite similar to the *Fast Reservation Protocol* in ATM or *ATM Block Transfer with Immediate Transmissions* (ATM-IT) [24]. However, in the former, all packets or cells in a burst are processed together as a whole by the switch, while in the later, each packet or cell is stored and forwarded individually by the switch.

1.2 Previous Work

1.2.1 Burst Offset

The burst offset is the interval of time between the transmission by the source node of the first bit of the control packet and the transmission of the first bit of the data burst. Based on the length of the burst offset, optical burst switching can be classified into the following three classes.

- No Reservation: The burst is sent immediately after the control packet. That is, the offset is only the transmission time of the control packet. This scheme is practical only when the switch configuration time and the switch processing time of a control packet are very short. This class is very close to the optical packet switching. The optical burst switching scheme called *Tell And Go (TAG)* [32] belongs to this class.

- One-Way Reservation: A burst is sent shortly after the control packet, and the source node does not wait for the acknowledgement back from the destination node. Therefore, the size of the offset is between transmission time of the control packet and the round-trip delay of the control packet. Different optical burst switching mechanisms may choose different offset values in this range.

  In *just-enough-time* (JET) proposed by Qiao and Yoo [25] , the offset is selected in a manner that takes into account the processing delays of the control packet at the intermediate switches. Thus when the burst arrives at the intermediate switches, those switches are already configured. At the destination node, the burst arrives just after it has been configured. Yoo *et al.* [38] also described a priority scheme for JET to support multi-class traffic. Specifically, a burst with a higher class is assigned with an additional offset. They also analyzed the performance of this priority scheme assuming no class interference. Dolzer and Gauger [11] proposed an iteration algorithm to analyze approximately this priority scheme considering the class interference, and they found that the burst interarrival time distribution has only a small impact on the burst loss probability, while the burst length distribution and the ratio of the mean burst length of the classes have a great impact on the burst loss probability.

  Verma *et al.* [28] and Chaskar *et al.* [9] presented a traffic shaping scheme which randomizes the offset to reduce the burst loss probability.

- Two-Way Reservation: The offset is the time required to receive an acknowledgement from the destination. This class is very close to optical circuit switching in that it incurs a round-trip delay to set up the transmission, and since the control packet reserves resources, delivery of the burst is guaranteed. The optical burst switching scheme called *Tell And Wait (TAW)* [32] belongs to this class. The major drawback of this class is the long offset time, which causes the long data delay.

Düser and Bayvel [13] proposed a wavelength-routed optical burst switching (WR-OBS), which uses two-way reservation. Different from other two-way reservations, which send out the control packet after aggregating the burst, WR-OBS sends out the control packet during the aggregation process. Specifically, at some point during the aggregation process, the source node sends a control packet to reserve the resources along the path to the destination node. The source node continues aggregating packets into the burst, until it receives the acknowledgement. The burst is then sent to the destination node. Therefore, the average data delay is reduced. However, the source node must estimate the size of the final burst by monitoring the buffer filling statistics, when it sends out the control packet.

1.2.2 Wavelength Reservation Schemes

This subsection describes different wavelength reservation schemes for optical burst switching based on one-way reservation. Depending on when the reservation starts and ends, Baldine *et al.* [4] described the following four types of wavelength reservation schemes as shown in Figure 1.1.

- Scheme 1: Explicit setup and explicit release. In this scheme, the `setup` message (control packet) contains the offset of the burst, but not the duration of the burst. The reservation starts immediately after the switch receives the `setup` message, and ends after the `release` message arrives. Therefore, only a single on/off bit is required to record the status of a single wavelength. "On" means that the wavelength is busy, and "off" means that the wavelength is free. The on/off bit is triggered by the `setup` and `release` messages.

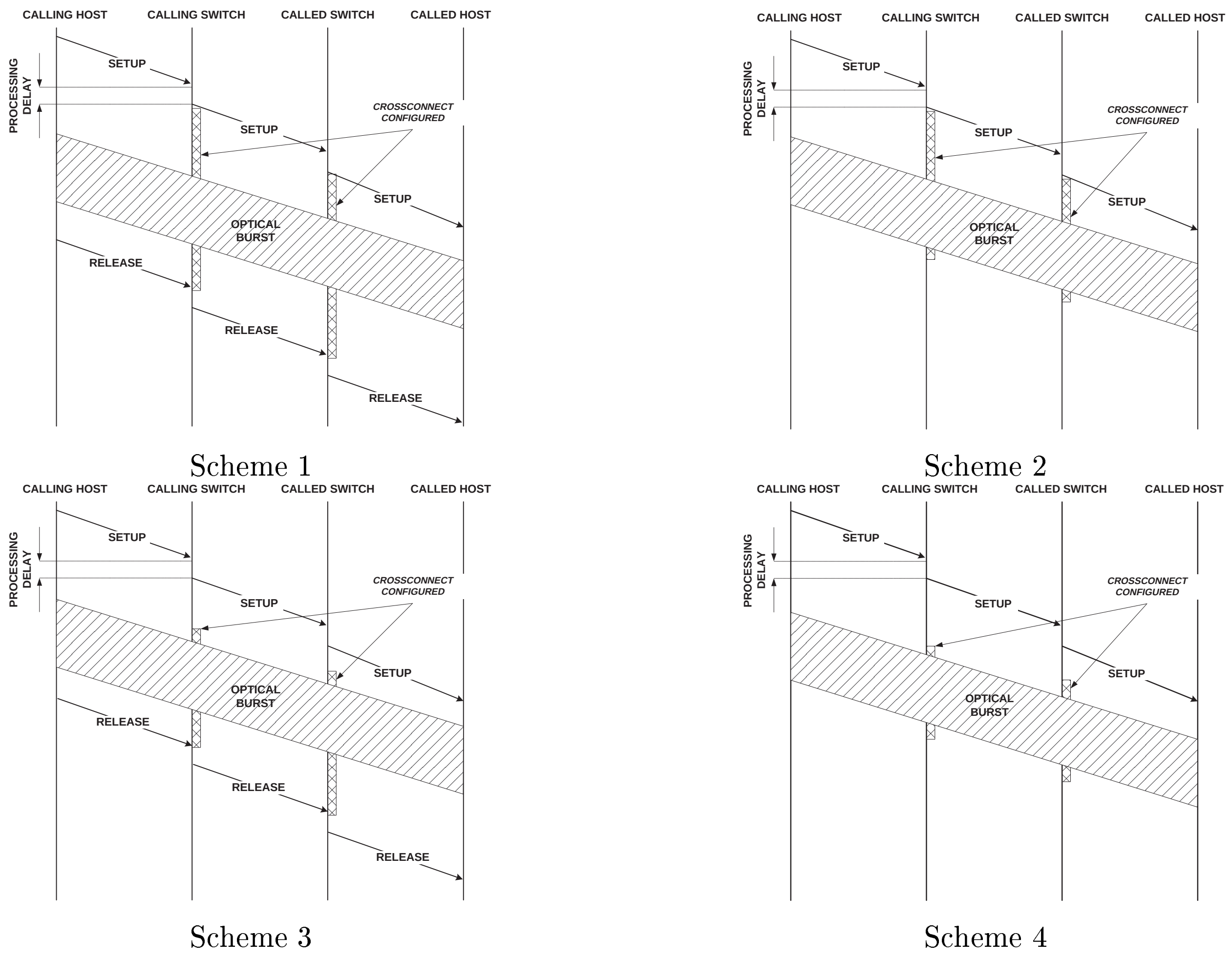

Figure 1.1: Four types of wavelength reservation schemes

- Scheme 2: Explicit setup and estimated release. In this scheme, the `setup` message contains both the offset and the duration of the burst. Each wavelength is associated with a deadline, which indicates when the wavelength will become free. The reservation starts both after the switch receives the `setup` message, and after the deadline. The reservation ends at the end of the burst, which is calculated using the duration of the burst.

- Scheme 3: Estimated setup and explicit release: In this scheme, the `setup` message contains the offset of the burst, but not the duration of the burst. The reservation starts at the beginning of the burst, which is calculated using the offset of the burst, and ends after the `release` message arrives.

- Scheme 4: Estimated setup and estimated release. In this scheme, the `setup` message contains both the offset and the duration of the burst. The reservation starts at the beginning of the burst, which is calculated using the offset of the burst, and ends at the end of the burst, which is calculated using the duration of the burst. Each wavelength is associated with a vector of time period, which indicates the period the wavelength is busy.

Just-in-time (JIT) proposed by Wei and McFarland [30] belongs to scheme 1. The Horizon scheme proposed by Turner [27] for a scalable burst switch architecture belongs to scheme 2. *just-enough-time* (JET) proposed by Qiao and Yoo [25] belongs to scheme 4. Gauger *et al.* [15] introduced a new reservation scheme based on JET for optical switches with optical buffers. Xiong *et al.* [33] and Callegati *et al.* [8] studied the LAUC-VF scheme (Latest Available Unused Channel with Void Filling), which belongs to scheme 4. Yang *et al.* [36] proposed a new wavelength reservation algorithm based on LAUC-VF to support DiffServ.

Dolzer *et al.* [12] compared the burst loss probabilities of scheme 1, 2, and 4. They found that with variable offset time, scheme 4 achieves the lowest burst loss probability, followed by scheme 2, and scheme 1 achieves the highest burst loss probability. If the offset time is constant, then scheme 4 and 2 achieve the same burst loss probability. However, the lower burst loss probability comes at a price of higher complexity at the scheduler in the switch, as well as significant amounts of memory on the signaling board [19].

The `JumpStart` project [1] is an ARDA-supported research project between NCSU and MCNC, addressing the design, specification, performance evaluation, and hardware implementation of a signaling protocol for OBS networks. The signaling protocol follows the just-in-time (JIT) approach, and part of this signaling scheme has been reported by Baldine *et al.* [4]. The `JumpStart` project [1] is designed to support both scheme 1 and scheme 2, both unicast and multicast, both short bursts and lightpaths, both persistent path setup and on-the-fly path setup, and both best-effort and quality-of-service.

1.2.3 Contention Resolution

Contention resolution is necessary in order to handle the case where more than one burst are destined to go out of the same output port at the same time. In an optical burst switch, techniques designed to address this problem include the following four methods.

- Exploiting the wavelength domain: In WDM, several wavelengths run on a fiber link that connects two optical switches. This can be exploited to minimize contention as follows. Let us assume that two bursts are destined to go out of the same output port at the same time. Then, they can be still transmitted out but on two different wavelengths. This method may have some potential in minimizing contention, particularly since the number of wavelengths that can be coupled together onto a single fiber continues to increase. For instance, it is expected that in a year there will be as many as 200 wavelengths per fiber.

- Optical buffering: Optical buffering currently can only be implemented using optical delay lines (ODL). An ODL can delay a burst for a specified amount of time, which is related to the length of the delay line. Delay lines may be acceptable in prototype switches, but they are not commercially viable. The alternative, of course, is to convert the optical packet to the electrical domain and store it electronically. This is not an acceptable solution, since electronic memories cannot keep up with the speeds of optical networks.

- Deflection routing: Deflection routing is ideally suited to switches that have little buffer space or no buffer space. When there is a conflict between two bursts, one will be routed to the correct output port, and the other will be routed to any other available output port. In this way, no or little buffer is needed. However, the deflected burst may end up following a longer path to its destination. As a result, the end-to-end delay for a burst may be unacceptably high. Also, bursts will have to be re-ordered at the destination since they are likely to arrive in an out-of-sequence manner.

- Burst segmentation: In burst segmentation, when contention happens, a node does not drop the entire burst, however, it breaks the burst into multiple segments, and only the overlapping segments are dropped. In this way, even though the burst loss probability does not change, but the loss probability of the data contained in the burst is reduced.

Yoo *et al.* [38] studied the effect of the optical delay line on the burst loss probability, and they found that optical delay lines can reduce the burst loss probability, and the length of the optical delay line required to achieve a given burst loss probability can be significantly reduced as the number of wavelength increases. Kim *et al.* [16] presented a deflection routing algorithm. Besides the normal offset time, each burst is assigned a "routing offset time" to accommodate the extra burst delay of the deflection routing. They found that by introducing a 10% "routing offset time", the burst loss probability could be significantly reduced. Detti *et al.* [10] proposed the optical composite burst switching (OCBS). It uses both wavelength dimension and burst segmentation to solve contention. They found that burst segmentation can reduce the burst loss probability, and the larger the number of packets in a burst, the better performance the OCBS can obtain. Vokkarane *et al.* [29] studied the contention resolution using both burst segmentation and deflection routing.

1.2.4 Analytical Model

All the performance studies of OBS networks proposed in the literature are based on either simulation or simple analytical models. In [9, 12, 11, 28], an output port of an OBS node is analyzed assuming Poisson arrivals and no buffering. Under these assumptions, an output port can be modeled by a finite number of servers, each representing a wavelength, with no queue. Then, the probability that a burst destined to this output port is lost can be obtained from the Erlang B formula. In [27, 38], an output port is analyzed assuming Poisson arrivals and buffering. It is then modeled by an $M/M/m/K$ queue, where m is the number of wavelengths and $K - m$ is the capacity of the buffer. Wei and McFarland [31] considered multiple classes of bursts, each of which is a Poisson arrival process.

However, It is well-known that the Poisson process is not a good model for wide area traffic [22], and it is unlikely that the burst arrival processes in future optical networks will be accurately characterized by the Poisson model. Another problem is that in the Poisson process, an arrival occurs instantaneous, and the service time required by the arrival is independent of the inter-arrival time of two consecutive bursts. However, in optical burst switching, the service time of a burst is the duration of the burst, which is *not* independent on the inter-arrival time of the bursts. For example, the inter-arrival time of between a very long burst and the following burst must be very long, too. Therefore, more sophisticated models are required in order to advance our understanding of the performance and the potential of OBS networks.

In [10], an OBS node is analyzed assuming the On/Off traffic, which is more realistic than the Poisson process. However, the arriving burst is assigned a destination output port following the uniform distribution. This is not a practical assumption, since in most cases, a particular output port has more traffic than other output ports. It also assumes that all input wavelengths have the same On/Off process. That is not a practical assumption, either. Finally, the problem with On/Off process is that it only models a single class of bursts.

1.3 Thesis Organization

To the best of our knowledge, all the performance studies of OBS networks in the literature are for optical networks with a mesh topology, and none of them study the optical networks with a ring topology. However, ring networks represent a significant investment on the part of carriers, and are currently being upgraded to support WDM. Therefore, performance studies of OBS networks with a ring topology are necessary. Chapter 2 considers an OBS network with a ring topology. Several access protocols are proposed to solve receiver collisions, and their performance in terms of throughput, delay, and fairness is analyzed by simulation. A new offset calculation is also proposed to simplify the protocol design.

In chapter 3, we propose a burst arrival process described by a 3-state Markov process, which is more realistic than Poisson process. It permits short and long bursts to be modeled, and the parameters of the model can be selected to capture a wide range of behaviors in the arrival stream.

Chapter 4 considers an edge node of an OBS mesh network with traffic modeled by the burst arrival process described in chapter 3. The arriving burst is assigned a destination output port following the arbitrary distribution. The OBS mesh network consists of OBS nodes interconnected by bi-directional fiber links. An OBS node consists of a non-blocking space-division switch fabric with no optical buffers. The message between a user and the edge node follows the signaling protocols defined in the `JumpStart` project [1]. The edge node is modeled as a closed non-product-form queueing network consisting of special nodes with orbiting customers. We developed algorithms for the analysis of both the single-class and multi-class queueing networks. The single-class queueing network is solved using Marie's method with a novel expression for the conditional throughput of the special nodes with orbiting customers. The multi-class queueing network is analyzed by decomposition. Comparisons against simulation data suggest that our algorithms have a reasonable fast speed and a good accuracy.

In chapter 5, we studied the OBS edge node with a large number of wavelengths. Based on our observations when the number of wavelength increases, we proposed an approximation algorithm for the analysis of an edge OBS node with a large number of wavelengths.

Finally, a summary of research contributions and suggested future work are given in chapter 6.

Chapter 2

Access Protocols for an OBS Ring Network

2.1 Introduction

Optical burst switching has been studied in the context of wide area networks with a mesh topology [33, 25, 27, 28]. In this chapter, we study OBS access protocols for WDM ring networks. Our focus on ring topologies is motivated by the wide deployment of SONET/SDH rings. These networks represent a significant investment on the part of carriers, and are currently being upgraded to support WDM. To the best of our knowledge, this is the first study of burst switching protocols specifically for ring networks. Our vision of the OBS ring is that it will be used to transport different types of traffic, such as IP, ATM, and Frame Relay traffic, and also HDTV and sensor traffic that may not be transported over IP, ATM or Frame Relay. The objective of this chapter is to analyze the performance and fairness of five different OBS access protocols. How these protocols can be used to provide different classes of services to different applications is beyond the scope of this chapter.

This chapter is organized as follows. Section 2.2 describes the optical burst switching used in the ring network, and proposed a new offset calculation method that simplifies the protocol design. Section 2.3 describes the ring network we consider and the basic operation of burst switching in such an environment. Section 2.4 provides a detailed description of the various burst switching access protocols studied in this chapter. Section 2.5 presents the simulation results on the performance of these burst switching access protocols, and finally Section 2.6 provides some concluding remarks. The material in this chapter was presented in Networking 2002 [34].

2.2 Optical Burst Switching

There are several variants of burst switching, mainly differing on the length of the offset. In the burst switching scheme called *Tell And Go (TAG)* [32], the burst is transmitted immediately after the burst header packet. That is, the offset is only the transmission time of the burst header packet. This scheme is practical only when the switch configuration time and the switch processing time of a burst header packet are very short. At the other extreme, the *Tell and Wait (TAW)* [32] scheme requires the offset to be at least equal to the time required to receive an acknowledgement from the destination. TAW is equivalent to circuit switching in that it incurs a round-trip delay to set up the transmission, and since the burst header packet reserves resources, delivery of the burst is guaranteed. Another advantage of TAW is that it eliminates receiver collisions, since a node returns an acknowledgment only for bursts it is prepared to accept.

An intermediate burst switching scheme, known as *Just Enough Time (JET)* [25], selects the offset in a manner that takes into account the processing delays of the burst header packet at the intermediate switches. Let $T_i^{(p)}$ denote the processing delay of a burst header packet at an intermediate switch, $T_d^{(p)}$ denote the processing delay of a burst header packet at the destination switch, and $T_d^{(s)}$ denote the time to setup (configure) the destination switch. Then, the offset value for JET is :

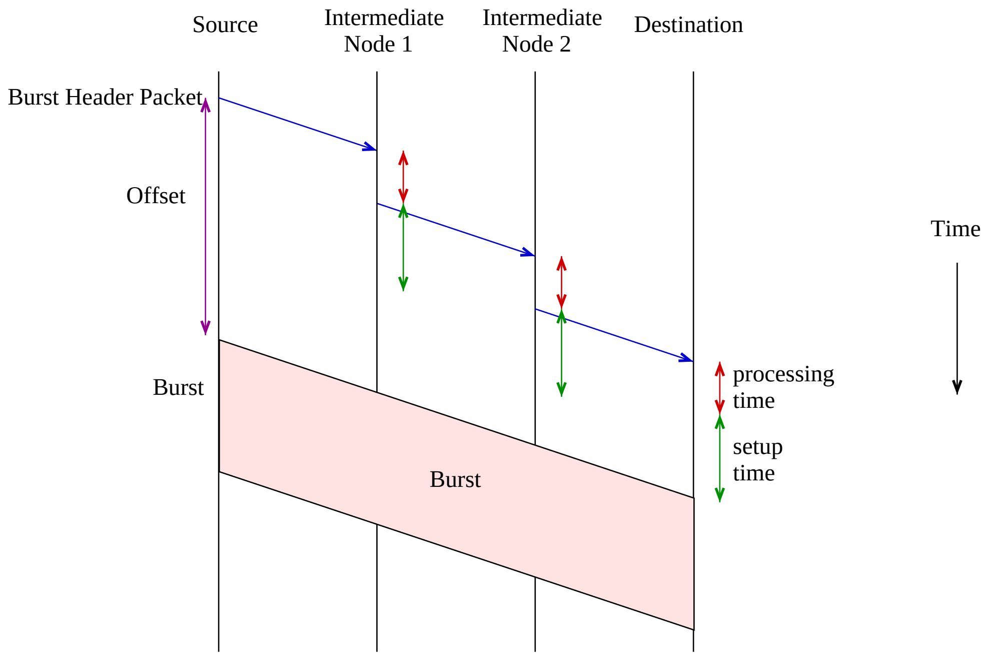

Figure 2.1: Offset calculation in the JET OBS protocol

$$\text{offset}_{\text{JET}} = \left(\sum_i T_i^{(p)}\right) + T_d^{(p)} + T_d^{(s)} \tag{2.1}$$

The offset calculation for the JET protocol is illustrated in Figure 2.1 for a path that includes two intermediate switching nodes between the source and destination of the burst. As can be seen, the offset needs to be long enough to account for the processing time of the burst header packet at the two intermediate nodes and the destination plus the switch setup time at the destination. If the offset time is less than that, then there is a possibility that the burst may arrive at a node before the node is ready to switch the burst.

One issue that arises in computing the offset under JET is determining the number of intermediate switching nodes (hops) between the source and destination. In OBS networks, information about the number of hops in a path may not, in general, be readily available; even when such information is somehow known, because of the effects of routing changes, it is not guaranteed to be valid when used. Thus, it is desirable to use an offset value that does not depend on the path used and does not require the exchange of information among network nodes.

As we can see from expression (2.1), the part of the offset value that depends on the path between the source and destination is the sum of the processing times at intermediate nodes. Given the recent advances in hardware implementation of communication protocols, it is reasonable to assume that the processing time $T_i^{(p)}$ in (2.1) will be very short for most common functions of the signaling protocol (i.e., no exception conditions). In this case, fiber delay lines of reasonable length may be used at intermediate nodes to delay each incoming burst by an amount of time equal to $T_i^{(p)}$. Given such fiber delays, the first term in the right hand side of (2.1) can be omitted when computing the offset. We call this new scheme the *Only Destination Delay (ODD)* protocol, and its offset is given by:

$$\text{offset}_{\text{ODD}} = T_d^{(p)} + T_d^{(s)} \tag{2.2}$$

Furthermore, instead of using destination-specific values for the processing and switching delays in (2.2), one may use a constant offset value by taking the maximum of these values over all destinations. A constant offset that does not depend on the path (number of hops) to the destination significantly simplifies the design and implementation of signaling protocols and optical switches for burst switching networks.

2.3 The Network Under Study

2.3.1 Ring and Node Architecture

We consider N OBS nodes organized in a unidirectional ring, as shown in Figure 2.2. The ring can be a metropolitan area network (MAN) serving as the backbone that interconnects a number of access networks, and transporting multiple types of traffic from users, such as IP traffic, ATM traffic, Frame Relay traffic, HDTV traffic,

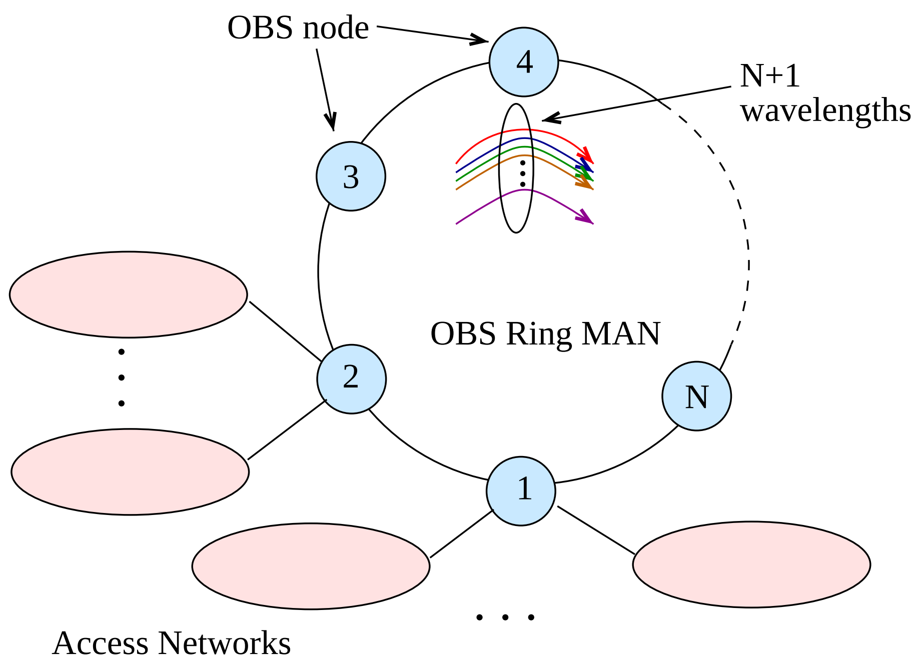

Figure 2.2: OBS ring MAN

and sensor traffic. Each fiber link between two consecutive OBS nodes in the ring can support $N+1$ wavelengths. Of these, N wavelengths are used to transmit bursts, and the $(N+1)$-th wavelength is used as the control channel.

Each OBS node is attached to one or more access networks. In the direction from the access networks to the ring, the OBS node acts as a concentrator. It collects and buffers electronically data, transmitted by users over the access networks, which need to be transported over the ring. Buffered data are subsequently grouped together and transmitted in a burst to the destination OBS node. A burst can be of any size between a minimum and maximum value. Bursts travel as optical signals along the ring, without undergoing any electro-optic conversion at intermediate nodes. In the other direction from the ring to the access networks, an OBS node terminates optical bursts destined to itself, electronically processes the data contained therein, and delivers them to users in its attached access networks.

The architecture of an OBS node is shown in Figure 2.3. Each node is equipped with one optical add-drop multiplexer (OADM), and two pairs of optical transceivers.

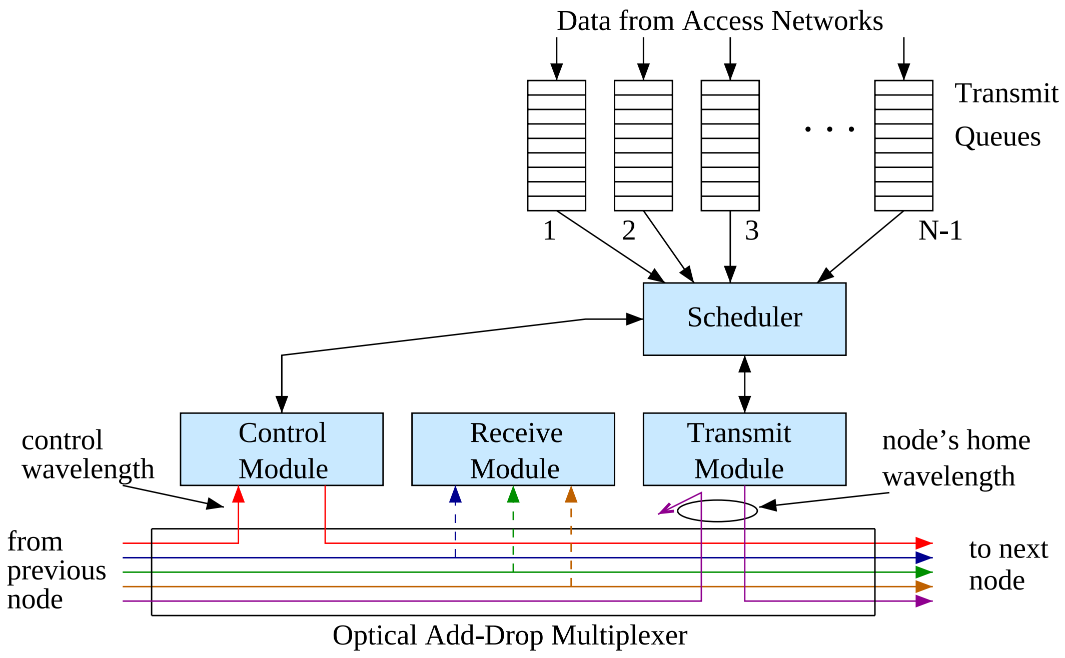

Figure 2.3: OBS node architecture (Delay lines are not shown)

The first pair consists of a receiver and transmitter fixed tuned to the control wavelength, and are part of the control module in Figure 2.3. The control wavelength is dropped by the OADM at each node, and added back after the control module has read the control information and (possibly) has inserted new information (the following subsection provides more details on the operation of the control wavelength).

The second pair of transceivers consists of a transmitter that is fixed tuned to the node's *home wavelength*, and an agile receiver (or a receiver array) that can receive from all N wavelengths that transmit bursts. Each OBS node has a dedicated home wavelength on which it transmits its bursts. The OADM at each node removes the optical signal from the node's home wavelength by dropping the corresponding wavelength, as Figure 2.3 illustrates. The OADM also drops the optical signal on other burst wavelengths, whenever they contain bursts for this node. In the case where multiple bursts arrive, each on a different wavelength, at an OBS node, the receive module in Figure 2.3 employs a collision resolution strategy to determine which burst will be accepted.

To support ODD, an extra fiber delay line (not shown in Figure 2.3) is added into the node to delay outgoing bursts on all wavelengths except the control wavelength and the node's home wavelength.

Data waiting for transmission is organized into (logical) transmit queues according to their destination. The data buffer at each OBS node is shared by $N - 1$ transmit queues, each corresponding to one of the $N - 1$ destination nodes. The order in which transmit queues are served is determined by the scheduler module in Figure 2.3. The transmit queues are served in a Round-Robin manner.

2.3.2 Control Wavelength Operation

The control wavelength is used for the transmission of control slots. In a ring with N nodes, N control slots, one for each node, are grouped together in a *control frame* which continuously circulates around the ring. Depending on the length of the circumference of the ring, there may be several control frames circulating simultaneously. In this case, control frames are transmitted back-to-back on the control wavelength.

Each node is the owner of one control slot in each control frame. Each control slot contains several fields, as Figure 2.4 illustrates. The format and type of the fields depend on the OBS protocol used (for more details, refer to the description of the protocols in Section 2.4). In general, however, each control slot includes fields for the destination address, the offset, and the burst size. Other fields, such as a token field, may be included for some of the protocols, as necessary.

When acting as a source, a node waits for the next control frame and writes the burst information (destination address, burst length, and, if applicable, the offset) in its own control slot. If it has nothing to transmit, it just clears all the fields in its control slot. At each node, the entire control frame is read first to determine whether any control slots indicate a burst transmission to this node. If so, and assuming that the node is not in the process of receiving another burst, it instructs its tunable receiver to tune to the appropriate wavelength to receive the burst; in other words, preemption is not allowed. In case of a receiver collision (i.e., when

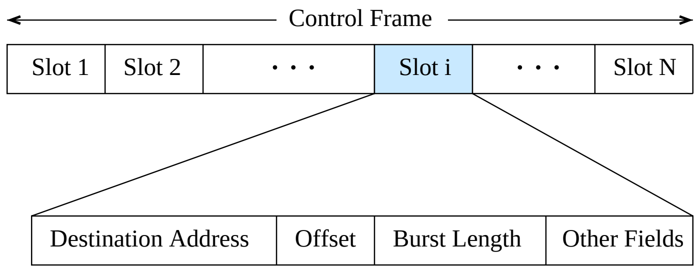

Figure 2.4: Structure of a control frame

the address of this node is specified in multiple control slots, which may give rise to overlapping transmissions), the destination node selects one of the bursts to receive. In an acknowledgment-based protocol, the node also modifies the appropriate field as an indication to the source node to transmit its burst.

We note that each node in the ring acts as a source node (inserting bursts in its home wavelength), as an intermediate node (passing through bursts traveling to downstream nodes), and as a destination node (terminating bursts sent to it). As a result, each node must read each control frame in its entirety before determining what action to take (i.e., whether to write in its own control slot to indicate its intention to transmit a burst, and/or whether to acknowledge the request of a burst transmission). Therefore, in a ring network the time to process a control frame is the same for intermediate and destination nodes (i.e., $T_i^{(p)} = T_d^{(p)}$). The control frame is delayed by this amount of time as it passes through each node. This delay is the sum of the control frame transmission time plus the time to process the control frame, and it can be kept short by employing a simple protocol implemented in hardware. A number of OBS protocols having these features are described in the next section.

2.4 OBS Protocols

Since each OBS node is assigned a unique home wavelength, bursts may be lost due to receiver collisions. This occurs when two or more source nodes transmit (each on its home wavelength) bursts to the same destination node, and the burst transmissions overlap in time. In this chapter, we proposed a number of different access protocols that differ mainly in the way that receiver conflicts are resolved. These protocols can be classified in the following three classes, depending on who is responsible to resolve receiver collisions.

- Source Node. In this class of protocols, a source node resolves receiver collisions using the information transmitted on the control wavelength.
- Destination Node. In this class of protocols, a source node must get permission from the destination node, before it can send its burst. The destination node schedules all incoming requests so that to avoid collisions.
- Other. In this class of protocols, neither the source node nor the destination node are responsible for receiver collision resolution. For example, a common method in ring networks is to use tokens to resolve receiver collisions.

Our emphasis is on protocols that use few rules, are simple to implement in hardware (i.e., they can operate at wire speeds) and are distributed in nature (i.e., each node locally executes an identical copy of the protocol and makes transmit decisions by its local knowledge). We have deliberately avoided protocols that are centralized in nature, or they require the collection of transmit queue sizes, or they require network-wide synchronization (e.g., TDM-based schemes).

In this chapter, we propose five access protocols, namely: RR/R, RR/P, RR/NP, RR/ACK, and RR/Token. The first three protocols belong to the “source node” class, RR/ACK belongs to the “destination node” class, and the last one belongs to “other” class. Before we proceed with the description of the five protocols, a short discussion on the assumptions we make is necessary. We define a burst as an encapsulation of IP packets, ATM cells, Frame Relay frames, or some other types of packets containing data. A burst format is needed so that the destination node can correctly extract the data from the received burst. The format of a burst is outside the scope of this work. While any burst format incurs overheads that affect performance measures such as throughput and delay, the various protocols are affected in the same degree. Since we are interested in the *relative* performance of the five protocols, we have ignored this burst format overhead.

A transmit queue is *eligible* for service if its size is larger than `MinBurstSize`, or the first data of the transmit queue has waited for more than `TimeOut` time. If the size of the *eligible* transmit queue is less than `MaxBurstSize`, then a burst that includes all data in the transmit queue is constructed. Otherwise, a burst of at most size `MaxBurstSize` is constructed, and the data remaining in the transmit queue is served at a later time.

In the following five subsections, we describe the proposed OBS access protocols. Numerical results are given in Section 2.5.

2.4.1 Round-Robin with Random Selection (RR/R)

The first protocol we consider uses a round-robin scheduler at each node to serve the transmit queues, and lets each receiver randomly select a burst from the bursts that arrive simultaneously. Thus, we call this protocol *Round-Robin with Random Selection (RR/R).* More specifically, the operation of the protocol at node i is as follows.

- At the transmitting side, the scheduler of node i visits all eligible transmit queues in a round-robin fashion. Suppose that, at time t_1, transmit queue j is selected for service, then node i waits for the first control frame that arrives after time t_1. When the frame arrives, node i writes the burst information and destination address j in its own control slot (i.e., the i-th slot of the control frame). After a delay equal to the offset value, node i transmits the burst on its home wavelength.

- At the receiving side, when a control frame arrives at node i, it scans the control slots of the control frame, checking for any slot that has i in the destination address field. If more than one such slots are found, node i randomly selects one of them, say k (since all the corresponding bursts will arrive at node i at the same time for both JET and ODD, and at most one of them can be accepted).

In this case, all bursts to node i except the burst from node k will be lost. Node i then checks whether its receiver is free at the time when the burst from node k arrives at node i, and checks whether its receiver has enough time to tune to another wavelength. If so, it instructs its receiver to tune to node k's home wavelength in order to receive the burst transmission. Otherwise, it gives up on the burst from node k.

Because of the randomness involved in resolving receiver conflicts, RR/R is a fair protocol. However, burst loss may occur due to these conflicts.

2.4.2 Round-Robin with Persistent Service (RR/P)

The *Round-Robin with Persistent Service (RR/P)* protocol is similar to the RR/R protocol, but it is designed to eliminate receiver conflicts that can be detected prior to the transmission of a burst. The operation of this protocol at node i is as follows.

- At the transmitting side, node i maintains a variable `EarliestFreeTime(j)` for each destination node j, which specifies the earliest time at which the receiver of node j would be free. This variable is updated by monitoring the burst information in control slots that have j in the destination address field.

  The scheduler at node i visits all eligible transmit queues in a round-robin fashion. Suppose that, at time t_1, transmit queue j is selected for service, then node i waits for the first control frame that arrives after time t_1. Suppose it arrives at time t_2, then node i updates the variable `EarliestFreeTime(j)` based on relevant information (if any) in the control frame. Node i also computes the time t_3 that the first bit of its burst would arrive at node j. t_3 is calculated as follows.

$$t_3 \;=\; t_2 \;+\; T_i^{(p)} \;+\; \text{offset} \;+\; \delta_{ij} \tag{2.3}$$

  where δ_{ij} is the burst propagation delay from node i to node j. If `EarliestFreeTime(j)` plus the receiver tuning time at node j is less than t_3, then node i writes its burst information in its own control slot, and sends the burst after a delay equal to the offset. If, on the other hand, `EarliestFreeTime(j)` plus the receiver tuning time at node j is greater than t_3, then node i knows that sending its burst will result in a receiver conflict. In this case, node i does not transmit the burst; instead it waits for the next control frame and repeats the process of transmitting the burst to node j. This is the *persistent* feature of the protocol, in that the round-robin scheduler does not proceed to serve the next transmit queue until the burst to node j has been sent.

We note that deferring the transmission of a burst based on a calculation of the earliest free time for receiver j does not altogether eliminate receiver collisions. Suppose that two nodes simultaneously determine (based on information they read in *different* control frames) that it is safe to send a burst to some destination j. This simultaneous transmission may result in a receiver conflict, which neither of the nodes is able to predict. When the downstream node later receives the control frame with the upstream nodes burst information, it will detect the conflict. Despite this fact, the downstream node proceeds with its burst transmission, and its scheduler also proceeds to serve the next eligible transmit queue after queue j.

- At the receiving side, the operation of the protocol is identical to RR/R.

RR/P does eliminate some receiver collisions, but it does not completely eliminate receiver collisions.

2.4.3 Round-Robin with Non-Persistent Service (RR/NP)

The operation of the *Round-Robin with Non-Persistent Service (RR/NP)* protocol is identical to the operation of the RR/P protocol with one exception. Suppose that at time t_1 node i has selected transmit queue j for service using the RR scheduler. Suppose also that once the first control frame arrives after time t_1, the node determines that transmitting a burst to j would result in a collision. The node refrains from transmitting the burst, but, instead of continuing its attempt to serve transmit queue j (the persistent feature of RR/P), its scheduler proceeds to serve the next eligible transmit queue upon arrival of the next control frame.

The RR/NP protocol may result in lower delay than RR/P. However, since a node gives up its burst transmission whenever it determines that it will lead to a collision, RR/NP may lead to the starvation of certain transmit queues, and thus, it has fairness problems. Specifically, a node's priority to transmit to a given destination depends on the relative location in the ring. Node i has the highest (lowest) priority to transmit bursts to node $(i \ominus 1)$ (respectively, node $(i \oplus 1)$), where $\ominus$ and $\oplus$ denote subtraction and addition, respectively, modulo N.

