IMPLEMENTATION OF OPTICAL BURST SWITCHING FRAMEWORK IN PTOLEMY SIMULATOR

Shakeel Ahmad*, Sireen Malik**
Department of Communication Networks
Hamburg University of Technology, Germany
Email: {shakeel.ahmad, s.malik}@tu-harburg.de

Abstract
We present an architecture for Optical Burst Switching (OBS) node and the implementation of OBS for a discrete event simulator like Ptolemy. This implementation allows to do performance analysis of OBS. Currently we implemented Just-In-Time (JIT) reservation scheme. However the software implementation provides flexibility to use other reservation schemes.

Keywords: Optical burst switching, OBS, Implementation of OBS in Ptolemy simulator, Just-In-Time, JIT.

1. INTRODUCTION

The number of internet 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. But because of the absence of stable optical buffer and optical logic at each of the intermediate nodes optical-electronic-optical (OEO) conversion is required. Presently there is a great mismatch between the electronic processing speed and optical transmission speeds which creates a bottleneck. To avoid OEO conversion, researchers proposed All-Optical-Networks (AON) [1] where user’s data would travel completely in optical domain. With regard to AONs, Optical-Packet-Switched-Networks (OPSNs) seems to be the most suitable candidate for future optical network. However in view of the present technological limitations Optical Burst Switching (OBS) is the most suitable AON control framework. It combines the best feature of both circuit switching and packet switching [2].

![Figure 1: Node and network architecture for OBS](image)

Figure 1 shows a general diagram for node and architecture of OBS. At the edge node upper layer user data is collected and aggregated into fixed/variable (depending upon aggregation algorithm) sized units called bursts. Once a burst is ready edge node creates burst header control (BHC) and sends it into the network on dedicated control channel. The purpose of sending the BHC is to establish an optical path so that the following burst could pass through with out buffering. At each of the intermediate nodes BHC is converted back to electronic domain, information is extracted from it and if reservation can be made then this BHC is forwarded to the next node. Edge node then transmits the burst after an offset time without knowing that whether a complete path has been established or not. This is known as one-way-reservation. Due to one-way-reservation OBS suffers from high burst drop rate. Predicting the effects of different OBS parameters like offset time, Maximum Burst Assembly Time (MaxBAT), Maximum Burst Length (MaxBL), number of wavelengths $w$ and...
reservation scheme on the burst loss probability by analytical means is not a trivial task. Therefore most of the times, simulation means are used either to predict or verify the analytical results.

In section 2, we will present OBS framework and develop a simulation model for OBS node. Section 3 will cover the implementation details of this model in Ptolemy simulator. In section 4 we will discuss verification of the simulation results. At last, Appendix A will give a brief overview of Ptolemy simulator.

2. OBS FRAMEWORK

In our design, every OBS is supposed to have two types of interfaces: one interface is for the IP plane and the other is for the OBS plane. Each OBS takes and delivers IP traffic from and to the IP layer through its Add and Drop ports respectively as shown in Figure 2. In addition to these Add/Drop ports, OBS may have a number of optical ports communicating with other OBS nodes.

At the edge node, once a burst is ready, aggregation unit creates a BHC and sends it towards reservation unit requesting to reserve the resources. At reservation unit these BHCs are queued if necessary and are served according to reservation algorithm (JIT in this case). If BHC is accepted then reservation unit instructs the 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. This acknowledgment tells the burst assembly unit that on which wavelength it should send the following burst [6]. Burst assembly unit sends the related burst after an offset time only if a positive acknowledgment is received.

When a core node receives BHC, it tries to allocate the wavelength and it forwards the BHC to the next node only if allocation of wavelength on this node is successful. On receiving the following burst it is passed through transparently to the next node without getting any acknowledgment from the next node (one-way-reservation).

When a burst arrives at destination OBS node it is sent towards deaggregation unit where it is deaggregated into IP packets which are eventually transmitted out to the destination IP Router.

A block diagram of an OBS node is shown in Figure 3. IP packets coming from IP node go directly to aggregation unit where they are assembled into bursts according to their destination. The burst aggregation algorithm is shown in Figure 6. Burst traffic shaping can be controlled by the values of MaxBAT and MaxBL.

For signaling and control, distributed signaling with one-way-reservation has been used. The signaling as used in JumpStart project [4] employing Just-In-Time (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 [5]. We have followed the same approach.

The format of BHC as used in JumpStart project has total length of 70 bytes [7]. The same length has been used here however we used only the relevant fields (as shown in Figure 4.) necessary to setup connections for unicast short term bursts.

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 5.

For reservation scheme JIT with Explicit connect and Implicit Release has been implemented. 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.

