Design and implementation of optical burst switching (OBS)

Student Project / Studienarbeit

Shakeel Ahmad

Tutor: M.Sc. Sireen Malik
Declaration

With these words I declare that the work in this project has been done by myself using only the specified sources as listed in the bibliography.
Acknowledgment

I am deeply obliged to M.Sc. Sireen Malik for supervising this project. He took a keen interest in the project. He remained a source of guidance and constant inspiration throughout the period of the project. Thanks are also due to Prof. Dr. Ulrich Killat for providing me a chance to work on this project in his department and reviewing the project.
List of Figures

1.1 Optical network evolution .................................................. 2
1.2 Node and network architecture for optical burst switching ........ 3
2.1 OBS Signaling with one-way reservation .................................. 9
2.2 Reservation schemes .......................................................... 11
2.3 Immediate reservation scheme .............................................. 12
2.4 Wavelength reservation with JIT .......................................... 13
2.5 Delayed reservation scheme ............................................... 14
2.6 Delayed reservation without void filling (Horizon) ................. 16
2.7 Delayed reservation with void filling (JET) ............................ 17
3.1 Conceptual diagram of OBS design ...................................... 21
3.2 Block diagram of OBS node ............................................... 22
3.3 Signaling for connection setup .......................................... 23
3.4 Format of BHC (Jumpstart project) ..................................... 24
3.5 Format of BHC used in OBS project ..................................... 24
3.6 Algorithm for burst assembly ............................................. 27
4.1 UML diagram for the OBS Project ....................................... 30
4.2 State diagram of reservation unit ....................................... 35
4.3 Algorithm for processing BHC .......................................... 36
4.4 Simulation model for OBS Node ........................................ 39
5.1 Simulation scenario OBS bottleneck with Poisson sources ...... 40
5.2 Simulation scenario OBS bottleneck with TCP sources ........... 41
5.3 Loss probability versus load for various burst sizes ............... 42
5.4 Effect of number of wavelengths on losses at constant capacity .... 44
5.5 Effect of burst assembly time-out on goodput ......................... 46
5.6 TCP goodput versus expected throughput for different number of wavelengths .................................................. 47
5.7 TCP goodput versus losses for different number of wavelengths .... 48
5.8 Topology of German Network ............................................ 49
5.9 Simulation scenario ‘German Network’ implemented with OBS .... 50
List of Tables

5.1 Effect of burst size on drop probability for various load factors .... 42
5.2 Effect of number of wavelengths on losses for constant Capacity ...43
5.3 Effect of maximum burst assembly time on TCP throughput for
different number of wavelengths.........................................45
5.4 Effect of number of wavelengths on losses for constant capacity.... 47
Chapter 1

Introduction

The current Internet is suffering from its own success [1]. The number of users and the variety of applications demanding more and more bandwidth keeps on increasing day by day. These ever-increasing demands need ever-increasing bandwidth. Here optical communication comes into the picture. It provides huge amount of bandwidth and leads to the popular concept of optical Internet. The potential of optical fiber was realized fully when wavelength division multiplexing (WDM) was invented. It was determined that with wavelengths and values typically used in optical networks today it is theoretically possible to transmit data rates of up to 100 terabit per second [2]. It can be seen that at present commercial optical systems, however, do not use WDM in the optimal way and their maximum transmission data rate is at just less than 2 Terra bit per second [3]. One example is the SONET rings, which use WDM simply as a transport medium. Here optical signal have to be terminated at every hop in order to extract SONET control information by doing optical-to-electronic-to-optical (OEO) conversion. The biggest problem in this case is the great mismatch between the electronic processing speed and optical transmission speed that leads to an undesired bottleneck at each SONET node.

To overcome the problem of OEO conversion, researchers proposed All-Optical-Networks (AON) [3] where user’s data would travel completely in optical domain. The elimination of OEO conversion would lead to quite high data rates because processing delays at intermediate nodes would have been completely removed. In past few years AON have been studied extensively but still we lack the reliable, simple, scaleable, adaptable and robust AON control framework.

Research is being done on how exactly to design the networking layers in the optical Internet which would minimize current trends of over layering and protocol functions overlapping. For example, currently it is common to
find IP-over-ATM-over-SONET networks that is a wasteful overhead of control traffic. Therefore most of the recent designs for the future optical Internet consider a simpler approach of two-layer model where the data layer is directly over an optical layer.

1.1 Evolution of optical networks

The Evolution of optical networks as classified in [3] is shown in Figure 1.1.

Among the AON evolution first is the Wavelength-Routed-Optical-Networks (WRONs) whose operation consists of setting up long term circuit connections between the network nodes. These circuit connections are called
‘light paths’. The main constraint with WRONs is scalability. When network grows up the limited number of wavelengths becomes bottleneck to establish full mesh network. Consequently for each topology network architects have to solve the NP-hard problem of routing and wavelength allocation (RWA) of the limited number of wavelengths in an optimized way to improve utilization [3]. Another constraint with WRONs is its quasi-static nature due to which it is not suitable for dynamic changes in the traffic which is a key feature of Internet traffic. Today WRONs architecture is supported by the IETF’s GMPLS standards [5,6].

Optical-Packet-Switched-Networks (OPSNs) would be the most sophisticated and seem to be an ideal architecture for future optical networks. As clear from name the traffic in OPSNs is carried in optical packets along with in-band control information. But the problem with OPSNs is the lack of optical logic and practical optical buffer.

With respect to current technology presently Optical Burst Switching (OBS) is the most suitable AON control framework. It combines the best feature of both circuit switching and packet switching [7].

Figure 1.2: Node and network architecture for optical burst switching [15].

A general diagram for node and architecture of OBS is shown in Figure 1.2. At the edge node upper layer user data is collected and aggregated
into variable sized units called bursts. Size of the burst may be fixed or variable depending upon aggregation algorithm. Edge node also creates control packet, or burst header control (BHC), which is sent into the network a time, called offset, prior to their corresponding bursts on dedicated control channel. In the following chapters terms ‘control packet’ and BHC will be used interchangeably. The purpose of sending the control packet is to establish an optical path so that the following burst could pass transparently through all the intermediate nodes. At each node control packet is converted back to electronic domain, information is extracted from this control packet, and if reservation can be made then this control packet is forwarded to the next hop. After the offset time, burst is transmitted without knowing that whether reservation could be made or not.
Chapter 2

OBS Framework and Design Parameters

The basic requirement of the OBS control framework is that it should support all-optical WDM network efficiently. Separation of control and data planes is an important feature of OBS design. There are two possibilities to achieve separation between control and data channel [3]. One is to dedicate a specific wavelength in an all-optical-network to be used as an out-of-band signaling channel and other choice is to use completely separate control channel as an IP or ATM network. In both cases, separation of control and data channels in both physical and time space is one of the main advantages of OBS. It facilitates efficient electronic control and allows a great flexibility in user data format and rate [17].