As in RR/P, RR/NP does not completely eliminate receiver collisions.

2.4.4 Round-Robin with Tokens (RR/Token)

This protocol uses tokens to resolve receiver collisions at the receivers. Different from traditional token-based protocols, such as the IBM token ring and FDDI, which are single token access protocols, this protocol uses multiple tokens (Cai *et al.* [7] proposed a multiple token access protocol for a different WDM ring architecture). There are N tokens, one for each destination node. A token may be either available or in use. The status of token j is indicated in a binary field (located in the "other fields") of the j-th control slot. If it is available, then the binary field is set to one. Otherwise, it is set to zero. If token j is available, then this will be marked in the j-th control slot of only one control frame. In the remaining control frames, this binary field will be set to zero. A node can only transmit to a destination node j, if it captures the j-th token. The transmit queues at each node are served in a Round-Robin manner. Thus, we call this protocol *Round-Robin with Tokens (RR/Token).*

The operation of the protocol at node i is as follows.

- At the transmitter side, node i monitors each received control frame. If it finds an available token, node i removes it from the control frame, and puts it in its FIFO token queue. Node i also serves the transmit queues in the arrival order of tokens. More specifically, suppose that the first token in the token queue is token j, node i first checks whether transmit queue j is eligible for service. If not, node i releases token j, i.e. it removes it from its token queue, and it places it in the next control frame, and it then proceeds with the next token in the queue. Otherwise, node i constructs the burst to node j, writes the burst information in the next control frame, and sends it after a delay equal to the offset value. Once the burst transmission is complete, node i releases token j to the next control frame. It then proceeds to serve the transmit queue corresponding to the next token in the token queue. Since every node has a FIFO token queue, the order in which tokens circulate around the ring is fixed. Recall that there are only N tokens, one for each destination node. Therefore, transmit queues are served in a Round-Robin manner.

- At the receiver side, node i checks each incoming control frame for any control slot indicating a burst transmission to this node. If such a control slot is found, node i instructs its receiver to tune to the appropriate home wavelength for receiving the burst.

Because of the token operation, there will be at most one burst transmission arriving at a destination node at any time. That is, RR/Token is a receiver collision free protocol.

2.4.5 Round-Robin with Acknowledgement (RR/ACK)

The *Round-Robin with Acknowledgment (RR/ACK)* protocol is based on the Tell and Wait (TAW) scheme [32]. A source node i first sends a request (including destination and size) to transmit a burst to the destination node j. When node j receives the request, it calculates an offset value, and sends it back to node i in the offset field of control slot i. We note that a source node is not allowed to have more than one outstanding request; in other words, it is not permitted to send out another request to a different destination node while it is waiting for an acknowledgement. This rule avoids transmitter conflicts, i.e., the situation in which a source node receives acknowledgements from two or more different destinations which may cause overlapping burst transmissions.

- At the transmitter side, node i selects a transmit queue j using Round-Robin among all `eligible` transmit queues. It then waits for the next incoming control frame, and it writes a request in its own control slot i. The request consists of the destination address (in this case, j), and the burst length. Note that the source node i does *not* write the offset field in the control slot; the offset value will be provided by the destination node as the acknowledgment. After node i receives the acknowledgment from node j one round-trip time later, it instructs its transmitter to send out the burst at the time specified by node j in the acknowledgement.

  Let τ be the round-trip delay of a control frame (i.e., the propagation time around the ring plus the sum of the processing time of a control frame at each node in the ring). Let t be the time (in the future) at which the current burst transmission to node j will complete. In order to improve the utilization of the ring under the RR/ACK scheme, we define the *next safe request point* for node i as $t - \tau$. Node i will wait until time $t - \tau$ before it submits a new request for transmission to another destination k. Thus, when node i receives the acknowledgment from node k at time $(t-\tau)+\tau = t$, the burst transmission to node j will be complete and its transmitter will be free to transmit a burst to node k.

- At the receiver side, node j acknowledges (i.e. fills in the corresponding offset field) each request it receives in a first-come, first-served manner. Specifically, after acknowledging a request from node i, node j computes the time t' at which it will receive the last bit of node i's burst. Node j's receiver is free after time t'. When the next request arrives, say, from node k, node j sends an offset that is computed such that the first bit of node k's burst will arrive at node j after time t' plus the tuning latency of the receiver.

No	Protocol Name	Offset Calculation
1	RR/R	ODD
2	RR/P	ODD
3	RR/NP	ODD
4	RR/Token	ODD
5	RR/R	JET
6	RR/P	JET
7	RR/NP	JET
8	RR/Token	JET
9	RR/ACK	TAW

Table 2.1: OBS protocols used in the simulation

RR/ACK is a receiver collision free protocol.

2.5 Numerical Results

In this section we use simulation to compare the protocols listed in Table 2.1. For each of the four protocols RR/R, RR/P, RR/NP, and RR/Token, we consider two variants: one in which the offset calculation is based on ODD, using expression 2.2, and one in which the offset calculation is based on JET, using expression 2.1. Recall that the main difference between the two offset calculations is that the ODD offset includes only the processing and setup delay at the destination, while the JET offset includes additional terms representing the processing delays at intermediate nodes. As we shall see, the ODD offset calculation results in smaller delay for all four protocols, and higher throughput for the RR/Token protocol, the only receiver collision free protocol. Moreover, ODD makes it possible to design a delay fair protocol. However, the reader should keep in mind that this performance improvement is achieved at the expense of more complex burst switching nodes, since the latter must implement fiber delay lines to delay incoming bursts for an amount of time equal to the processing delay of a burst header packet. Finally, we also simulate the RR/ACK protocol which is a tell-and-wait (TAW) protocol.

In our simulation study we consider a ring network with 10 nodes, each with an electronic buffer of 10 MBytes. The distance between two successive nodes in the ring is taken to be 5 kilometers. We assume that the control wavelength runs at 622 Mbps, while each burst wavelength runs at 2.5 Gbps. Each control slot in a control frame is 100 bytes long regardless of the protocol used in the ring. That is, the duration of a control slot is 1.286 μs. The processing time of a control frame at both the intermediate ($T_i^{(p)}$) and destination nodes ($T_d^{(p)}$) is set to be 10 slot times, or 12.86 μs, and the setup time at the destination nodes $T_d^{(s)}$ is 1 μs.

We assume that data arrives in packets, and the packet arrival process to each node is described by a modified Interrupted Poisson Process (IPP) [14]. This modified IPP is an ON/OFF process, where both the ON and the OFF periods are exponentially distributed. Packets arrive back to back during the ON period at the rate of 2.5 Gbps. No packets arrive during the OFF period. The packet size is assumed to follow a truncated exponential distribution with an average size of 500 bytes and a maximum size of 5000 bytes. The last packet in an ON period may be truncated so that its last bit arrives at the end of the ON period. We use the squared coefficient of variation, c^2, of the packet inter-arrival time to measure the burstiness of the arrival process. c^2 is defined as the ratio of the variance of the packet inter-arrival time divided by the squared mean of the packet inter-arrival time. We use the expression for the c^2 of an IPP, where the packet size is not truncated. We have

$$c^2{}_{IPP} \quad = \quad 1 \; + \; \frac{2 \; \lambda \; \mu_1}{(\mu_1 \; + \; \mu_2)^2} \tag{2.4}$$

where $1/\lambda \; = \;$ (500 bytes) / (2.5 Gbps) $\; = \;$ 1.6 μs, and $1/\mu_1$ and $1/\mu_2$ are the mean times of the ON and OFF periods, respectively. We have found experimentally that it is very close to the c^2 of the modified IPP used in this simulation. To completely characterize the arrival process, we use the above expression for the c^2 and another equation that involves the mean times of the ON and OFF periods. We define the quantity

$$\text{Average Arrival Rate} \quad = \quad (2.5 \text{ Gbps}) \; \times \; \frac{\mu_2}{\mu_1 \; + \; \mu_2} \tag{2.5}$$

Given the c^2 and the average packet arrival rate, we can calculate the quantities μ_1 and μ_2, and therefore the arrival process is completely characterized.

In all the figures given in this section, simulation results are plotted along with 95% confidence intervals estimated by the method of batch mean. The number of batches was set to 30, with each batch run lasting until each node has transmitted at least 10,000 bursts. As the reader will notice, however, most confidence intervals are very narrow and are barely visible in these figures.

In Section 2.5.1 we present a comparison of the performance of the RR/R, RR/P, RR/NP, and RR/Token protocols with ODD offsets. In Section 2.5.2 we investigate the impact of the offset calculation JET versus ODD on the performance of the protocols. In Section 2.5.3, we compare RR/ACK that uses the TAW scheme with RR/Token that uses the ODD offset. In these three sections, the traffic to the ring is symmetric. That is, each node is fed with an arrival process that has the same parameters, and a packet arriving at a node is assigned a destination node following the uniform distribution. In Section 2.5.4, we study the performance of the access protocols assuming asymmetric traffic.

2.5.1 Performance of Protocols with ODD Offset

Effect of Average Arrival Rate

In this section, we investigate the performance of the first four protocols listed in Table 2.1 for which the calculation of the offset is based on ODD. Specifically, we are interested in five performance measures, namely: throughput, loss, delay, fairness, and buffer requirement. These performance measures are estimated by varying the average arrival rate from 0.5 Gbps to 2.0 Gbps with an increment of 0.3 Gbps. (The average arrival rate we refer to, is the average arrival rate into a single node). Packets arriving at a node are assigned a destination node following the uniform distribution. c^2 of the packet inter-arrival time at each node is set to 20. We also set `MaxBurstSize` to 112 Kbytes, `MinBurstSize` to 16 Kbytes, and `TimeOut` to 4 ms, which is about ten times the round-trip delay of the control frame.

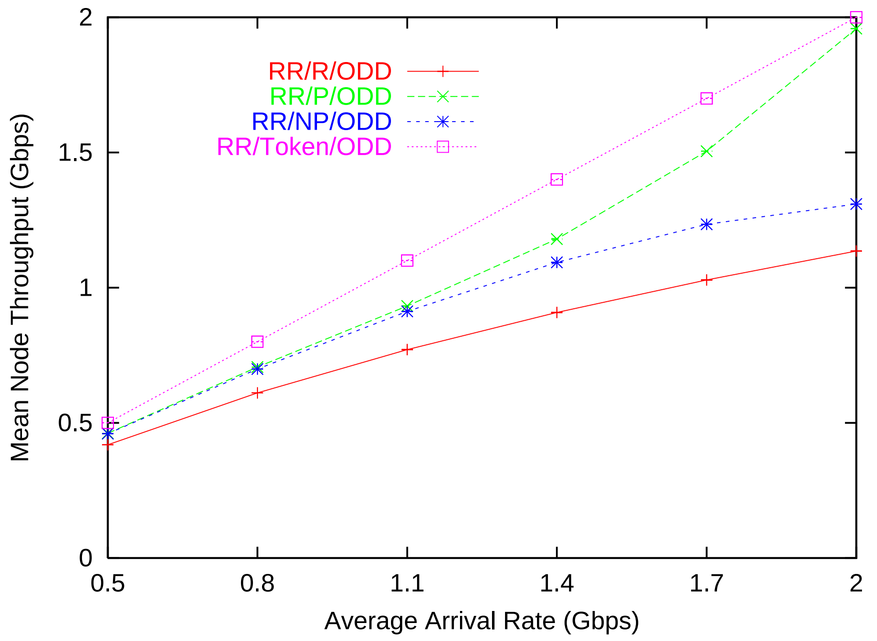

Figure 2.5: ODD: Mean node throughput vs. average arrival rate

Figure 2.5 plots the mean node throughput versus the average arrival rate for all four protocols. The mean node throughput is defined as the average number of bits received by all nodes in a unit time divided by the number of nodes. We observe that RR/Token, a receiver collision free protocol, achieves the highest throughput. Among the three protocols in which receiver collisions are possible, RR/P achieves the highest throughput, followed by RR/NP and RR/R.

We distinguish between two types of loss. First, packets arriving to find a full buffer at the source node are dropped. In our simulation experiments, we observed that only RR/Token has a 0.01% packet loss rate (i.e., the number of packets lost divided by the number of all packets arrived) due to buffer overflow, when the average arrival rate is 2.0 Gbps. That means RR/Token requires a larger buffer than the other three protocols.

The second type of loss occurs when a burst is dropped at the destination due to a receiver collision. Figure 2.6 plots the burst loss rate due to receiver collisions versus the average arrival rate. The burst loss rate is the total number of lost bursts in all

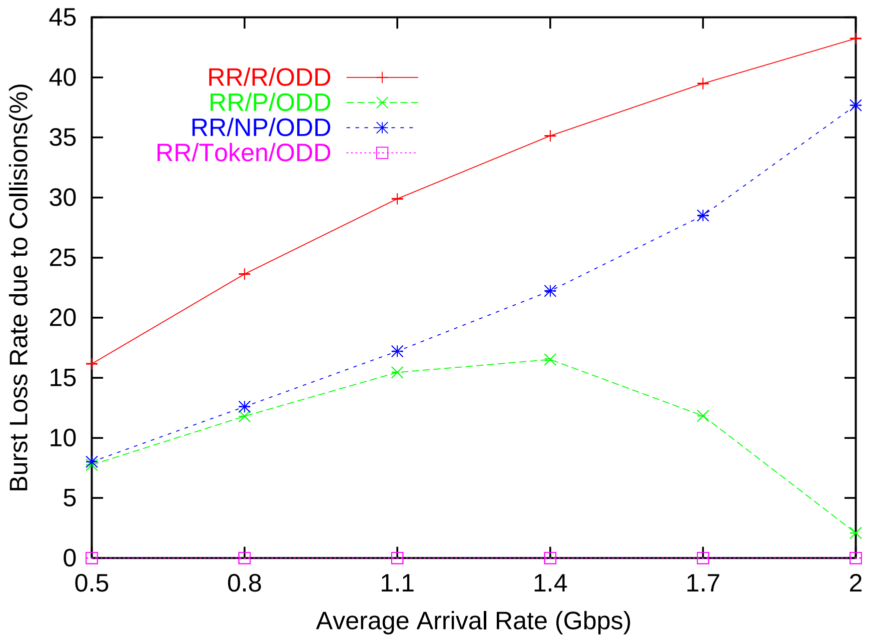

Figure 2.6: ODD: Burst loss rate due to receiver collisions vs. average arrival rate

nodes divided by the total number of transmitted bursts on the ring. As a receiver collision free protocol, RR/Token never incurs loss due to receiver collisions. For the other three protocols, RR/P has the least burst loss rate, followed by RR/NP and RR/R.

Next, we give an intuitive explanation of the burst loss plots in Figure 2.6. Recall that a burst loss means that two or more bursts overlap in time. Therefore, intuitively the more irregular the burst size, the larger the burst loss rate. We use the c^2 of the burst size to measure how irregular the burst size is. We found that if all other parameters are kept the same, a larger burst size c^2 leads to a larger burst loss rate due to receiver collisions. Figure 2.7 shows the c^2 of the burst sizes as a function of the average arrival rate. We note that the plots in both Figures 2.6 and 2.7 have the same pattern. As the average arrival rate increases, the c^2 of the burst size of RR/R and RR/NP increases, and so does the burst loss rates. As for RR/P, as the average arrival rate increases, the burst size c^2 first increases, then peaks at 1.4 Gbps, and finally it decreases. The burst loss rate follows the same pattern. The reason for the

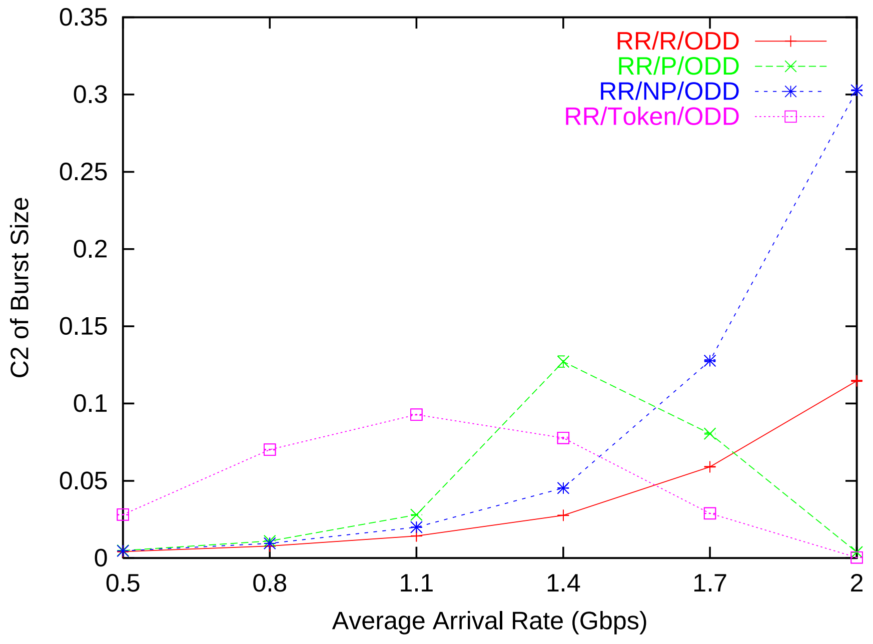

Figure 2.7: ODD: c^2 of burst size vs. average arrival rate

change in the c^2 of burst size is that when the burst size reaches a specific point, the `MaxBurstSize` starts to limit the c^2 of burst size.

From the simulation, we also found that the burst loss rate due to receiver collisions of RR/P depends not only on the c^2 of the burst size, but also on another important parameter, the `EnoughData` probability. Recall that in a node, a transmit queue is not eligible for service unless its size is at least equal to the value of `MinBurstSize`. Therefore, when a node turns to serve a transmit queue, the transmit queue may or may not be eligible for service. The probability that a transmit queue is eligible for service when a node turns to serve it is the `EnoughData` probability. We found that for RR/P, an `EnoughData` probability equal to or very close to one leads to a lower burst loss rate due to receiver collisions than an `EnoughData` probability close to zero. Figure 2.8 shows the `EnoughData` probability versus the average arrival rate. The `EnoughData` probability of RR/P increases as the average arrival rate increases. Especially, it reaches almost 1 when the average arrival rate reaches 1.7 Gbps.

Figure 2.9 plots the mean packet delay versus the average arrival rate. The mean

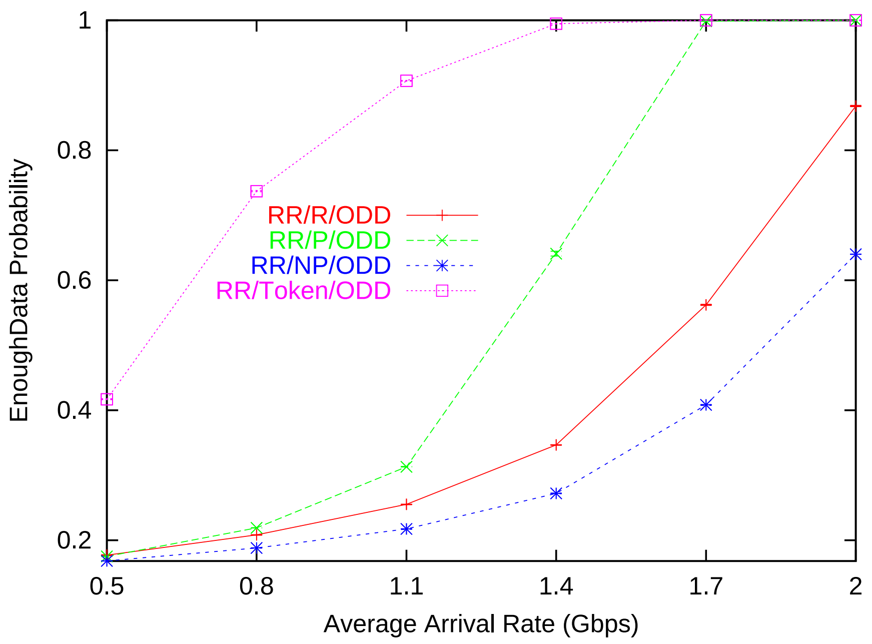

Figure 2.8: ODD: `EnoughData` probability vs. average arrival rate

packet delay is the average packet delay over all transmit queues and nodes, where the packet delay includes both the queueing and the propagation delay. The packet queueing delay is defined as the time interval from the instance that the packet arrives at a node to the instance that the packet leaves the node. RR/R has the least delay, followed by RR/NP, RR/P and RR/Token. We observe that, as the average arrival rate increases, the mean packet delay in all protocols first decreases, and then it increases. This is due to the fact that when the traffic intensity is low, the time for a transmit queue to reach the `MinBurstSize` accounts for the major part of the packet delay. Therefore, as the average arrival rate increases, the time for a transmit queue to reach `MinBurstSize` decreases, which causes the mean packet delay to decrease. The 95% percentile packet delay was also calculated in the simulation. Since the plot trend is the same as that of the mean packet delay, the figure is not shown here.

Let us now compare the four protocols in terms of fairness. We distinguish two types of fairness, namely, throughput fairness and delay fairness. We define the *throughput fairness index of a node i* as the c^2 of the throughput from node i to all

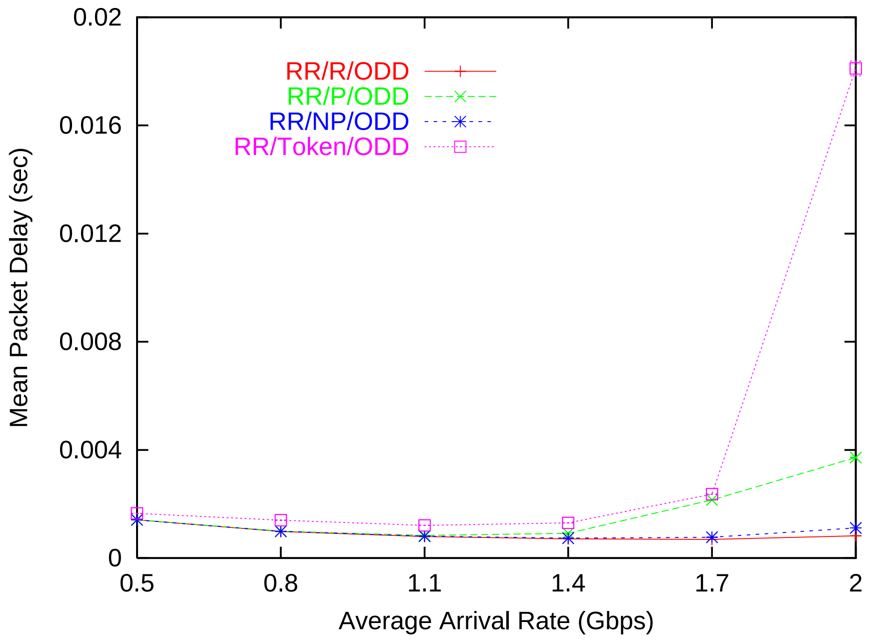

Figure 2.9: ODD: Mean packet delay vs. average arrival rate

other nodes.

$$\text{Throughput Fairness Index of Node } i \quad = \quad \left(\sum_{j=1, j\neq i}^{10} (H_{ij} \; - \; \overline{H_i})^2\right) \times \frac{1}{\overline{H_i}^2} \qquad (2.6)$$

where H_{ij} is the throughput from node i to node j, i.e. the average number of bits transmitted by node i and received by node j in a unit time, and $\overline{H_i} = (\sum_{j=1,j\neq i}^{10} H_{ij})/9$. We then define the *throughput fairness index of a protocol* as the average of the throughput fairness indeces of all nodes. According to this definition, the smaller the throughput fairness index of a protocol, the better the throughput fairness of the protocol.

Figure 2.10 shows the throughput fairness index of the four protocols versus the average arrival rate. We observe that RR/R and RR/Token have values very close to zero, meaning that they are throughput fair protocols. In Appendix A, we give additional figures of the throughout from node 0 to all other nodes under the four protocols. We observed that both RR/NP and RR/P protocol provide better throughput to nodes closer to the source than to nodes far away. This follows directly from the

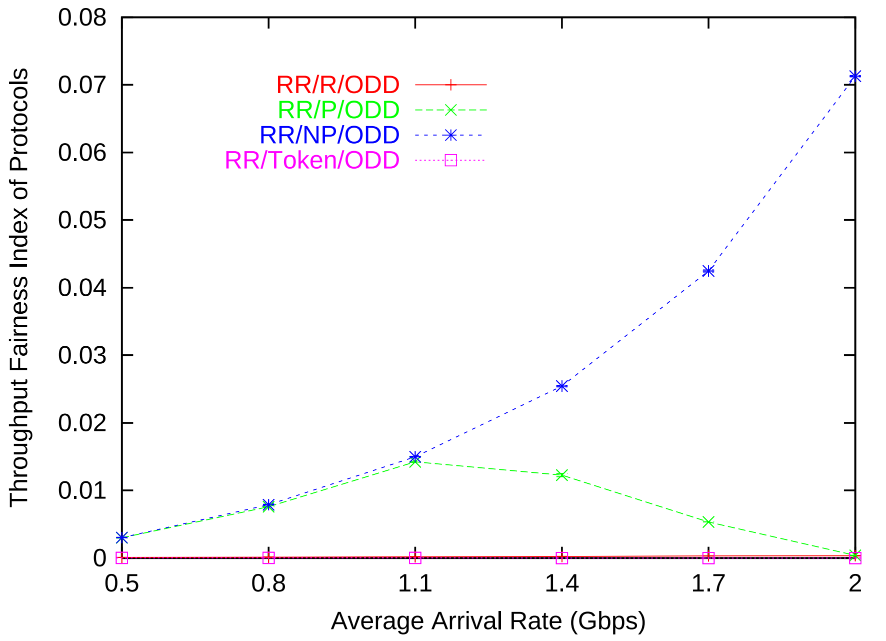

Figure 2.10: ODD: Throughput fairness index of protocols vs. average arrival rate

operation of RR/NP and RR/P described in Section 2.4.3 and 2.4.2.

The second type of fairness we consider is related to delay. For this, we define the *delay fairness index of a node i* as the c^2 of the mean packet queueing delay of the transmit queues. We have

$$\text{Delay Fairness Index of Node } i \quad = \quad \left(\sum_{j=1, j \neq i}^{10} (W_{ij} - \overline{W_i})^2 \right) \times \frac{1}{\overline{W_i}^2} \tag{2.7}$$

where W_{ij} is the mean queueing delay of a packet in transmit queue j in node i, and $\overline{W_i} = (\sum_{j=1,j\neq i}^{10} W_{ij})/9$. We also define the *delay fairness index of a protocol* as the average of the delay fairness indeces of all nodes. (Note that in defining the fairness index we use the queueing delay only, not the total delay which also includes the propagation delay which depends on the destination node). According to this definition, the smaller the delay fairness index of a protocol, the better the delay fairness of the protocol. Specifically, if the delay fairness index of a protocol is zero, the protocol is perfectly fair since the queueing delay of a packet is insensitive to the source and destination of the packet. For unfair protocols, access to the burst

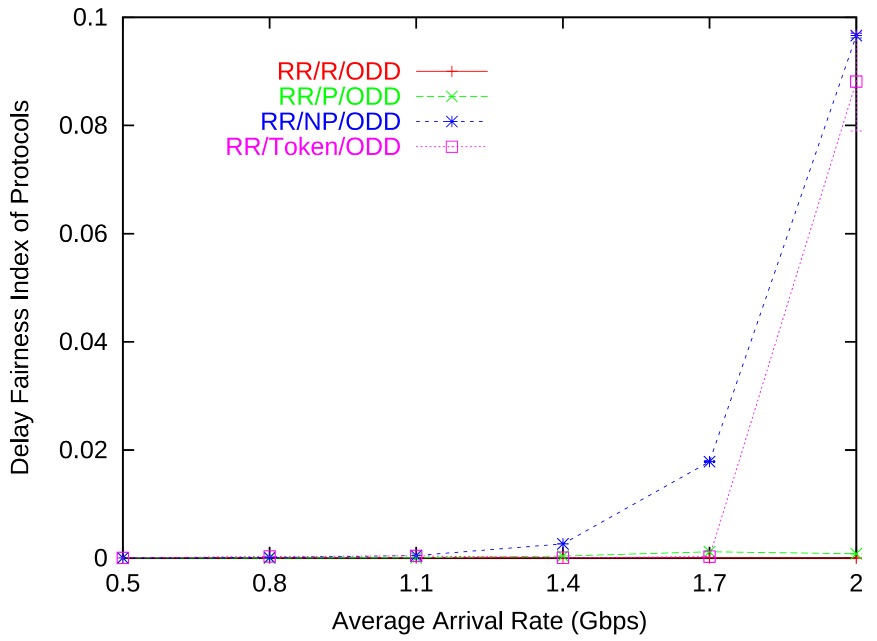

Figure 2.11: ODD: Delay fairness index of protocols vs. average arrival rate

wavelengths may depend on factors such as the relative position of the source and destination nodes in the ring. In this case, some transmit queues may take longer to serve than others, increasing the queueing delay of the respective packets relative to others, and thus, increasing the delay fairness index of the node and protocol.

Figure 2.11 shows the delay fairness index of the four protocols versus the average arrival rate. We observe that only RR/R has delay fairness index values very close to zero, meaning that it is the only fair protocol in terms of delay. In Appendix A, we give additional figures of the mean packet queueing delay of each transmit queue in node 0 for all protocols. We observed that RR/NP provides better delay access to wavelengths of nodes far away than to wavelengths of nodes close to the source of bursts, and RR/P and RR/Token do not always provide the best or worst delay access to a specific node. For further details, the reader is referred to Appendix A.

Overall, based on the above experimentation, RR/Token achieves the highest mean node throughput, followed by RR/P, RR/NP and RR/R. RR/R has the smallest mean packet delay, followed by RR/NP, RR/P and RR/Token. RR/R also requires the smallest mean buffer requirement, followed by RR/NP, RR/P and RR/Token. The burst loss rate due to receiver collisions for the protocols which are not receiver collision free depends on the burst size c^2. The burst loss rate of RR/P also depends on the `EnoughData` probability. Only RR/R is a delay fair protocol, while both RR/R and RR/Token are throughput fair protocols.

Effect of `MaxBurstSize`

We varied the value of `MaxBurstSize` from 32 Kbytes to 112 Kbytes with an increment of 16 Kbytes. `MinBurstSize` is 16 Kbytes. The average arrival rate to each node is 1.7 Gbps, c^2 of the packet inter-arrival time at each node is 20, and, and `TimeOut` is 4 ms. A packet arriving at a node is assigned a destination node following the uniform distribution.

Simulation results showed that an increase in `MaxBurstSize` leads to an increase in the burst size c^2 and to a small change of the `EnoughData` probability, which lead to the increase in the burst loss rate due to receiver collisions, and finally lead to the decrease in the throughput of RR/R, RR/NP, and RR/P, as shown in Figure 2.12. However, the decrease in the throughput of RR/R and RR/NP is very small. RR/Token requires a large `MaxBurstSize` so that no packets will be lost due to buffer overflow.

Figure 2.13 plots the mean burst delay against `MaxBurstSize`. We observe that the delay of RR/R and RR/NP is not sensitive to the `MaxBurstSize`. We also observe that as the `MaxBurstSize` increases, the delay of RR/P and RR/Token first decreases, and then increases. The intuitive reason for the decrease is that a larger `MaxBurstSize` means that a node can transmit more packets at a time, which makes the delay go down. The intuitive reason for the increase is that a larger `MaxBurstSize` permits other nodes to transmit more packets at a time, so that the node must wait for a longer time to transmit its burst, which causes the delay to increase. Finally, we observe that only a very small `MaxBurstSize` can lead to a very long delay.

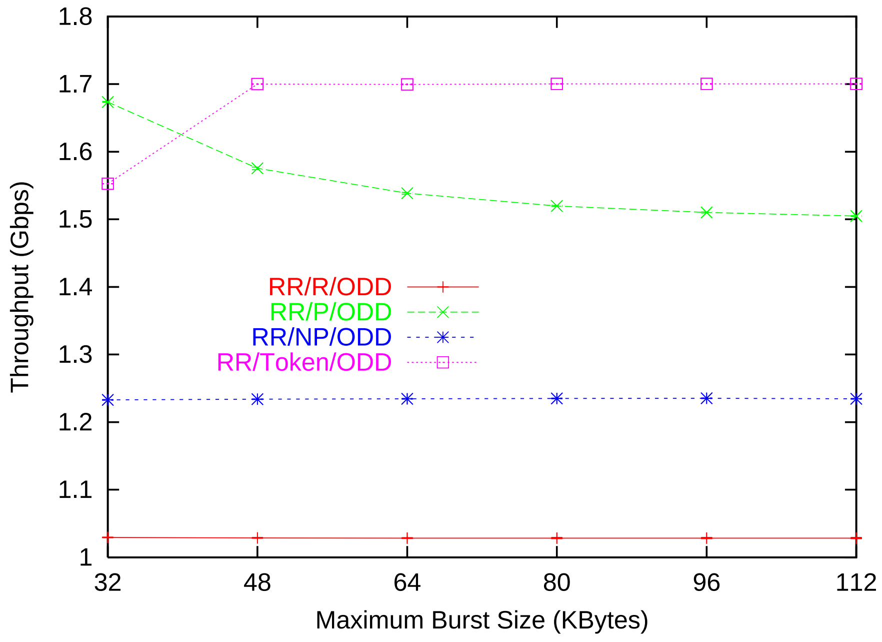

Figure 2.12: ODD: Mean node throughput vs. maximum burst size

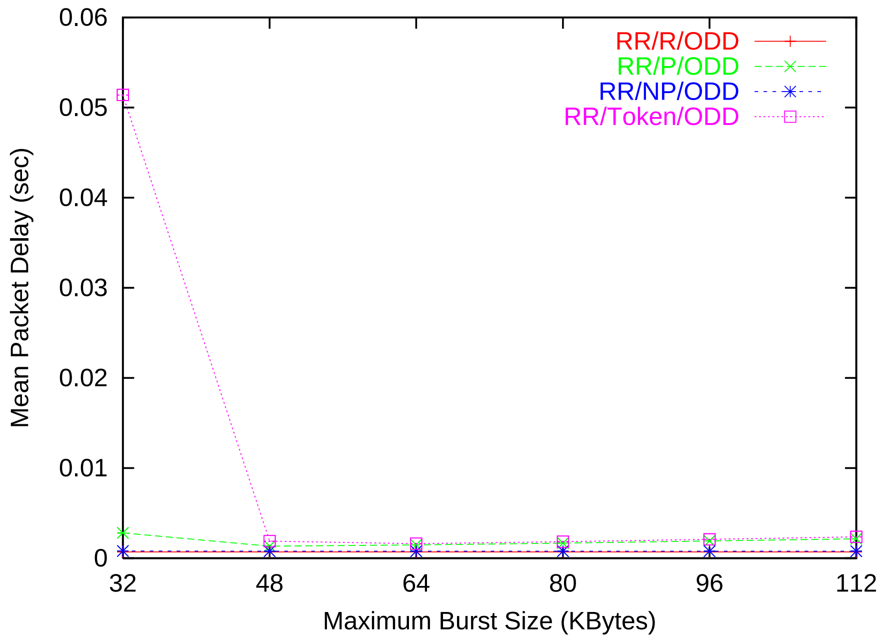

Figure 2.13: ODD: Mean packet delay vs. maximum burst size

Effect of `MinBurstSize`

We varied `MinBurstSize` from 16 Kbytes to 96 Kbytes with an increment of 16 Kbytes. `MaxBurstSize` is 112 KBytes. The average arrival rate to each node is 1.7 Gbps, c^2 of the packet inter-arrival time at each node is 20, and `TimeOut` is 4 ms. A packet arriving at a node is assigned a destination node following the uniform distribution.

Simulation results showed that an increase in `MinBurstSize` leads to a decrease in the burst size c^2 and a decrease in `EnoughData`. For RR/R and RR/NP, the decrease in the burst size c^2 leads to a small decrease in the burst loss rate due to collisions, which finally leads to a small increase in the mean node throughout, as shown in Figure 2.14. However, for RR/P, a big decrease in the `EnoughData` probability leads to an increase in the burst loss rate due to receiver collisions, which finally leads to a decrease in the mean node throughout. Changes in `MinBurstSize` do not lead to any change in the mean node throughput of RR/Token. Increases in `MinBurstSize` also lead to increases in the mean packet delay of all protocols, as shown in Figure 2.15.

2.5.2 JET vs. ODD

In this section we focus on the difference between the JET and ODD offset calculations. In our comparisons, we will only consider two protocols: RR/Token and RR/R. RR/Token is selected since it is free of receiver collisions, while RR/R is selected as a representative protocol among the three protocols that suffer from receiver collisions.