Figure 5: Signaling for connection setup

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 \( P_{in} \) and wavelength \( \lambda_y \) then from forwarding table output port \( P_{out} \) is determined and we 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 by using wavelength conversion. Wavelength conversion can be set ON or OFF. Figure 7 explains the JIT reservation mechanism where at time \( t_1 \), BHC arrives and requests to reserve the wavelength for a burst coming on input port \( N \), incoming wavelength W and the first bit of burst is arriving at time \( t_2 \).

2.1 Fixed maximum burst length and maximum burst assembly time algorithm for Aggregation

Event: IP packet Arrives
If (timer not running)
{   Start Timer;
}
Add IP packet to burst;
Update Burst Length;
If (Burst Length == Max Burst Length)
{   Send BHC;
   Send Burst after offset Time;
   Reset Timer and Burst Length;
}
Event: Timer Expires
Send BHC;
Send Burst after offset time;
Reset Timer and Burst Length;

Figure 6: Algorithm for burst assembly.

2.2 JIT Algorithm for reservation

Figure 7: JIT algorithm for reservation
3. IMPLEMENTATION IN PTOLEMY SIMULATOR

First we developed a simulation model for the OBS node as discussed in section 2. Then we implemented this model for Ptolemy Simulator. Appendix A gives a brief overview about Ptolemy Simulator. The UML diagram for the software implementation is shown in Figure 8.

Figure 8: UML diagram for the OBS Node

Star DeAggregationStar (please note that ‘De’ with the name of the star is due to discrete-event domain) is responsible for the task of aggregating the packets into bursts. This star uses two helping classes namely Aggregation class and OpticalBurst class to perform its duties.

Aggregation class creates and initializes destination based queues and provides an interface for communication between queues and DeAggregationStar. 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.

Class OpticalBurst is the entity that 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.

Star DeOBS has two components namely reservation unit and optical cross connect switch fabric. Optical cross connect is N * M blocking free cross connect switch where N and M represents number of incoming and outgoing optical ports connected to the switch respectively. 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 i.e., N=M. The abstraction used to simulate multiple optical ports having multiple wavelengths is to implement a (N * W) * (N * 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 dynamically. Moreover, in each optical port first wavelength is always reserved for control channel and all remaining wavelengths are reserved for data channel.

Reservation unit which is implemented inside the switch by using a class ReservationJIT can be further divided into two functional parts. One is the control and the other is database. The 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 (in our case it uses JIT reservation scheme). If reservation can be made then it instructs the switch fabric to make necessary connection (switch fabric takes time T_{oxc} to be configured), stores information about this reservation in the database and forwards the BHC 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 has well-defined interface for reservation unit. Although JIT have been implemented in the project, but later on it is possible to implement any other reservation scheme without making any changes to the star DeOBS. All what is needed is to provide new implementation for the same interface.

Star DeDEAggregationStar as the name suggests takes bursts intended for local delivery, deaggregate them into IP Packets and sends them out to the IP Router through Drop port.

A snap shot of the simulation model of OBS node which was developed for Ptolemy Simulator is shown in Figure 9. It contains two galaxies namely ‘OBS Port’ and ‘BHC Queue’. OBS Port 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 to capture statistics on the link and one Delay star to simulate propagation delay.
Galaxy ‘BHC Queue’ consists of a queue to store BHC and a StateServer which simulates processing delay. After passing through StateServer BHC enters OBS through its control port for reservation.

![Simulation model for OBS node](image)

**Figure 9: Simulation model for OBS node**

### 4. VERIFICATION

In order to verify, we took a known set of results for JIT as described in [8] and reproduced the results based on the same assumptions and parameters e.g., Figure 10. The output port of an OBS node can be modeled as multiple server loss system therefore Erlang-B formula can be used to obtain the burst drop probability.

\[
Erl(\rho, m) = \frac{\rho^m / m!}{\sum_{i=0}^{\infty} \rho^i / i!}
\]

Burst drop probability obtained from simulation matched with the results already reported in [8]. It verifies that simulation is yielding expected results. Although in this simulation the arrival process of bursts is set to be poissonian (to verify the implementation), however, our implementation has the capability to take IP packets directly and to aggregate them into bursts based on OBS parameters like MaxBAT and MaxBL which means no assumptions are made on the arrival or service processes.

### 5. CONCLUSIONS AND FUTURE WORK

In this paper an Optical-Burst-Switching (OBS) framework was designed and implemented in Ptolemy simulator using JIT as reservation scheme. It allows the designer to do performance analysis of OBS.

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

Although JIT reservation scheme was implemented but other reservation schemes can also be implemented. 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.

We ran simulation to see the effect of number of wavelengths on burst drop probability. The results match with the results previously reported in literature [8].

**Burst Drop Probability versus Num Wavelengths for \(\rho = 32\)**

![Burst Drop Probability versus Num Wavelengths](image)

**Figure 10: Burst drop probability versus Num Wavelengths**

### References


APPENDIX A

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/