The OBS node that interacts with IP layer is called edge node and the one that does not interact with IP layer is called core node. The edge node has the capability to accept IP packets from IP layer directly in electronic form in addition to the normal OBS traffic (bursts and signaling in optical domain) from other OBS nodes. Edge OBS node is equipped with a burst assembly unit in which IP packets are aggregated into bursts, queued according to the destination (Destination Based Queuing DBQ). Once a burst is complete a BHC is generated and sent on a dedicated control channel. After offset time burst is transmitted. BHC contains information about offset time, routing, incoming wavelength and duration for the following burst.

The offset time allows the BHC to reserve the needed resources along the path towards destination. The BHC is processed electronically at each OBS node along the path from source to destination. The intermediate OBS nodes need this offset time to configure the optical cross connect switch so that the following burst could pass though them transparently without the need of any buffer. This type of reservation is called one-way-reservation.
One-way-reservation scheme might lead to high burst drop probability. Bursts might be dropped if the corresponding BHC could not do necessary reservation. There are two reasons for unsuccessful reservation:

- One reason is the contention which means that a BHC is trying to reserve some resources that are already reserved. Conflict might be in time domain or in frequency domain.
- A finite capacity server processes BHC electronically. If there is overload on this channel BHC might be lost or late processed. If it is late processed then the estimation about the arrival of burst will be wrong. In worst case burst might arrive before the BHC is processed.

2.1 Connection setup mechanism

There are three main components in setting up a connection for each burst.

- Signaling
- Routing
- Wavelength allocation

2.1.1 Signaling

Signaling describes the protocol by which an edge node communicates with the network to request connections. Utilization efficiency of the resources depends upon this protocol. One-way-signaling procedure is used to setup burst transmission path throughout the network. Figure 2.1 shows the steps involved in this type of signaling but before describing these steps some terms need to be explained.

- $T_{\text{oxc}}$

This is the amount of time, optical cross connect (OXC) takes to configure its switch fabric in order to setup the connection between an input port and an output port. It can be said that it is a delay incurred
between the instant the OXC receives the command to setup a connection from an input port to an output port until the time a connection within the switch fabric have been established and ready to accept the burst [9]. Further it is supposed that this configuration delay is independent of the pair of input-output port that is required to be connected. It is also independent from the state of the optical switch at the time it gets connection request.

- $T_{\text{setup}}(x)$

This is the time an OBS node takes to process BHC under reservation scheme $x$ (details of reservation schemes will be discussed later). This time depends upon the reservation scheme being used. For a given reservation scheme it is assumed that this time is constant across all burst.

- $T_{\text{offset}}(x)$

This is the offset value of a burst when using reservation scheme $x$. Its value depends upon the reservation scheme being used, number of nodes burst has traveled already and other factors like whether offset is being used as service differentiation or not.

The basic concern in setting up the offset value is to make sure that first bit of the burst would reach the destination node shortly after the switch is configured. The time lag between BHC and first bit of burst shrinks as the two propagate towards destination. It is shown in Figure 2.1. It is because BHC suffers from processing delay at each OBS node along the path whereas burst passes transparently. If $k$ is the number of OBS nodes in the path of a burst from source to destination then the minimum value of offset that will guarantee that first bit of burst will arrive at the destination node shortly after the switch is configured, can be calculated as follows [9].

$$T_{\min offset}(x) = k \cdot T_{\text{setup}}(x) + T_{\text{toxc}}$$

(2.1)

The Steps involved in signaling can be described with reference to Figure 2.1 as follows:
OBS

- Send a BHC into the network, which is electronically processed at each OBS node along the path towards destination of burst. Purpose of this BHC is to reserve resources along the path to establish an all-optical path.

- Each of the intermediate OBS nodes processes this BHC and tries to reserve resources according to some certain reservation scheme (will be discussed in article 2.3). It forwards the BHC to next OBS node only if it could reserve resources successfully otherwise it discards the BHC which results in dropping of the following burst.

- After an offset time, burst is sent towards the destination without knowing that whether a complete path for this burst has been established or not.
2.1.2 Routing

A requirement for the OBS routing is the ability to explicitly pre-determine the routes at the edge of the network. Routing algorithm should be capable of predetermining the route at the edge OBS node. This type of routing is called source routing. Source routing is done based upon a derivative of multi-protocol label switching (MPLS) as given in [18]. It says that shaper or burst
assembly unit asks to the OBS about the wavelength on which it should send bursts. In the OBS fabric wavelength allocations are done only in forward direction. Source routing provides information like hop count before transmitting a burst that is used to determine offset time.

2.1.3 Wavelength assignment
(with or without wavelength conversion)

When a request arrives at an OBS node asking it to reserve wavelength, there are two possibilities [3].

- First one is the wavelength continuity allocation scheme, where the burst travels on only one wavelength throughout its path from source to destination. This scheme results in high burst loss because it decreases the probability that there will be a free wavelength available on all the fiber links along the route.

- Other scheme assumes that on every OBS node, along the path to destination, wavelength conversion capability is available. So if two incoming bursts are coming on same wavelength but two different optical In ports and they want to go same output optical port, they can be assigned two different wavelengths on the same output port by doing wavelength conversion for one of them. Wavelength conversion capability enhances the wavelength utilization [3] and burst drop probability decreases.

2.2 Scheduling of resources: reservation and release

Under this section most of the discussion is referred to [9]. When a BHC arrives at OBS node requesting for an outgoing link the fashion in which outgoing wavelength is reserved varies in different presented OBS architectures. In general, reservation scheme can be categorized as shown in Figure 2.2.
2.2.1 Immediate reservation scheme

Immediate reservation scheme can be described as:

An output wavelength is reserved immediately after the arrival of a BHC. However if a wavelength cannot be reserved at that time then the BHC and the following burst both are dropped.

The operation of JIT can be described with the help of Figure 2.3. Let BHC arrives at an OBS node along the path to destination at time \( t \). It will take time \( T_{\text{setup}} \) to process the BHC. So at time \( t + T_{\text{setup}} \), a wavelength is immediately reserved for the upcoming burst and the operation to configure the optical cross connect (OXC) fabric to switch the burst is initiated. This configuration will be completed at time \( t + T_{\text{setup}} + T_{\text{oxc}} \) and at this time switch is ready to accept the burst. Note that burst will not arrive at the OBS node under consideration until time \( t + T_{\text{offset}} \). This results in idle time for the reserved wavelength.

\[
\text{Idle Time} = T_{\text{offset}} - T_{\text{setup}} - T_{\text{oxc}}
\] (2.2)
The time lag between BHC and burst decreases as we go farther into the network therefore this idle time also decreases as we go deeper into the network.