Simulation experiments were carried out with the same parameters as in section 2.5.1. The results showed that, compared to ODD, JET leads to a longer mean packet delay for all protocols (see Figure 2.16), which in turn leads to a larger mean buffer requirement (see Figure 2.17), and to a larger packet loss rate due to buffer overflow (see Figure 2.18). Therefore, as a receiver collision protocol, RR/Token has a lower mean node throughput with JET than with ODD. Moreover, JET naturally leads to delay unfair protocols, but does not change the throughput fairness property

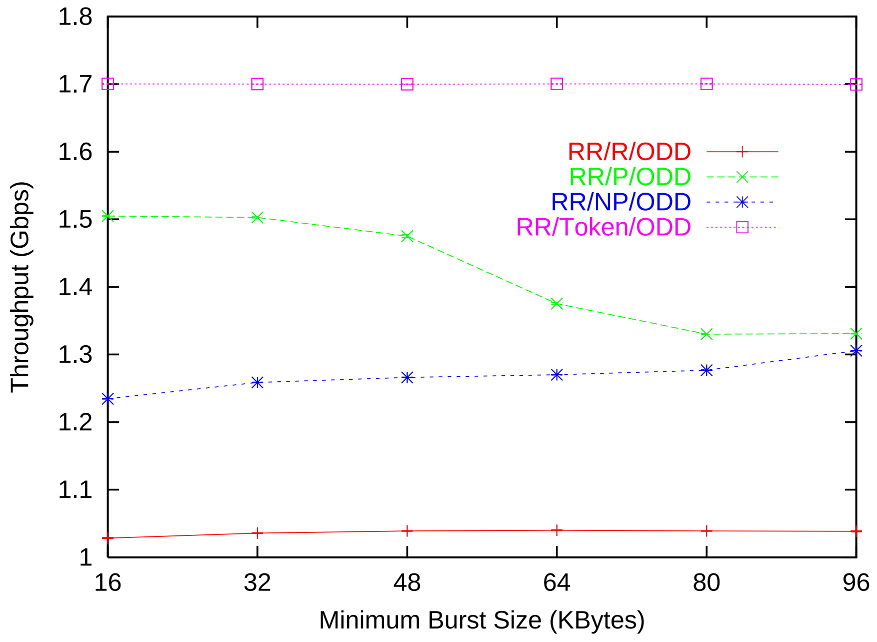

Figure 2.14: ODD: Mean node throughput vs. minimum burst size

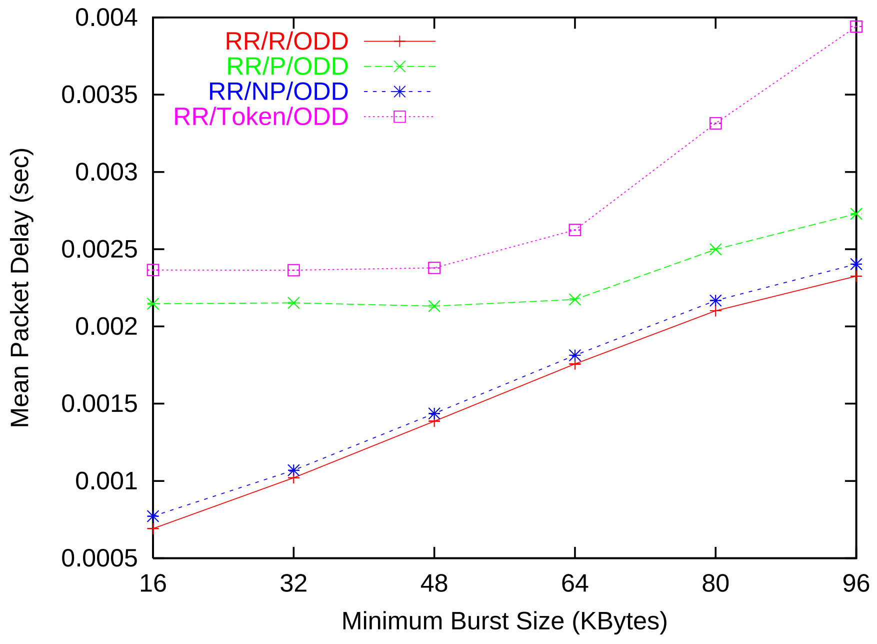

Figure 2.15: ODD: Mean packet delay vs. minimum burst size

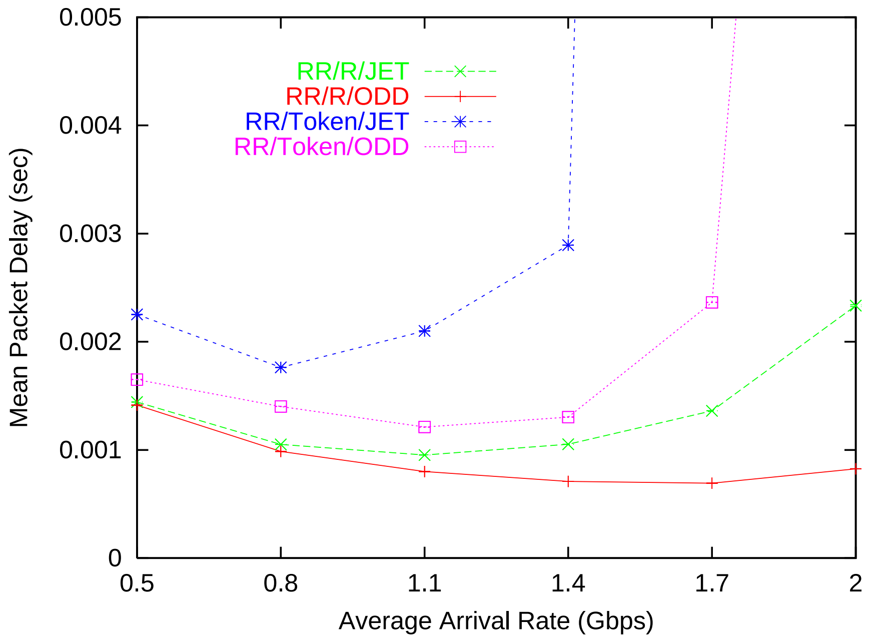

Figure 2.16: JET: Mean packet delay vs. average arrival rate

of the protocols.

The effect of `MaxBurstSize` was also investigated. The results showed that all protocols are more sensitive to `MaxBurstSize` with JET than with ODD. A much larger `MaxBurstSize` is required in JET than in ODD, in order to get a higher mean node throughput and lower mean packet delay.

Results also showed that both ODD and JET are not very sensitive to `MinBurstSize`. As the `MinBurstSize` increases, for RR/R, ODD and JET are almost same. But for RR/Token, ODD is always much better than JET in both the mean node throughput and the mean packet delay.

2.5.3 TAW vs. ODD

RR/ACK is the only protocol using TAW, and it is receiver collision free. We compared RR/Token using ODD with RR/ACK using TAW using the same parameters as in section 2.5.1.

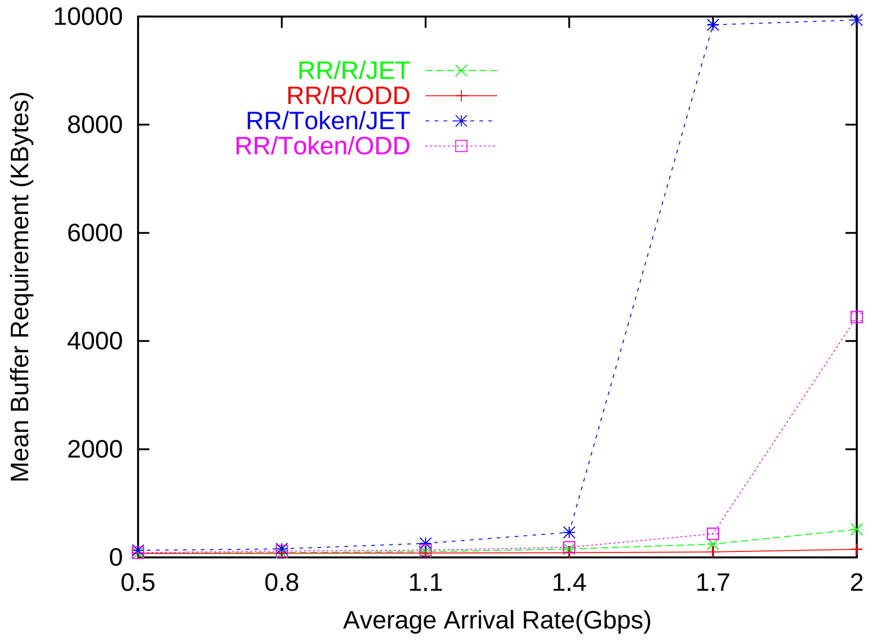

Figure 2.17: JET: Mean buffer requirement vs. average arrival rate

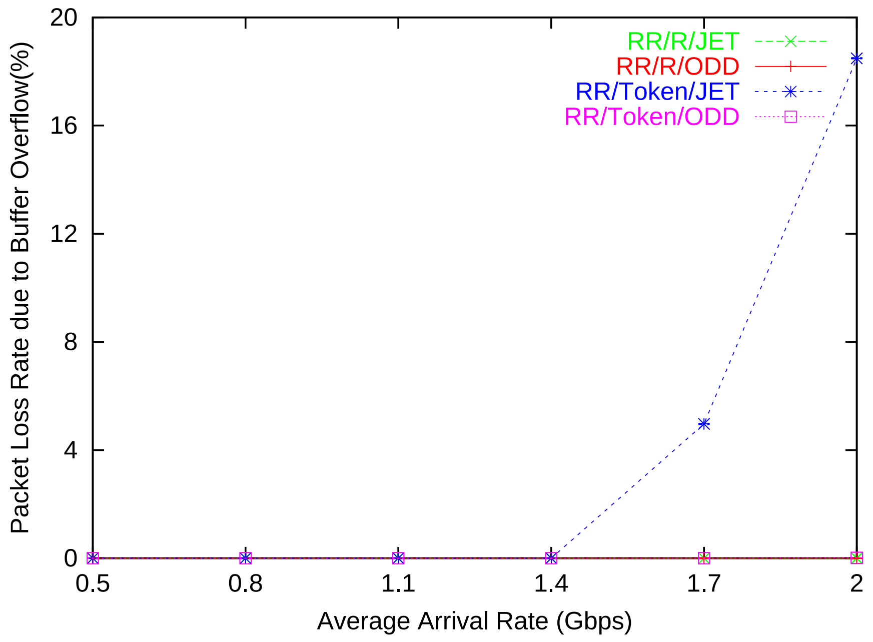

Figure 2.18: JET: Packet loss rate due to buffer overflow vs. average arrival rate

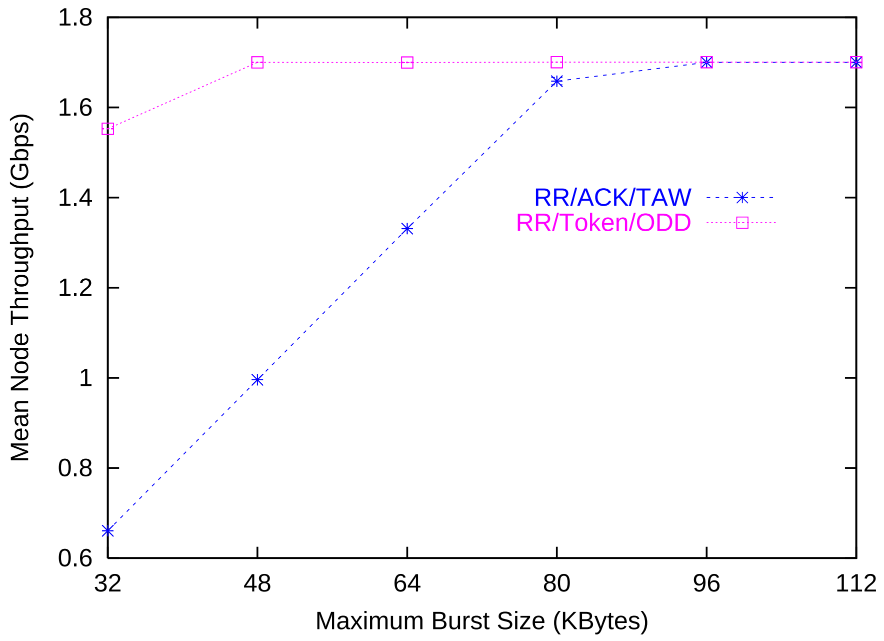

Figure 2.19: TAW: Mean node throughput vs. maximum burst size

Simulation results showed that when the `MaxBurstSize` is small, RR/Token with ODD gets both a higher mean node throughput and a lower mean packet delay than RR/ACK with TAW, as shown in Figure 2.19 and 2.20, respectively. When the `MaxBurstSize` is large, in most cases, both protocols have similar mean node throughput and mean packet delay. However, when both the average arrival rate and the `MaxBurstSize` are very large, RR/ACK gives a higher mean node throughput and lower mean packet delay than RR/Token. As for fairness, RR/ACK is a throughput fair protocol, but not a delay fair protocol.

2.5.4 Asymmetric Traffic

In this section, we investigate the performance of the protocols under asymmetric traffic. We vary the average arrival rate from 0.5 Gbps to 1.3 Gbps with an increment of 0.2 Gbps, and set the c^2 of the packet inter-arrival time to 20, `MaxBurstSize` to 112 Kbytes, `MinBurstSize` to 16 Kbytes, and `TimeOut` to 4 ms. At all nodes except

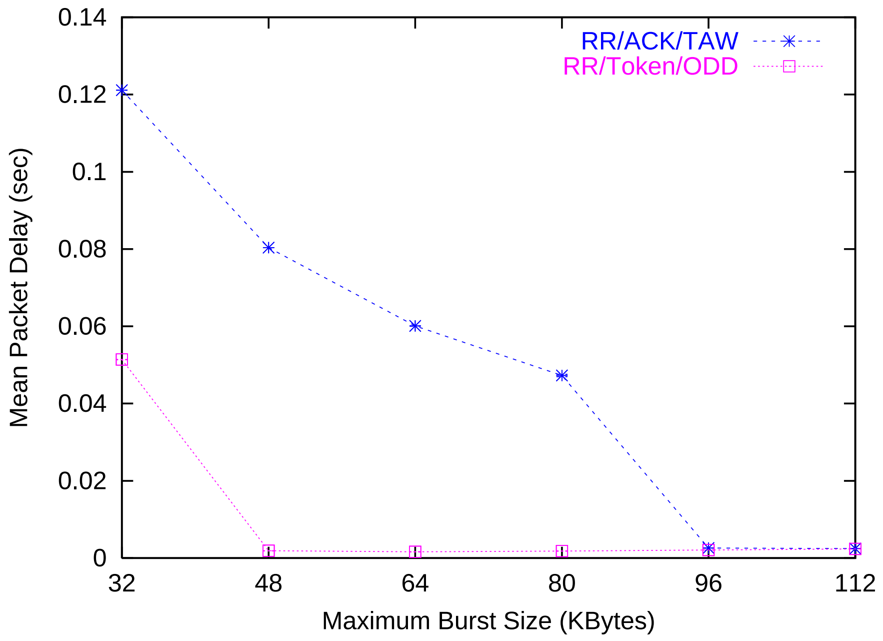

Figure 2.20: TAW: Mean packet delay vs. maximum burst size

node 0, the probability that a packet is destined to node 0 is 1/6, and the probability to other nodes is uniformly distributed. At node 0, a packet is assigned a destination node following the uniform distribution.

Among the four protocols based on ODD, we found that RR/Token achieves the highest mean node throughput, followed by RR/P, RR/NP and RR/R (see Figure 2.21), and RR/R has the smallest mean packet delay, followed by RR/NP, RR/P and RR/Token (see Figure 2.22). We also compared the simulation results of RR/R and RR/Token based on ODD with that of RR/R and RR/Token based on JET, and we found that JET leads to a longer mean packet delay for both protocols. Finally, we found that RR/ACK with TAW achieves the same mean node throughput as RR/Token with ODD, but has a longer mean packet delay than RR/Token with ODD.

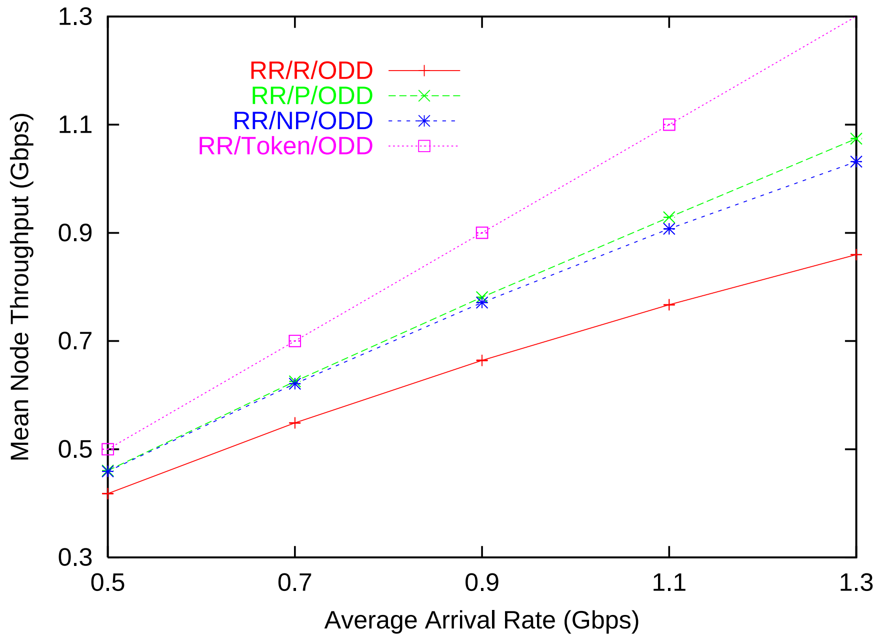

Figure 2.21: ASYM: Mean node throughput vs. average arrival rate

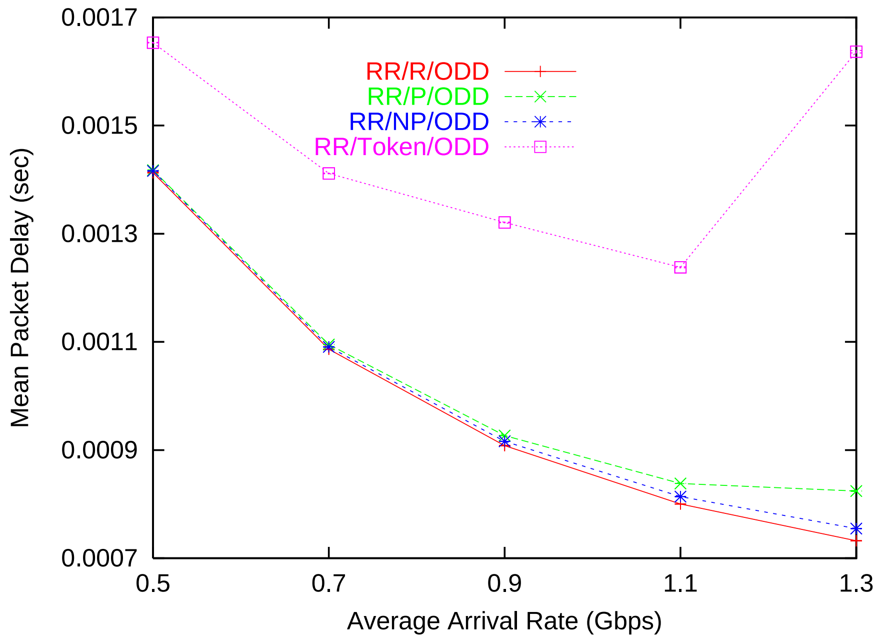

Figure 2.22: ASYM: Mean packet delay vs. average arrival rate

2.6 Concluding Remarks

This chapter described a WDM metro ring architecture with optical burst switching. Several access protocols are proposed and their performance is analyzed by simulation.

Based on our experimentation, we found that, RR/Token achieves the highest mean node throughput, followed by RR/P, RR/NP and RR/R. RR/R has the smallest mean packet delay, followed by RR/NP, RR/P and RR/Token. `MaxBurstSize` affects both the mean node throughput and the mean packet delay, but only a very small `MaxBurstSize` leads to a much lower mean node throughput under RR/Token, and a much longer mean packet delay under RR/P and RR/Token. Increases in `MinBurstSize` lead to an increase in the mean packet delay of all protocols, but do not affect the mean node throughput of RR/Token. We also observed that JET leads to a longer mean packet delay and to a larger packet loss rate due to buffer overflow than ODD. The protocols become more sensitive to `MaxBurstSize` with JET than with ODD. Compared to RR/Token with ODD, RR/ACK with TAW achieves better performances when both the `MaxBurstSize` and the average arrival rate are very large. In the simulations with symmetric traffic, we found that only RR/R is a delay fair protocol, while both RR/R and RR/Token are throughput fair protocols.

Finally, we note that the results were obtained by setting `MaxBurstSize` and `MinBurstSize` to values which are not very small. If we set them to very small values (for example, set `MinBurstSize` to zero), we will get different results. However, very small values of `MaxBurstSize` and `MinBurstSize` are not reasonable, since some optical device overheads (e.g., the setup delay of a node) may greatly degrade the performance of the ring.

Chapter 3

Burst Arrival Process

3.1 Model Description

In most analytical models of OBS networks proposed in the literature [9, 12, 11, 28, 27, 38, 30], it is assumed that the burst arrival process is Poisson. However, it is well-known that the Poisson process is not a good model for wide area traffic [22], and it is unlikely that the burst arrival processes in future optical networks will be accurately characterized by the Poisson model. Another problem is that in the Poisson process, an arrival occurs instantaneous, and the service time required by the arrival is independent of the inter-arrival time of two consecutive bursts. However, in optical burst switching, the service time of a burst is the duration of the burst, which is *not* independent on the inter-arrival time of the bursts. For example, the inter-arrival time of between a very long burst and the following burst must be very long, too. Therefore, more sophisticated models are required in order to advance our understanding of the performance and the potential of OBS networks.

We use the three-state Markov process shown in Figure 3.1 to model arrivals on a given burst wavelength. The arrival process may be in one of three states: `short burst`, `long burst`, or `idle`. If it is in the `short burst` (respectively, `long burst`)

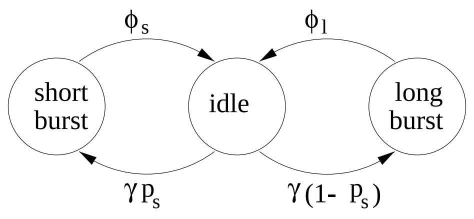

Figure 3.1: The burst arrival process

state, then a short (respectively, long) burst is being transmitted on this wavelength. If it is in the `idle` state, then no burst is being transmitted on this wavelength. The duration of a burst, whether short or long, is assumed to be exponentially distributed. In this model, we assume that the wavelength becomes idle after the transmission of each burst. That is, bursts are not transmitted back-to-back. This assumption can be easily removed by modifying the three-state Markov process in Figure 3.1 to include transitions between the `short burst` and `long burst` states. Also, more complicated burst arrival processes can be modeled by introducing additional states and appropriate transitions between them. For instance, instead of using only two burst lengths (short and long), we may introduce multiple burst lengths, each associated with a different state of the Markov process. Non-exponentially distributed burst lengths can also be accounted for by describing the length of a burst by a Coxian distribution. The analysis of the queueing network model that represents an edge OBS node, which we develop in the following chapters, can be extended in a straightforward manner to include these more general burst arrival processes. However, incorporating more general arrival processes in the model does introduce additional complexities in the expressions we use. Therefore, to keep the analysis simple we only consider the three-state Markov process in Figure 3.1.

The burst arrival process of Figure 3.1 is characterized completely by the following four parameters:

- $1/\gamma$, the mean duration of the `idle` state,
- $1/\phi_s$ and $1/\phi_l$, the mean durations of the `short burst` and `long burst` states, respectively, and

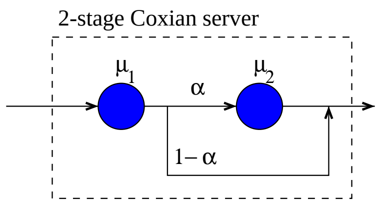

Figure 3.2: 2-stage coxian server

- p_s, the probability that a burst is a small burst

3.2 Burst Duration

Since a burst may be either short or long with probability p_s and $1 - p_s$, respectively, the burst duration distribution is a two-stage hyperexponential distribution. It is well-known that this distribution is equivalent to a two-stage Coxian distribution with the squared coefficient of variation larger than or equal to 1 (see Figure 3.2). We will let μ_1 and μ_2 denote the service rate of the first and second stages of the corresponding Coxian server, respectively, and a denote the probability that, upon completion of the first service stage, the customer in the Coxian server will proceed to the second stage. The values of μ_1, μ_2, and a are uniquely determined by the values of $1/\phi_s, 1/\phi_l$, and p_s as follows [23]. In this thesis, we will work with the Coxian representation of the burst duration.

$$\mu_1 = \phi_s, \quad \mu_2 = \phi_l, \quad a = \frac{(1 - p_s)(\phi_s - \phi_l)}{\phi_s} \tag{3.1}$$

Let B be the random variable denoting the burst duration. The mean $E(B)$ and the squared coefficient of variation $c^2(B)$ of the burst duration are given by the following expressions:

$$E(B) = \frac{p_s}{\phi_s} + \frac{1 - p_s}{\phi_l} \tag{3.2}$$

$$c^2(B) = \frac{2}{E^2(B)} \left(\frac{p_s}{\phi_s^2} + \frac{1 - p_s}{\phi_l^2} \right) - 1 \tag{3.3}$$

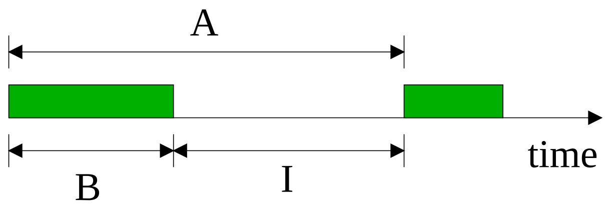

Figure 3.3: Relationship between random variables A, B, and I

Most of the analytical models [9, 28, 27, 38, 30] of OBS networks in the literature assume that the burst duration follows exponential distribution (with $c^2(B) = 1$). However, it is well-known that the packet size is not exponentially distributed, and it is unlikely that the burst duration will be exponentially distributed. Therefore, our model which permits a hyperexponential distribution (with $c^2(B) \geq 1$) is more realistic.

3.3 Burst Interarrival Time

Let A, B, and I be random variables denoting the burst interarrival time, burst duration, and idle time, respectively. Their relationship is shown in Figure 3.3. Let $L_A(s)$, $L_B(s)$, and $L_I(s)$ denote their Laplace transform, respectively. We have that:

$$\begin{aligned} L_B(s) &= \tfrac{\phi_s}{\phi_s+s} + (1-p_s)\tfrac{\phi_l}{\phi_l+s} \\ L_I(s) &= \tfrac{\gamma}{\gamma+s} \\ L_A(s) &= L_B(s)L_I(s) \\ &= \left(p_s\tfrac{\phi_s}{\phi_s+s} + (1-p_s)\tfrac{\phi_l}{\phi_l+s}\right)\tfrac{\gamma}{\gamma+s} \end{aligned} \tag{3.4}$$

By differentiating $L_A(s)$, we obtain the first two moments of the interarrival time A as follows:

$$E[A] = p_s\frac{1}{\phi_s} + (1-p_s)\frac{1}{\phi_l} + \frac{1}{\gamma} \tag{3.5}$$

$$E[A^2] = 2p_s\left(\frac{1}{\phi_s^2} + \frac{1}{\gamma\phi_s} + \frac{1}{\gamma^2}\right) + 2(1-p_s)\left(\frac{1}{\phi_l^2} + \frac{1}{\gamma\phi_l} + \frac{1}{\gamma^2}\right) \tag{3.6}$$

Then, the squared coefficient of variation of the inter-arrival time of successive bursts (short or long), $c^2(A)$, is given by:

$$c^2(A) = \frac{E(A^2)}{E^2(A)} - 1 \tag{3.7}$$

The squared coefficient of variation $c^2(A)$ is a measure of the burstiness of the arrival process. Unlike the Poisson process which is smooth ($c^2(A) = 1$), one may introduce any degree of burstiness into the arrival process of Figure 3.1 by appropriately selecting the parameters of the three-state Markov process. In appendix B, we show how $c^2(A)$ and $c^2(B)$ change with γ, ϕ_s, ϕ_l and p_s.

We note that the inter-arrival times of successive bursts are i.i.d. It is possible to introduce correlation among the inter-arrival times by allowing bursts to arrive back-to-back, as we explained above in section 3.1. In this thesis, we do not consider correlated inter-arrival times.

3.4 Control Parameters

Below, we define three sets of parameters to control the burst arrival process. The first set is just the four parameters described in section 3.1. They are γ, ϕ_s, ϕ_l, and p_s. The second set includes the following parameters.

- l, the load of the burst arrival process, which is the percentage of time that there is a burst (short or long) being transmitted on the wavelength,
- s, the mean duration of a burst,
- r, the ratio of the long burst duration to the short burst duration, and
- p_s, the probability that a burst is a small burst

Given the first set of parameters, the second set of parameters can be obtained as follows.

$$\begin{cases} l &= \frac{p_s/\phi_s+(1-p_s)/\phi_l}{1/\gamma+p_s/\phi_s+(1-p_s)/\phi_l} \\ s &= \frac{ps}{\phi_s}+\frac{1-p_s}{\phi_l} \\ r &= \frac{\phi_s}{\phi_l} \\ p_s &= p_s \end{cases} \tag{3.8}$$

Given the second set of parameters, the first set of parameters can be obtained as follows.

$$\begin{cases} \gamma &= \frac{l}{(1-l)s} \\ \phi_s &= \frac{p_s+(1-p_s)r}{s} \\ \phi_l &= \frac{\phi_s}{r} \\ p_s &= p_s \end{cases} \tag{3.9}$$

The third set includes $E(B)$, $E(A)$, $c^2(A)$, and p_s, where $E(B)$, $E(A)$, and $c^2(A)$ are defined in section 3.2 and section 3.3. Given the third set of parameters, the first set of parameters can be obtained as follows.

$$\begin{cases} \gamma &= \frac{l}{E(A)-E(B)} \\ \phi_s &= \frac{-r_b+\sqrt{{r_b}^2-4r_ar_c}}{2r_a} \\ \phi_l &= \frac{(1-p_s)\phi_s}{\phi_sE(B)-p_s} \\ p_s &= p_s \end{cases} \tag{3.10}$$

where r_a, r_b, and r_c are defined as follows.

$$\begin{cases} r_a &= -c^2(A)E^2(A)(1-p_s)+E^2(A)(1-p_s)-2E(A)E(B)(1-p_s)+2E^2(B) \\ r_b &= -4E(B)p_s \\ r_c &= 2p_s \end{cases} \tag{3.11}$$

and $r_b^2 \geq 4r_ar_c$.

We would like to mention that given the values of $E(B)$, $E(A)$, and $c^2(A)$, then $c^2(B)$ can be uniquely determined by the following equation.

$$c^2(B) = \frac{(c^2(A)+1)E^2(A)-2E(A)(E(A)-E(B))}{E^2(B)}-1 \tag{3.12}$$

We note that the parameters of the first set are independent of each other. For example, the choice of ϕ_l does not depend on the values of γ, ϕ_s, and p_s. The parameters of the second set are also independent of each other. For example, the choice of r does not depend on the values of l, s, and p_s. However, the parameters of the third set are dependent of each other. For example, the choice of $c^2(A)$ depends on the values of $E(B)$, $E(A)$, and p_s. For instance, if $E(B) = 1$, $E(A) = 1.2$, and $p_s = 0.9$, then $c^2(A)$ must be smaller than 14.

Chapter 4

An Edge Node of an OBS Mesh Network

4.1 Introduction

In this chapter, we develop for the first time a queueing network model of an edge OBS node using the burst arrival process described in the previous chapter. The edge OBS node serves a number of users, each connected to the OBS node by a fiber link, which can support multiple wavelengths. Each wavelength is associated with a separate burst arrival process. We consider an OBS edge node both with and without converters, and we model it as a closed non-product-form queueing network, which we analyze by decomposition. We develop algorithms for both the single-class and multi-class cases; in the former, all wavelengths have the same arrival process, while in the latter, each wavelength has a different arrival process. In particular, we present a new computationally efficient method for analyzing multi-class queueing networks by decomposing them into a set of two-class networks. This method is not limited to the model at hand, and it can be applied to general multi-class queueing networks.

Finally, we use our algorithms to gain new insight into the performance of an edge OBS node.

Following this introduction, we describe briefly the operation of an edge OBS node in section 4.2. In section 4.3, we describe a queueing network model of the edge OBS node. Sections 4.4 and 4.5 describe a method for analyzing this queueing network without and with wavelength converters, respectively, assuming a single-class of customers. That is, all the wavelengths are associated with the same burst arrival process. In section 4.6, we describe a decomposition method for analyzing a multi-class generalization of this queueing network. In section 4.7, we consider a number of limiting cases, for which we can obtain simple analytic expressions. We validate the accuracy of the algorithm in section 4.8 by comparing it to simulation results, and we conclude the chapter in section 4.9.

4.2 The Edge OBS Node

In this chapter, we model an edge OBS node employing the `JumpStart` JIT signaling protocol. The `JumpStart` project [1] is a joint NCSU/MCNC research effort addressing the design, specification, performance evaluation, and hardware implementation of a signaling protocol for OBS networks. The signaling protocol follows the just-in-time (JIT) approach, and is based on the work by Wei and McFarland [30]. We now describe the aspects of the `JumpStart` signaling protocol that are necessary for modeling an edge OBS node; for the full details, the interested reader is referred to [4].

We consider an OBS network consisting of OBS nodes (switches) interconnected by bidirectional fiber links, as shown in Figure 4.1. Each fiber link between a user and an OBS edge node, or between two adjacent OBS nodes, can support $W+1$ wavelengths. Of these, one wavelength (referred to as *control wavelength*) is used to transmit control packets, and the other W wavelengths (referred to as *burst wavelengths*) are used to transmit data bursts. A user is equipped with $W+1$ pairs of optical transceivers, each fixed tuned to one of the $W+1$ wavelengths. (Alternatively, a user may be

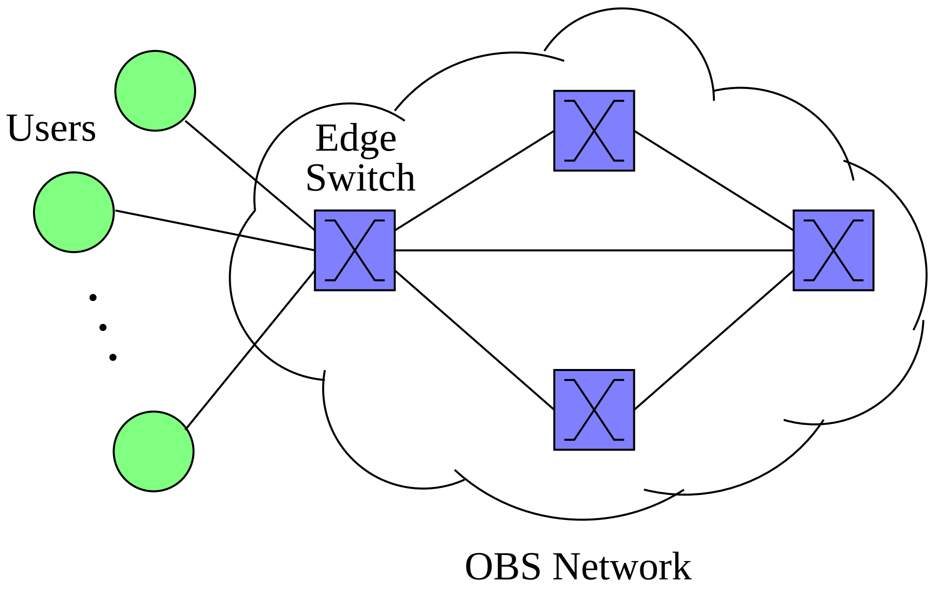

Figure 4.1: Users connected to an edge switch of an OBS network

equipped with fewer than $W + 1$ pairs of tunable or fixed transceivers; in this case, however, it is not possible for a user to access all wavelengths at the same time.)

Following the `JumpStart` JIT signaling protocol [4], a user first sends a `setup` message to its edge OBS node. The `setup` message includes the source and destination addresses, the wavelength on which the source prefers to transmit the burst, and other information. We assume that an OBS node consists of a non-blocking space-division switch fabric, with no optical buffers. If the edge node can switch the burst on the specified wavelength, it returns a `setup ack` message to the user. The `setup ack` message contains the offset field that informs the user how long it should wait before transmitting its burst. It is possible, however, that a `setup` message be refused if the preferred wavelength on the destination output port is busy, or in the case of full wavelength converters, if all the wavelengths on the destination output port are busy. In this case, the edge node returns a `reject` message. The user goes through a random delay, and it then re-transmits the `setup` message. In our model, we assume that the user continues to re-transmit the `setup` message until it receives a `setup ack` message, although this assumption can be easily removed.

In order to reduce the user delay in the case of an edge node with no wavelength converters, the `JumpStart` signaling protocol provides for an alternative way to send a burst. When an edge node receives a `setup` message, and the wavelength indicated by the user is busy on the destination port, it then checks whether other wavelengths are free on the destination port. If so, the edge node, rather than returning a `reject` message, it returns a `setup ack` message with which it informs the user to transmit its burst on one of the free wavelengths. Note that, if we assume that the time for a wavelength converter to convert an optical signal to another wavelength is negligible (compared to the propagation delay and the burst duration), then the throughput of this method is exactly the same as that of the previous method under the assumption that the edge node is equipped with wavelength converters. Therefore, we only model the previous method in this paper.

We assume that the node allocates resources within its switch fabric for a burst at the moment that it decides to accept the `setup` message. An alternative approach is to allocate the necessary resources near the time the edge node expects the burst to arrive. Assuming that the estimate regarding the burst arrival time is accurate, the latter approach minimizes the holding time of the resource for a burst. On the other hand, it also requires a complex scheduling algorithm, as well as significant amounts of memory on the signaling board that processes the OBS signals [4, 19]. Therefore, in `JumpStart` we have decided to follow the former approach, which is the one we model in this work.

Another design issue is related to the time when the node frees the resources allocated to a burst. One way of resolving this problem is for the source to indicate the length of the transmission of the burst in the `setup ack` message. Assuming that the node knows when the burst will start to arrive, it can then calculate the time when it will free its resources. Alternatively, the user does not communicate to the edge node the length of its burst, but it simply sends a `release` message to the node to indicate the end of its transmission. Upon receipt of the `release` message, the node frees the resources allocated to the burst. The latter solution seems to be easier to implement, but it gives rise to a larger number of signaling messages. Our model can take into account either method, due to the inherent abstractions in the

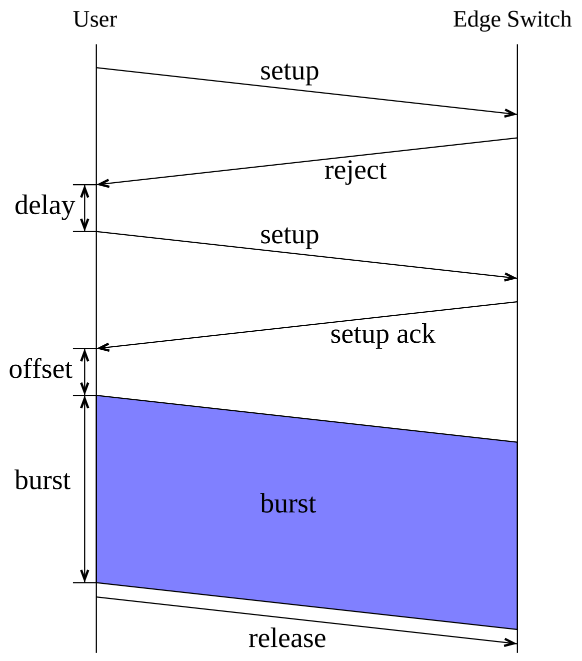

Figure 4.2: Signaling messages in `JumpStart`

underlying queueing network.

The sequence of messages exchanged between a user and its edge node is shown in Figure 4.2. A user can be seen as being in one of the following three states: **(1)** *idle*, i.e., no bursts to transmit; **(2)** *busy* transmitting a burst; or **(3)** *blocked*, i.e., undergoing a delay before it re-transmits a `setup` message. If a user can simultaneously transmit bursts on different wavelengths, then it can be in a different state for each burst wavelength.

4.3 A Queueing Network Model of the Edge OBS Node

As shown in Figure 4.1, an edge OBS node is connected to a number of users and to a number of other OBS nodes. Consequently, it receives bursts both from users and other OBS nodes. In this work, we assume that there is no burst traffic from other OBS nodes to the edge OBS node, and we only consider the burst traffic from the users to the edge node [1]. Let P and N denote the number of input (or output) ports of an edge node and the number of the users connected to the edge node, respectively. Note that, $P \geq N$.

