Analysis of an Edge-Core Joint Node in OBS Networks

Farrukh Zeeshan Khan, Faisal Hayat, Muhammad Afzal, Syed Ahsan

Department of Computer Science & Engineering
University of Engineering and Technology, Lahore (Pakistan)
fsl.hayat@gmail.com

Abstract: Optical burst switching (OBS) has been introduced as a short term implementable solution for future all-optical networks. Its performance evaluation received a considerable attention and does not seem to be weaken. In prevalent studies, an OBS network is strictly divided into edge and core domains, while practically for future all-optical mesh network deployments; this is not valid for most of nodes. In these networks, not only a few but probably majority of the nodes will have to combine both functionalities to provide flexible operation. Such nodes are termed herein joint edge-core OBS nodes. In this paper, an architecture of a joint edge-core OBS node is studied and a new scheduling algorithm have been proposed to multiplex the local assembled traffic along with the transit traffic. Thorough simulations have been done on benchmark networks to show the significance of edge scheduling in joint edge-core OBS nodes.

[Farrukh Zeeshan Khan. **Analysis of an Edge-Core Joint Node in OBS Networks.** *Life Sci J* 2012;9(4):67-73] (ISSN:1097-8135). http://www.lifesciencesite.com. 11

Keywords: Optical burst switching, Joint edge-core node

In a prevalent number of OBS performance studies, it is taken for granted that an OBS network is strictly divided into the edge and the core part. This means that the network consists of the nodes only assembling packet traffic into bursts (edge nodes) and the nodes only switching the burst traffic along the transmission path (core nodes), see Fig. 1a. However, this assumption cannot be valid for practical future deployments in dynamically reconfigurable networks with mesh topology. In these networks, not only a few but probably majority of the nodes will have to combine both functionalities to provide flexible operation, as shown in Fig. 1b. Such nodes are termed herein joint edge-core OBS nodes. The Fig. 2 explains the difference between two kind of networks, the conventional and network consists of joint OBS nodes in addition with pure edge or core nodes. A significant challenge in their design is that the local traffic must be multiplexed on output wavelengths channels with the transit traffic cutting through the node and a mechanism is needed to control the channel. Otherwise, a high local load may cause high losses of the external traffic and, vice versa, too intensive transit traffic may greatly delay the transmission of locally assembled bursts. Both phenomenadegrade the performance of OBS. However, the increase of losses for transit traffic is expected to have more adverse effects. Burst.

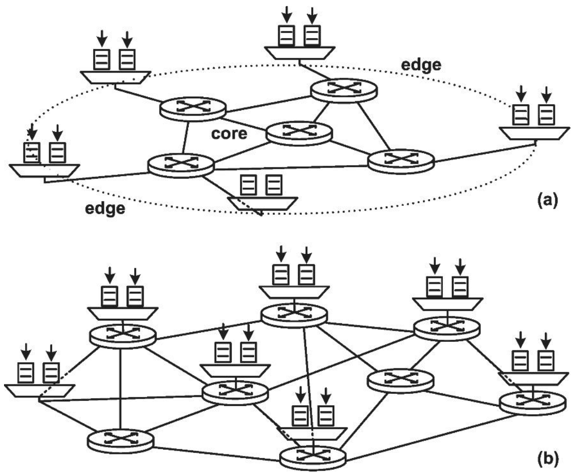

Fig. 1. OBS network with (a) separation of edge and core parts (b)joint node

Bursts that are lost not only waste the reserved path bandwidth but also invoke the retransmission and reordering delays in higher network layers. These effects are well-documented in numerous studies on Transmission Control Protocol over OBS, see for example [4], [3], [5] and references therein. On the contrary, waiting in the transmission buffer, even when excessive, is easier to monitor and does not propagate out of OBS layer to such an extent, thus assuring more stability to particular end-to- end packet flows.