The functionality of immediate reservation becomes clear from Figure 2.4 by considering the operation of a single output wavelength at an OBS node. Every output wavelength has only two possible states, whether it is reserved or free. Reserved state is composed of two parts: idle reserved (shown as IR in the Figure) and busy reserved (shown as BR in Figure). Idle reserved is the time for which it is idle but appears to be busy for any new
connection request. Busy reserved is the time the corresponding burst takes to pass through it.

Figure 2.4: Wavelength reservation with JIT.

Figure 2.4 shows that a BHC arrives at time $t_1$. Let's suppose that an outgoing wavelength is free. At that time BHC is accepted and status of wavelength is changed to reserved. After a time $\text{offset}$ first bit of corresponding burst arrives at time $t_2$ and last bit arrives at $t_3$ at which time the status of the wavelength becomes free again. Any new BHC arriving between time $t_1$ and $t_3$ when the status of burst is reserved would be discarded because the wavelength is already reserved.

Suppose that BHC arrives at time $t_4 > t_3$ when the wavelength is free. It would be accepted for reservation. Consequently, the burst corresponding to this BHC becomes $(i+1)$th burst successfully departed from this node. But this burst may not necessarily be the $(i+1)$th burst because some BHC might have arrived before $t_3$ and were discarded.

From Figure 2.4 it is clear that immediate reservation scheme is simple. This is divided into periods during which wavelength is free followed by periods during which wavelength is free. Service is provided in FIFO fashion, in a sense that bursts are served in the order in which their BHC arrive.
2.2.2 Delayed reservation scheme

The Horizon and JET protocols are based on delayed reservation which can be described as follow:

*An output wavelength is reserved for a burst just before arrival of first bit of the burst. However if upon arrival of the BHC it is determined that no wavelength can be reserved at the appropriate time then the BHC is discarded* and the following burst is dropped [9].

![Figure 2.5: Delayed reservation scheme.](image)
With reference to Figure 2.5, let us assume that a BHC arrives at an OBS node at time $t$ saying that first bit of the burst will arrive at time $(t + T_{offset})$. Assuming the request can be accepted, wavelength is reserved for the corresponding burst starting at time $t' = t + T_{offset} - T_{oxc}$. At time $t'$ OBS node orders its OXC fabric to configure its switch element to carry burst and this configuration operation completes just before the arrival of the first bit of burst.

Another important distinguishing feature between immediate reservation protocol and delayed reservation protocol is the fact that the former scheme allows only single outstanding reservation for each of the output wavelength whereas the later scheme allows multiple BHCs to make future reservations for the same outgoing wavelength provided that these reservation do not overlap in time. It is also worth mentioning that an output wavelength is reserved for an amount of time equal to the length of burst plus switch configuration time $T_{oxc}$.

From Figure 2.5 it can also be observed that a void is created at the output wavelength between time $t + T_{setup}$, when the processing of BHC is completed, and the time $t' = t + T_{offset} - T_{oxc}$. Here $t$ is the time when output wavelength is actually reserved for the burst. If the offset value is taken as required by minimum value according to expression 2.1, then the length of this void at some OBS node $n$ is equal to $r \times T_{setup}$ where $r$ is the number of OBS nodes in the path from $n$ to the destination of the burst. Hence it can be concluded that length of void decreases as the burst travels along the path.

### 2.2.2.1 Delayed reservation without void filling

Delayed reservation schemes, e.g., Horizon [9,11] that does not employ void filling, are relatively less complex than those schemes which employ void filling. In Horizon scheme each of the outgoing wavelength is associated with a time Horizon for burst scheduling purpose. The time Horizon of a wavelength is defined as:

*The earliest time after which there is no planned use of the wavelength.*
Delayed reservation scheme without void filling can be described as:

An outgoing wavelength is reserved for a burst only if the arrival time of the first bit of burst is later than the time Horizon of the wavelength. However if upon arrival of the BHC it is determined that the arrival time of the burst is earlier than the smallest time Horizon of any wavelength then the BHC and the following burst both are dropped.

After scheduling a burst on a wavelength, the Horizon of that wavelength is updated to the time instant when last bit of burst will depart plus $T_{\text{oxc}}$. Therefore a burst can only be scheduled if the first bit of the burst is arriving after all burst currently scheduled on this wavelength have departed.

In Figure 2.6 two bursts have been shown which are transmitted successfully on a given wavelength. At time $t_1$, BHC arrives for burst $i$, wavelength is reserved for this burst until time $t_4$, at which time last bit of burst will pass. After $t_4$, another time $T_{\text{oxc}}$ is needed to configure the switch before any new burst can be accepted. Therefore the time Horizon of this wavelength is set to $t_4 + T_{\text{oxc}}$, which is $t_5$.

Now say that before first bit of burst $i$ arrives, another control message arrives at time $t_2 < t_3$ requesting to reserve wavelength for $(i+1)$th burst. Only if the first bit of $(i +1)$th burst is arriving after the time Horizon of this wavelength then wavelength can be reserved for this burst and the time Horizon of this wavelength will be updated accordingly. Otherwise BHC will be discarded and the corresponding burst will be dropped.
2.2.2.2 Delayed reservation with void filling

A well-known example of delayed reservation with void filling is JET [12]. With reference to [9] delayed reservation with void filling scheme can be explained as:

Upon arrival of a BHC, an outgoing wavelength is reserved if:

- Condition 1: First bit of the burst is arriving after the time Horizon of wavelength
- Condition 2: First bit of the burst is coinciding with the gap called void, and last bit is arriving a time $T_{occ}$ before the end of void.

If upon arrival of a control message it is determined that none of the above conditions is being fulfilled for any outgoing wavelength then that BHC is discarded and corresponding burst is dropped.

![Figure 2.7: Delayed reservation with void filling (JET).](image)

It is worth mentioning that any OBS Node employing Horizon would discard bursts which are accepted due to condition 2. This is the main distinguishing point between Horizon and JET. Consequently JET is expected to out-perform Horizon in terms of burst loss probability. But cost has to be paid in terms of complexity of the reservation algorithm. Void filling algorithm must maintain an extensive database containing starting and end
points of all wavelengths which is much more complex implementation as compared to Horizon or JIT.