The traffic on each incoming wavelength from a user to the edge node is generated by the burst arrival process described in chapter 3. Since each user can simultaneously transmit bursts on all its W burst wavelengths, the user is associated with W different burst arrival processes. Therefore, an edge node with N users has a total of NW burst arrival processes. Recall that a `setup` message is refused if, at the time it arrives at the edge node, the output port is busy transmitting another burst. In this case, the corresponding burst arrival process undergoes an exponential delay, and then the user re-transmits its `setup` message. Thus, at any time, there may be a number of burst arrival processes undergoing an exponential delay for each output wavelength.

4.3.1 The Edge OBS Node without Converters

Let us first consider an edge OBS node with no converters. In this case, a burst on an incoming wavelength can only be switched to the *same* wavelength on each output port, and user bursts arriving to the edge switch on different wavelengths do not interfere with each other. Consequently, the edge node can be decomposed exactly into W sub-systems, one per burst wavelength. Each sub-system $w, w = 1, \cdots, W$, is a $P \times P$ switch with N users, but each input and output port has a single wavelength, which corresponds to wavelength w of the original edge switch. Therefore, each sub-system has N burst arrival processes.

The queueing network model of a sub-system is shown in Figure 4.3; it consists of

[1]This is a reasonable assumption for an edge OBS node. Most traffic from other OBS nodes to the edge node is in the direction *from* the OBS network *to* the users, while in this work we are interested in modeling the performance of an edge node in the direction *from* the users *to* the OBS network. We are currently extending the techniques we present in this work to analyze a network of OBS nodes in order to investigate the performance of burst traffic as it travels along an end-to-end path of OBS nodes.

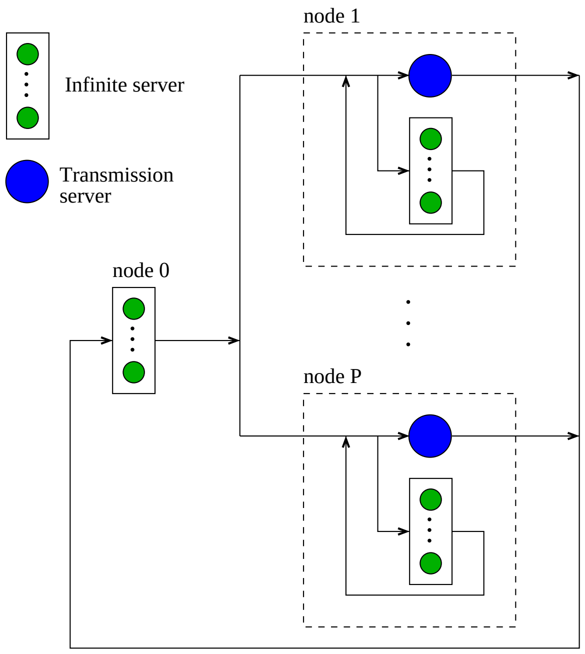

Figure 4.3: Queueing network model of a sub-system of an edge switch without converters

$P+1$ nodes numbered $0, 1, \cdots, P$. Node 0 is an infinite server node, and it represents the burst arrival processes which are in the `idle` state. Node $i, i = 1, \cdots, P$, represents the (single) wavelength on output port i. Each node i consists of a single *transmission server* and an *infinite server*. The customer (if any) occupying the transmission server represents the burst arrival process whose burst is being transmitted by output port i. The customers (if any) in the infinite server represent those burst arrival processes which are undergoing a delay before their users re-transmit the corresponding `setup` messages. The total number of customers in this closed queueing network model of a sub-system is equal to N (i.e., it is equal to the total number of burst arrival processes in the sub-system).

Let us now follow the path of a customer through the queueing network model in Figure 4.3. Let us assume that the customer starts in the `idle` state, i.e., it is in node 0. The time it spends in the `idle` state is exponentially distributed with mean $1/\gamma$. Upon completion of its service at node 0, it moves to node i with probability p_i; this corresponds to the transmission of a `setup` message for a burst with output port i. If the single transmission server at node i is free, the customer enters service immediately. The service time is exponentially distributed with a mean of $1/\phi_s$ or $1/\phi_l$ with probabilities p_s or $1-p_s$, corresponding to the transmission of short or long burst, respectively. If the transmission server is busy (i.e., output port contention occurs), the customer enters the infinite server at node i, where it undergoes an exponential delay with mean $1/\omega$; this delay models the delay until the retransmission of the `setup` message. Upon completion of the exponential delay, the customer again tries to seize the transmission server. If the transmission server is busy, the customer joins the infinite server again, and it undergoes another delay, and so on, until it succeeds to get hold of the transmission server. The customers in the infinite server are often referred to in the literature as *orbiting* customers. Note that it is possible for the transmission server to become idle while there are one or more customers orbiting. In this case, it is possible that a new customer arrives from node 0 and starts service immediately.

In the case where all N customers have the same burst arrival process, the closed queueing network model can be seen as consisting of a single class of N customers and $P+1$ nodes. If each customer has a different burst arrival process, then the queueing network becomes a multi-class queueing network with $P+1$ nodes and N classes, where each class contains exactly one customer.

4.3.2 The Edge OBS Node with Converters

Let us now consider an edge OBS switch with converters. In this case, a `setup` message for output port i of the switch is accepted as long as at least one wavelength is free on this output port. Otherwise, the `setup` message is rejected, and the user undergoes a delay before retransmitting the message. Clearly, the above decomposition of an edge switch into sub-systems per wavelength is no longer possible, since user bursts arriving on different wavelengths may interfere with each other. However, the edge switch *as a whole* can be modeled by a closed queueing network very similar to the one shown in Figure 4.3. The new queueing network consists of $P+1$ nodes and a total of NW customers (since there are now NW arrival processes). Node 0

Parameter	**Description**
N	number of users connected to an edge switch
P	number of input (output) ports of an edge switch
W	number of burst wavelengths in a fiber
$1/\gamma$	mean duration of the `idle` state
$1/\phi_s$	mean duration of the short burst
$1/\phi_l$	mean duration of the long burst
$1/\omega$	mean orbiting time of a user
p_s	probability that a burst is a small burst
p_i	probability that the destination output port of a burst is i
$E(B)$	mean duration of a burst (short or long)
μ_1, μ_2, a	parameters of the 2-stage Coxian distribution of the burst size

Table 4.1: Notation used in the analysis

in the new queueing network is identical to node 0 in the network of Figure 4.3. Similarly, each node $i, i = 1, \cdots, P$, in the new queueing network corresponds to each of the output ports of the edge switch. The main difference is that each node $i, i = 1, \cdots, P$, consists of an infinite server and W (rather than one) transmission servers, each corresponding to one of the W wavelengths of output port i.

In the following sections, we describe a technique for solving the queueing network in Figure 4.3. We note that, despite the rich literature in queueing network analysis, this particular queueing network with orbiting customers has not been analyzed before. The notation used in the analysis is summarized in Table 4.1. In section 4.4, we analyze the queueing network assuming a single class of customers (i.e., all burst arrival processes are identical) and no wavelength converters. In section 4.5, we analyze the network assuming a single class of customer and wavelength converters. Finally, the analysis of the multi-class network is presented in section 4.6.

4.4 Analysis of the Single-Class Queueing Network w/o Converters

The queueing network shown in Figure 4.3 is a non-product-form queueing network with Coxian service times. We analyze it using Marie's algorithm [17, 18]. The idea in Marie's method is to replace each non-BCMP node by an equivalent node with a load-dependent exponential service rate, obtained by calculating the conditional throughput of the non-BCMP node in isolation under a load-dependent arrival rate. In the following subsection, we calculate the conditional throughput of each node i, $i = 1, \cdots, P$. Node 0 is an infinite server that is a BCMP node, so we do not need to construct a flow equivalent node for it. We note that, to the best of our knowledge, Marie's method has not been applied to nodes with orbiting customers. Consequently, the derivation in the next subsection of a flow equivalent server for such a node is a new contribution.

4.4.1 The Flow Equivalent Server

Let us consider node $i, i = 1, \cdots, P$, of the queueing network shown in Figure 4.3. Let $\lambda_i(n_i)$ be the arrival rate into this node when there are a total of n_i customers in the node. We also assume that the service time is a two-stage Coxian distribution with parameters $\mu_{(i,1)}$, $\mu_{(i,2)}$, and a_i. The state of node i can be described by the triplet: $(n_i^{(t)}$, k_i, $n_i^{(o)})$, where $n_i^{(t)} = 0, 1$, indicates whether the transmission server is busy or not, $k_i = 0, 1, 2$, is the Coxian phase of the transmission server ($k_i = 0$ if and only if $n_i^{(t)} = 0$), and $n_i^{(o)} = 0, 1, \cdots, N - 1$, gives the number of orbiting customers occupying the infinite server. The state transition diagram of node i is shown in Figure 4.4. In order to simplify the notation, and since we are only concerned with the analysis of node i in isolation, we drop the index i in Figure 4.4 and throughout the rest of this subsection.

Let $p(n^{(t)}, k, n^{(o)})$ be the steady-state probability of the state $(n^{(t)}, k, n^{(o)})$. From

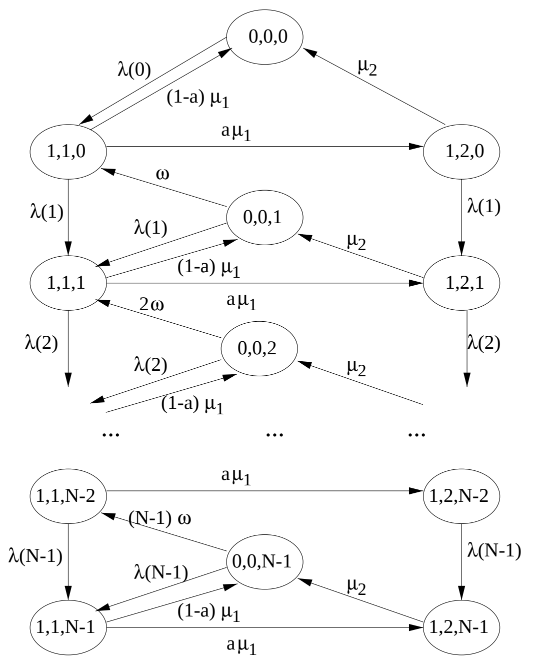

Figure 4.4: State transition rate diagram of node i, $i = 1, \cdots, P$, of the queueing network of Figure 4.3

Figure 4.4, we have the following global balance equations:

$$p(0,0,n)(\lambda(n)+n\omega) = p(1,1,n)(1-a)\mu_1 + p(1,2,n)\mu_2, \qquad 0 \le n < N \tag{4.1}$$

$$p(1,1,n-1)(\lambda(n)+\mu_1) = p(0,0,n)n\omega + p(1,1,n-2)\lambda(n-1) + p(0,0,n-1)\lambda(n-1), \quad 0 < n \le N \tag{4.2}$$

$$p(1,2,n-1)(\lambda(n)+\mu_2) \;=\; p(1,1,n-1)a\mu_1 + p(1,2,n-2)\lambda(n-1), \quad 0 < n \le N \tag{4.3}$$

Let $p(n)$ denote the steady-state probability that there are a total of n customers in the node. We have that:

$$p(n) = \begin{cases} p(0,0,0) & n = 0 \\ p(0,0,n) + p(1,1,n-1) + p(1,2,n-1) & 0 < n < N \\ p(1,1,N-1) + p(1,2,N-1) & n = N \end{cases} \tag{4.4}$$

Let $v(n)$ denote the conditional throughput of the node, calculated as follows:

$$v(n) \;=\; \begin{cases} 0 & n = 0 \\ \frac{p(1,1,n-1)}{p(n)}(1-a)\mu_1 + \frac{p(1,2,n-1)}{p(n)}\mu_2 & n > 0 \end{cases} \tag{4.5}$$

We now have the following two theorems:

Theorem 1 $p(n-1)\lambda(n-1) \;=\; p(n)\upsilon(n), \qquad 0 < n \leq N$

Proof: By adding equations (4.1), (4.2), and (4.3) together, and after simplifying the result using equations (4.4) and (4.5), we obtain:

$$\begin{aligned} p(0)\lambda(0) &= p(1)\upsilon(1), \quad n = 0 && (4.6) \\ p(n)\lambda(n) + p(n)\upsilon(n) &= p(n-1)\lambda(n-1) + p(n+1)\upsilon(n+1), \quad 0 < n < N && (4.7) \\ p(N)\upsilon(N) &= p(N-1)\lambda(N-1), \; n = N && (4.8) \end{aligned}$$

Using equations (4.6) and (4.7) recursively, we finally get:

$$p(n-1)\lambda(n-1) = p(n)\upsilon(n), \quad 0 < n \leq N \tag{4.9}$$

completing the proof of the theorem. ■

Theorem 2 *The conditional throughput $\upsilon(n)$ of the node is given by the expression:*

$$\upsilon(n) \;=\; \begin{cases} 0, & n = 0 \\ \frac{\mu_1\omega(\lambda(1)-a\lambda(1)+\mu_2)}{(\lambda(1)+\omega)(\lambda(1)+a\mu_1+\mu_2)}, & n = 1 \\ \frac{n\mu_1\omega(\lambda(n-1)+(n-1)\omega)(\lambda(n)-a\lambda(n)+\mu_2)}{(\lambda(n)+n\omega)((n-1)\omega(\mu_1+\mu_2+\lambda(n)-\upsilon(n-1))+\lambda(n-1)(a\mu_1+\mu_2+\lambda-\upsilon(n-1)))}, & \textit{otherwise} \end{cases} \tag{4.10}$$

Proof: By means of expression (4.5), we can rewrite expression (4.1) as follows:

$$\begin{aligned} p(0,0,n)(\lambda(n)+n\omega) &= p(1,1,n)(1-a)\mu_1 + p(1,2,n)\mu_2 \\ &= p(n+1)\upsilon(n+1) \\ &= p(n)\lambda(n) && (4.11) \end{aligned}$$

We can also rewrite expression (4.2) as:

$$\begin{aligned} p(1,1,n-1)(\lambda(n)+\mu_1) &= p(0,0,n)n\omega + p(1,1,n-2)\lambda(n-1) + p(0,0,n-1)\lambda(n-1) \\ &= p(0,0,n)n\omega + \frac{p(1,1,n-2)+p(0,0,n-1)}{p(n-1)}p(n-1)\lambda(n-1) \\ &= p(0,0,n)n\omega + \frac{p(1,1,n-2)+p(0,0,n-1)}{p(n-1)}p(n)\upsilon(n) && (4.12) \end{aligned}$$

Using expressions (4.4), (4.5), (4.11), and (4.12), we get the following group of equations for $n > 1$:

$$\begin{cases} p(n) & = & p(0,0,n) + p(1,1,n-1) + p(1,2,n-1) \\ p(n-1) & = & p(0,0,n-1) + p(1,1,n-2) + p(1,2,n-2) \\ \upsilon(n) & = & \frac{p(1,1,n-1)}{p(n)}(1-a)\mu_1 + \frac{p(1,2,n-1)}{p(n)}\mu_2 \\ \upsilon(n-1) & = & \frac{p(1,1,n-2)}{p(n-1)}(1-a)\mu_1 + \frac{p(1,2,n-2)}{p(n-1)}\mu_2 \\ p(0,0,n)(\lambda(n) + n\omega) & = & p(n)\lambda(n) \\ p(0,0,n-1)(\lambda(n-1) + (n-1)\omega) & = & p(n-1)\lambda(n-1) \\ p(1,1,n-1)(\lambda(n) + \mu_1) & = & p(0,0,n)n\omega + \frac{P(1,1,n-2)+p(0,0,n-1)}{p(n-1)}p(n)\upsilon(n) \end{cases} \tag{4.13}$$

Now, assuming that $p(0,0,n)$, $p(0,0,n-1)$, $p(1,1,n-1)$, $p(1,1,n-2)$, $p(1,2,n-1)$, $p(1,2,n-2)$, and $\upsilon(n)$ are unknown variables, we can solve the group of equations (4.13) to obtain the following expression for $\upsilon(n)$ $(n > 1)$:

$$\upsilon(n) = \frac{n\mu_1\omega(\lambda(n-1) + (n-1)\omega)(\lambda(n) - a\lambda(n) + \mu_2)}{(\lambda(n) + n\omega)((n-1)\omega(\mu_1 + \mu_2 + \lambda(n) - \upsilon(n-1)) + \lambda(n-1)(a\mu_1 + \mu_2 + \lambda - \upsilon(n-1)))} \tag{4.14}$$

Similarly, we can solve the following group of equations for $p(0,0,1)$, $p(1,1,0)$, $p(1,2,0)$, and $\upsilon(1)$:

$$\begin{cases} p(1) & = & p(0,0,1) + p(1,1,0) + p(1,2,0) \\ \upsilon(1) & = & \frac{p(1,1,0)}{p(1)}(1-a)\mu_1 + \frac{p(1,2,0)}{p(1)}\mu_2 \\ p(0,0,1)(\lambda(1) + \omega) & = & p(1)\lambda(1) \\ p(1,1,0)(\lambda(1) + \mu_1) & = & p(0,0,1)\omega + p(1)\upsilon(1) \end{cases} \tag{4.15}$$

to obtain the following expression for $\upsilon(1)$:

$$\upsilon(1) = \frac{\mu_1\omega(\lambda(1) - a\lambda(1) + \mu_2)}{(\lambda(1) + \omega)(\lambda(1) + a\mu_1 + \mu_2)} \tag{4.16}$$

Since $\upsilon(0)$ is obviously equal to 0, the proof is complete. ■

We use the conditional throughput $\upsilon(n)$ as the load-dependent service rate $\mu(n)$ of the node in the iterative algorithm described in the next subsection.

We use the following approximate expressions to calculate the parameters of the transmission servers, since they are consistent with expression (4.33) used for the multi-class cases of this queueing network.

$$\begin{cases} \mu_1 & = & \frac{2}{E(B)} \\ \mu_2 & = & \frac{1}{E(B)c^2(B)} \\ \alpha & = & \frac{1}{2c^2(B)} \end{cases} \tag{4.17}$$

4.4.2 The Iterative Algorithm

We use Marie's algorithm [17, 18] to analyze the queueing network of Figure 4.3. The main steps of the algorithm are summarized as follows:

- **Step 1.** Initialize the service rate $\mu_i(n_i)$ of flow equivalent server $i, i = 1, \cdots, P$, to $1/E(B)$, for $n_i > 0$, and set the service rate $\mu_0(n_0)$ of flow equivalent server 0 to γn_0.
- **Step 2.** For each node $i, i = 1, \cdots, P$, do the following steps:
  - **Step 2.1.** Calculate the arrival rate $\lambda_i(n_i)$ of node i by short-circuiting node i in the substitute product-form closed queueing network, where each node j has an exponential service time of $\mu_j(n_j)$.
  - **Step 2.2.** Calculate the conditional throughput $v_i(n_i)$ of node i using Theorem 2.
  - **Step 2.3.** Calculate the steady-state probability $p_i(n_i)$ of node i using Theorem 1.
- **Step 3.** Check the following two convergence conditions. If both are satisfied, then stop. Otherwise, set $\mu_i(n_i)$ to $v_i(n_i)$ for all $i = 1, \cdots, P$, and go back to Step 2.
  1. The first convergence condition ensures that the sum of the mean number of customers at all nodes is equal to the number of customers in the queueing

network:

$$\left| \frac{N - \sum_{i=0}^{P} \sum_{j=0}^{N} j p_i(j)}{N} \right| < \epsilon \tag{4.18}$$

2. The second convergence condition makes sure that the conditional throughputs of each node are consistent with the topology of the queueing network:

$$\left| \frac{r_i - \frac{1}{P+1} \sum_{j=0}^{P} r_j}{\frac{1}{P+1} \sum_{j=0}^{P} r_j} \right| < \epsilon, \qquad i = 0, 1, \cdots, P \tag{4.19}$$

where

$$r_i = \begin{cases} \frac{1}{p_i} \sum_{j=0}^{N} p_i(j) \mu_i(j), & i = 1, \cdots, P \\ \sum_{j=0}^{N} p_i(j) \mu_i(j), & i = 0 \end{cases} \tag{4.20}$$

4.5 Analysis of the Single-Class Queueing Network with Converters

As we discussed in section 4.3, the only difference between the queueing network model of an edge switch with wavelength converters and the queueing network of Figure 4.3 (which models a single wavelength sub-system of an edge switch without wavelength converters) is that, in the former model, each node $i, i = 1, \cdots, P$, has W transmission servers, while in the latter model each node i has a single transmission server. Unfortunately, when each node i has multiple transmission servers, we cannot obtain a closed-form solution for the conditional throughput of the node. Instead, we solve each node i numerically using the Gauss-Seidel method [23], to get $p_i(n_i)$, the steady probability that node i has n_i customers. Then, we calculate the conditional throughput $\upsilon_i(n_i)$ as follows:

$$\upsilon_i(n_i) = \frac{p_i(n_i - 1) \lambda_i(n_i - 1)}{p_i(n_i)} \tag{4.21}$$

Finally, we use the same iterative algorithm described in section 4.4.2, to analyze this more general queueing network.

4.6 Analysis of the Multi-Class Queueing Network with or w/o Converters

In this section, we extend our analysis of the queueing network model of the edge OBS switch to the case where each customer has a different burst arrival process. This is taken into account by associating each customer with a different class. The resulting queueing network is a closed non-product-form queueing network with multiple classes, each of which has only a single customer. The number of classes is $C = NW$, where N is the number of users connected to the edge OBS node and W is the number of wavelengths per fiber. We note that, for realistic values of N and W, the number of classes can be very large (i.e., in the order of 100s).

This type of multi-class closed non-product-form has been studied in the literature [21, 5], mainly by extending Marie's algorithm [17]. Neuse and Chandy [21] proposed an algorithm called the *heuristic aggregation method* (HAM) to solve such a queueing network. HAM is a natural extension of Marie's method [17], but it involves two types of time-consuming computations which limit its applicability to networks with a very small number of classes only. First, it involves the numerical analysis of a node with multi-class load-dependent arrivals and a two-stage Coxian service time. Second, it involves the computation of the normalization constant in a multi-class queueing network. Baynat and Dallery [5], presented an alternative extension of Marie's method to multi-class queueing networks. Specifically, to avoid the computation of the normalization constant of a multi-class network, they decompose a C-class network into C single-class networks. The interaction of the customers in different classes is taken into account in the analysis of each node in isolation. They also proposed a class aggregation technique that reduces significantly the complexity of the analysis of a node. However, the arrival rate to the aggregate class is calculated by aggregating the arrival rate to each individual class [5, Equation (14)]. This aggregation process takes time that increases exponentially with the number of the classes. Consequently, while Baynat and Dallery's method is much faster than Neuse and Chandy's HAM, it still cannot be used in networks with a large number of classes.

Based on the above observations, we first show in section 4.6.1 how to use HAM to solve networks with only two classes of customers, and in section 4.6.2 we present a new method for solving queueing networks with more than two classes of customers. Specifically, we decompose a network with multiple classes of customers into a set of two-class networks, each of which is solved using HAM. We also employ a class aggregation technique to reduce the complexity of the analysis of a node. However, we use the convolution algorithm [23] to calculate the arrival rate to the aggregate class, not Baynat and Dallery's method [5], since, as we mentioned above, the latter is not scalable.

4.6.1 The Two-Class Queueing Network

In this subsection, we assume that there are only two classes of customers in the queueing network of Figure 4.3, namely, class 1 and 2. Also, we assume that each node $i, i = 1, \cdots, P$, of the queueing network consists of $W \geq 1$ transmission servers. Therefore, the analysis applies to edge OBS switches with converters ($W > 1$) or without ($W = 1$).

We first construct a flow equivalent node for each node $i, i = 1, \cdots, P$, of the queueing network in Figure 4.3. Let $\lambda_i^{(1)}(n_i^{(1)}, n_i^{(2)})$ and $\lambda_i^{(2)}(n_i^{(1)}, n_i^{(2)})$ denote the arrival rate of class 1 and class 2 customers, respectively, to node i when there are $n_i^{(1)}$ class 1 customers and $n_i^{(2)}$ class 2 customers in the node. We also assume that the service time of class j, $j = 1, 2$, at the transmission server is a two-stage Coxian distribution with parameters $\mu_{(i,1)}^{(j)}$, $\mu_{(i,2)}^{(j)}$, and $a_i^{(j)}$. The state of node i can be described by the vector:

$$(n_i^{(t1,1)}, n_i^{(t2,1)}, n_i^{(o,1)}, n_i^{(t1,2)}, n_i^{(t2,2)}, n_i^{(o,2)}) \tag{4.22}$$

where

- $n_i^{(t1,1)}$ and $n_i^{(t1,2)}$ are random variables representing the number of class 1 and class 2 customers, respectively, being served by the transmission servers in phase one,

- $n_i^{(t2,1)}$ and $n_i^{(t2,2)}$ are random variables representing the number of class 1 and class 2 customers, respectively, being served in phase two, and
- random variables $n_i^{(o,1)}$ and $n_i^{(o,2)}$ represent the number of orbiting customers of class 1 and 2, respectively.

Let $p_i(n_i^{(t1,1)}, n_i^{(t2,1)}, n_i^{(o,1)}, n_i^{(t1,2)}, n_i^{(t2,2)}, n_i^{(o,2)})$ be the steady-state probability that node i is in state $(n_i^{(t1,1)}, n_i^{(t2,1)}, n_i^{(o,1)}, n_i^{(t1,2)}, n_i^{(t2,2)}, n_i^{(o,2)})$. We use the Gauss-Seidel method to calculate the steady-state probability numerically. We then obtain the conditional throughput $\upsilon_i(n_i)$ of node i as:

$$\upsilon_i(n_i) = \frac{p_i(n_i-1)\lambda_i(n_i-1)}{p_i(n_i)} \tag{4.23}$$

where $p_i(n_i)$ and $\lambda_i(n_i)$ are calculated by:

$$p_i(n_i) = \sum_{n_i^{(t1,1)}+n_i^{(t2,1)}+n_i^{(o,1)}+n_i^{(t1,2)}+n_i^{(t2,2)}+n_i^{(o,2)}=n_i} p_i(n_i^{(t1,1)}, n_i^{(t2,1)}, n_i^{(o,1)}, n_i^{(t1,2)}, n_i^{(t2,2)}, n_i^{(o,2)}) \tag{4.24}$$

$$\lambda_i(n_i) = \frac{1}{p_i(n_i)} \sum_{n_i^{(1)}+n_i^{(2)}=n_i} p_i(n_i^{(t1,1)}, n_i^{(t2,1)}, n_i^{(o,1)}, n_i^{(t1,2)}, n_i^{(t2,2)}, n_i^{(o,2)})[\lambda_i^{(1)}(n_i^{(1)}, n_i^{(2)})+\lambda_i^{(2)}(n_i^{(1)}, n_i^{(2)})] \tag{4.25}$$

where $n_i^{(1)} = n_i^{(t1,1)} + n_i^{(t2,1)} + n^{(o,1)}$, and $n_i^{(2)} = n_i^{(t1,2)} + n_i^{(t2,2)} + n_i^{(o,2)}$.

The conditional throughput $\upsilon_i(n_i)$ is used as the load-dependent service rate $\mu_i(n_i)$ of the flow equivalent server of node i.

We solve this two-class product-form network consisting of the flow equivalent servers using the convolution algorithm to obtain the arrival rates $\lambda_i^{(1)}(n_i^{(1)}, n_i(2))$ and $\lambda_i^{(2)}(n_i^{(1)}, n_i^{(2)})$ to each node $i, i = 1, \cdots, P$. This process is repeated until convergence, as dictated by Marie's algorithm described in section 4.4.2.

4.6.2 An Iterative Algorithm for Analyzing More Than Two Classes

As we observed above, the complexity of HAM increases exponentially with the number of classes, thus it can only be applied to networks with a small number of classes. We now introduce a new method for solving queueing networks with a large number of classes. The main idea of our algorithm is to approximate the original multi-class network with a set of two-class networks, each of which is solved using HAM. Below, we first describe a mechanism for aggregating a number of classes into a single class, and subsequently we describe an iterative algorithm for analyzing multi-class queueing networks.

Class Aggregation

Let C denote the number of classes in a network, $C = NW$. For a network with C classes, we create C two-class networks. For each two-class network $c, c = 1, \cdots, C$, the first class is class c in the original network, and the second class is the aggregate class of all the other classes in the original network. For the aggregate class, we have to specify the branching probability $p_i^{(agg)}$ that an aggregate class customer leaving node 0 will enter node i, the parameters of the service rate $\mu_{(i,1)}^{(agg)}$, $\mu_{(i,2)}^{(agg)}$, and $\alpha_i^{(agg)}$ of the aggregate class at the transmission server of node $i, i = 1, \cdots, P$, and the parameters of the service rate $\mu_0^{(agg)}$ of the aggregate class at node 0.

Assuming that we know the mean response time $T_i^{(k)}$ of class k at node i, then we can calculate the throughput $H^{(k)}$ of class k in the network as follows:

$$H^{(k)} = \frac{N^{(k)}}{T_0^{(k)} + \sum_{i=1}^{P} T_i^{(k)} \times p_i^{(k)}} \tag{4.26}$$

where $N^{(k)}$ is the number of class k customers, (i.e., $N^{(k)} = 1$ for all k), and $p_i^{(k)}$ is the branching probability in the original network that a class k customer leaving node 0 will enter node i.

For a given class c, the branching probabilities $p_i^{(agg)}$ of the aggregate class can be calculated by:

$$p_i^{(agg)} = \frac{\sum_{k \neq c} (H^{(k)} p_i^{(k)})}{\sum_{k \neq c} H^{(k)}} \tag{4.27}$$

We employ the class aggregation technique in Baynat and Dallery's method [5] to obtain the parameters of the service time distribution of the aggregate class at the transmission server of node i, $i = 1, \cdots, P$. Since the distribution of each class is a two-stage hyper-exponential distribution, the distribution of the aggregate class is also a hyper-exponential distribution, but with more than two stages. The first two moments $E_i(Agg)$ and $E_i(Agg^2)$ of the distribution of the aggregate class at node i can be easily calculated as follows.

$$E_i(Agg) = \frac{\sum_{k\neq c} H^{(k)} p_i^{(k)} E(B^{(k)})}{\sum_{k\neq c} H^{(k)} p_i^{(k)}} \tag{4.28}$$

$$E_i(Agg^2) = \frac{\sum_{k\neq c} H^{(k)} p_i^{(k)} E^2(B^{(k)})(1 + c^2(B^{(k)}))}{\sum_{k\neq c} H_i^{(k)} p_i^{(k)}} \tag{4.29}$$

$$c_i^2(Agg) = \frac{E_i(Agg^2) - E_i^2(Agg)}{E_i^2(Agg)} \tag{4.30}$$

where $E(B^{(k)})$ and $c^2(B^{(k)})$ are the mean and the squared coefficient of variation of the burst duration of class k, respectively. $E(B^{(k)})$ and $c^2(B^{(k)})$ can be calculated using expressions (3.2) and (3.3), respectively. We approximate the service time distribution of the aggregate class as a two-stage Coxian distribution by moment matching. We use Marie's method [18] to match the first two moments, as follows:

$$\mu_{(i,1)}^{(agg)} = \frac{2}{E_i(Agg)} \tag{4.31}$$

$$\mu_{(i,2)}^{(agg)} = \frac{1}{E_i(Agg) c_i^2(Agg)} \tag{4.32}$$

$$\alpha_i^{(agg)} = \frac{1}{2c_i^2(Agg)} \tag{4.33}$$

The parameters of the service rate $\mu_0^{(agg)}$ of the aggregate class at node 0 can be obtained as follows:

$$1/\mu_0^{(agg)} = \frac{\sum_{k\neq c} H^{(k)}/\gamma^{(k)}}{\sum_{k\neq c} H^{(k)}} \tag{4.34}$$

where $1/\gamma^{(k)}$ is the mean duration of the idle state of class k.

The Iterative Algorithm

As we described in the previous subsection, if we know the mean response time of each class at each node in a network with C classes, then we can decompose the network into C two-class queueing networks. Using HAM, we can solve each of these

C two-class queueing networks, and then we can re-calculate the mean response time of each class at each node. We repeat this process until it converges. The following steps summarize our iterative algorithm.

- **Step 1.** Initialize the mean response time $T_i^{(c)}$ to 1 for all i, c, $i = 0, 1, \cdots, P, c = 1, \cdots, C$. Initialize the load-dependent service rate $\mu_i(n_i)$ of node $i, i = 1, \cdots, P$, in each two-class network to $\frac{\sum_i H^{(c)} p_i^{(c)}}{\sum_i H^{(c)} p_i^{(c)} E(B^{(c)})}$ for $n_i > 0$.

- **Step 2.** For each $i = 0, 1, \cdots, P$, and each $c = 1, \cdots, C$, do

  - **Step 2.1.** Calculate the throughput of each class using expression (4.26).

  - **Step 2.2.** Aggregate all classes except class c into one class using expressions (4.27), (4.33), and (4.34) to obtain the two-class queueing network c, which consists of class c and the aggregate class.

  - **Step 2.3.** Solve the two-class product-form queueing network c using the convolution algorithm to obtain the arrival rate to node i of both class c and the aggregate class.

  - **Step 2.4.** Solve node i numerically using the Gauess-Seidel method to obtain the steady-state probabilities $p_i(n_i^{(t1,1)}, n_i^{(t2,1)}, n_i^{(o,1)}, n_i^{(t1,2)}, n_i^{(t2,2)}, n_i^{(o,2)})$.

  - **Step 2.5.** Calculate the conditional throughput $v_i(n_i)$ of node i using expression (4.23), and use this value as the load-dependent service rate $\mu_i(n_i)$ of the flow equivalent server of node i in the two-class network c.

  - **Step 2.6.** Set variable $OLDT_i^{(c)}$ to $T_i^{(c)}$.

  - **Step 2.7.** Calculate the *new* mean response time $T_i^{(c)}$ of class c at node i.

- **Step 3.** Check whether the mean response times $T_i^{(c)}, i = 0, 1, \cdots, P, c = 1, \cdots, C$, satisfy the following convergence criterion. If so, then stop. Otherwise, repeat from Step 2.

$$\frac{\sqrt{\sum_{i,c}(T_i^{(c)} - OLDT_i^{(c)})^2}}{\sqrt{\sum_{i,c}(T_i^{(c)})^2}} < \epsilon \tag{4.35}$$

4.7 Limiting Cases

In this section, we consider several limiting cases for which we can obtain closed-form expressions.

4.7.1 Infinite Orbiting Time

Let us first consider the case where the mean orbiting time $1/\omega$ tends to infinity. For simplicity, we consider the single-class queueing network with converters analyzed in section 4.5. In the limiting case, it is easy to see that some customers will be orbiting for ever, whereas the remaining customers will be circulating through the network between node 0 and the transmission servers. We expect that, in steady state, there will be exactly W customers which are circulating within the network. If, at some time, there are more than W customers circulating within the network, then there is a non-zero probability that one of these customers will find all transmission servers busy when it arrives at a node, and it will become orbiting for ever. The number of circulating customers will continue to decrease until it reaches W. In this case, none of these W customers will ever find all transmission servers busy when it arrives at a node, since each node $i, i = 1, \cdots, P$, has W transmission servers.

The sojourn time of one of these W customers is $E(B) + 1/\gamma$. In view of this, the throughput of node i is $p_i W/(E(B) + 1/\gamma)$. The mean number of customers orbiting at node i is approximately $(NW - W)p_i$. Using Little's theorem, the mean waiting time of a customer at node i is $(N-1)(E(B) + 1/\gamma)$.

4.7.2 Single Hot Spot

Let us now consider the special case where there is one hot output port, say output port k, and all traffic goes to this output port (i.e., $p_k = 1$, and $p_i = 0$, for all $i \neq k$). The corresponding queueing network is shown in Figure 4.5. It consists of an idle node (node 0) and a node corresponding to output port k.

For simplicity, we analyze the queueing network in Figure 4.5 with a single class

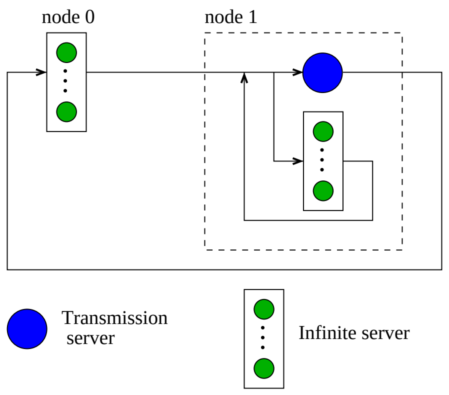

Figure 4.5: Queueing model of an edge switch with a hot spot

of customers. Let $T^{(t)}$ denote the mean service time at the transmission server (i.e., $T^{(t)} = E(B)$), $T^{(i)}$ denote the mean service time at the idle server (i.e., $T^{(i)} = 1/\gamma$), and $T^{(w)}$ denote the mean waiting time of a customer at the orbiting server before it succeeds to receive service at the transmission server (that is, it is equal to mean orbiting time times the mean orbiting number). Note that the mean sojourn time of a customer is $T^{(t)}+T^{(i)}+T^{(w)}$. Let ρ denote the utilization of the transmission server. In case of multiple transmission servers (e.g., in an edge switch with converters), ρ is the mean utilization of all transmission servers. Let Q denote the mean number of customers at the orbiting server. Also, we use ω, N, W as defined in Table 4.1.

The mean number of customers at the orbiting server, Q, is given by:

$$Q = NW\frac{T^{(w)}}{T^{(t)} + T^{(i)} + T^{(w)}} \tag{4.36}$$

The mean number of customers at the transmission server can be calculated in a similar manner, and it is equal to $NW\frac{T^{(t)}}{T^{(t)}+T^{(i)}+T^{(w)}}$. Since the mean number of customers at the transmission server is also equal to ρW, we obtain the following expression:

$$\rho = N\frac{T^{(t)}}{T^{(t)} + T^{(i)} + T^{(w)}} \tag{4.37}$$

Now, we calculate ρ approximately as follows. We first consider the extreme case that the utilization of output port k is almost 1. In this case, there are always orbiting customers. However, the utilization cannot reach 1, because when a transmission server becomes free, it takes a mean time equal to $1/(Q\omega)$ for the first orbiting customer to enter the transmission server. Thus, the utilization of output port k can be calculated approximately as:

$$\rho = \frac{T^{(t)}}{T^{(t)} + 1/(Q\omega)} \tag{4.38}$$

For the case where the utilization of output port k is not very close to 1, we use the following expression, which is an intuitive extension of expression (4.38).

$$\rho \;=\; \frac{T^{(t)}}{T^{(t)} + \rho/(Q\omega)} \tag{4.39}$$

Now, we have three unknown variables: Q, ρ, and $T^{(w)}$. By solving equations (4.36), (4.37), and (4.39), we obtain:

$$\begin{cases} T^{(w)} &= (\frac{N}{\rho} - 1)T^{(t)} - T^{(i)} \\ Q &= W(N - \rho - \rho\frac{T^{(i)}}{T^{(t)}}) \\ \rho &= \frac{(N+1)W\omega T^{(t)} + W\omega T^{(i)} - \sqrt{((N+1)W\omega T^{(t)} + W\omega T^{(i)})^2 - 4NW\omega T^{(t)}(W\omega T^{(t)} + W\omega T^{(i)} - 1)}}{2(W\omega T^{(t)} + W\omega T^{(i)} - 1)} \end{cases} \tag{4.40}$$

When the number of wavelengths is very large (i.e., $W \to \infty$), we have that $\rho = 1$. This result is due to the fact that, when there is an infinite number of orbiting customers (note that $\lim_{W\to\infty} Q = \infty$), the mean time $1/(Q\omega)$ for the first orbiting customer to enter the transmission server becomes zero. In this case, we obtain:

$$\begin{cases} T^{(w)} &= (N - 1)T^{(t)} - T^{(i)} \\ Q &= \infty \\ \rho &= 1 \end{cases} \tag{4.41}$$

4.8 Numerical Results

In this section, we examine the accuracy of our approximation algorithm, by comparing the approximate results to results obtained from a simulation program of an edge OBS switch. In all the figures given in this section, simulation results are plotted along with 95% confidence intervals estimated by the method of batch mean. The number of batches was set to 30, with each batch run lasting until each

Name	Description
Mean waiting time of a user	Average waiting time of the user before transmitting a burst to the switch
Utilization of a wavelength	Percentage of time that the wavelength is busy
Throughput of a wavelength	Number of bursts transmitted on the wavelength in a unit of time
Utilization of an output port	Average utilization of all wavelengths in the output port
Throughput of an output port	Sum of throughput of all wavelengths in the output port
Mean waiting time of an output port	Average waiting time of all users before transmitting a burst to the output port
Switch utilization	Average utilization of all output ports of the switch
Switch throughput	Sum of throughput of all output ports of the switch
Switch mean waiting time	Average waiting time of all users before transmitting a burst to the switch

Table 4.2: Performance measures in each experiment

wavelength has transmitted at least 100,000 bursts. As the reader will notice, however, most confidence intervals are very narrow and are barely visible in these figures.

For each experiment, we are interested in the performance measures listed in Table 4.2.

The edge switch that we consider has 8 fiber links for users and 2 other links connecting it to other 2 OBS nodes. As mentioned before, we assume that there is no traffic from the other 2 OBS nodes to this edge switch. We only model traffic generated by the users, which is destined to the 2 OBS nodes, and to the users themselves. We have $P+1$ nodes in our queueing network model, where $P=10$. Nodes 1 to 8 represent the output ports connected to the users, nodes 9 and 10 represent the output ports connected to the other two OBS nodes, and node 0 represents the burst arrival processes in the idle state.

4.8.1 Edge Switch with Homogeneous Users and w/o Converters

In this subsection, we assume that the edge switch is not equipped with wavelength converters. As described in section 4.3, the analysis of an edge node can be decomposed into multiple sub-systems, one per burst wavelength. Therefore, in this subsection, we analyze the edge switch with only one wavelength for each input and output port (i.e. $W = 1$). We also assume that all wavelengths have identical burst arrival processes. That is, the queueing network model is a single-class queueing network with $N = 8$ customers.

Unless noted otherwise in the rest of this subsection, we set the mean duration of a short burst $1/\phi_s$ to 1, the mean duration of the long burst $1/\phi_l$ to 100 (i.e. $\phi_l = 0.01$), the probability p_s that a burst is a short burst to 0.8, the mean duration of the idle state $1/\gamma$ to 10 (i.e. $\gamma = 0.1$), the mean orbiting time $1/\omega$ to 10 (i.e. $\omega = 0.1$), the probability p_i that a burst is destined to output port i to 0.1 for all $i = 1, 2, \cdots, 10$.

In the first experiment, we vary ϕ_l from 0.1 to 0.01, and increase p_s from 0.1 to 0.9 with an increment of 0.1. Figures 4.6 and 4.7 show the switch utilization and switch mean waiting time, respectively. We observe that the approximation method has a good accuracy (the percent error is less than 5%). We also observe that the error is larger with ϕ_l=0.01 than with ϕ_l=0.1. This is because a smaller ϕ_l leads to a larger squared coefficient of variation of the service time of the transmission server, as calculated in Equation 3.3. We also observe that as the p_s increases, the switch utilization goes down. This is because we keep γ unchanged (i.e. the mean duration of the idle state), the increase of p_s leads to the decrease of the percentage of time that the user is transmitting, and subsequently the decrease of the switch utilization. This is also the reason for the decrease of the switch utilization when we decrease the mean duration of the long burst.

In the second experiment, we increase γ from 0.001 to 0.01, 0.1, 1, and 10. Figures 4.8 and 4.9 show the switch utilization and switch mean waiting time, respectively. We observe that the error increases as the γ increases (i.e. as the burst traffic

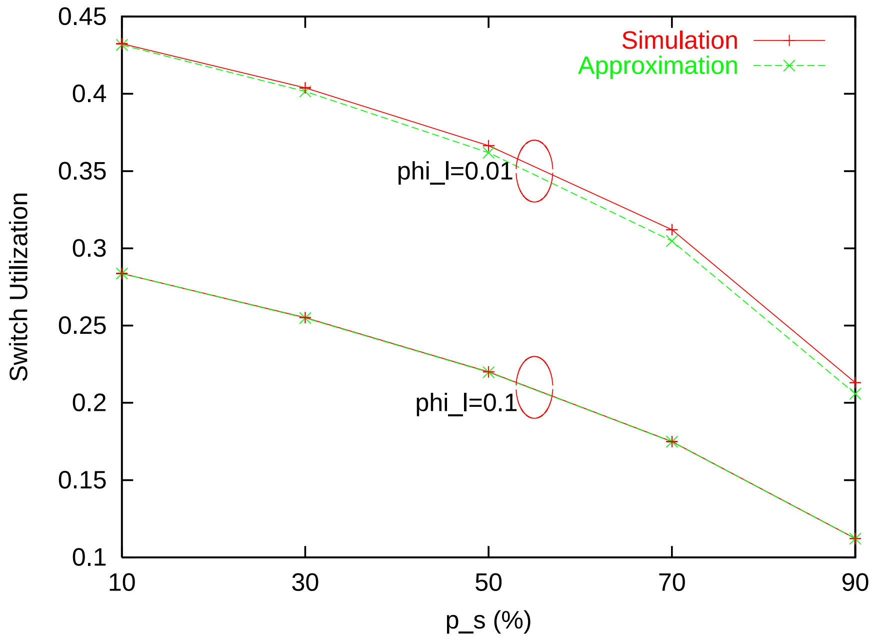

Figure 4.6: Switch utilization for ϕ_l=0.1, and 0.01

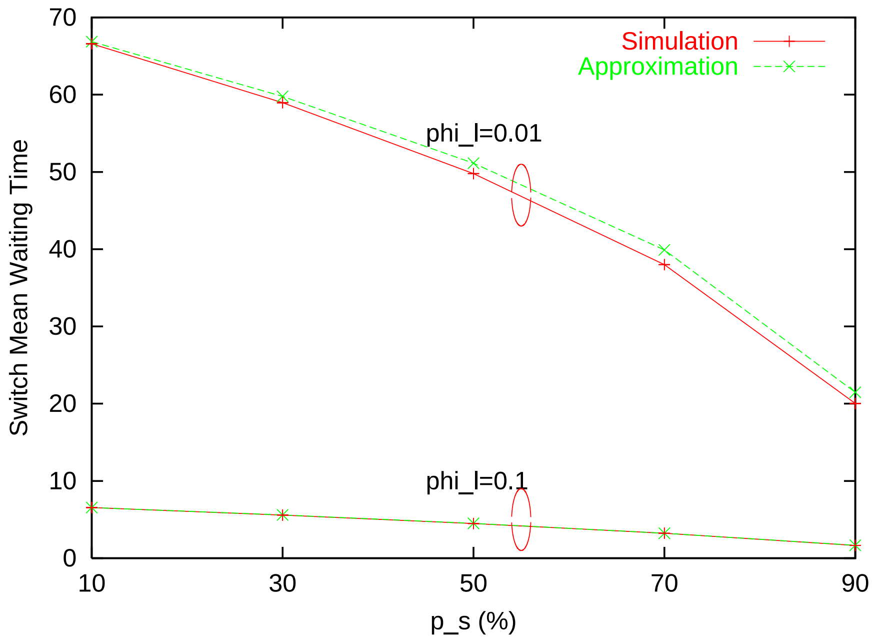

Figure 4.7: Switch mean waiting time for ϕ_l=0.1, and 0.01

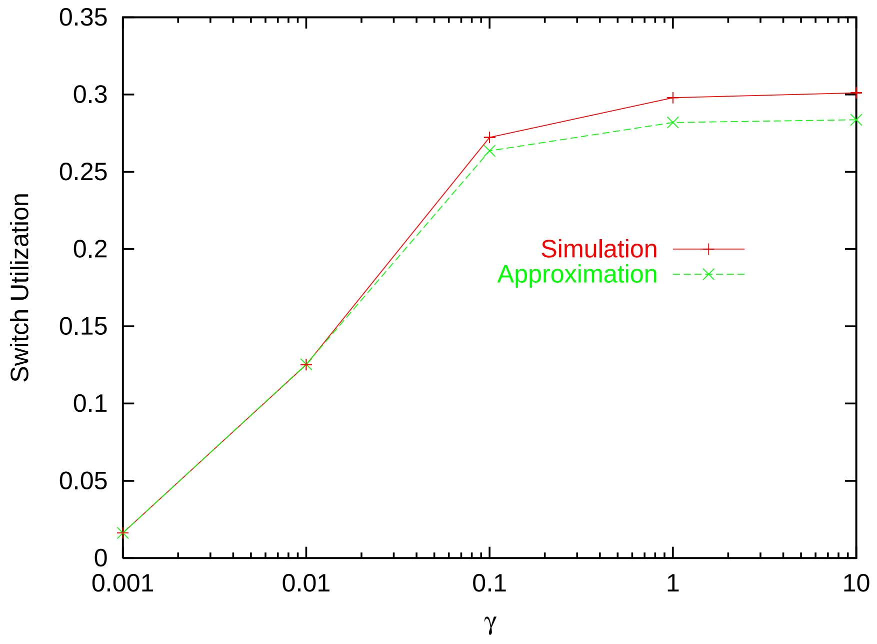

Figure 4.8: Switch utilization vs. γ

increases). The maximum error is about 10%. We find that the switch utilization and switch mean waiting time go up when γ increases. This is because when γ is increased, the mean duration of the idle state decreases. That is, the user will transmit more bursts.

In the third experiment, we increase ω from 0.0001 to 0.001, 0.01, 0.1, 1, and 10. Figures 4.10 and 4.11 show the switch utilization and the switch mean waiting time, respectively. The utilization and the waiting time with an infinite small ω are also plotted (based on the calculation in section 4.7.1). We observe that the switch utilization goes up as ω increases. This is because when ω increases, the orbiting customers attempt to seize the transmission server more often, and therefore the server utilization goes up. We also observe that when ω is 0.0001, the simulation result is close to the result of the limiting case, and the error of the approximation algorithm is very large. However, a very small ω is not reasonable in practice, since it leads to a lower utilization and a longer waiting time.

In the last experiment, we consider a hot spot case by increasing p_{10} from 0.1 to

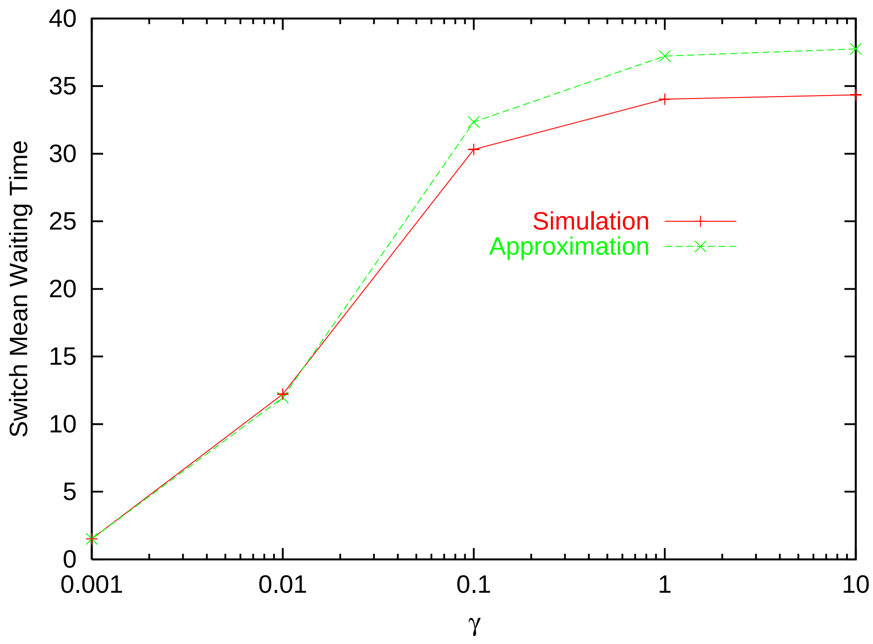

Figure 4.9: Switch mean waiting time vs. γ

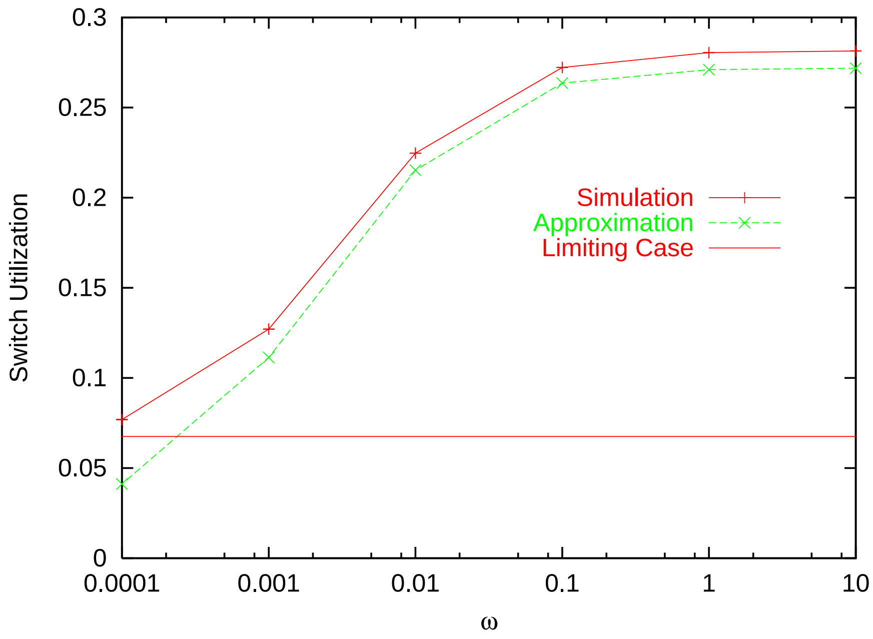

Figure 4.10: Switch utilization vs. ω

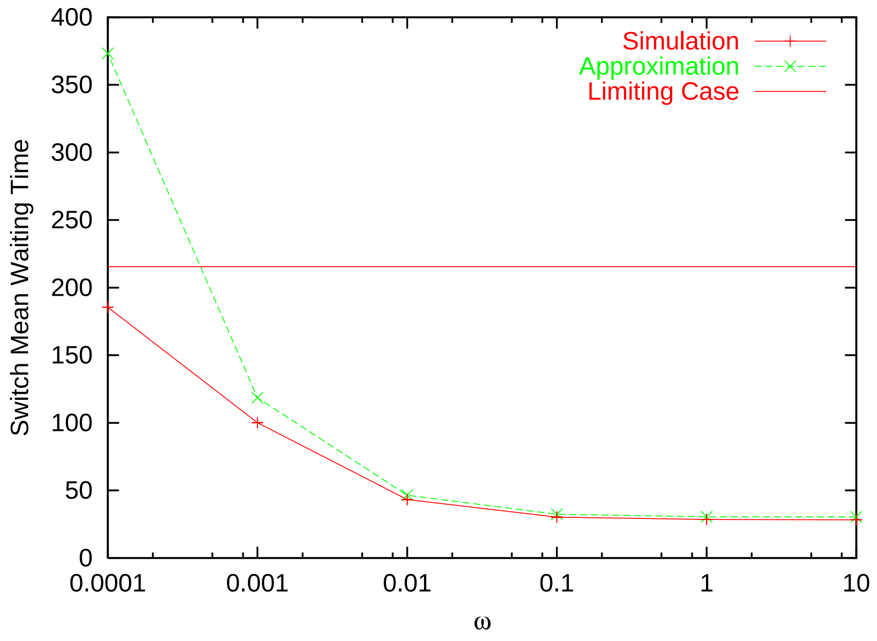

Figure 4.11: Switch mean waiting time vs. ω

0.9 with an increment of 0.2, and setting p_i $(i = 1, \cdots, 9)$ to $(1 - p_{10})/9$. Figures 4.12, 4.13, and 4.14 show the utilization of output port 1, output port 10 and the switch, respectively. Figures 4.15, 4.16, and 4.17 plot the mean waiting time of output port 1, output port 10 and the switch, respectively. We observe that the approximation algorithm has a better accuracy for output port 10 than for output port 1. The maximum error is about 10%.

In all experiments reported in this subsection, the simulation usually takes about half an hour to run, while the approximation algorithm only takes about 1 second.

4.8.2 Edge Switch with Homogeneous Users and with Converters

In this subsection, we assume that the edge switch is equipped with wavelength converters. We increase the number of wavelength W in each fiber from 1 to 3, 5,

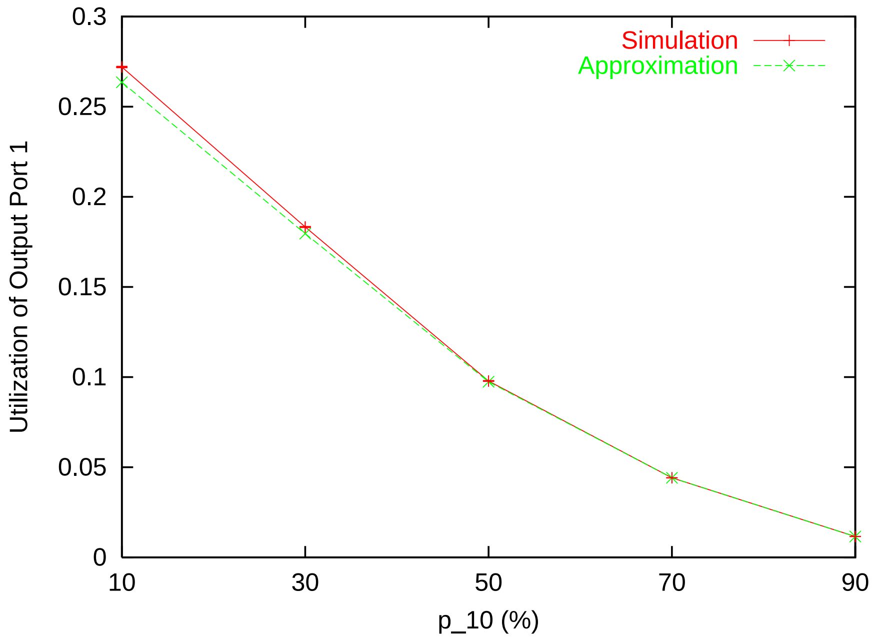

Figure 4.12: Utilization of output port 1 vs. p_{10}

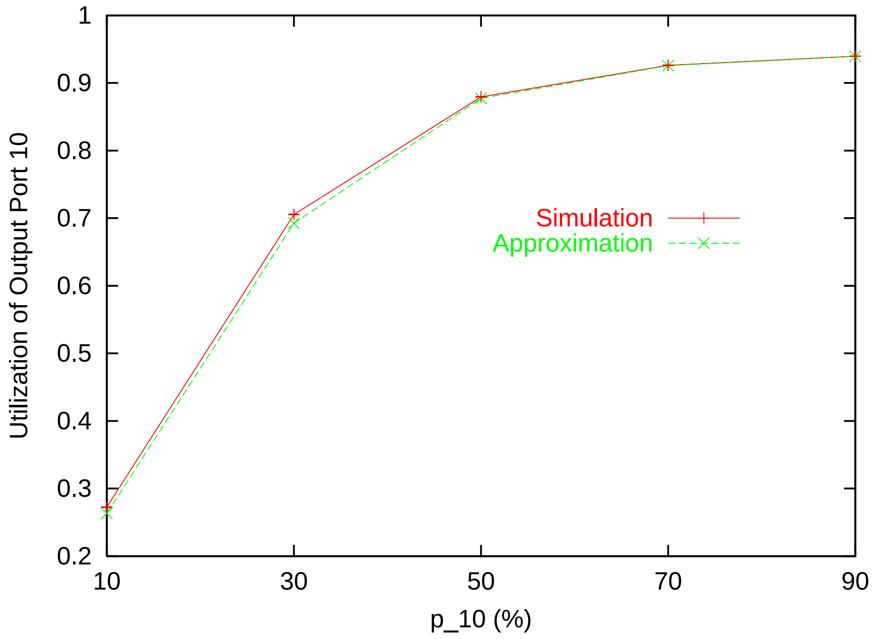

Figure 4.13: Utilization of output port 10 vs. p_{10}

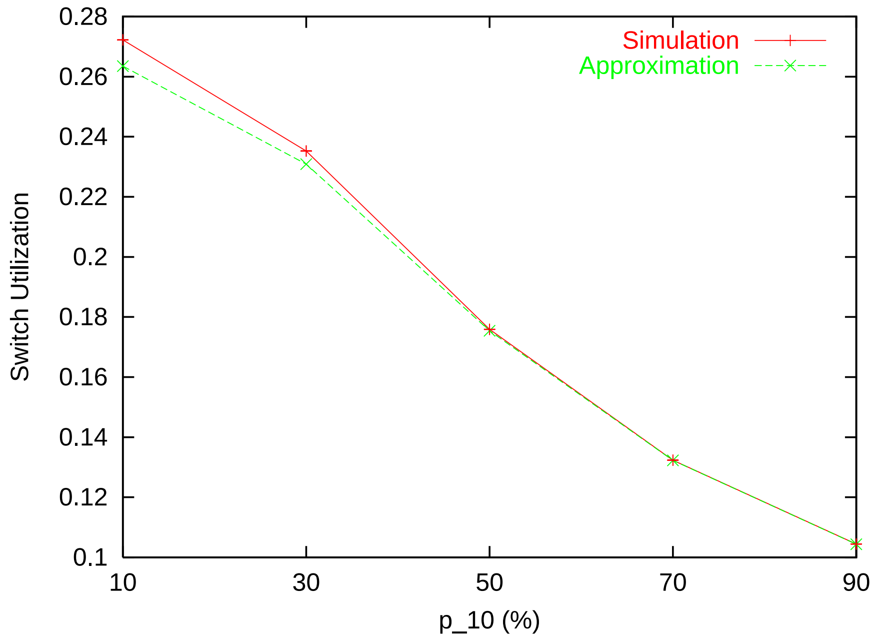

Figure 4.14: Switch utilization vs. p_{10}

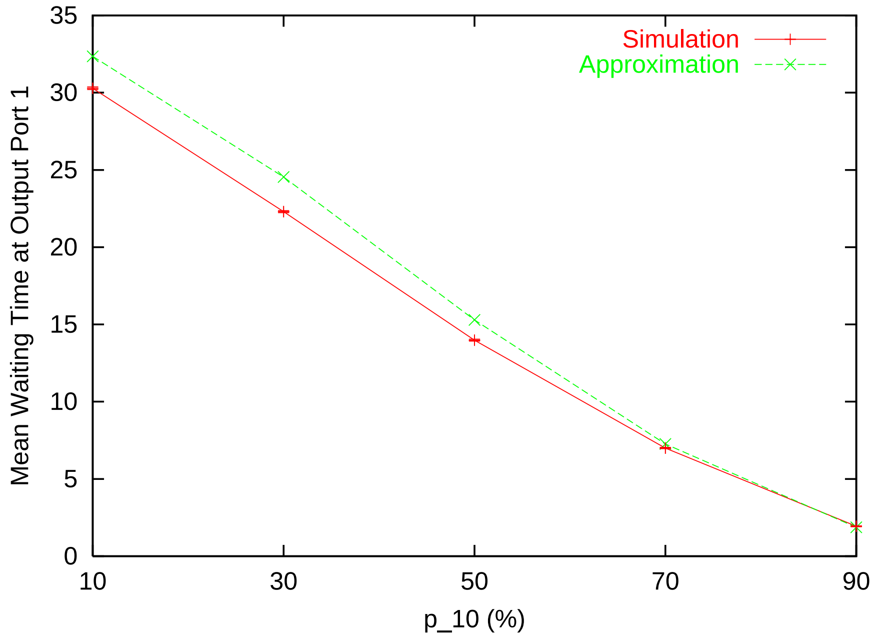

Figure 4.15: Mean waiting time of output port 1 vs. p_{10}

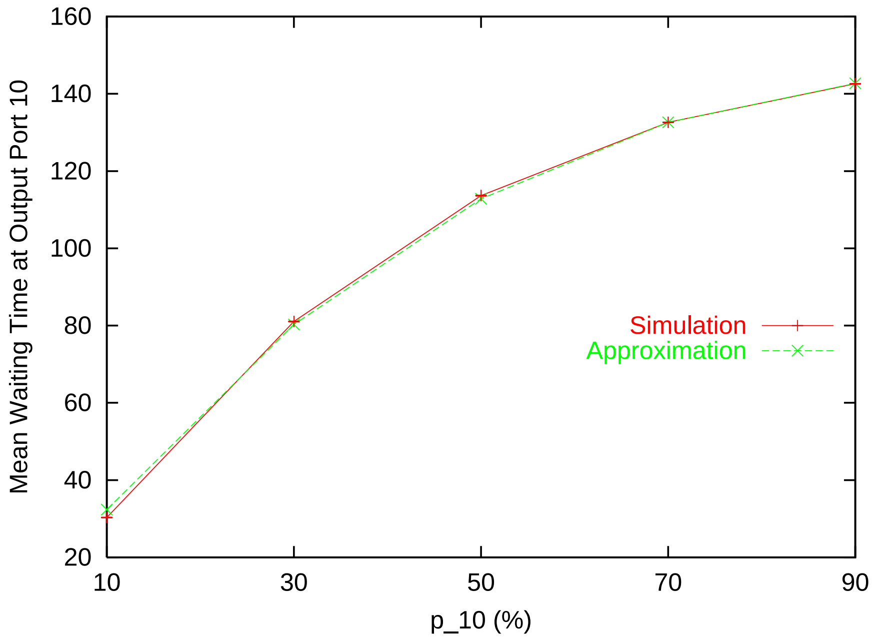

Figure 4.16: Mean waiting time of output port 10 vs. p_{10}

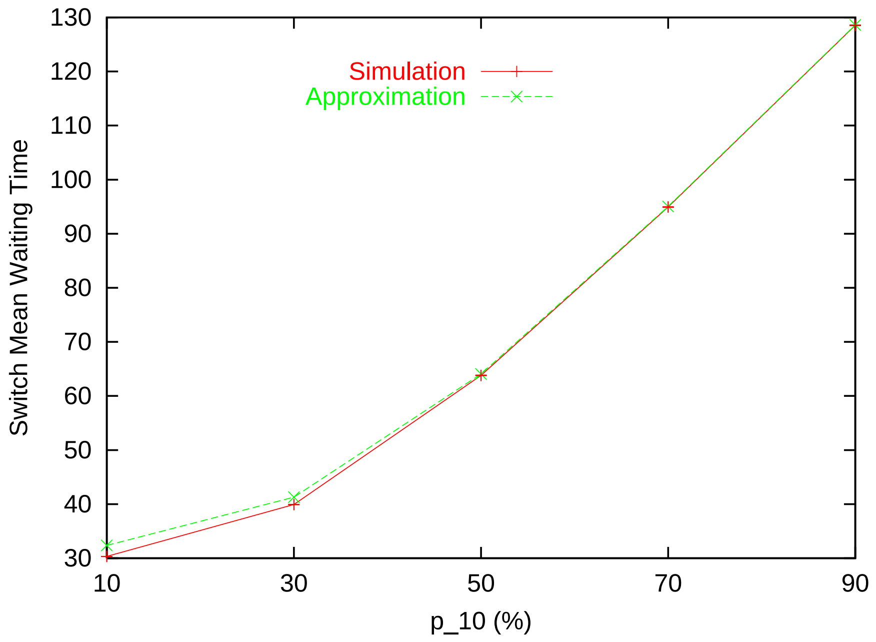

Figure 4.17: Switch mean waiting time vs. p_{10}

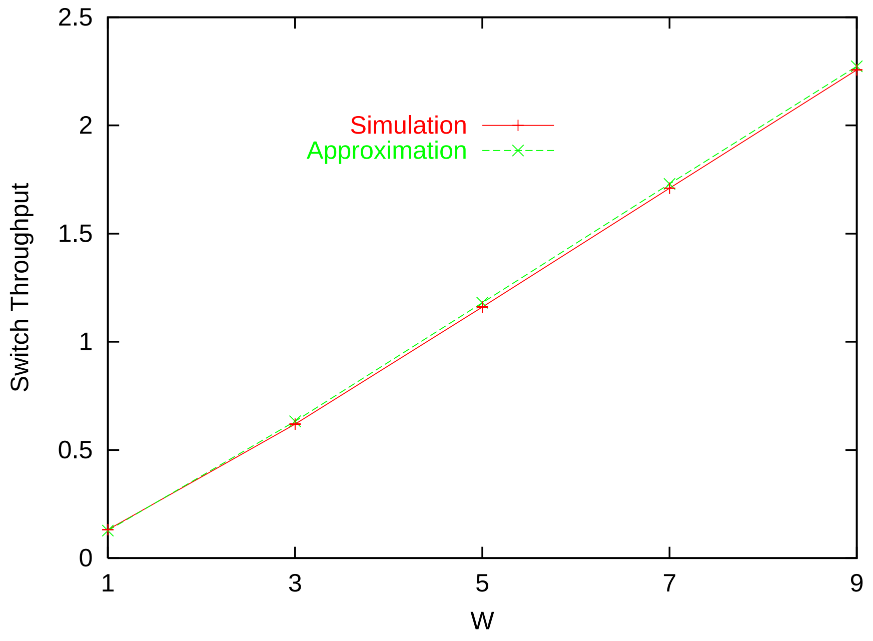

Figure 4.18: Switch throughput vs. W

7, and 9. We also assume that all wavelengths have identical burst arrival processes. Therefore, the corresponding queueing network model is still a single-class queueing network, but the number of customers is $8 \times W$ instead of 8, and the number of transmission servers is W instead of one.

We also set ϕ_s=1, ϕ_l=0.01, p_s=0.8, γ=0.1, ω=0.1, p_i=0.1 for all $i = 1, 2, \cdots, 10$. Figures 4.18, 4.19 and 4.20 show the switch throughput, switch utilization and switch mean waiting time, respectively. We observe that as W increases, the switch utilization and the switch throughput increase, while the switch mean waiting time decreases. We also observe that the approximation algorithm has a larger error (about 20%) of the switch mean waiting time when $W = 9$. However, the absolute difference between the simulation result and the numerical result is very small at that point. We also note that the approximation algorithm takes a much shorter time to get the result than the simulation. For example, when $W = 9$, the approximation algorithm takes about 1 minute to run, and the simulation takes about 7 hours.

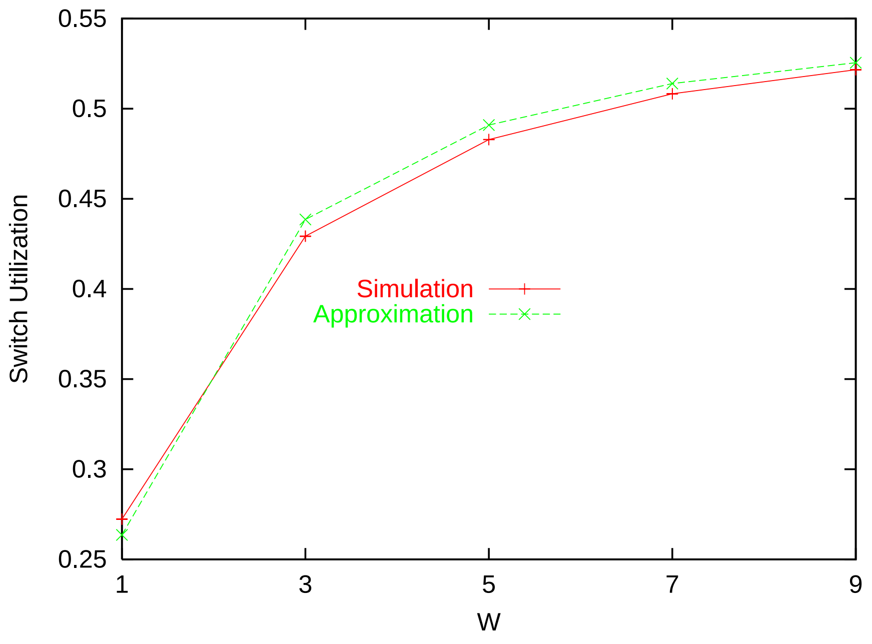

Figure 4.19: Switch utilization vs. W

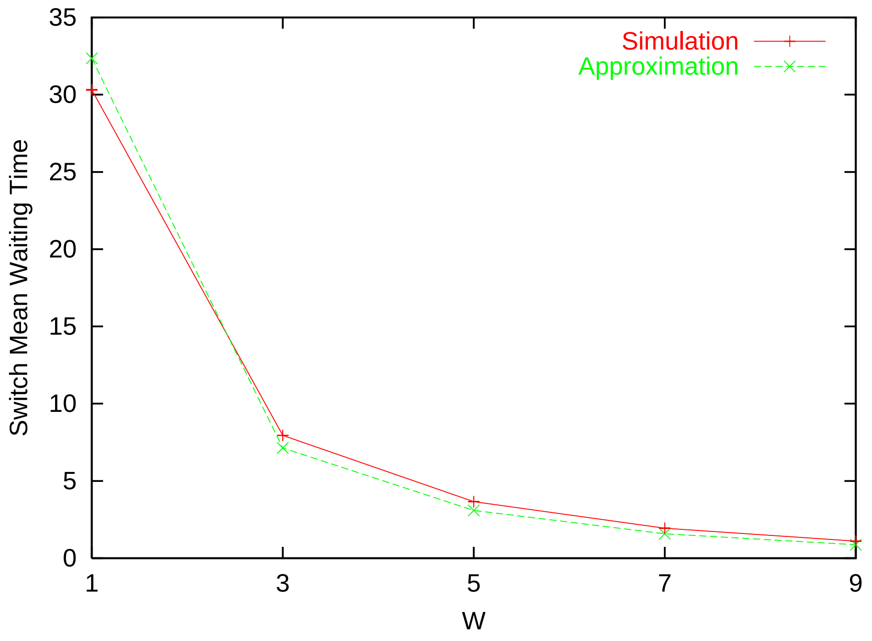

Figure 4.20: Switch mean waiting time vs. W

4.8.3 Edge Switch with Heterogeneous Users and w/o Converters

In this subsection, we assume that the edge switch is not equipped with wavelength converters. We use the same edge OBS node used in the experiments described in previous subsection 4.8.1. We assume that the edge switch has only one burst wavelength for each input and output port (i.e. $W = 1$), and each wavelength has a different burst arrival process. That is, the queueing network model is a multi-class queueing network with $N = 8$ classes, each of which has a single customer.

First, we consider a hot spot case. The edge switch has the same parameters as in the hot spot case in subsection 4.8.1, with the exception that a user can not transmit bursts to itself. Let $p_j(i)$ denote the probability that a burst from user i will go to output port j. We set $p_j(i)$ for all $i = 1, 2, \cdots, 8$ as follows.

$$p_j(i) \ = \ \begin{cases} p_{hot} & \text{if } j = 10 \\ \frac{1-p_{hot}}{8} & \text{if j} \neq \text{10 and j} \neq \text{i} \\ 0 & \text{if j = i} \end{cases} \tag{4.42}$$

We increase p_{hot} from 0.1 to 0.9 with an increment of 0.2, and set ϕ_s=1, ϕ_l=0.01, p_s=0.8, γ=0.1, ω=0.1. Figures 4.21 and 4.22 show the utilization of output port 1 and output port 10, respectively. Figures 4.23 and 4.24 plot the mean waiting time of output port 1 and output port 10, respectively. We observe that the maximum error of the approximation algorithm is about 10%.

Secondly, we consider the case where each user has a different burst size distribution. For simplicity, we set ϕ_s of all users to 1, γ of all users to 0.1, and only vary ϕ_l and p_s as in Table 4.3. We set $p_j(i)$ to 1/9 for all $j \neq i$, and to 0 for all $j = i$. We also set ω=0.1. Figures 4.25, 4.26, and 4.27 plot the mean waiting time of user 1, user 8 and the switch mean waiting time, respectively. Figure 4.28 shows the switch utilization. We observe that the maximum error of the approximation algorithm is about 5%.

In both experiments, the simulation usually takes half an hour, whereas the approximation algorithm takes about 1 second.