Figure 2.7 demonstrates the operation of void filling in JET. It shows two burst A and B transmitted on the same wavelength one after other. At time \( t_1 \), BHC for burst A arrives and it is accepted but no initiation of connections in optical cross bar takes place until time \( t_6 - T_{\text{oxc}} \). A void is created, which starts at \( t_1 \) and ends at time \( t_6 - T_{\text{oxc}} \). At time \( t_2 \), BHC for burst B arrives. At this time switch notices that an other burst is arriving before already scheduled burst A and runs void filling algorithm to check whether the new burst can be accepted or not. To accept burst B, there should be enough time between the last bit of burst B and the first bit of burst A for the switch to reconfigure its cross-connect fabric to setup a path for burst A. In this scenario burst B is accepted and it completes service before the arrival of first bit of bust.

2.2.3 Comparison of JIT, Horizon and JET

In reference [9] the comparison between JIT, Horizon and JET is discussed in detail. If the BHC processing time \( T_{\text{setup}}(x) \) is considered to be same for all three reservation schemes then in general, JET will outperform the other two in terms of burst loss probability followed by Horizon and then JIT. However in real world, relative performance of the three schemes is highly dependent on certain system parameters.

- \( T_{\text{oxc}} > kT_{\text{setup}} \Rightarrow \text{JET} = \text{Horizon} \)

  Referring to equation 2.1 and Figure 2.7, it is clear that if \( T_{\text{oxc}} \) is larger than the sum of processing times at all Nodes then void cannot be filled. It is because that to do void filling switch has to be configured two times. First time for the new burst filling the void and second time for the burst scheduled earlier. Total time for this operation is \( 2T_{\text{oxc}} \) and void is at most equal to \( T_{\text{offset}} = T_{\text{oxc}} + kT_{\text{setup}} \leq 2T_{\text{oxc}} \)

- **Minimum burst duration + \( T_{\text{oxc}} > kT_{\text{setup}} \Rightarrow \text{JET} = \text{Horizon} \)**

  Same argument as in above case can be given.
• $T_{\text{offset}} = \text{constant.} \Rightarrow \text{JET} = \text{Horizon}$

If offset value for all bursts is constant, then during offset time of a already scheduled burst say burst A, if a BHC arrives for burst B, then the first bit of burst B will be either arriving after the burst A finishes (Horizon) or burst B will be overlapping with burst B. In second case it will be discarded. Burst B can be scheduled only in first case, which is the case of Horizon. Hence no void filling is possible.

• Mean burst duration $\gg T_{\text{oxc}}$ && Mean burst duration $\gg T_{\text{setup}}$

If mean burst size is very large as compared to $T_{\text{oxc}}$ and $T_{\text{setup}}$, then from eq. 2, it is clear that mean burst size is also very large as compared to $T_{\text{offset}}$. In this case one may conclude that there will be very few opportunities for void filling or delayed reservation and performance for all three schemes will be almost similar.

2.3 Burst aggregation

General functionality of burst aggregation unit is to take traffic from upper layers e.g., IP layer sort it on the basis of destination address and put all traffic intended for the same destination into one group called burst.

Design of burst aggregation algorithm can greatly impact the overall performance of OBS network because it allows controlling the characteristics of bursts and shaping the burst arrival process to the OBS network. The algorithm used for the project will be explained in chapter 3.
Chapter 3

Design of OBS

3.1 Design issues and decisions

The basic function of an OBS network is:

- Take IP packets from IP layer.

- Convert these IP packets into OBS traffic (i.e., aggregating IP packets into bursts and generating BHC).

- Transport OBS traffic to destination OBS node.

- Deaggregate bursts into IP packets and deliver them back to IP layer.

The first problem was to define an interface between OBS and IP layer. This interface should be responsible for receiving IP packets from and delivering them back to IP layer. Please note that it is not necessary that the upper layer is always IP layer but as an application is being designed for “IP-Over-WDM” therefore in further discussion it will be supposed that upper layer is always IP layer. Figure 3.1 shows the basic conceptual diagram.

Every OBS has two types of interfaces: one interface is for IP plane and the other is for OBS plane. This interface should be transparent for IP layer. It means that IP node does not need to know any thing about OBS node. Each OBS takes and delivers IP traffic from and to IP layer through its add and drop ports respectively as shown in Figure 3.1. In addition to these add/drop ports, OBS may have a number of optical ports communicating with other OBS nodes.
Figure 3.1: Conceptual diagram of OBS design.

Defining interface between IP layer and OBS layer was not a trivial task. A block diagram of an OBS node has been shown in Figure 3.2. IP packets from IP node go directly to aggregation unit. Aggregation unit knows exactly which IP node is attached to which OBS node and by using this information it builds a map between IP addresses of IP node and address of OBS nodes which are attached to each other. Aggregation unit then assembles IP packets into burst according to DBQ. Once a burst is ready, aggregation unit creates a BHC and sends it towards reservation unit.

At reservation unit these BHC are queued if necessary and are served according to reservation algorithm e.g., JIT. If BHC is accepted then reservation unit instructs optical cross connect fabric to make necessary connection, BHC is forwarded to the next OBS node along the route and an acknowledgment is sent to the burst assembly unit. If acknowledgment is positive then burst assembly unit sends related burst after an offset time.
When a burst arrives at destination OBS node it is sent towards deaggregation unit where it is deaggregated into IP packets and transmitted to IP node through drop port.

![Block diagram of OBS node](image)

**Figure 3.2: Block diagram of OBS node**

### 3.2 Signaling

For signaling and control, **distributed signaling with one-way reservation** has been used. The signaling used in Jumpstart project employing JIT is a derivative of multi-protocol label switching (MPLS). The first message in a signaling flow serves the purpose of setting up label-switched path. This on-the-fly setup of the label switched path is the main difference between MPLS and the signaling used in Jumpstart [18]. We have followed the same approach.

When a burst is complete, BHC is sent to the reservation unit requesting for a connection. If reservation unit can reserve the resources it not only forwards the BHC but also sends a setup acknowledgment to the aggregation unit telling that on which wavelength burst should be sent [8].
3.3 Format of burst header control (BHC)

The format of BHC as used in Jumpstart project has been shown in Figure 3.4 [19]. There are many fields for different purposes e.g., a fixed length common header and variable length hard/soft path information elements (IEs). The size of BHC is 70 bytes.
However, we have used a simplified form of this format that is necessary to setup connections for unicast short term bursts. Size of the BHC is kept 70 bytes equal to the one in original format. It has been shown in Figure 3.5

![Figure 3.5: Format of BHC used in OBS project.](image)

BHC contains necessary information about the upcoming burst to reserve resources. **Input-port-identifier** and **input-wavelength-identifier** are input optical port and incoming wavelength for the upcoming burst respectively. The values of offset and burst-size are used to estimate the end of burst whereas **Route-string** is the pre-calculated path for the burst. Please note that when BHC is sent from aggregation unit to edge OBS node the field **input-wavelength-identifier** is empty. It is the reservation unit that decides about the wavelength on which aggregation unit should send the burst. Then reservation unit informs aggregation unit about **input wavelength identifier** through setup acknowledgment as shown in figure 3.2.
3.4 Buffering of IP packets at OBS edge node

IP packets arriving in the interval during which burst is waiting for acknowledgment from reservation unit cannot be added to the burst. It is because burst length information has already been sent in the BHC and burst length will become longer than the period for which reservation has been requested. Moreover it is supposed that there is only one DBQ for each OBS destination node so another DBQ cannot be initialized while one is already waiting for the acknowledgment. To solve this problem IP packet queuing is implemented at the edge node. This queue stores the packets which arrive in the acknowledgment-waiting interval. We did not investigate the impact of edge buffering as part of the project and relied on the results from reference [20].

3.5 Reservation mechanism

For reservation scheme JIT has been chosen Why it has been chosen, two arguments may be given.

- According to reference [9] it is proved that JIT will perform almost same as JET, or Horizon. Comparison among the three schemes has been discussed in detail under article 2.3.3.

- JIT is relatively simple to implement (KISS principle).

There are four variants of JIT [14].

- **Explicit connect and Explicit Release**: switching elements are configured for the upcoming burst immediately upon arrival of a BHC and remains configured until a release message is received.

- **Explicit connect and Implicit Release**: In this scheme no release message is sent. BHC carries information about the duration of burst and connection release time is estimated from arrival time of BHC, offset time and burst duration.

- **Implicit connect and Explicit Release**: This scheme is opposite of “Explicit connect and Implicit Release”. Instead of estimating end of
burst, start of burst is estimated from the information contained in the BHC. This scheme, however, requires a release message to make switch elements free so that they can be used to route other bursts.

- **Implicit connect and Implicit Release**: Here both start and end of burst is estimated from the information contained in the BHC.

In the implementation "Explicit connect and Implicit Release" has been used. An argument behind it is that extra release messages should be avoided because they might cause congestion in control plane. There is always a trade off between accuracy and complexity of algorithm. Tighter the estimate for the start and end of burst, smaller the overhead of keeping switching elements configured will be. In reference [14] these schemes have been discussed in detail.

### 3.6 Routing

Source routing as explained in article 2.1.2 have been chosen and implemented because to estimate initial offset one must know number of hops between source and destination beforehand which is easily possible with source routing.

### 3.7 Wavelength assignment

In the design, every OBS node has wavelength conversion capability. When a request arrives for reservation saying that a burst is coming on input port \( P_{in} \), wavelength \( \lambda_y \) then from forwarding table output port \( P_{out} \) is determined and an outgoing wavelength is assigned as follows.

- First try to assign the same outgoing wavelength \( \lambda_y \) as the incoming wavelength.

- However, if on output port \( P_{out} \), \( \lambda_y \) is already reserved then try to assign next available wavelength.

- Wavelength conversion can be set ON or OFF.
3.8 Burst assembly algorithm

Burst aggregation algorithm can directly affect the burst characteristics. There are two major concerns: burst assembly time and burst size, which are controlled by burst assembly algorithm. The burst assembly algorithm used has been shown in Figure 3.6. The same implementation can be used either as ‘fixed burst length algorithm’ by disabling time out function or as ‘fixed burst assembly time out algorithm’ by setting maximum burst length very large.

**Fixed maximum burst length and maximum burst assembly time**

<table>
<thead>
<tr>
<th>Event: IP packet Arrives</th>
</tr>
</thead>
<tbody>
<tr>
<td>If (timer not running)</td>
</tr>
<tr>
<td>{</td>
</tr>
<tr>
<td>Start Timer;</td>
</tr>
<tr>
<td>}</td>
</tr>
<tr>
<td>Add IP packet to burst.</td>
</tr>
<tr>
<td>Update burst Length</td>
</tr>
<tr>
<td>If (burst Length == Max Burst Length)</td>
</tr>
<tr>
<td>{</td>
</tr>
<tr>
<td>Send BHC</td>
</tr>
<tr>
<td>Send Burst after offset Time.</td>
</tr>
<tr>
<td>Reset Timer.</td>
</tr>
<tr>
<td>}</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Event: Timer Expires</th>
</tr>
</thead>
<tbody>
<tr>
<td>Send BHC</td>
</tr>
<tr>
<td>Send Burst after offset time.</td>
</tr>
<tr>
<td>Reset Timer.</td>
</tr>
</tbody>
</table>

Figure 3.6: Algorithm for burst assembly
Chapter 4

Implementation

Discrete event simulation is the most suitable simulation methodology for the simulation of communication networks because it can be used to simulate asynchronous, non-periodic data transmission which makes it a good choice to simulate communication networks. Other benefits are graphical user interface, object-oriented programming language, object-oriented design of simulation entities and easy extensibility. These are the reasons which led to select Ptolemy as simulation environment.

Ptolemy is structured on solar system. The constituents of a simulation are stars and galaxies which finally combine to make universe. Star is the lowest level block while universe is the outermost block. Every star is associated with some functionality and it performs that functionality when it is fired. Interested reader may find a detailed description about Ptolemy under link http://ptolemy.eecs.berkeley.edu/

Now all the stars and helping classes which were developed for the project will be discussed. A UML diagram showing various classes and their relationship is shown in Figure 4.1.

4.1 AggregationStar

This is the first object, which interacts with the incoming IP traffic. It is a multipurpose star and its job includes:
**Assembling IP packets into bursts:**

AggregationStar serves as an interface between IP and OBS layer as it takes IP packets at input and gives bursts at output. Bursts are assembled on the basis of OBS destination node. For this purpose, a map is built between IP nodes and OBS nodes attached together. This map tells which OBS node is attached to a particular IP node.

**Signaling:**

Once a burst is ready, according to burst aggregation algorithm, this star creates burst header control, inserts information about burst like bursting length, offsetting time, and sends it to the switch for reservation as explained in article 3.2.

**Source routing:**

This star is also responsible for providing complete path from source OBS node to destination OBS node in the burst header control.

### 4.1.1 Helping classes

There are two helping classes, which DeAggregationStar uses to perform its tasks.