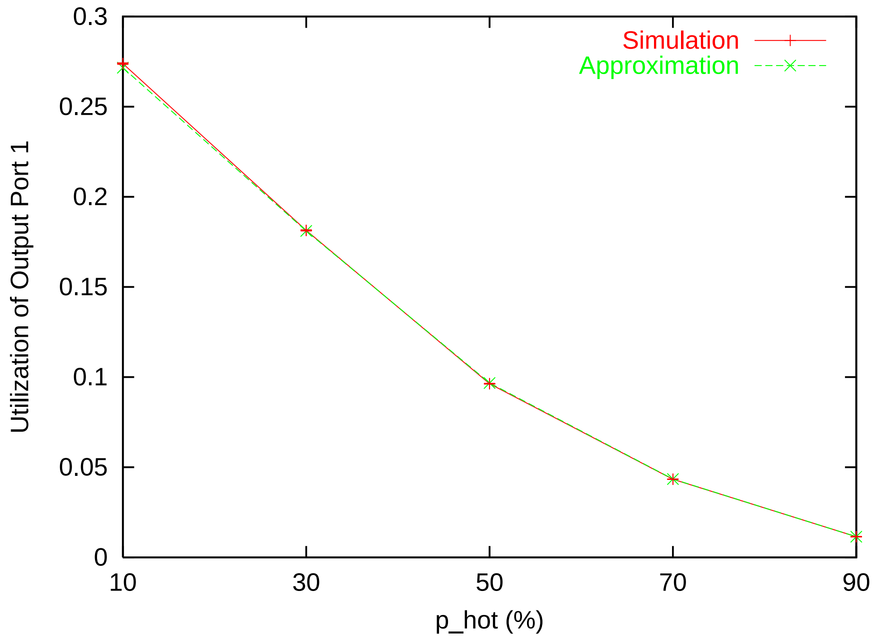

Figure 4.21: Utilization of output port 1 vs. p_{hot}

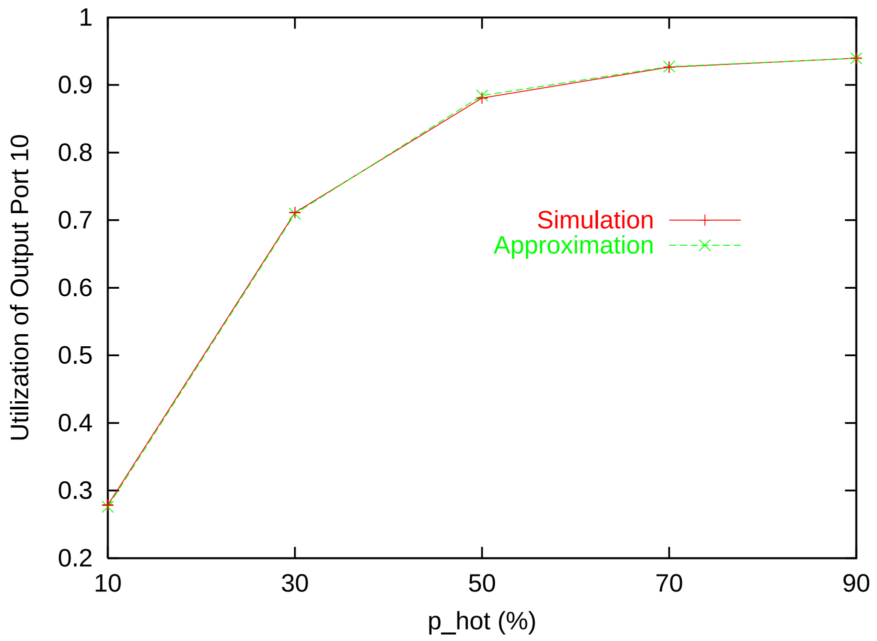

Figure 4.22: Utilization of output port 10 vs. p_{hot}

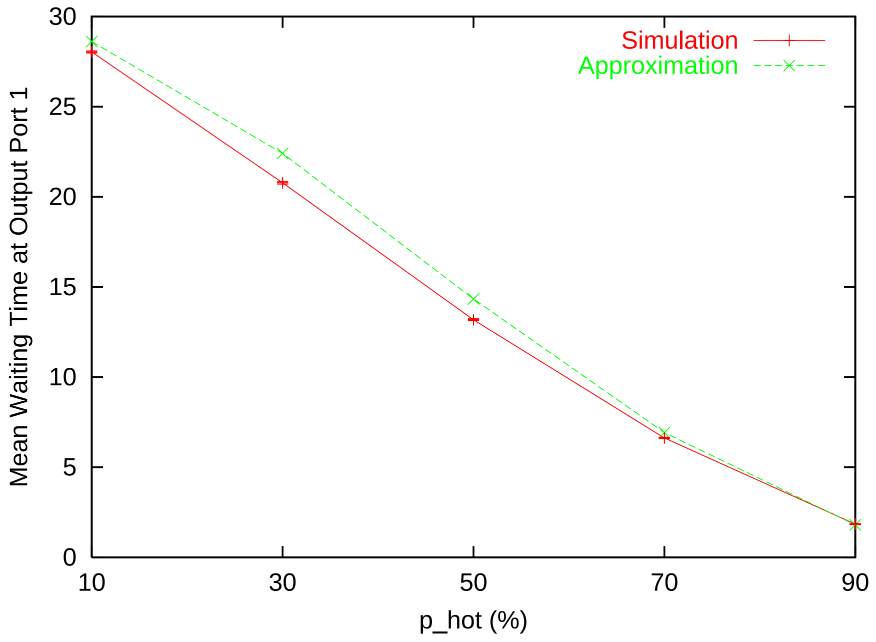

Figure 4.23: Mean waiting time of output port 1 vs. p_{hot}

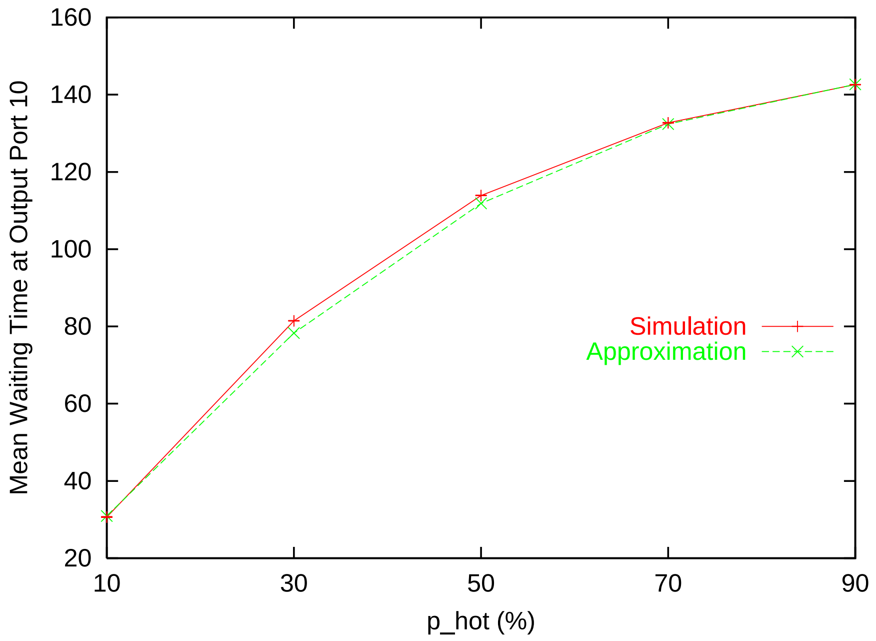

Figure 4.24: Mean waiting time of output port 10 vs. p_{hot}

		user 1	user 2	user 3	user 4	user 5	user 6	user 7	user 8
case 1:	ϕ_l	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5
	p_s	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6
case 2:	ϕ_l	0.5	0.5	0.5	0.5	0.1	0.1	0.1	0.1
	p_s	0.6	0.6	0.6	0.6	0.7	0.7	0.7	0.7
case 3:	ϕ_l	0.5	0.5	0.5	0.1	0.1	0.1	0.05	0.05
	p_s	0.6	0.6	0.6	0.7	0.7	0.7	0.8	0.8
case 4:	ϕ_l	0.5	0.5	0.1	0.1	0.05	0.05	0.01	0.01
	p_s	0.6	0.6	0.7	0.7	0.8	0.8	0.9	0.9

Table 4.3: Case where all users have different burst size distributions

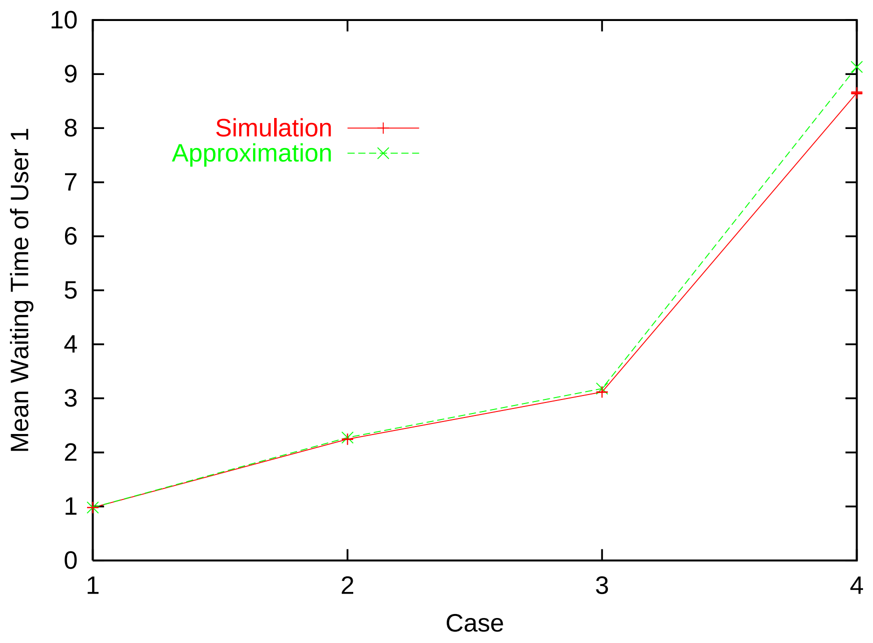

Figure 4.25: Mean waiting time of user 1 for each case

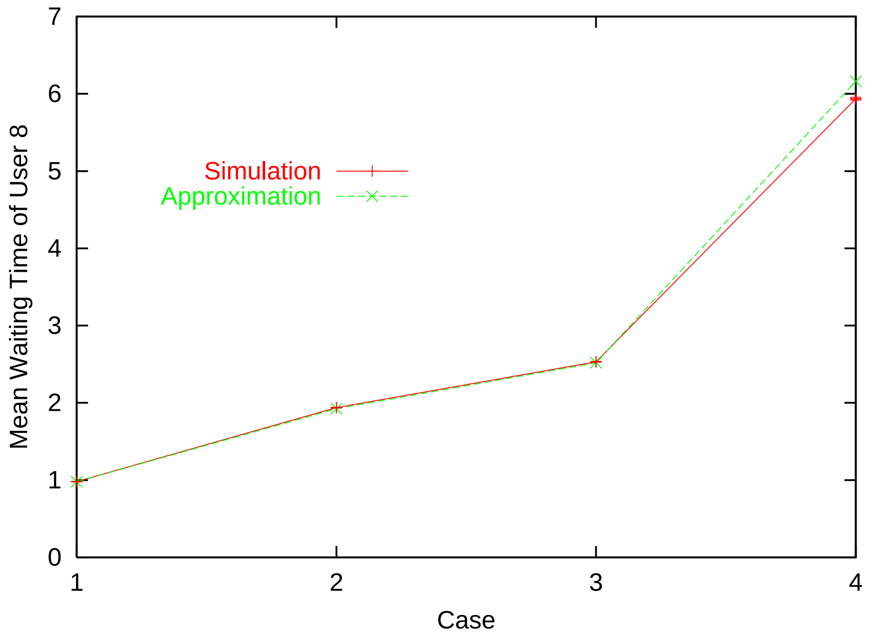

Figure 4.26: Mean waiting time of user 8 for each case

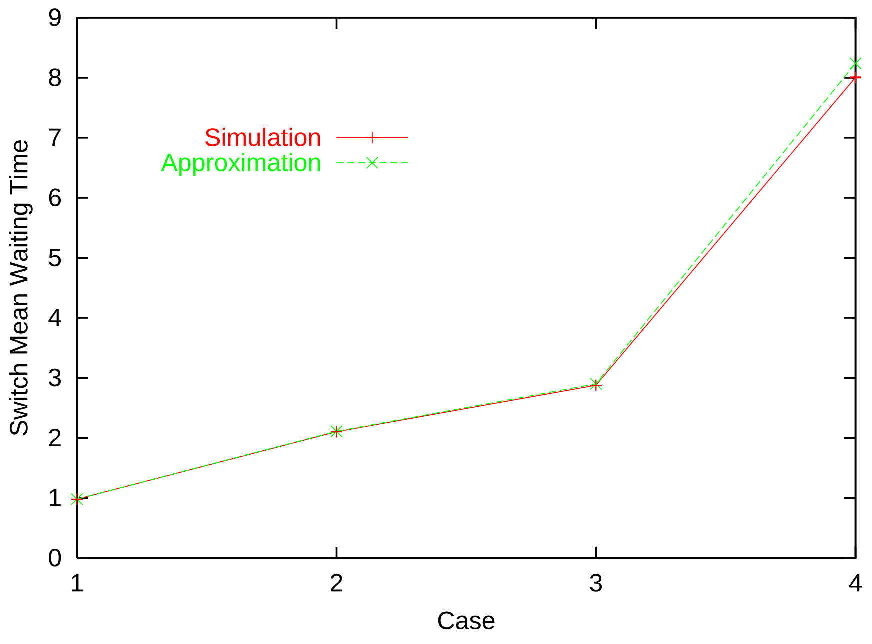

Figure 4.27: Switch mean waiting time for each case

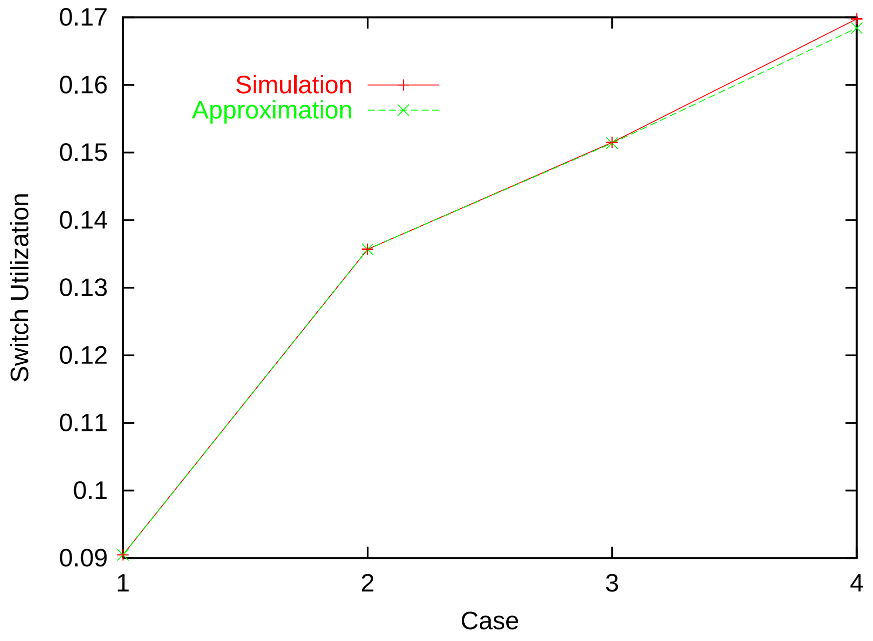

Figure 4.28: Switch utilization for each case

4.8.4 Edge Switch with Heterogeneous Users and with Converters

In this subsection, we consider the same edge switch analyzed in the previous subsection, but we assume that it is equipped with wavelength converters, each fiber supports W burst wavelengths, and the number of the transmission server is W. We also assume that different wavelengths have different burst arrival processes. That is, the queueing network has $N \times W = 8 \times W$ classes and one customer per class. For simplicity, we assume that the burst arrival processes associated with the W wavelengths of the same user have the same parameters.

First, we consider a hot spot case. We increase W from 1 to 9 with an increment 2. For all burst arrival process, we set ϕ_s=1, ϕ_l=0.01, p_s=0.8, γ=0.1, and set $p_j(i)$ using equation 4.42 with p_{hot}=0.3. We also set ω=0.1. Figures 4.29, 4.30 and 4.31 show the switch throughput, switch utilization and switch mean waiting time, respectively. We observe that the approximation algorithm has a good accuracy, and the maximum

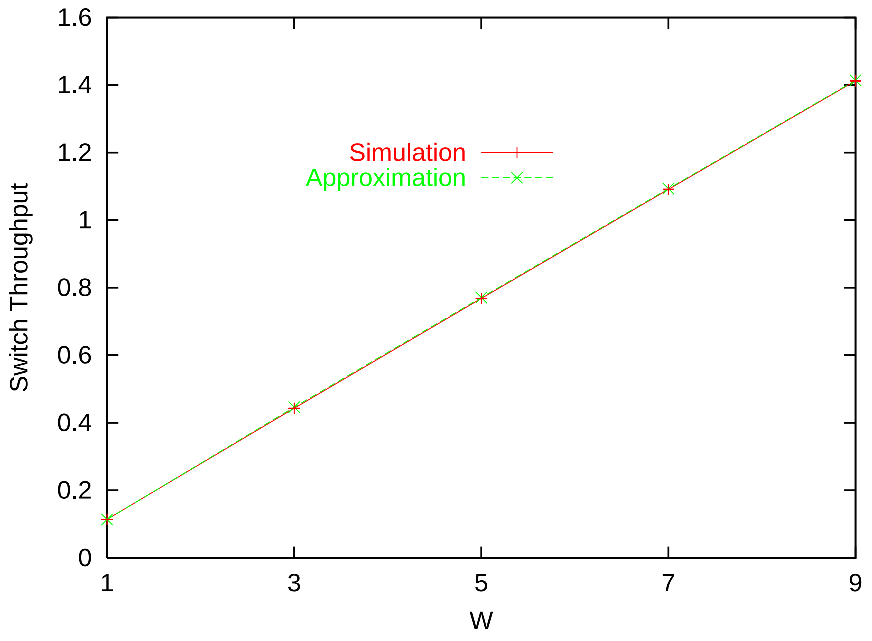

Figure 4.29: Switch throughput vs. W

error is about 2%. We note that when $W = 9$, the simulation takes about 12 hours, and the approximation algorithm takes about 2 hours (the number of iterations is 3).

Secondly, we consider the case where all users have different burst size distributions. We increase W from 1 to 9 with an increment 2. We set ϕ_s of all users to 1, γ of all users to 0.1, and set ϕ_l and p_s to the case 4 in Table 4.3. We set $p_j(i)$ to 1/9 for all $j \neq i$, and to 0 for all $j = i$. We also set ω=0.1. Figures 4.32, 4.33 and 4.34 show the switch throughput, switch utilization and switch mean waiting time, respectively. We observe that the approximation algorithm has a good accuracy. We also note that when $W = 9$, the simulation takes about 9 hours, and the approximation takes about 5 hours (the number of iterations is 5).

We would like to mention that when $W = 9$, the corresponding queueing network has $NW = 8 * 9 = 72$ classes, with one customer per class. In both experiments, the algorithm takes several hours. This is because the 72-class queueing network is decomposed into 72 two-class queueing networks, and then all these 72 two-class queueing networks need to be solved. If we know in advance that some burst arrival

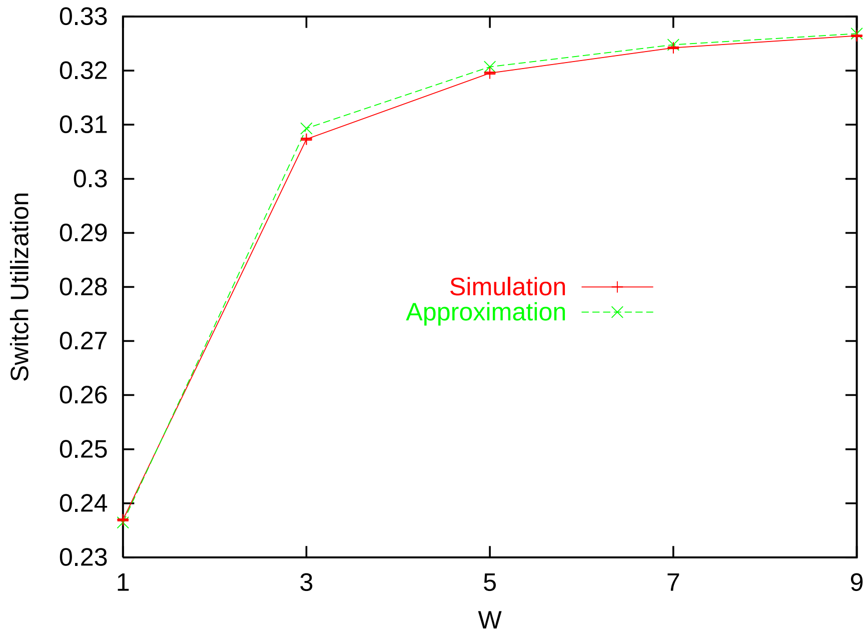

Figure 4.30: Switch utilization vs. W

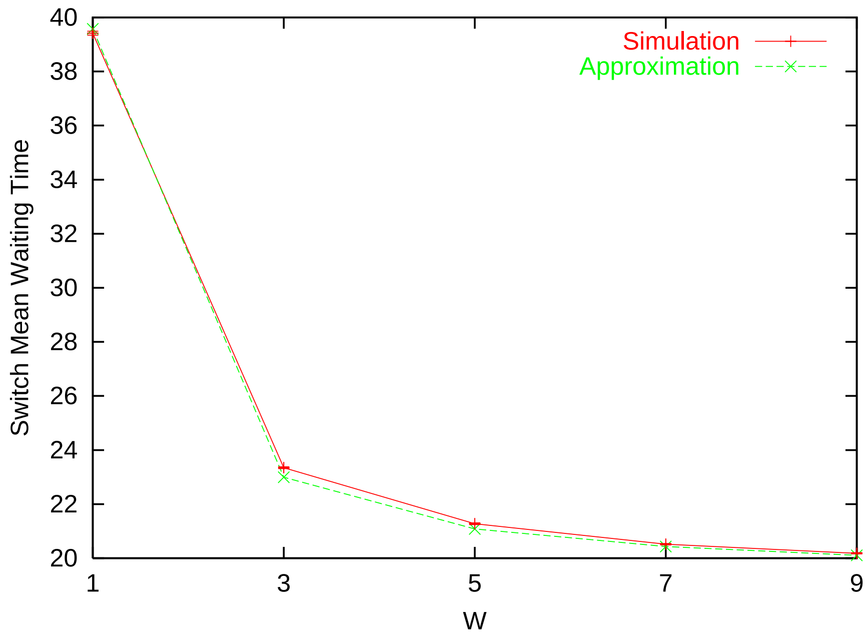

Figure 4.31: Switch mean waiting time vs. W

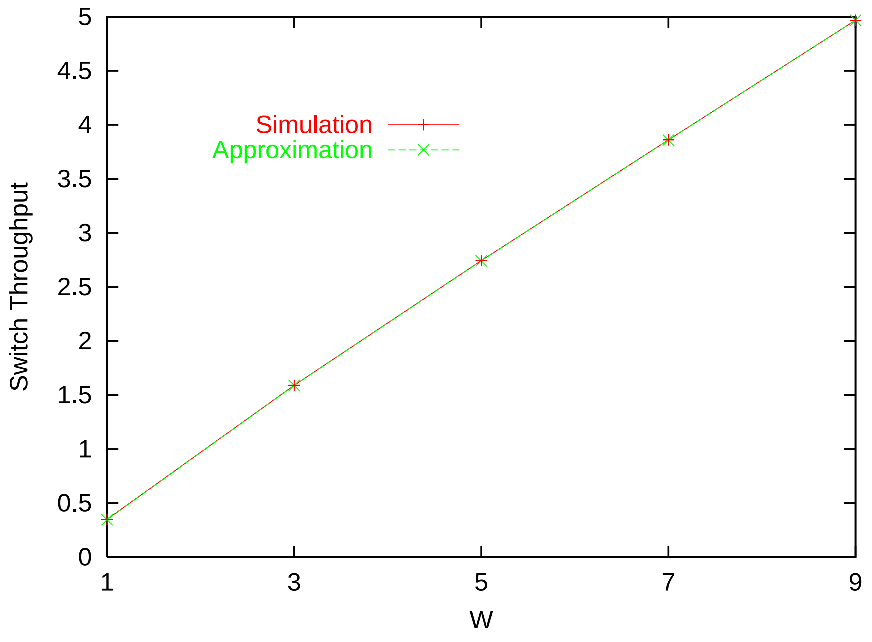

Figure 4.32: Switch throughput vs. W

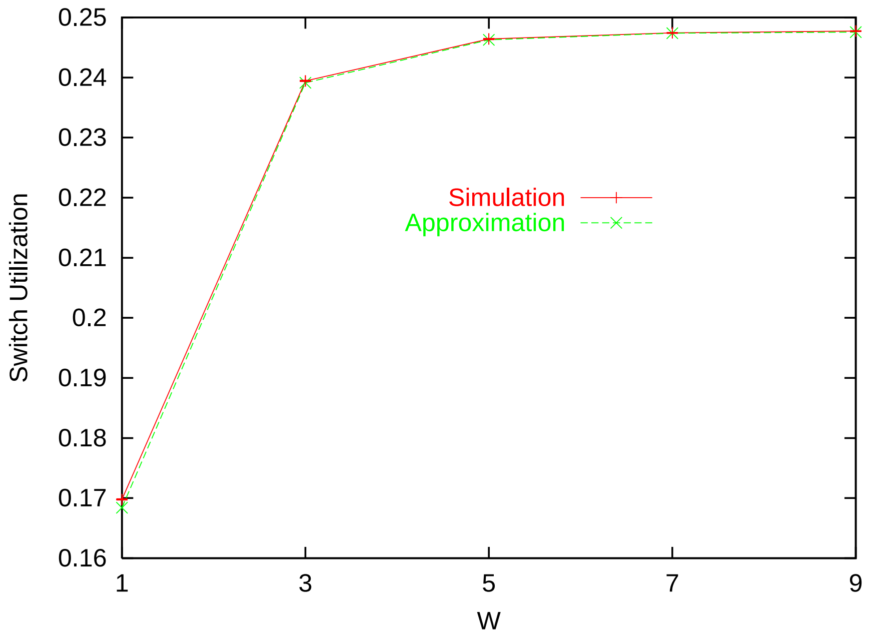

Figure 4.33: Switch utilization vs. W

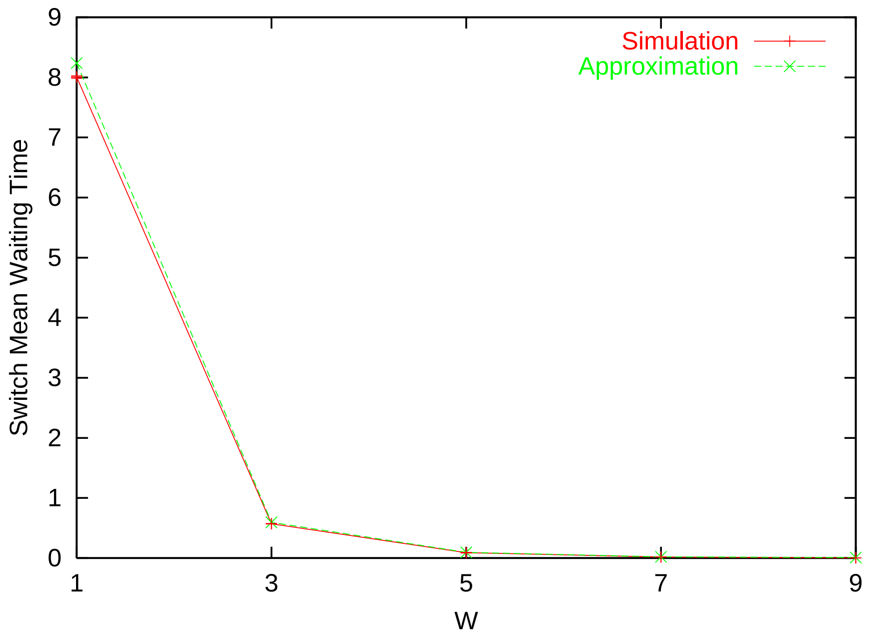

Figure 4.34: Switch mean waiting time vs. W

processes have the same parameters, then we can construct a queueing network with less than NW classes. For example, in these two experiments, the burst arrival processes associated with the W wavelengths of the same user have the same parameters, we can construct a queueing network with only $N = 8$ classes, with $W = 9$ customers per class. In this case, the algorithm will take much less time, because the 8-class queueing network will be decomposed into 8 two-class queueing networks, and then only 8 two-class queueing networks need to be solved.

4.8.5 Limiting Cases

In this subsection, we consider the limiting cases discussed in section 4.7. Since we already discussed the limiting case when the orbiting time is infinite in subsection 4.8.1, we only discuss the limiting case of a single hot spot in this subsection. We assume that the edge switch is equipped with wavelength converters, each fiber supports W burst wavelengths, and the number of the transmission server is W. We

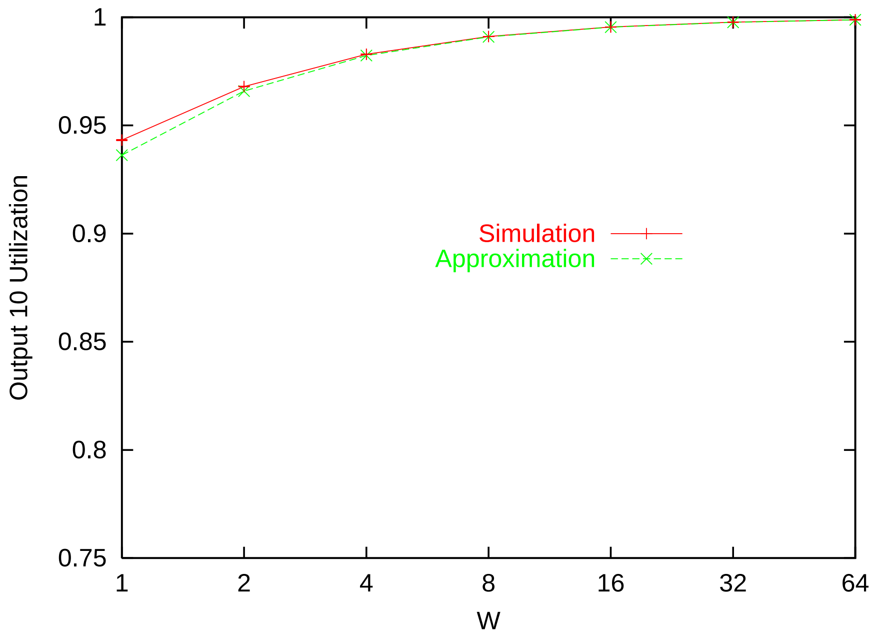

Figure 4.35: Utilization of output port 10 vs. W

also assume that all wavelengths have the same burst arrival process. That is, the queueing network has a single class.

Limited by the simulation time, we only increase W from 1 to 64. We set $\phi_s = 1$, $\phi_l = 0.01$, $\gamma = 0.1$, and $p_s = 0.8$. We set p_i to 100% for $i = 10$, and to 0 for all $i \neq 10$. We also set ω=0.1. Figure 4.35 shows the utilization of output port 10. We observe that the utilization increases as the number of wavelength increases, and it gets close to 1 after $W \geq 16$. Figure 4.36 shows the mean waiting time of the output port 10. Figure 4.37 shows the throughput of output port 10. We observe that the approximation has a good accuracy. Finally, note that as W increases, the simulation time grows very quickly. When $W = 64$, the simulation takes 3 days, whereas the approximation time takes less than 1 second.

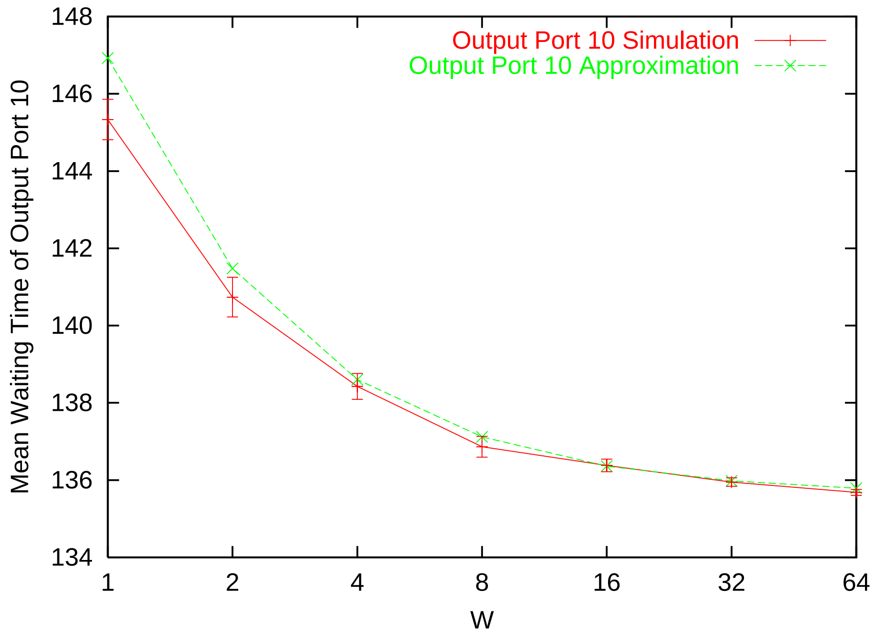

Figure 4.36: Mean waiting time of output port 10 vs. W

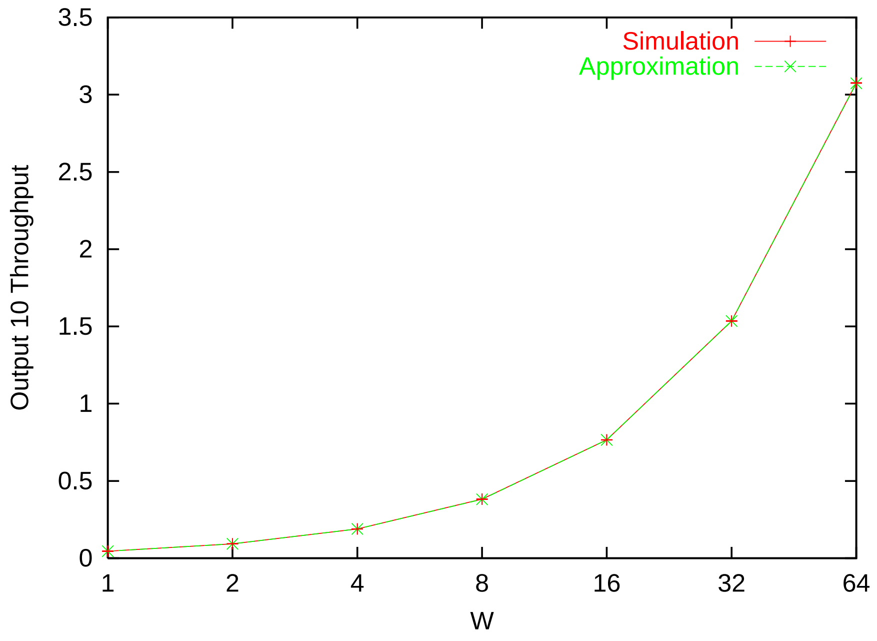

Figure 4.37: Throughput of output port 10 vs. W

4.9 Conclusions

In this chapter, we consider an edge optical burst switching (OBS) node with or without converters, and with no buffering. The OBS node is modeled as a closed non-product form queueing network. Comparisons against simulation data suggest that our algorithm has a reasonable fast speed and a good accuracy, except in the limiting case when the orbiting time is infinite. However, this limiting case is not a practical case. We also analyzed a single hot spot case.

Let us take a look at the experiments described in this chapter with $P = 10$, $N = 8$, and $W = 9$. In the case of the single-class queueing network without wavelength converters, the simulation usually takes half an hour, and the algorithm usually takes 1 second. For the single-class queueing network with wavelength converters, the simulation usually takes 7 hours, and the algorithm usually takes 1 minute. In the case of the multi-class queueing network without wavelength converters, the simulation usually takes half an hour, and the algorithm usually takes 1 second. For the multi-class queueing network with wavelength converters, the simulation usually takes about 10 hours, and the algorithm takes about 4 hours. For the special case, when all traffic goes to a single hot spot, it always takes less than 1 second to get the analytical result using the closed-form expressions described in section 4.7.2.

We note that in our models analyzed in this chapter, we assume that a user continues to transmit the `setup` message until it gets a `setup ack` message back from the edge node. In real life, a user may drop the burst, and go back to the `idle` state, if it receives a `reject` message. If the destination of the next burst is always the same as that of the one just dropped, then the edge node can still be modeled by the queueing network described in this chapter, by setting $\omega = \gamma$. If the destination of the next burst is chosen randomly without any reference to the previous destination, then the edge node can be modeled by a closed queueing network with blocking with repetitive service with random destination, and reversible routing matrix. If we assume that the burst size is exponentially distributed, then this queueing network has a product-form solution. For more details, the reader is referred to Perros [23].

Chapter 5

An Edge Node with a Large Number of Wavelengths

The algorithm described in the previous chapter has a good accuracy for analyzing the performance of an OBS edge node. However, it is still computationally intensive, and it cannot be used to solve an OBS edge node with a very large number of wavelengths, say 128 wavelengths per fiber. In this chapter, we propose an approximation algorithm for the analysis of an edge node with a large number of wavelengths. This chapter is organized as follows. In section 5.1, we present some observations regarding the behavior of the queueing network when the number of the wavelengths increases. A new algorithm, based on these observations, is presented for a single-class and multi-class queueing network in section 5.2 and section 5.3, respectively. Finally, numerical results are shown in section 5.4.

5.1 Observations when the Number of Wavelengths is Large

In this section, we discuss two observations regarding the OBS queueing network, when the number of wavelengths is very large. The algorithms described later in this chapter are based on these two observations.

5.1.1 Exponentially Distributed Burst Size

In this chapter, we will use the second set of parameters, as described in section 3.4, to control the burst arrival process. They are: the load l of the burst arrival process, the mean duration s of bursts, the ratio r of the mean duration of long bursts to the mean duration of short bursts, and the probability p_s that a burst is a short burst.

The first observation is that as the number of the wavelengths increase, the queueing network with the same mean burst size, but with different ratio r and probability p_s, tends to have the same performance. That is, when the number of the wavelengths is large enough, the ratio r and the probability p_s almost do not affect the performance of a queueing network.

Figure 5.1 shows the switch mean waiting time for $r = 1, 10, 100$, as the number of wavelengths W is increased from 2 to 32. The number of users $N = 4$, the number of output (input) ports $P = 5$, and the mean orbiting rate $\omega = 0.1$. All burst arrival processes are identical with load $l = 0.7$, mean burst size $s = 1$, probability $p_s = 0.9$, and probability $p_i = 1.0$ for $i = 5$, and $p_i = 0$ for $i = 1, 2, 3, 4$. We observe that when $W = 2$, the difference between the switch mean waiting times is very large. As W increases, this difference decreases, and when W is equal to 32, the difference becomes very small.

Figure 5.2 shows the switch mean waiting time for $p_s = 0.1, 0.5, 0.9$, as the number of wavelengths W varied from 2 to 32. The parameters of the experiment are the same as the parameters in the previous experiment, with the exception of ratio r which is fixed to 100. We observe that when $W = 2$, the difference between the switch mean

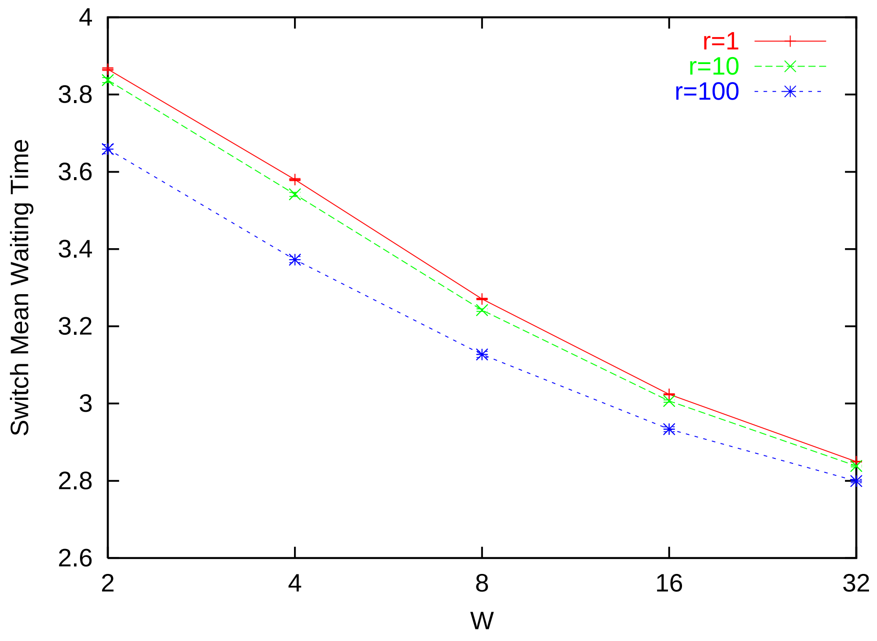

Figure 5.1: Switch mean waiting time for different r vs. W

waiting times is very large. As W increases, this difference decreases, and when W is equal to 32, the difference becomes very small.