- **Aggregation class:**

  Aggregation class creates and initialize destination based queues and provides an interface for communication between queues and star. Basically these queues are instances of class OpticalBurst. While initializing it provides information like source node, destination node, offset time and routing to every queue. This is one time task and helps to speed up the simulation.

- **OpticalBurst class:**

  This is the entity which represents a queue. IP packets are added to this queue to form a burst. When burst is ready then on receiving a positive acknowledgment from reservation unit, it is transmitted by using future time stamping. Future time stamping uses global queue of the...
scheduler and reduces number of events considerably by avoiding to re-fire the star. The class ‘OpticalBurst’ also performs buffering of IP packets which arrive in the acknowledgment-waiting-interval. It has been explained under article 3.4.

Figure 4.1: UML diagram for the OBS project
4.1.2 Parameters of AggregationStar

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Default Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OBSNode</td>
<td>Int</td>
<td>1</td>
<td>It shows the OBS Node number, which is also its address. It starts form 1 not from 0.</td>
</tr>
<tr>
<td>bHHoldTime</td>
<td>Float</td>
<td>0.0</td>
<td>Burst header hold time. If set to non zero positive value (in ms), then, once a burst is ready, wait a time equal to bHHoldTime before sending burst header control. If it is set to zero, then it has no effect.</td>
</tr>
<tr>
<td>maxBAT</td>
<td>Float</td>
<td>-1</td>
<td>Max burst assembly time in ms. If set to −1, then this feature is deactivated, otherwise it determines burst assembly time out value.</td>
</tr>
<tr>
<td>Tsetup</td>
<td>Float</td>
<td>0.001</td>
<td>Burst header control Processing Time in ms. Here it is used in estimating basic offset</td>
</tr>
<tr>
<td>Toxc</td>
<td>Float</td>
<td>0.02</td>
<td>Switch fabric configuration time in ms. Here it is used in</td>
</tr>
<tr>
<td>Parameter</td>
<td>Type</td>
<td>Value</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------------</td>
<td>--------------</td>
<td>-------</td>
<td>--------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>maxBurstLength</td>
<td>int</td>
<td>100</td>
<td>Max. Burst length in terms of number of IP packets a burst may have. It is used as an indication that burst is ready.</td>
</tr>
<tr>
<td>destNodeNetMask</td>
<td>string</td>
<td>0.255.255.255</td>
<td>This parameter comes from IP plan used to determine IP node number from an IP Address.</td>
</tr>
<tr>
<td>obsNodeAddrArray</td>
<td>intarray</td>
<td>1</td>
<td>An array containing address of all OBS nodes in the topology. Address of an OBS node is simply an integer starting from 1.</td>
</tr>
<tr>
<td>minBurstLength</td>
<td>int</td>
<td>1</td>
<td>It puts a lower bound on the number of IP packets a burst might have.</td>
</tr>
<tr>
<td>obsRouteArray</td>
<td>stringarray</td>
<td>obsRouteArray</td>
<td>An array whose elements are strings describing complete route from this OBS node to all other possible destination OBS nodes, e.g., 1-2-5 is the route from OBS node 1 to 32.</td>
</tr>
</tbody>
</table>
OBS node 5.

<table>
<thead>
<tr>
<th>IpNodeAddrArray</th>
<th>stringarray</th>
<th>0.0.0.0</th>
<th>Containing IP addresses of all IP Routers attached to OBS nodes.</th>
</tr>
</thead>
<tbody>
<tr>
<td>obsAttachToIPNodeArray</td>
<td>intarray</td>
<td>1</td>
<td>An array whose element at index i shows the address of OBS node attached to the IP Node at index i in IpNodeAddrArray</td>
</tr>
</tbody>
</table>

### 4.2 Splitter

This star works like an optical filter tuned to the wavelength of control channel. It splits control channel from each input optical port and diverts them to the reservation unit. At this point control packets coming on all these control channels are converted to electronic form and queued up in the reservation unit to request reservation.

#### 4.2.1 Parameters of Splitter

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Default Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>numWaveLength</td>
<td>int</td>
<td>5</td>
<td>Number of wavelength per fiber link (WDM). It is supposed to be same for all fiber links in the topology. Only first wavelength is used for control plane and the remaining wavelengths are used for data channel</td>
</tr>
<tr>
<td>numPorts</td>
<td>int</td>
<td>1</td>
<td>Number of optical ports i.e., number of fiber links attached to this OBS node.</td>
</tr>
</tbody>
</table>
This star has two components namely reservation unit and optical cross connect switch fabric.

- **Optical cross connect:**

  It is $N \times M$ blocking free cross connect switch where $N$ represents number of optical ports or number of fiber links connected to the switch. Each optical port or fiber link has $W$ wavelengths (WDM). For sake of simplicity it is assumed that number of input optical ports is equal to the number of output optical ports. The abstraction used to simulate multiple optical ports having multiple wavelengths is to implement a $(N \times W) \times (N \times W)$ switch and then group a number of ports (which is equal to the number of wavelengths per fiber) to make a logical optical port. It has flexibility to scale number of optical ports and number of wavelengths per optical port. Moreover, in each optical port first wavelength is always reserved for control channel and all remaining wavelengths are reserved for data channel. Another supposition is that all optical ports have same number of wavelengths and all data channels and control channels are of same speed.

- **Reservation unit:**

  Reservation unit which is implemented inside switch by using a class `ReservationJIT` can be further divided into two functional parts. One is control and other is database. Task of control unit is to processes burst control header and determine whether reservation for the demanded resources can be made or not by using some reservation scheme e.g., in our case it uses JIT reservation scheme. If reservation can be made then it instructs the switch fabric to make necessary connection, stores information about this reservation in the database and forwards the control packet after doing necessary changes. These changes include dropping of this hop from routing list and updating offset time. Note that maintaining a database about reservation is only necessary for software implementation so that when a burst actually arrive its reservation could be confirmed from database. If it is not confirmed positively then this burst is dropped. Star DeOBS is well-
defined interface for reservation unit. Although JIT have been implemented in the project, but later on it is possible to implement JET, Horizon or XYZ without making any change to the star DeOBS. All what is needed is to provide new implementation for the same interface.

4.3.1. How reservation works

Reservation mechanism can be understood with the help of state diagram in Figure 4.2 and reservation algorithm in Figure 4.3.

![State diagram of reservation unit](image.png)

Figure 4.2: State diagram of reservation unit
At time $t_1$, BHC arrives and requests to reserve wavelength for a burst coming on input port $N$, incoming wavelength $W$ and the first bit of burst is arriving at time $t_2$.

**Algorithm for processing BHC**

![Algorithm for processing BHC](image)

Figure 4.3: Algorithm for processing BHC
### 4.3.2 Parameters for OBS

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Default Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>speedofWavelength</td>
<td>float</td>
<td>1e9</td>
<td>Transmission speed on a single wavelength in bps.</td>
</tr>
<tr>
<td>thisOBSNode</td>
<td>int</td>
<td>1</td>
<td>Address of this OBS node. All OBS nodes are assigned an address which is simply a integer starting from 1.</td>
</tr>
<tr>
<td>Tsetup</td>
<td>float</td>
<td>0.001</td>
<td>Burst header control Processing Time in reservation unit. Its units are ms. Here it is used in reservation server to determine how many Burst header control packets can be processed in one second.</td>
</tr>
<tr>
<td>Toxc</td>
<td>float</td>
<td>0.02</td>
<td>Switch fabric configuration time in ms. Here it is used in reservation algorithm.</td>
</tr>
<tr>
<td>numWaveLength</td>
<td>int</td>
<td>5</td>
<td>Total number of wavelength in one fiber link. Used to configure optical cross connect switch.</td>
</tr>
<tr>
<td>numPorts</td>
<td>int</td>
<td>1</td>
<td>Number of optical ports i.e., number of fiber links attached to this OBS node plus one to account for the additional fiber link coming from aggregation unit.</td>
</tr>
</tbody>
</table>
FWLConversion | int | TRUE | It shows whether wavelength converters are available or not. Although its an int but it was used like Boolean. As in Ptolemy no Boolean data type is available.

obsNodeAddrArray | intarray | 1 | It is an array containing addresses of all OBS nodes in the topology. Address of an OBS node is simply an integer starting from 1.

obsPortNumArray | intarray | 1 | It is an array showing which OBS nodes are attached to which ports. It is used to build forwarding table. Format is such that OBS node at index i in obsNodeAddrArray is attached to port number mentioned at index i in obsPortNumArray.

4.4 DeAggregationStar

This star provides an interface between OBS plan and IP plan at drop port. It takes bursts intended for local delivery, deaggregate bursts into IP Packets and sends them out to the IP Router attached with this OBS node. There is only one parameter with name ‘dropPortRate’ and float type. Its value is in Mbps and it is very important to set its value equal to the input link rate of IP node.
4.5 Simulation Model for OBS Node

A snapshot of the simulation model which was developed for OBS Node is shown in Figure 4.4. It shows the implementation details of the OBS node for Ptolemy simulator. All the stars have already been discussed.

- **OBS Port galaxy:**
  contains replacement block for higher order function (HOF) which allows dynamic instantiation of replacement blocks according to the number of optical ports and number of wavelengths per optical port. Replacement block consists of one StatServer and one Delay star.

- **BHC Queue:**
  This galaxy consists of a queue to store BHC and a StateServer which simulates processing delay. After stateServer BHC enters OBS through its control port for reservation.

Figure 4.4: Simulation model for OBS node
Chapter 5

Simulation Setup and Results

The biggest problem in OBS is high burst drop probability and in this chapter effects of various parameters like burst size, burst assembly time-out and number of wavelengths on burst drop probability will be studied. There are two simulation scenarios used to get results and they are shown in Figure 5.1 and 5.2. Basically both scenarios are exactly same except for the only difference that in former Poisson sources have been used while in later TCP sources have been used. In both scenarios, delays are same between all links and equal to 5 ms except on the bottleneck link (between node 3 and 4) which 10 ms.

Figure 5.1: Simulation scenario OBS bottleneck with Poisson sources.
5.1 Effect of burst size on drop probability

Effect of burst size on drop probability can be described with two aspects.

- Shorter the burst size, higher will be the number of BHCs for a constant throughput that might lead to congestion in control plan. BHCs will suffer from queuing delay before entering to the reservation unit. Due to this queuing delay bursts might arrive before the switch is configured and hence will be dropped. In worst case it might create a situation where a burst is arriving at an OBS node before the corresponding BHC have been processed. Obviously this burst would be dropped because there is no reservation for it.

- On the other hand, from utilization point of view, it can be seen that utilization \( U \) of optical links can be given as:

\[
U = \frac{burstduration}{burstduration + \text{offset}} \quad \text{5.1}
\]
For a given offset and load, utilization increases by increasing burst size resulting in a decrease in drop probability.

To observe the effect of burst size on drop probability simulation scenario as shown in Figure 5.1 has been used and packets of constant size (1480 Bytes) are arriving as Poisson process. They are aggregated for a fixed number of packets and the resulting burst departure process is n-Erlang where n is the number of packets aggregated into burst. It means that aggregation unit waits for n packets to be arrived before the burst is sent. Trials have been made for 4 wavelengths each of capacity 25 Mbps

Table 5.1: Effect of burst size on drop probability for various load factors.

<table>
<thead>
<tr>
<th>Load factor</th>
<th>Drop probability n= 40</th>
<th>Drop probability n= 80</th>
<th>Drop probability n= 120</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.50</td>
<td>2.45e-5</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>0.60</td>
<td>1.36e-3</td>
<td>1.36e-5</td>
<td>1e-7</td>
</tr>
<tr>
<td>0.65</td>
<td>5.57e-3</td>
<td>1.672e-4</td>
<td>0.808e-5</td>
</tr>
<tr>
<td>0.70</td>
<td>1.783e-2</td>
<td>1.5e-3</td>
<td>1.8e-4</td>
</tr>
</tbody>
</table>

Figure 5.3: Loss probability versus load for various burst sizes.
Graph in Figure 5.3 clearly shows that drop probability decreases by increasing burst size. However burst size cannot be increased arbitrarily because it would increase aggregation time resulting in higher end-to-end (e2e) delay. Therefore upper limit of burst size comes from maximum permissible e2e delay for a given load. Effects on e2e delay have not been studied in this project.

5.2 Effect of number of wavelengths on losses for constant capacity

The purpose of this trial is to investigate what difference it makes if a single wavelength of capacity \( c \) is used and then it is replaced by \( w \) wavelengths each of capacity \( c/w \) so that the total capacity remains constant. By using scenario shown in Figure 5.1 simulation was run for a constant offered traffic using Poisson source at a load factor of 0.7 and total capacity of 100 Mbps. Burst drop probability was measured by increasing number of wavelengths and decreasing the capacity per wavelength so that total capacity remains constant.