Overall, as W increases, the queueing networks with the same mean s, but with different ratio r and probability p_s, tend to have the same performance. If we fix ratio r to 1, and probability p_s to 0.5, then we have a burst arrival process in which the size of both short bursts and long bursts is exponentially distributed with the same mean s. That is, when we analyze an OBS edge switch with a very large number of wavelengths, we can use the exponential distribution to approximate the original hyper-exponential distribution of the burst size. The algorithm for the single-class queueing network described in section 5.2 is based on this observation.

A similar observation was reported in Bolch *et al* [6]. They found using simulation that if arbitrarily distributed service times are replaced by exponentially distributed service times in a closed queueing network, the deviation of the performance measures is tolerable. It is referred to as the `robustness` of the closed queueing network. In their experiments, for a network with only -/G/1 FCFS nodes, if they analyze it by

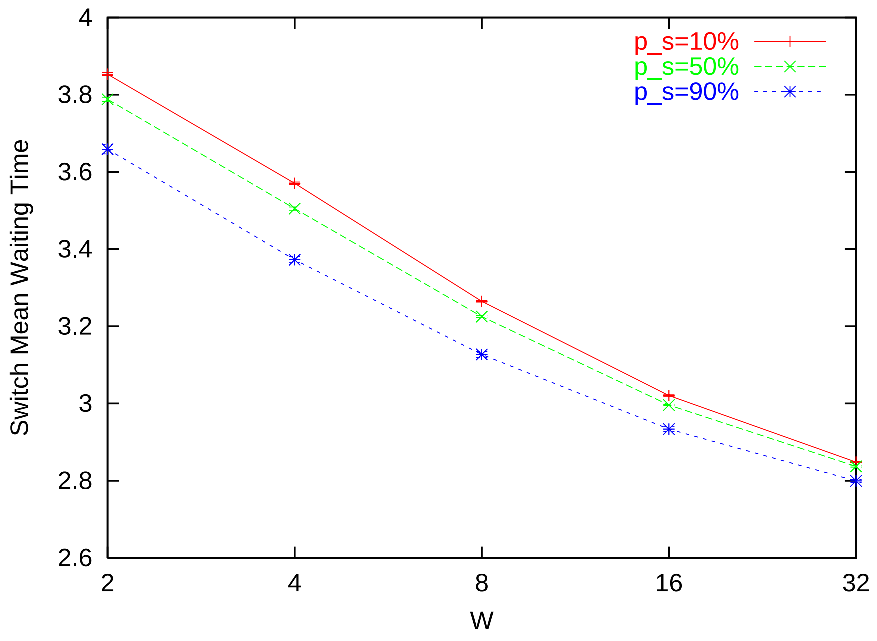

Figure 5.2: Switch mean waiting time for different p_s vs. W

replacing -/G/1 node by -/M/1 node, then the mean deviation is about 6%. For a network with only -/G/m node, the mean deviation is about 2%. Especially, for a network with only -/G/∞ node, the mean deviation is zero. They summarized their observations as follows

- This approximation works better, if the c^2 of the service time is less than 1, and/or is not very large.
- This approximation works better, if there is a bottleneck in the queueing network.
- The larger the number of customers, the better the approximation.

Note that, in our case, as W increases, the number of both the customers and the servers increases.

	user 1	user 2	user 3	user 4
load l	0.2	0.4	0.6	0.8
mean burst size s	2	4	6	8
ratio r	20	40	60	80
probability p_s	0.2	0.4	0.6	0.8
probability p_i	1 if $i = 5$ 0 otherwise	1 if $i = 5$ 0 otherwise	1 if $i = 5$ 0 otherwise	1 if $i = 5$ 0 otherwise

Table 5.1: Parameters of the burst arrival processes

5.1.2 Arrivals See the Same Mean Waiting Time

The second observation is for the OBS node with multiple wavelengths, each associated with different burst arrival processes and converters. This node is modeled by a queueing network with $N \times W$ classes, with one customer per class. For presentation purposes, we number sequentially all the classes of the queueing network, so that the class corresponding to wavelength j $(j = 1, 2, \cdots, W)$ of user i $(i = 1, 2, \cdots, N)$ is identified by the number $(j-1) \times N + i$. For example, the class corresponding to wavelength 1 of user 1 is class 1, and the class corresponding to wavelength 1 of user 3 is class 3. We observed that as the number of the wavelengths increases, the mean waiting time of a class at a node $i = 1, 2, \cdots, P$ becomes closer to the mean waiting time of all classes at the same node.

Figure 5.3 shows the mean waiting time of class 1, 2, 3, 4 and of all classes at output port 4, as the number of wavelengths W is increased from 2 to 32. The number of users $N = 4$, the number of output (input) ports $P = 5$, and the mean orbiting rate $\omega = 0.1$. For simplicity, we assume that the wavelengths of the same user have identical burst arrival processes. The parameters of the burst arrival processes are listed in Table 5.1.

We observe that when $W = 2$, the difference between the mean waiting time of different classes is very large. As W increases, this difference decreases. When W is equal to 32, the mean waiting times of class 1, 2, 3, and 4 all become very close to the mean waiting time of all classes. This observation implies that when we analyze an OBS edge switch with a very large number of wavelengths, we can use a single-

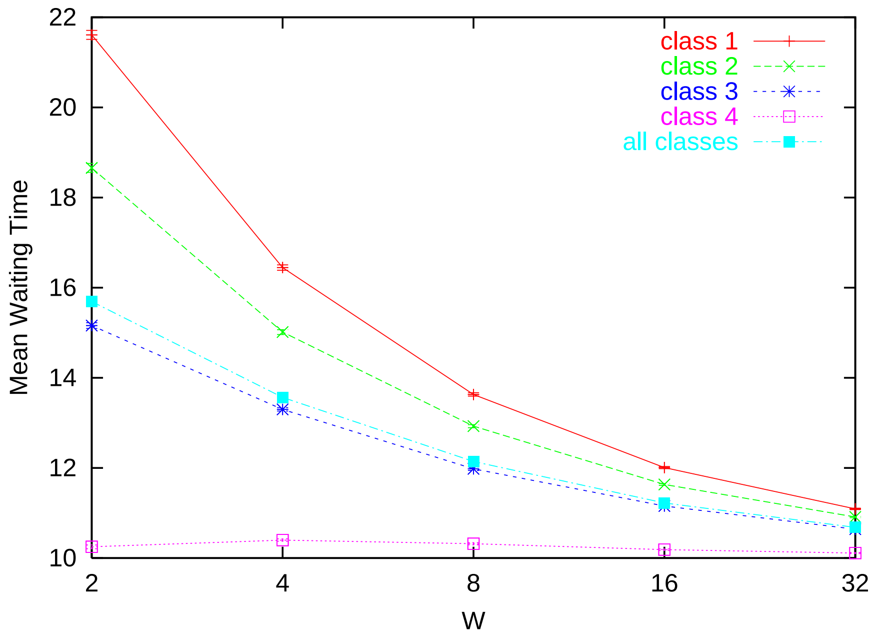

Figure 5.3: Mean waiting time of different classes vs. W

class queueing network to approximate the multi-class queueing network described in the previous chapter. The algorithm for a multi-class queueing network described in section 5.3 is based on this observation.

This observation can be intuitively explained as follows. For clarity and simplicity, suppose we have a product-form closed queueing network with two classes and with a population vector (n_1, n_2), where n_1 and n_2 are the number of customers of class 1 and 2, respectively. The closed queueing network consists of M/M/1 FIFO nodes and M/G/∞ nodes. At M/M/1 nodes, both classes have the same service time distribution, but at M/G/∞, classes have different service time distributions. Now, let's look at a M/M/1 node i, and suppose the service rate at node i is μ_i.

Let $W_{ij}(n_1, n_2)$ denote the mean waiting time of a class-j customer at node i in a network with a population vector (n_1, n_2), and $N_i(n_1, n_2)$ denote the mean customer number at node i in a network with a population vector (n_1, n_2). By `arrival theorem`, the distribution of the number of customers seen by a class-j customer at the time of arrival to node i is the same as that of the number of customers at node

i with one less class-j customers in the network. The mean waiting time of a class-1 customer and a class-2 customer can be obtained as follows.

$$\begin{aligned} W_{i1}(n_1,n_2) &= N_i(n_1-1,n_2)/\mu_i \\ W_{i2}(n_1,n_2) &= N_i(n_1,n_2-1)/\mu_i \end{aligned} \tag{5.1}$$

In a special case, where $n_1 = 1$ and $n_2 = 1$, the mean waiting time of a class-1 customer and a class-2 customer can be obtained as follows.

$$\begin{aligned} W_{i1}(1,1) &= N_i(0,1)/\mu_i \\ W_{i2}(1,1) &= N_i(1,0)/\mu_i \end{aligned} \tag{5.2}$$

A population vector $(0,1)$ means that there is only a class-2 customer in the network, and a population vector $(1,0)$ means that there is only a class-1 customer in the network. Since class 1 and 2 have different service time distribution at M/G/∞ nodes, $N_i(0,1)$ is different from $N_i(1,0)$. Therefore, $W_{i1}(1,1)$ is different from $W_{i2}(1,1)$.

In another extreme case, where $n_1 = 1000$ and $n_2 = 1000$, the mean waiting time of a class-1 customer and a class-2 customer can be obtained as follows.

$$\begin{aligned} W_{i1}(1000,1000) &= N_i(999,1000)/\mu_i \\ W_{i2}(1000,1000) &= N_i(1000,999)/\mu_i \end{aligned} \tag{5.3}$$

A population vector $(999,1000)$ means that there are a total of 1999 customers in the network, including 999 class-1 customers and 1000 class-2 customers. A population vector $(1000,999)$ means that there are a total of 1999 customers in the network, including 1000 class-1 customers and 999 class-2 customers. We can see that there are 1998 common customers between these two population vectors. That is, 1998/1999=99.95% customers are same in both population vectors. Therefore, intuitively, the percentage difference between $N_i(999,1000)$ and $N_i(1000,999)$ is smaller than the percentage difference between $N_i(0,1)$ and $N_i(1,0)$. Then, intuitively, the percentage difference between $W_{i1}(1000,1000)$ and $W_{i2}(1000,1000)$ is smaller than the percentage difference between $W_{i1}(1,1)$ and $W_{i2}(1,1)$. That is, the mean waiting time of different classes tends to become same as the number of customers increases.

5.2 Analysis of the Single-Class Queueing Network

This queueing network is a non-product form queueing network, and we analyze it using Marie's algorithm [17, 18]. The idea in Marie's method is to replace each non-BCMP node by an equivalent node with a load-dependent exponential service rate, obtained by calculating the conditional throughput of the non-BCMP node in isolation under a load-dependent arrival rate. In the following subsection, we calculate the conditional throughput of each node i, $i = 1, \cdots, P$. Node 0 is an infinite server, that is, a BCMP node, so we do not need to construct a flow equivalent node for it.

5.2.1 The Conditional Throughput

Let us consider node i, $i = 1, \cdots, P$, of the queueing network shown in Figure 5.4. Let $\lambda_i(n_i)$ be the arrival rate into this node when there are a total of n_i customers in the node. We assume that the service time of the transmission server is exponentially distributed with mean $1/\mu_i$, i.e. $1/\mu_i = s$. The state of node i can be described by $(n_i^{(t)}, n_i^{(o)})$, where $n_i^{(t)} = 0, 1, .., W$ indicates the number of busy transmission servers, and $n_i^{(o)} = 0, 1, .., (N-1)W$ gives the number of orbiting customers occupying the infinite server. In order to simplify the notation and since we are only concerned with the analysis of node i in isolation, we drop the index i throughout the rest of this subsection.

Let $v(n)$ denote the conditional throughput of the node with n customers. It can be obtained as follows. (If there is only one transmission server, $v(n)$ can be obtained by theorem 2)

$$v(n) \;=\; \frac{p(n-1)\lambda(n-1)}{p(n)} \tag{5.4}$$

Where $p(n)$ is the steady-state probability that there are a total of n customers in the node. It can be calculated as follows:

$$p(n) \;=\; \sum_{n^{(t)}+n^{(o)}=n} p(n^{(t)}, n^{(o)}) \tag{5.5}$$

Where $p(n^{(t)}, n^{(o)})$ is the steady-state probability of the state $(n^{(t)}, n^{(o)})$. It can be obtained by solving the node numerically. For $N = 2$ and $W = 2$, the state transition

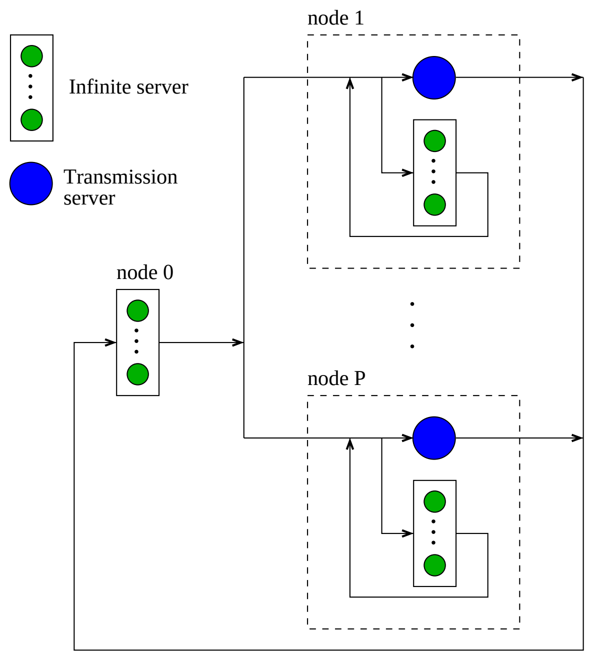

Figure 5.4: Queueing model of an edge switch

diagram is shown in Figure 5.5, and the transition rate matrix **Q** is shown in Table 5.2. We note that it is a block tri-diagonal matrix, and each diagonal block is a tri-diagonal matrix. Therefore, we use block Gauss-Seidel method, see Stewart [26], to solve it. Note that, since each diagonal block is a tri-diagonal matrix, it is very easy to get its LU decomposition, and then each block equation can be solved by a forward and backward substitution.

5.2.2 The Iterative Algorithm

The queueing network is analyzed following Marie's algorithm as follows:

- step 1: Initialize the service rate $\mu_i(n_i)$ of flow equivalent server i, $i = 1, 2, \cdots, P$, to n_i/s, and set the service rate $\mu_0(n_0)$ of flow equivalent server 0 to γn_0.
- step 2: For each node i, $i = 1, 2, \cdots, P$, do the following steps:
  - step 2.1: Calculate the arrival rate $\lambda_i(n_i)$ of node i by short-circuiting node

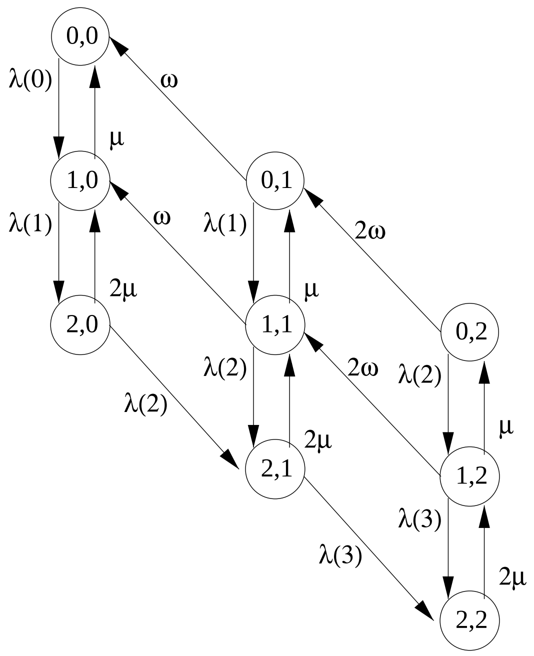

Figure 5.5: State transition diagram of the node

	00	10	20	01	11	21	02	12	22
00	$-\lambda(0)$	$\lambda(0)$							
10	μ	$-\mu-\lambda(1)$	$\lambda(1)$						
20		2μ	$-2\mu-\lambda(2)$			$\lambda(2)$			
01		ω		$-\omega-\lambda(1)$	$\lambda(1)$				
11			ω	μ	$-\omega-\mu-\lambda(2)$	$\lambda(2)$			
21					2μ	$-2\mu-\lambda(3)$			$\lambda(3)$
02					2ω		$-2\omega-\lambda(2)$	$\lambda(2)$	
12						2ω	μ	$-2\omega-\mu-\lambda(3)$	$\lambda(3)$
22								2μ	-2μ

Table 5.2: Transition rate matrix Q of the node

i in the substitute product-form closed queueing network, where each node j has an exponential service time of $\mu_j(n_j)$.

  - step 2.2: Calculate the steady-state probability $p_i(n_i^{(t)}, n_i^{(o)})$ of node i using the block Gauss-Seidel method.
  - step 2.3: Calculate the conditional throughput $\upsilon_i(n_i)$ of node i using equation 5.4.

- step 3: Check the following two convergence conditions. If both convergence criterions are satisfied, then stop. Otherwise, set $\mu_i(n_i)$ to $\upsilon_i(n_i)$ for all $i = 1, 2, \cdots, P$, and go back to step 2.

  - Convergence condition 1: to make sure that the sum of the mean number of the customers at all nodes is equal to the total number of the customers in the queueing network.

$$\left| \frac{NW - \sum_{i=0}^{P} \sum_{j=0}^{NW} j p_i(j)}{NW} \right| < \epsilon \tag{5.6}$$

  - Convergence condition 2: to make sure that the conditional throughputs of each node are consistent with the topology of the queueing network.

$$\left| \frac{r_i - \frac{1}{P+1} \sum_{j=0}^{P} r_j}{\frac{1}{P+1} \sum_{j=0}^{P} r_j} \right| < \epsilon \quad \text{for } i = 0, 1, \cdots, P \tag{5.7}$$

where

$$\begin{aligned} r_i &= \tfrac{1}{p_i} \sum_{j=0}^{NW} p_i(j)\mu_i(j) && \text{if } i = 1, \cdots, P \\ r_i &= \sum_{j=0}^{NW} p_i(j)\mu_i(j) && \text{if } i = 0 \end{aligned}$$

5.2.3 The Mean Waiting Time

The mean waiting time $T_i^{(w)}$ of a customer at node i, $i = 1, 2, \cdots, P$, is the average time from the instance when the customer enters the node to the instance when the customer gets service at the transmission server. The mean response time $T_i^{(r)}$ of a customer at node i, $i = 1, 2, \cdots, P$, is the average time from the instance when the customer enters the node to the instance when the customer leaves the node. We have

$$T_i^{(w)} = T_i^{(r)} - 1/\mu_i \tag{5.8}$$

The mean response time $T_i^{(r)}$ can be obtained by the Little's law as follows.

$$T_i^{(r)} = N_i^{(r)}/\lambda_i \tag{5.9}$$

Where $N_i^{(r)}$ is the mean number of customers in node i, and λ_i is the mean arrival rate into node i. They can be obtained as follows.

$$N_i^{(r)} = \sum_{j=0}^{(N-1)W} \sum_{k=0}^{W} (k+j) p_i(k,j) \tag{5.10}$$

$$\lambda_i = \sum_{j=0}^{(N-1)W} \sum_{k=0}^{W} \lambda_i(k+j) p_i(k,j) \tag{5.11}$$

Note that, the utilization of the transmission server at node i is $\lambda_i/(W\mu_i)$. It can also be obtained as follows.

$$\frac{1}{W} \sum_{j=0}^{(N-1)W} \sum_{k=0}^{W} k p_i(k,j) \tag{5.12}$$

Therefore, we have

$$\lambda_i/(W\mu_i) = \frac{1}{W} \sum_{j=0}^{(N-1)W} \sum_{k=0}^{W} k p_i(k,j) \tag{5.13}$$

We solve it for μ_i, and we get

$$1/\mu_i = \frac{1}{\lambda_i} \sum_{j=0}^{(N-1)W} \sum_{k=0}^{W} k p_i(k,j) \tag{5.14}$$

By substituting equation 5.9 and 5.14 to equation 5.8, we get

$$T_i^{(w)} = \frac{1}{\lambda_i} \sum_{j=0}^{(N-1)W} \sum_{k=0}^{W} j p_i(k,j) \tag{5.15}$$

5.3 Analysis of the Multi-Class Queueing Network

In this section, we extend our analysis of the queueing network model of the edge OBS switch to the case where each customer has a different burst arrival process. This is taken into account by associating each customer with a different class. The resulting queueing network is a closed non-product form queueing network with multiple classes, each of which has only a single customer. The number of classes is $N \times W$. For example, the number of classes in the queueing network model of an OBS node with 8 users and 128 wavelengths is 1024.

In section 5.1, we observed that as the number of the wavelengths increases, the mean waiting time of a class at a node becomes closer to the mean waiting time of all classes at the node. This observation implies that when we analyze a queueing network with a very large number of classes, we can use a single-class queueing network to approximate the multi-class queueing network. Below, in subsection 5.3.1, we describe an aggregation technique for constructing a single-class queueing network, which is equivalent to the multi-class queueing network under study, and in subsection 5.3.2, we present an iterative algorithm for analyzing the multi-class queueing network.

5.3.1 Class Aggregation

For the equivalent single-class queueing network, we have to specify the branching probability $p_i^{(agg)}$ that a customer leaving node P will enter node i, and the mean service rate $\mu_i^{(agg)}$ at node i, $i = 0, 1, 2, \cdots, P$.

Let $p_i^{(j)}$ denote the branching probability that a class j ($j = 1, 2, \cdots, NW$) customer leaving node P will enter node i in the original multi-class queueing network, $E(B^{(j)})$ denote the mean burst size of class j, and $1/\gamma^{(j)}$ denote the mean duration of the idle state of class j.

Suppose we know the mean waiting time $T_i^{(w)}$ of all class customers at node i, then we can approximately calculate the throughput $H^{(j)}$ of class j in the network as follows.

$$H^{(j)} \;=\; 1/(\frac{1}{\gamma^{(j)}} + \sum_{i=1}^{P} (T_i^{(w)} + E(B^{(j)}))p_i^{(j)}) \tag{5.16}$$

In the equivalent single-class queueing network, the branching probabilities $p_i^{(agg)}$, $i = 1, \cdots, P$, can be calculated as follows.

$$p_i^{(agg)} \;=\; \frac{\sum_j H^{(j)} p_i^{(j)}}{\sum_j H^{(j)}} \tag{5.17}$$

The mean service rate $\mu_i^{(agg)}$, $i = 1, \cdots, P$ can be obtained as follows.

$$1/\mu_i^{(agg)} \;=\; \frac{\sum_j H^{(j)} p_i^{(j)} E(B^{(j)})}{\sum_j H^{(j)} p_i^{(j)}} \tag{5.18}$$

The mean service rate $\mu_0^{(agg)}$ can be obtained as follows.

$$1/\mu_0^{(agg)} \;=\; \frac{\sum_j H^{(j)}/\gamma^{(j)}}{\sum_j H^{(j)}} \tag{5.19}$$

5.3.2 The Iterative Algorithm

As shown in the previous subsection, if we know the throughput of each class in a multi-class queueing network, we can aggregate it into a single-class queueing network. However, in order to calculate the throughput of each class in the multi-class queueing network, we need the mean waiting time of all class customers, which can be obtained by solving the single-class queueing network. Therefore, we use an iterative algorithm to solve the multi-class queueing network as follows.

- step 1: Initialize the throughput $H^{(j)}$ of each class j to 1.
- step 2: Construct the equivalent single-class queueing network by aggregating all classes into one class using equations 5.17, 5.19, and 5.18.
- step 3: Solve the equivalent single-class queueing network using the algorithm described in section 5.2.

- step 4: Calculate the mean waiting time of all class customers using equation 5.15
- step 5: Set $H^{(j)}_{old}$ to $H^{(j)}$, $j = 1, 2, \cdots, NW$
- step 6: Calculate $H^{(j)}$ using equation 5.16, $j = 1, 2, \cdots, NW$
- step 7: Check whether the throughput $H^{(j)}$ satisfy the following convergence criterion. If yes, then stop. Otherwise, go to step 2.

$$\sqrt{\sum_j (H^{(j)} - H^{(j)}_{old})^2} \Big/ \sqrt{\sum_j \left(H^{(j)}\right)^2} < \epsilon \tag{5.20}$$

5.4 Numerical Results

In this section, we examine the accuracy of our approximation algorithm, by comparing the approximate results to results obtained from a simulation program of an edge OBS switch. In all the figures given in this section, simulation results are plotted along with 95% confidence intervals estimated by the method of batch mean. The number of batches was set to 30, with each batch run lasting until each wavelength has transmitted at least 100,000 bursts. As the reader will notice, however, most confidence intervals are very narrow and are barely visible in these figures.

The edge switch that we consider is equipped with wavelength converters, and has 8 fiber links for users and 2 other links connecting it to 2 other OBS nodes. For the approximation algorithm, we increase the number of wavelength W in each fiber from 2 to 4, 8, 16, 32, 64, and 128. However, limited by the running time of the simulation program, we only run simulations with W equal to 2, 4, 8, 16, and 32.

As described in section 4.3, we assume that there is no traffic from the other two OBS nodes to this edge switch. We only model traffic generated by the users, which is destined to the 2 OBS nodes, and to the users themselves. We have $P + 1$ nodes in our queueing network model, where $P = 10$. Nodes 1 to 8 represent the output ports connected to the users, nodes 9 and 10 represent the output ports connected to the other two OBS nodes, and node 0 represents the burst arrival processes in the idle state. When all wavelength are associated with the identical burst arrival process, the corresponding queueing network is a single-class queueing network. When all wavelengths are associated with different burst arrival processes, the corresponding queueing network has $N \times W = 8 \times W$ classes and one customer per class.

Traffic Load

In this subsection, we describe three experiments. In the first experiment, we assume that all wavelengths are associated with the same burst arrival process with the following parameters: load $l = 0.1$, mean burst size $s = 1$, burst size ratio $r = 100$, short burst probability $p_s = 0.9$, and destination probability $p_i = 0.3$ if $i = 9, 10$, otherwise $p_i = (1 - 0.3 - 0.3)/8 = 0.05$. The mean orbiting time $1/\omega$ is set to be 10 times of the mean burst size (i.e. $1/\omega = 10 * 1 = 10$, or $\omega = 0.1$). The second and the third experiments have the same parameters as the first experiment, except that load l is set to 0.5 and 0.9, respectively.

Figure 5.6 shows the switch mean waiting time obtained from these three experiments. As we expected, we observe that the higher the load, the longer the mean waiting time. We also observe that the mean waiting time decreases, as the number of wavelength increases. Recall that in a switch with wavelength converters, a user has to wait for retransmitting of the *setup* message, if and only if all wavelengths at the destination output port are busy. Intuitively, the larger the number of wavelengths, the smaller the probability that all wavelengths are an output port are busy. Therefore, as the number of wavelengths increases, the mean waiting time decreases.

We observe that in all three experiments, the error decreases as W increases. When $W = 32$, the approximation results are almost the same as the simulation results. We also note that when $W = 32$, the simulation takes about 60 hours, whereas the approximation takes about 30 seconds.

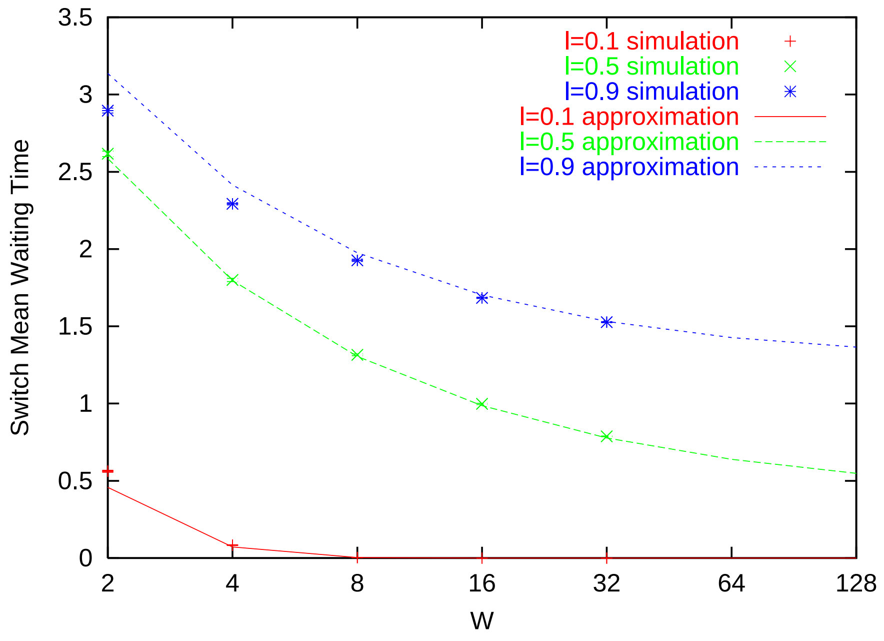

Figure 5.6: Traffic Load: Switch mean waiting time vs. W

Destination Probability

In this subsection, we give the results obtained from three experiments. In the first experiment, we assume that all wavelengths are associated with the same burst arrival process with the following parameters: load $l = 0.9$, mean burst size $s = 1$, burst size ratio $r = 100$, short burst probability $p_s = 0.9$, and destination probability $p_i = 0.1$ if $i = 9, 10$, otherwise $p_i = (1 - 0.1 - 0.1)/8 = 0.1$. The mean orbiting time $1/\omega$ is set to be 10 times of the mean burst size (i.e. $\omega = 0.1$). The second experiment has the same parameters as the first one, except that the destination probability $p_i = 0.2$ if $i = 9, 10$, otherwise $p_i = (1 - 0.2 - 0.2)/8 = 0.075$. In the third experiment, destination probability $p_i = 0.3$ if $i = 9, 10$, otherwise $p_i = (1 - 0.3 - 0.3)/8 = 0.05$.

Figure 5.7 shows the switch mean waiting time for these three experiments. We observe that the larger the p_9 and p_{10}, the longer the mean waiting time. Note that, when p_9 and p_{10} are 0.1, the switch has a uniform traffic. This is, a switch with a hot spot traffic has a longer mean waiting time than a switch with a uniform traffic.

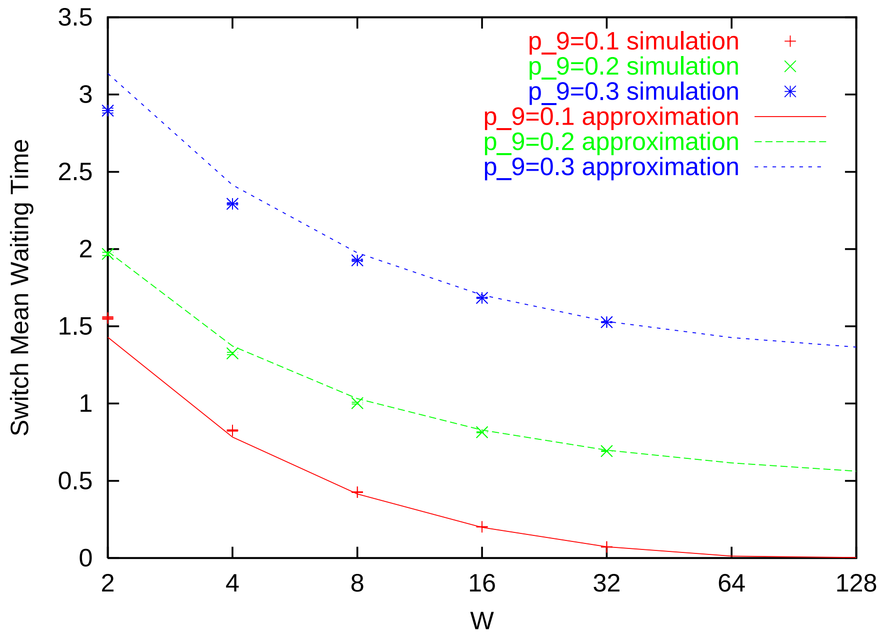

Figure 5.7: Destination Probability: Switch mean waiting time vs. W

We observe that in all three experiments, the error decreases as W increases. When $W = 32$, the approximation results are almost the same as the simulation results. We also note that when $W = 32$, the simulation takes about 60 hours, whereas the approximation takes about 30 seconds.

Orbiting Rate

As before, we give results for three experiments. In all three experiments, we assume that all wavelengths are associated with the same burst arrival process with the following parameters: load $l = 0.9$, mean burst size $s = 1$, burst size ratio $r = 100$, short burst probability $p_s = 0.9$, and destination probability $p_i = 0.3$ if $i = 9, 10$, otherwise $p_i = (1 - 0.3 - 0.3)/8 = 0.05$. The mean orbiting time $1/\omega$ is set to be 1, 10,and 100 times of the mean burst size, respectively in the first, second, and third experiment. That is, $\omega = 1,\ 0.1,\ 0.01$, respectively in the first, second, and third experiment.

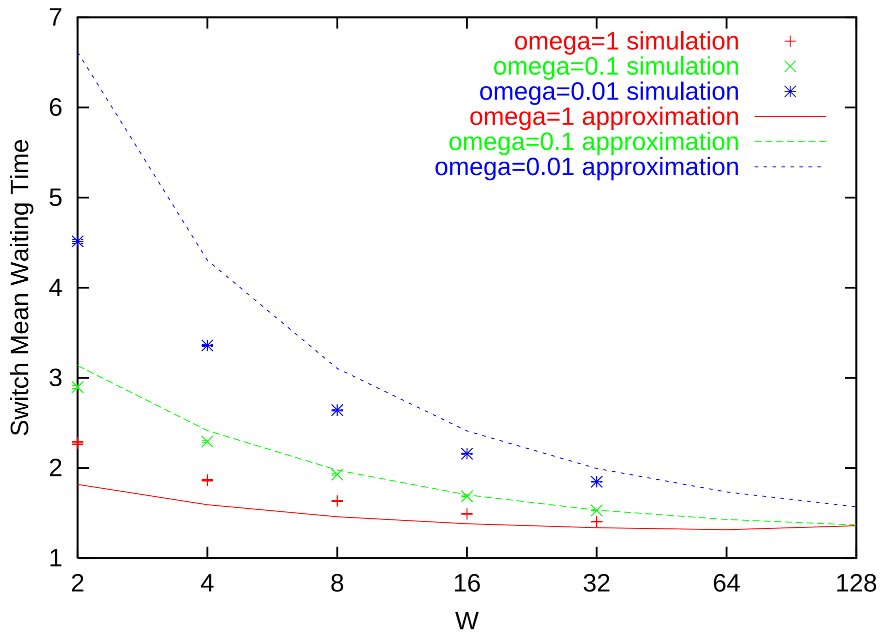

Figure 5.8: Orbiting Rate: Switch mean waiting time vs. W

Figure 5.8 shows the switch mean waiting time for these three experiments. As we expected, we observe that the smaller the orbiting rate (the longer the orbiting time), the longer the mean waiting time.

We observe that in all three experiments, the error decreases as W increases. When $W = 32$, the maximum percentile error is 8%. We also note that when $W = 32$, the simulation takes about 80 hours, whereas the approximation takes about 60 seconds.

Different Burst Arrival Processes

In this subsection, we assume that wavelengths are associated with different burst arrival processes. But for simplicity, we assume that the wavelengths of the same user are associated with the identical burst arrival processes. The parameters are listed in Tabel 5.3. The mean orbiting time $1/\omega$ is set to be 10 times of the mean burst size (i.e. $\omega = 0.1$).

Figure 5.9 shows the mean waiting time of class 2, 4, 6, and 8. Recall that, class

	user 1	user 2	user 3	user 4	user 5	user 6	user 7	user 8
load l	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8
mean burst size s	1	2	3	4	5	6	7	8
ratio r	10	20	30	40	50	60	70	80
probability p_s	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8
probability p_9, p_{10}	0.1	0.1	0.2	0.2	0.3	0.3	0.4	0.4
other p_i	0.1	0.1	0.075	0.075	0.05	0.05	0.025	0.025

Table 5.3: Parameters of burst arrival processes

2, 4, 6, and 8 correspond to the first wavelength of user 2, 4, 6, and 8, respectively. We observe that the error decreases as W increases. When $W = 32$, the maximum percentile error is 12%. We also note that when $W = 32$, the simulation takes about 60 hours, whereas the approximation takes about 5 minutes.

5.5 Conclusions

In this chapter, we consider an edge OBS node with a large number of wavelengths, which is modeled as a closed non-product form queueing network with a large number of classes. Our approximation algorithm is very fast, and has a good accuracy when the number of wavelengths is large. Let us take a look at the experiments described in this chapter with $P = 10$, $N = 8$, and $W = 32$. In the case of the single-class queueing network, the simulation usually takes 60 hours, the algorithm described in the previous chapter usually takes 1 hour, and the approximation algorithm described in this chapter usually takes 1 minute. In the case of the multi-class queueing network, the simulation takes about 60 hours, the algorithm described in the previous chapter takes more than 60 hours, and the approximation algorithm described in this chapter takes 5 minutes.

Finally, we would like to mention that for a node with a very large number of wavelengths, the corresponding closed queueing network has also a very large number of customers (in the hundreds). In this case, we may get underflows or overflows when calculating the normalization constant. That is, the value of the normalization

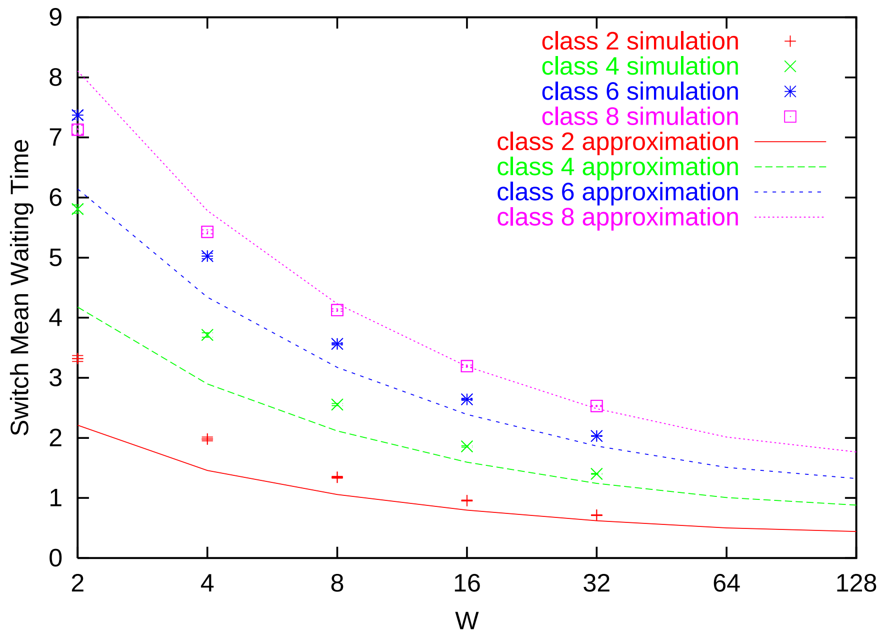

Figure 5.9: Different Burst Arrival Processes: Mean waiting time vs. W

constant may be out of the range of the JAVA `double` type, whose largest positive finite value is 1.8E308, and whose smallest positive value is 4.9E-324. In order to solve this problem, we developed a new JAVA data type `BigDouble`, whose largest positive finite value is 1.8E2147483647, and whose smallest positive value is 4.9E-2147483647.