<table>
<thead>
<tr>
<th>Number of wavelength</th>
<th>Drop probability</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.05473</td>
</tr>
<tr>
<td>2</td>
<td>0.03843</td>
</tr>
<tr>
<td>3</td>
<td>0.03572</td>
</tr>
<tr>
<td>4</td>
<td>0.029</td>
</tr>
<tr>
<td>5</td>
<td>0.0270</td>
</tr>
<tr>
<td>6</td>
<td>0.0260</td>
</tr>
</tbody>
</table>

It is clear from the graph shown in Figure 5.4 that by increasing the number of wavelengths although capacity per wavelength decreases but drop probability also decreases. Basically service rate is decreased and number of servers is increased. This increases the probability that there is a free wavelength available when a BHC arrives for reservation, no matter how slow it is. Please note that it was an independent trial and does have any link with other trials. In all other trials, except this, capacity per wavelength is considered to be constant e.g., increasing number of wavelengths two folds capacity also increases two folds.
Figure 5.4: Effect of number of wavelengths on losses at constant capacity

5.3 OBS with TCP

The performance of network is more sensitive towards burst assembly time than burst length. It is because TCP congestion control and retransmission mechanism is based on timers and therefore intuitively burst assembly time out mechanism should be adopted for burst assembly mechanism [13].

Next important question is that how much this maximum burst assembly time out should be. Its value, on one hand should be high enough so that there is no congestion in control plane and on other hand it should be low enough so that it should not trigger unnecessary retransmission. Simulation has been conducted to determine an optimum value for maximum burst assembly time.
5.3.1 Effect of burst assembly time-out on TCP goodput

Simulation scenario has been shown in Figure 5.2.

- Number of wavelengths = 4
- Capacity per wavelength = 25 Mbps
- Number of flows = 2 (each of 35 Mbps)
- Load factor = 0.7

Burst assembly time-out is varied and throughput is measured. This throughput is measured at application layer and it does not include retransmissions. In other words it is goodput. A graph has been plotted between maximum burst assembly time-out and goodput in Figure 5.5.

Initially goodput rises up with the increase of time-out rapidly and after 10 ms, increment rate decreases and after 20 ms it becomes straight. At lower burst assembly time-out bursts and associated BHC packets are created at a higher rate that results in congestion in control plane due to which burst drop probability increases. By increasing burst assembly time-out congestion in control plane reduces due to which losses reduces and goodput increases. However after 20 ms there is no effect on goodput because the losses that were occurring due to congestion in control plane have been vanished. It does not mean that all losses have been vanished, still there are some losses occurring due to wavelength contention due to which goodput saturates and never reach the expected throughput (70 Mbps). For further trials with TCP 20 ms will be used for burst assembly time-out. The optimal value of time-out (20 ms) is valid only for the simulation scenario used here and it is not an absolute value. It might vary from node to node and from topology to topology. A mathematical methodology to predict optimal value for time-out is still a topic for future study.

Table 5.3: Effect of maximum burst assembly time on TCP throughput for different number of wavelengths.

<table>
<thead>
<tr>
<th>Maximum burst assembly time (ms)</th>
<th>Goodput for four wavelengths (Mbps)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>34</td>
</tr>
<tr>
<td>5</td>
<td>40</td>
</tr>
<tr>
<td>10</td>
<td>41.5</td>
</tr>
<tr>
<td>15</td>
<td>42</td>
</tr>
</tbody>
</table>
5.3.2 Effect of losses on TCP goodput

By using simulation scenario shown in Figure 5.2, trials were conducted by increasing the expected throughput for 4 and 6 wavelengths each of capacity 25 Mbps. Burst drop probability and TCP goodput was measured. A graph has been plotted between goodput and expected throughput for four and six wavelengths as shown in Figure 5.6. Here ‘expected throughput’ means ‘demand’ and it determines the number of TCP sources. A number of HTTP-TCP sources are used to generate traffic. The details about these resources are available in [21,22].
Figure 5.6 shows that initially goodput increases with expected throughput in a direct-proportional fashion. However after some point (goodput = 34, for 4 wavelength) slope of the graph starts decreasing. After this point although goodput is increasing but it is not increasing with expected throughput proportionally. This is because of losses. The graph shown in Figure 5.7 shows that as the throughput increases losses also increase.

Second observation is the effect of number of wavelengths. The direct-proportional relationship between goodput and expected throughput goes longer and TCP goodput increases by increasing the number of wavelengths. It is because losses are decreased by increasing the number of wavelengths as shown in Figure 5.7.

It can be concluded that losses have direct impact on TCP throughput.
5.4 Complete topology with OBS

After doing some performance analysis by using simple bottleneck, it is worth implementing OBS with some large real topology. For this purpose, GermanNetwork has been selected.

Total traffic volume for German network is 2396.2 Gbps [16], which has been scaled down by a factor of 100, thus making total volume for simulation about 20 Gbps. A snapshot of simulation scenario ‘German Network’ implemented with OBS has been shown in Figure 5.12.
There were two purposes to implement this topology.

- First purpose is to see that if there is any scalability issue with OBS implementation from software point of view. Simulation ran for 8 hours and no problem was reported. Source routing, forwarding tables, reservation mechanism, aggregation and deaggregation units are doing their function in a way they are supposed to do which really enhances my confidence in the software implementation.

- Second purpose was to see that if there is any speed gain from simulation point of view. It took 8 hours to simulate 20 Gbps for 10 seconds and maximum memory usage was 22% on a 4 GB memory machine. It is expected that due to aggregation, some speed gain in simulation of large networks may be achieved because it would reduce number of events. However exact statistics and further investigation that how much gain can be achieved exactly is left for future work.
Figure 5.9: Simulation scenario ‘German Network’ implemented with OBS.
Chapter 6

Conclusions and Future Work

In this student project an optical burst switching framework was designed and implemented in Ptolemy simulator using JIT as reservation scheme. Effects of some parameters namely burst size, burst assembly time out and number of wavelength on burst drop probability were studied. The performance of TCP on OBS was also studied.

OBS framework is a first time addition to Ptolemy library. Whole software design is pure object-oriented based to make easy derivation and extensibility. I hope it would provide a solid base for further OBS related analysis and studies.

Other reservation schemes like JET, Horizon or some new scheme would also be interesting to test. To implement any new reservation scheme, there is a well-defined interface and one has to just provide implementation of the same interface for new reservation scheme. To do this nothing else need to be changed except the implementation for the reservation interface.

Investigating the effects of burst size and burst assembly time out on TCP goodput in large topologies have been left for future work. Also end-to-end delay needed to be investigated.

Introducing fiber delay line (FDL) and investigating the effects of FDL on burst drop probability is another aspect for future work. Moreover by introducing FDL the same OBS framework can be used as a ground for optical packet switching (OPS).
Bibliography


Proceedings of the First International Workshop on Optical Burst Switching, October 16, 2003, Dallas, Texas (co-located with Opticomm 2003)