Chapter 6

Summary and Future Work

6.1 Summary of Research Contributions

In chapter 2, we studied for the first time WDM ring networks employing Optical Burst Switching. The studied OBS ring network consists of N OBS nodes connected by optical fibers, and each fiber can support $N + 1$ wavelength, including one control wavelength and N burst wavelengths. Each node in the OBS ring network is equipped with a tunable receiver and a fixed transmitter to access the burst wavelength. The major traffic loss in the OBS ring network is caused by the receiver collision, which happens when two or more bursts destined to the same node overlap in time.

We proposed a group of access protocols to solve receiver collisions, and these access protocols are simple and easy to implement. We also studied the performance of these access protocols in terms of throughput, packet delay, throughput fairness, and delay fairness under different network parameters: average packet arrival rate, maximum burst size, and minimum burst size. Finally, we proposed a new offset calculation method, which can significantly simplify the access protocol design, and reduce the packet delay for all access protocols.

In chapter 3, we proposed a burst arrival process described by a 3-state Markov process, which is more realistic than the well-known Poisson process. It models two classes of bursts, short bursts and long bursts, and each class of bursts has an exponentially distributed burst size. It can be easily extended to have more than two classes of bursts, where each class of bursts has a Coxian distributed burst size. The burst arrival process captures the burstiness of the arrival process, which is defined as the squared coefficient of the variation of the burst inter-arrival time. It can also be easily extended to capture the auto-correlation of the burst inter-arrival time by allowing back-to-back bursts.

In chapter 4, we analyzed the edge node in a WDM mesh network with Optical Burst Switching. The studied OBS mesh network consists of OBS nodes interconnected by bi-directional fiber links. An OBS node consists of a non-blocking space-division switch fabric, with no optical buffers. Each fiber can support $W + 1$ wavelengths, including one control wavelength and W burst wavelengths. The message between a user and the edge node follows the signaling protocols defined in the `JumpStart` project.

If an edge OBS node is not equipped with converters, a burst on an incoming wavelength can only be switched to the same wavelength on each output port, and bursts arriving to the edge switch on different wavelengths do not interfere with each other. Consequently, the edge node can be decomposed exactly into W sub-systems, one per burst wavelength. Each sub-system is modeled as a closed non-product-form queueing network, which consists of special nodes with orbiting customers. In the case where all incoming burst wavelength have the same burst arrival process, the closed queueing network model can be seen as consisting of a single class of customers. If each incoming burst wavelength has a different burst arrival process, then the queueing network becomes a multi-class queueing network, where each class contains exactly one customer corresponding to an incoming burst wavelength. If an edge OBS node is equipped with converters, the above decomposition of an edge switch into sub-systems per wavelength is no longer possible, since bursts arriving on different wavelengths may interfere with each other. However, the edge switch as a whole can be modeled by a closed queueing network very similar to the one described above.

Despite the rich literature in queueing network analysis, this particular queueing network with orbiting customers has not been analyzed before. We developed algorithms for both the single-class and multi-class queueing networks. The single-class queueing network is solved using Marie's method. In the case of no converters, we got the closed-form solution of the conditional throughputs of the special node with orbiting customers. The multi-class queueing network is analyzed by decomposition. Specifically, a multiple-class queueing network is decomposed into a set of two-class queueing network, and each of them is then solved by Neuse and Chandy's Heuristic Aggregation Method. We also employed a class aggregation technique to reduce the complexity of the analysis of a node, and we used the convolution algorithm to calculate the arrival rate of the aggregate class.

In chapter 5, we studied the OBS edge node with a large number of wavelengths. We found that as the number of wavelengths increases, the queueing networks with the same mean burst size, but with different burst size ratio and short burst probability, tend to have the same performance. We also found that as the number of wavelengths increases, different classes tend to have the same mean waiting time at the same node. Based on these observations, we proposed an approximation algorithm for the analysis of an edge OBS node with a large number of wavelengths.

6.2 Future Work

6.2.1 Performance Analysis of the OBS Ring Networks

In this thesis, we proposed some access protocols for an OBS ring network, and analyzed their performance by simulation. The next step is to study Quality-of-Service (QoS) issues. For example, we may introduce different priorities for different classes of bursts.

6.2.2 Queueing Analysis of the OBS Edge Node

In this thesis, we described a queueing network model for the edge node, and proposed some approximation algorithms to solve it. The next step is to obtain the departure process from the output port of the edge node, which can be used in the analysis of a queueing network representing of the mesh network.

6.2.3 Queueing Analysis of the OBS Mesh Networks

In this thesis, we proposed the algorithms and queueing models for the edge node of an OBS mesh network. The next step is to build a queueing model for an OBS mesh network, and design the approximation algorithm to solve it. For example, the network can be decomposed into small sub-networks, and the arrival processes of each sub-networks are the departure processes of the neighbor sub-networks. Each sub-network is analyzed separately, and the results from the sub-networks are combined together through an iterative scheme.

Appendix A

Fairness of Access Protocols

To further explore the nature of fairness of the access protocols described in chapter 2, we show the throughout from node 0 to all other nodes under RR/R, RR/NP, RR/P and RR/Token with 1.7 Gbps (the throughput from other nodes is very similar) in Figures A.1, A.2, A.3, and A.4, respectively. We observe that under RR/R, the throughput from node 0 to all other nodes is almost identical. The same applies to RR/Token. We also observe that under RR/NP and RR/P, the throughput is the highest for bursts destined to node 1, and it then decreases, as the destination of the burst is further away from node 0. Thus, both RR/NP and RR/P protocol provide better throughput to nodes closer to the source than to nodes far away. This follows directly from the operation of RR/NP and RR/P described in section 2.4.3 and 2.4.2.

To further explore the nature of delay fairness, we show the mean packet queueing delay of each transmit queue in node 0 for all protocols in Figures A.5 to A.9 (the mean queueing delays of packets originating at other nodes are very similar). In Figure A.5, regardless of the destination node, the mean packet queueing delays under RR/R with a 1.7 Gbps average arrival rate are almost identical. Figure A.6 shows the corresponding mean packet queueing delays under RR/NP protocol with a 1.7 Gbps average arrival rate. We observe that the mean queueing delay is the highest for packets destined to node 1 (the node immediately downstream from the

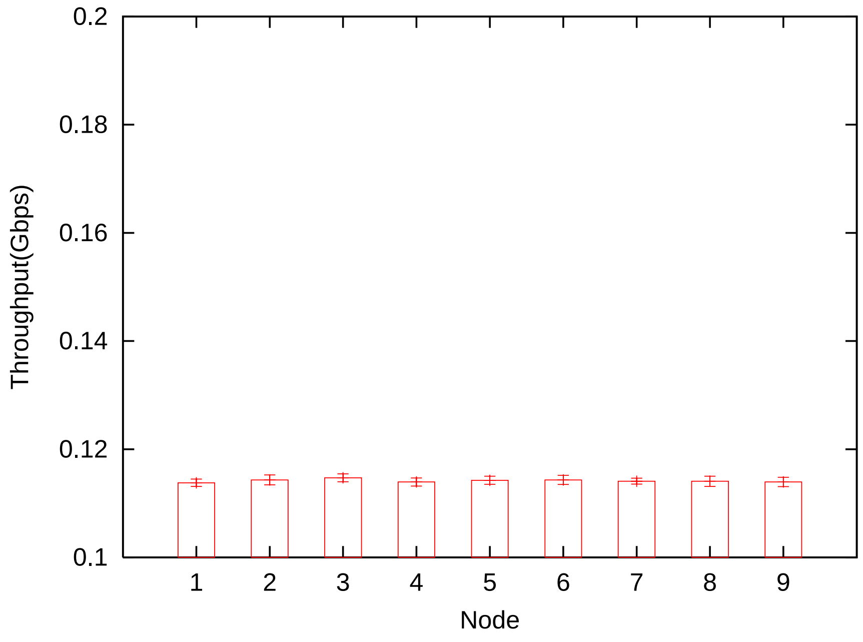

Figure A.1: Throughput from node 0 to other nodes, RR/R, 1.7 Gbps

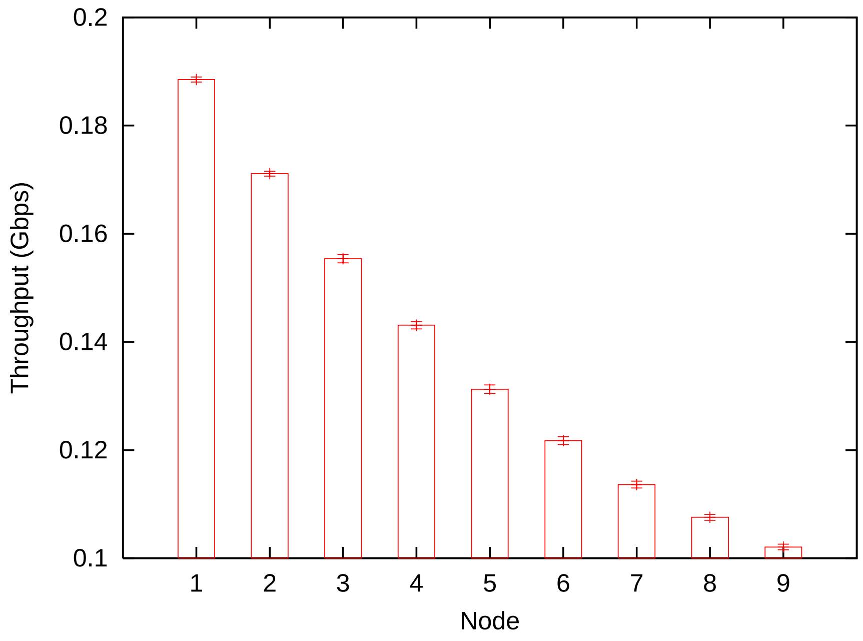

Figure A.2: Throughput from node 0 to other nodes, RR/NP, 1.7 Gbps

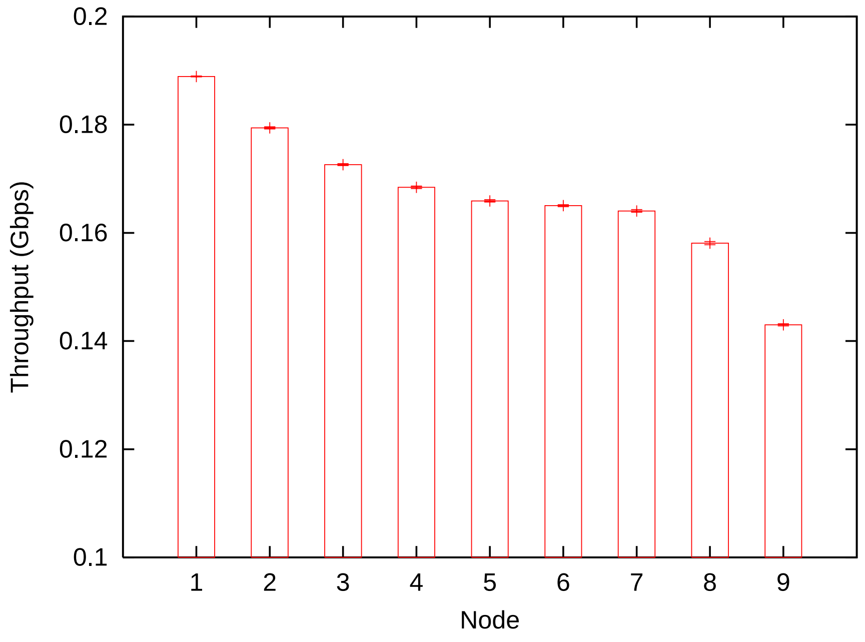

Figure A.3: Throughput from node 0 to other nodes, RR/P, 1.7 Gbps

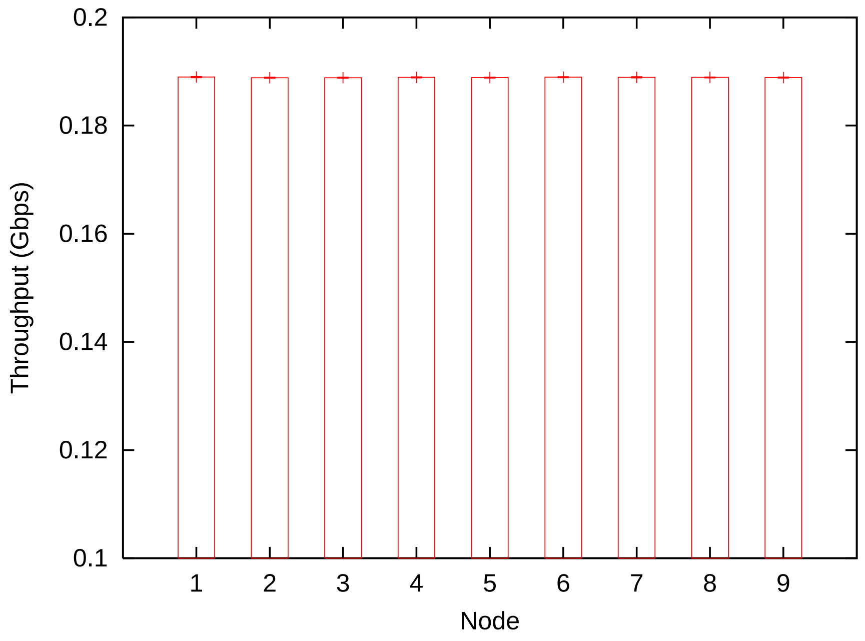

Figure A.4: Throughput from node 0 to other nodes, RR/Token, 1.7 Gbps

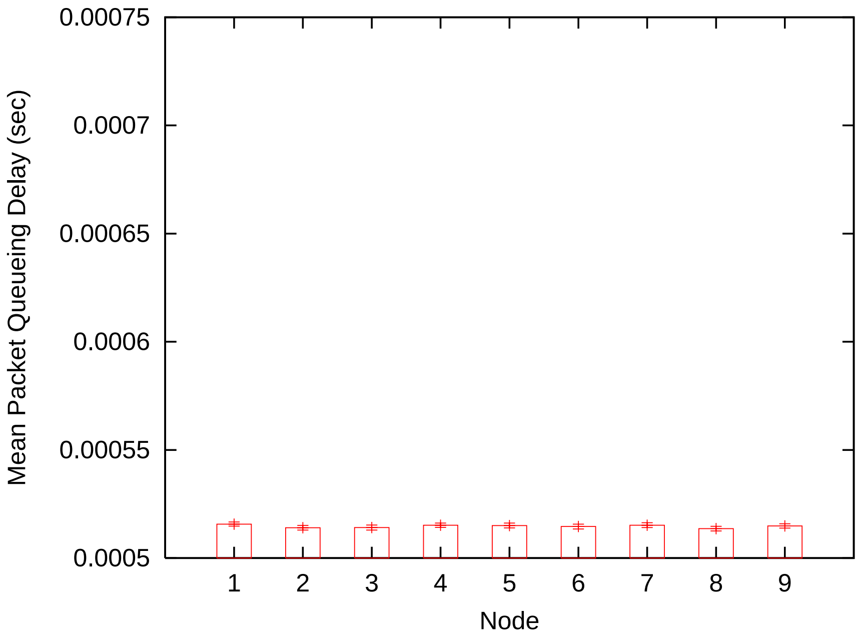

Figure A.5: Mean packet queueing delays in transmit queues in node 0, RR/R, 1.7 Gbps

source node 0), and it decreases as the destination of the packet is further away from node 0. Thus, the RR/NP protocol provides better delay access to wavelengths of nodes far away than to wavelengths of nodes close to the source of a packet. This behavior follows directly from the operation of the RR/NP protocol described in section 2.4.3. Figures A.7 and A.8 show the corresponding mean packet queueing delays under RR/P with 0.8 and 1.7 Gbps average arrival rates, respectively. We observe that the increasing and decreasing patterns in these two figures are not same. That is, RR/P does not always provide the best or worst delay access to a specific node. We also observe that in Figure A.8 the delays are not monotonically increasing with the distance of a node from node 0. Our simulations show that RR/Token does not always provide the best or worst delay access to a specific node, either. The mean packet queueing delay under RR/Token with a 1.7 Gbps average arrival rate is shown in Figure A.9 .

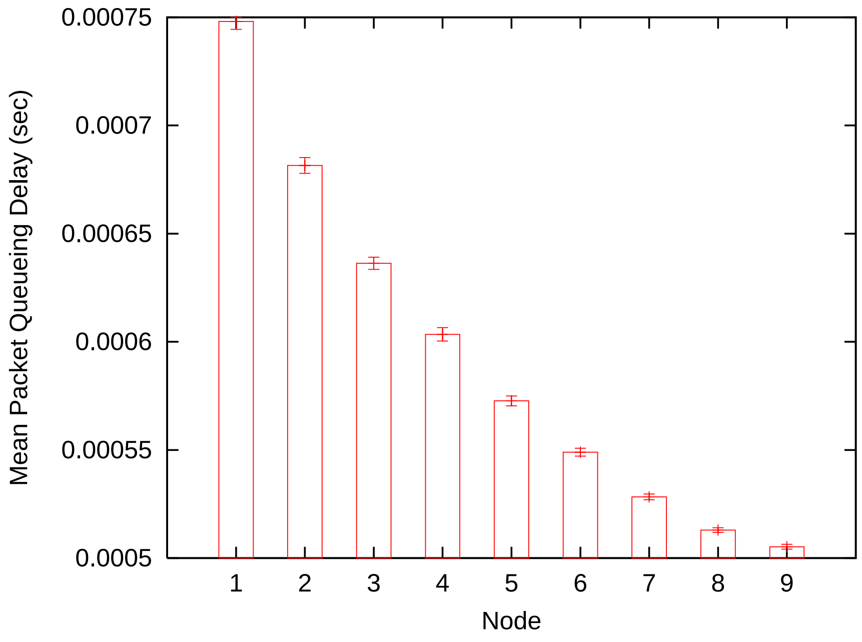

Figure A.6: Mean packet queueing delays in transmit queues in node 0, RR/NP, 1.7 Gbps

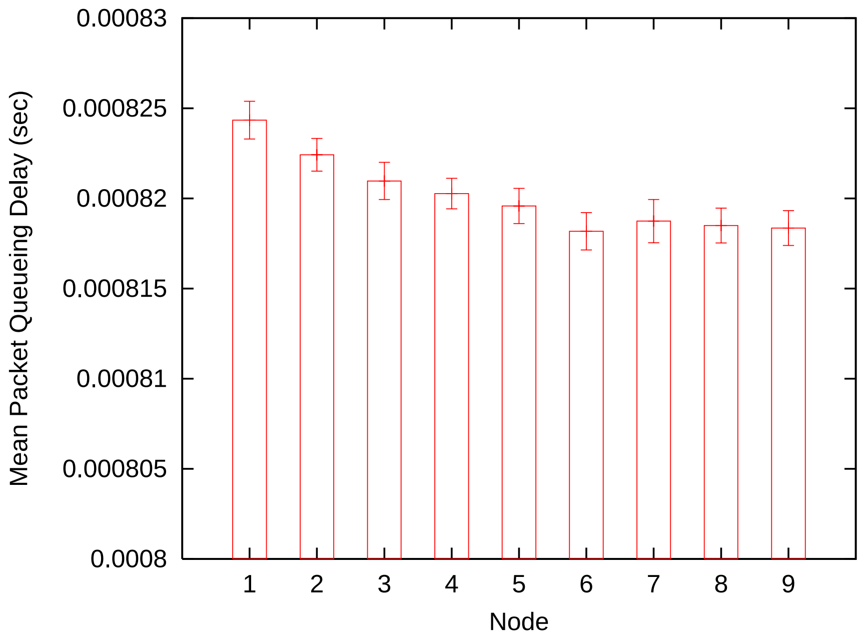

Figure A.7: Mean packet queueing delays in transmit queues in node 0, RR/P, 0.8 Gbps

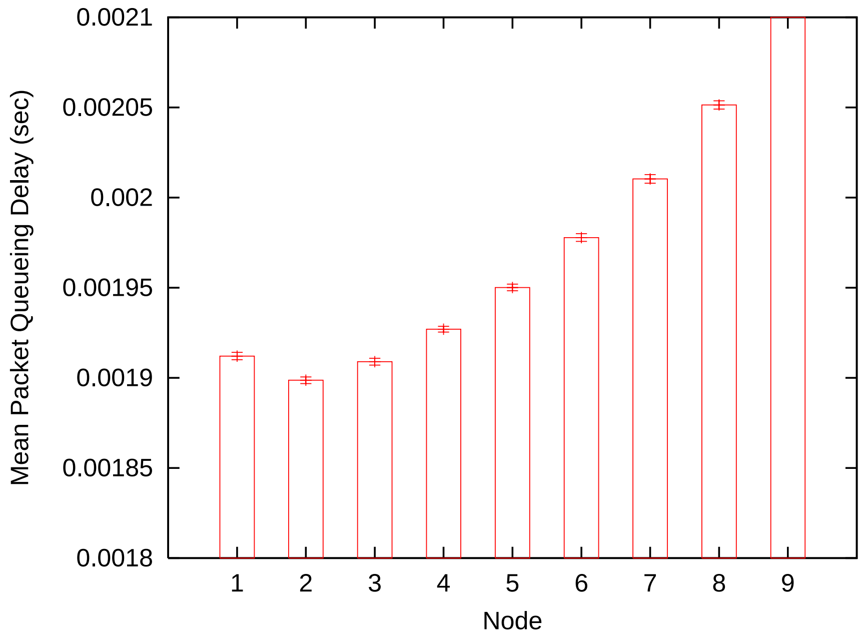

Figure A.8: Mean packet queueing delays in transmit queues in node 0, RR/P, 1.7 Gbps

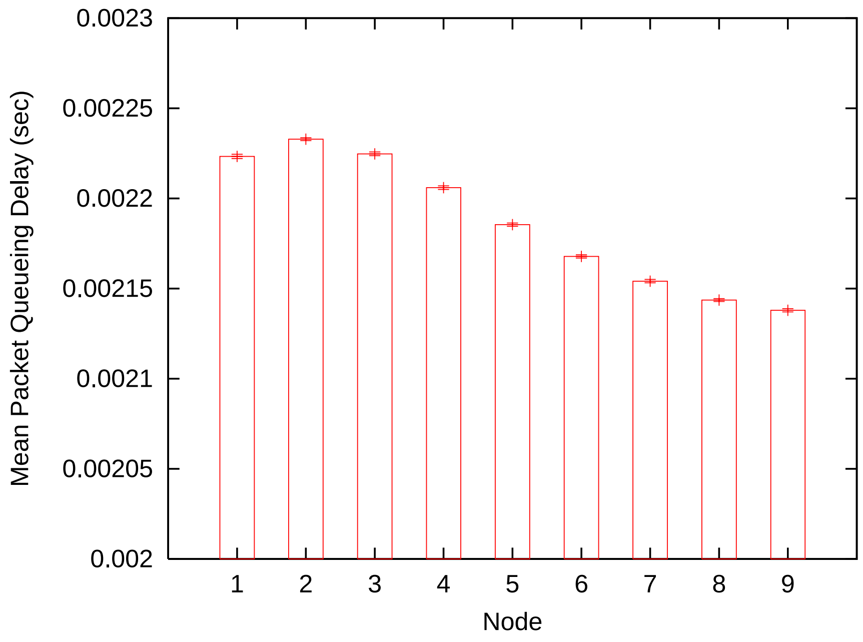

Figure A.9: Mean packet queueing delays in transmit queues in node 0, RR/Token, 1.7 Gbps

Appendix B

$c^2(A)$ and $c^2(B)$ of the Burst Arrival Process

In this appendix, we show how the $c^2(A)$ and $c^2(B)$ of the burst arrival process change with γ, ϕ_s, ϕ_l, and p_s. Figure B.1 shows $c^2(A)$ as a function of γ and ϕ_l. The plot were obtained by setting $\phi_s = 1$ and $p_s = 0.8$. We observe that when γ is very small, $c^2(A)$ is equal to 1 for all ϕ_l. This is because when γ is small enough, the burst arrival process becomes a Poisson process with a rate of γ. We also observe that with a fixed ϕ_l, $c^2(A)$ first decreases and then increases, as we increase γ. Finally, we observe that the larger the difference between ϕ_s and ϕ_l, the larger the $c^2(A)$.

Figure B.2 shows $c^2(B)$ as a function of γ and ϕ_l. The plot were obtained by setting $\phi_s = 1$ and $p_s = 0.8$. We observe that $c^2(B)$ is independent of γ. We also observe that the larger the difference between ϕ_s and ϕ_l, the larger the $c^2(B)$. Finally, we observe that $c^2(B)$ is equal to $c^2(A)$ shown in Figure B.1 when γ is very large. This is because when γ is large enough, the idle state becomes negligible so that the distribution of the burst interarrival times becomes the distribution of the burst durations.

Figures B.3 and B.4 show $c^2(A)$ and $c^2(B)$, respectively, as a function of p_s and

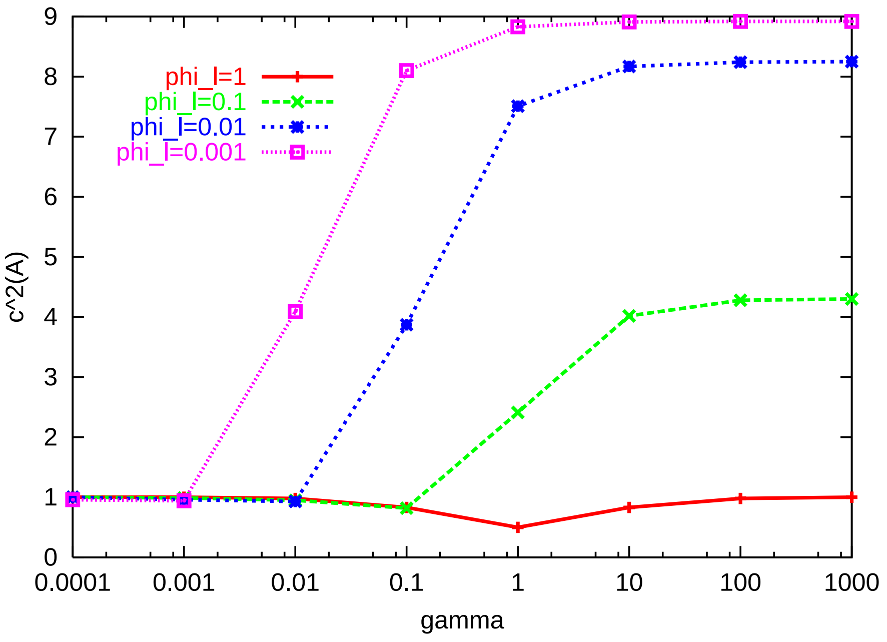

Figure B.1: $c^2(A)$ vs. γ

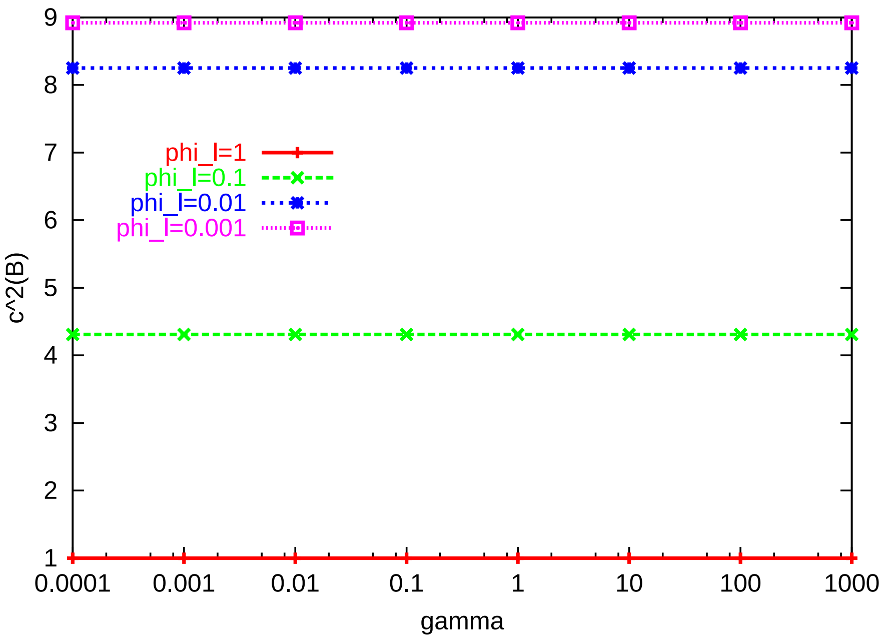

Figure B.2: $c^2(B)$ vs. γ

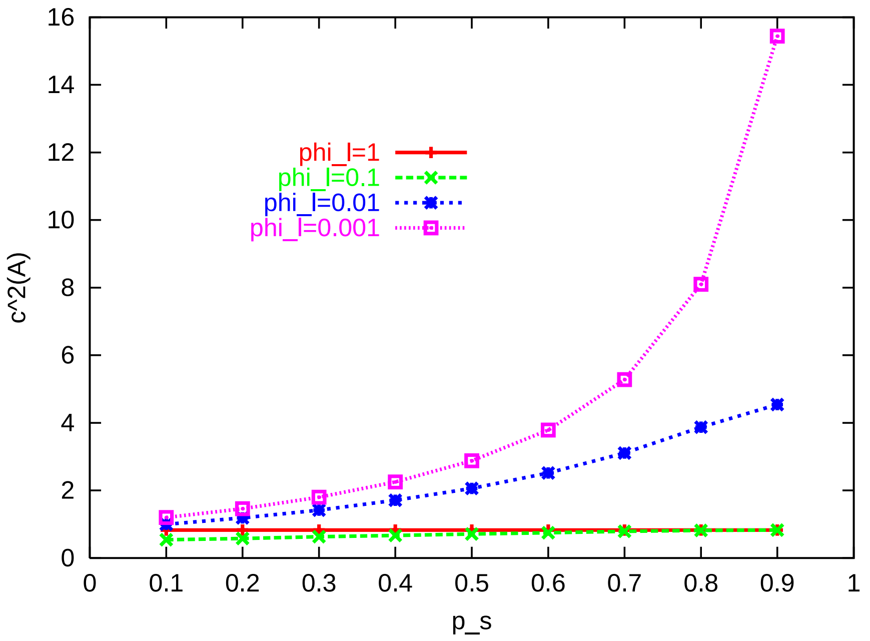

Figure B.3: $c^2(A)$ vs. p_s

ϕ_l, with $\phi_s = 1$ and $\gamma = 0.1$. We observe that the larger the p_s, the larger the $c^2(A)$ and $c^2(B)$.

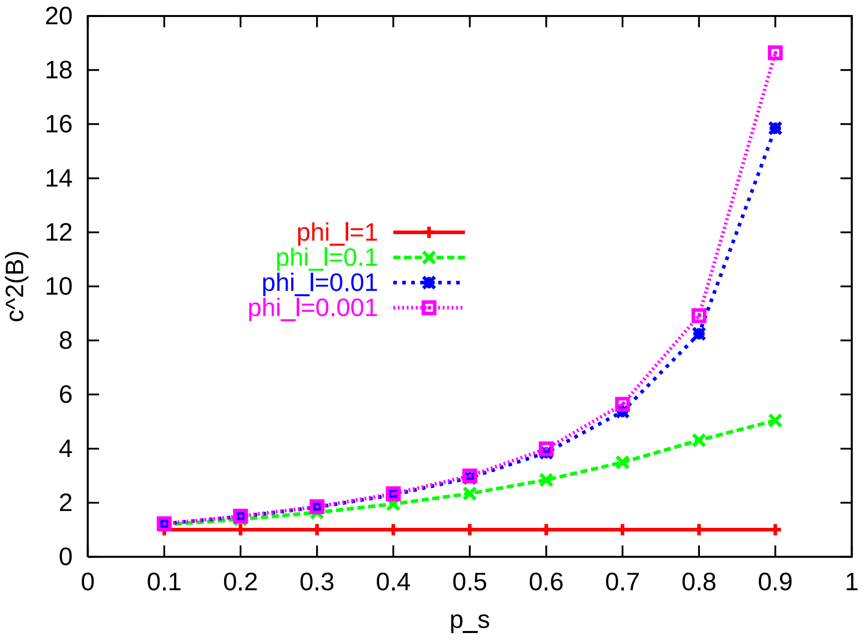

Figure B.4: $c^2(B)$ vs. p_s

Bibliography

[1] The `JumpStart` project. http://jumpstart.anr.mcnc.org.

[2] S. R. Amstutz. Burst switching - an introduction. *IEEE Communication Magazine*, 21(8):36–42, November 1983.

[3] S. R. Amstutz. Burst switching - an update. *IEEE Communication Magazine*, 27(9):50–57, September 1989.

[4] I. Baldine, G. N. Rouskas, H. G. Perros, and D. Stevenson. `JumpStart`: A just-in-time signaling architecture for WDM burst-switched networks. *IEEE Communications*, 40(2):82–89, February 2002.

[5] B. Baynat and Y. Dallery. A product-form approximation method for general closed queueing networks with several classes of customers. *Performance Evaluation*, 24:165–188, 1993.

[6] G. Bolch, S. Greiner, H. Meer, and K. S. Trivedi. *Queueing Networks and Markov Chains*. John Wiley and Sons, INC., Canada, 1998.

[7] J. Cai, A. Fumagali, and I. Chlamtac. The miltitoken interarrival time (mtit) access protocol for supporting variable size packets over wdm ring network. *IEEE Journal on Selected Areas in Communications*, 18(10):2094–2104, October 2000.

[8] F. Callegati, H. C. Cankaya, Y. Xiong, and M. Vandenhoute. Design issues of optical ip routers for internet backbone applications. *IEEE Communications Magazine*, pages 124–128, December 1999.

[9] H. M. Chaskar, S. Verma, and R. Ravikanth. A framework to support IP over WDM using optical burst switching. In *IEEE/ACM/SPIE Optical Network Workshop*, January 2000.

[10] A. Detti, V. Eramo, and M. Listanti. Performance evalution of a new technique for ip support in a wdm optical network: Optical composite burst switching (OCBS). *Journal of Lightwave Technology*, 20(2):154–165, February 2002.

[11] K. Dolzer and C. Gauger. On burst assembly in optical burst switching networks - a performance evaluation of Just-Enough-Time. In *Proceedings of the 17th International Teletraffic Congress*, pages 149–161, September 2001.

[12] K. Dolzer, C. Gauger, J. Späth, and S. Bodamer. Evaluation of reservation mechanisms for optical burst switching. *AEÜ International Journal of Electronics and Communications*, 55(1), January 2001.

[13] M. Düser and P. Bayvel. Analysis of a dynamically wavelength-routed optical burst switched network architecture. *Journal of Lightwave Technology*, 20(4):574–585, April 2002.

[14] W. Fischer and K. Meier-Hellstern. The Markove-Modulated Poisson Process (MMPP) cookbook. *Performance Evaluation*, 18:149–171, 1992.

[15] C. Gauger, K. Dolzer, and M. Scharf. Reservation strategies for FDL buffers in OBS networks. In *IEEE International Conference on Communications*, April/May 2002.

[16] S. Kim, N. Kim, and M. Kang. Contention resolution for optical burst switching networks using alternative routing. In *ICC 2002*. IEEE, April 2002.

[17] R. Marie. An approximate analytical method for general queueing networks. *IEEE Transactions on Software Engineering*, 5(5):530–538, September 1979.

[18] R. Marie. Calculation equilibrium probabilities for $\lambda(n)/C_k/1/N$ queues. *ACM Sigmetrics Performance Evaluation Review*, 9(2):117–125, 1980.

[19] P. Mehrotra, I. Baldine, D. Stevenson, and P. Franzon. Network processor design for use in optical burst switched networks. In *Proceedings of the International ASIC/SOC Conference*, September 2001.

[20] B. Mukherjee. *Optical Communication Networking*. McGraw-Hill, 1997.

[21] D. Neuse and K. M. Chandy. HAM: The heuristic aggregation method for solving general closed queueing network models of computer systems. *Performance Evaluation Review*, 11:195–212, 1982.

[22] V. Paxson and S. Floyd. Wide area traffic: The failure of poisson modeling. *IEEE/ACM Transactions on Networking*, 3(3):226–244, June 1995.

[23] H. Perros. *Queueing Networks with Blocking: Exact and Approximate Solutions*. Oxford University Press, 1994.

[24] H. Perros. *An Introduction to ATM Networks*. Wiley, New York, 2001.

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

[26] W. J. Stewart. *Introduction to the Numerical Solution of Markov Chains*. Princeton University Press, New Jersey, 1994.

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

[28] S. Verma, H. Chaskar, and R. Ravikanth. Optical burst switching: a viable solution for terabit IP backbone. *IEEE Network*, pages 48–53, November/December 2000.

[29] V. M. Vokkarane, J. P. Jue, and S. Sitaraman. Burst segmentation: An approach for reducing packet loss in optical burst switched networks. In *ICC 2002*. IEEE, 2002.

[30] J. Y. Wei and R. I. McFarland. Just-in-time signaling for WDM optical burst switching networks. *Journal of Lightwave Technology*, 18(12):2019–2037, December 2000.

[31] J. Y. Wei, J. L. Pastor, R. S. Ramamurthy, and Y. Tsai. Just-in-time optical burst switching for multiwavelength networks. In *IFIP TC6 WG6.2 Fifth International Conference on Broadband Communications*, pages 339–352. Kluwer Academic Publishers, November 1999.

[32] I. Widjaja. Performance analysis of burst admission control protocols. *IEEE Proceeding Of Communications*, 142:7–14, February 1995.

[33] Y. Xiong, M. Vandenhoute, and H.C. Cankaya. Control architecture in optical burst-switched WDM networks. *IEEE Journal on Selected Areas in Communications*, 18(10):1838–1851, October 2000.

[34] L. Xu, H. Perros, and G. Rouskas. A simulation study of access protocols for optical burst-switched ring networks. In *Networking 2002*, May 2002.

[35] L. Xu, H. G. Perros, and G. N. Rouskas. Techniques for optical packet switching and optical burst switching. *IEEE Communications*, 39(1):136–142, January 2001.

[36] M. Yang, S. Q. Zheng, and D. Verchere. A QoS supporting scheduling algorithm for optical burst switching DWDM networks. In *Global Telecommunications Conference*, pages 86–91, 2001.

[37] S. Yao, S. Dixit, and B. Mukherjee. Advances in photonic packet switching: An overview. *IEEE Communications*, 38(2):84–94, February 2000.

[38] M. Yoo, C. Qiao, and S. Dixit. QoS performance of optical burst switching in IP-over-WDM networks. *Journal on Selected Areas in Communications*, 18(10):2062–2071, October 2000.