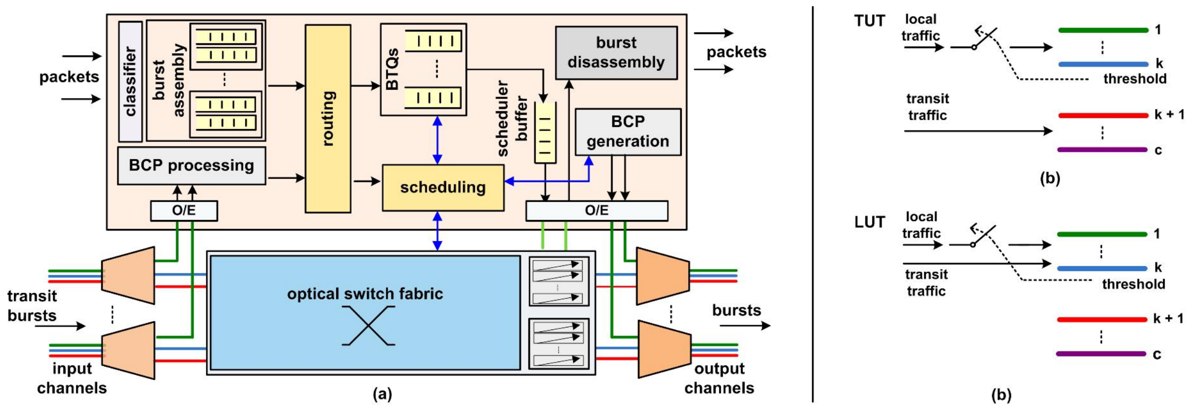

Fig. 2. Architecture of the joint OBS node

Therefore when contention for the channels occurs, transit traffic should be prioritized with no doubt. The issue of multiplexing of local and transit traffic in OBS has not been a subject of analysis. In several simulation studies, this fact was indeed assumed, the blocking of transit and vice-versa. The study gives many insights into the performance issues involved in the real implementation of OBS nodes.

2. Materials and Methods

2.1 Architecture of Joint Edge-Core Node

Edge-core joint node of OBS network is a combination of both edge and core nodes. It can assemble data bursts (DB) with the edge node functions and forward the transit bursts to the next node with the core node function. Figure 2 shows the complete architecture of ECJN. Packets arriving from different traffic sources are classified according to their traffic class and destination address and distributed into corresponding assembly queues. The bursts are assembled in edge node, according to size, time for hybrid based burst assembly mechanisms. The locally assembled bursts are then forwarded to burst transmission queues (BTQs), where they are buffered electronically and wait for the availability of the output channel. When the scheduling module finds output channel, it reserves the channel and forwards burst control packet (BCP) to the destination node, and forwards the burst from BTQ to the scheduler buffer, where the bursts wait electronically until their transmission time is reached. On the other hand, while performing the functions of the core node, the transit traffic (BCP) arrives at routing module, if the destination of arriving bursts is current node, the bursts are forwarded to the burst disassembly module, which disassembles the bursts into IP packets and forwards IP packets to their destinations. If transit bursts need to be forwarded to the next nodes, information is sent to the scheduling module, which looks for the availability to reserve output wavelength channel. If channel is found after wavelength conversion (if required), the wavelength is reserved for incoming bursts, otherwise the burst is dropped. Components of edge-core joint node are discussed in detail. Components of edge-core joint node are discussed in detail.

1) Classifier: The classifier receives the incoming packets from different packet sources, retrieves the packet header to extract destination, class of service and quality of service information and forwards the packet to the relevant assembler queue.

2) Burst Assembly: The burst assembly in a joint node is similar to the burst assembly in an edge node. After sorting of packets into assembler queues by the classifier, the burst assembly process starts from the arrival of first packet in each queue. The number of assembler queues depends upon the number of destination nodes in the network, and class of service. There exists one queue for each class of service for every destination node. The bursts are assembled according to size, time or hybrid based burst assembly technique.

3) Routing: The routing module in a joint node has to perform both edge and core node functions. As an edge node, when a new burst arrives, the routing module selects the most feasible path for the burst's destination, from the existing pre-calculated paths, or calculates a new path on each burst arrival. The decision of selection or calculation depends upon the adopted routing strategy. After the routing decision, the burst is forwarded to the scheduler module of the selected path's output link. In case of arrival of control packet of transit burst, while performing the function of core node's routing module, next hop of the burst is retrieved from the routing table and the control packet is forwarded to the scheduler module of next hop's output link. In case that the destination of transit burst is current node, the incoming burst is

forwarded to the burst disassembly module, where the packets in the burst are disassembled and forwarded to the respective packet destinations.

4) Burst Transmission Queues (BTQs): The locally assembled bursts dispatched from routing module to the scheduler module arrive into the burst transmission queues (BTQs) while the scheduler module searches for output wavelength for scheduling. Ideally there should be an infinite buffer availability for the delay traffic, but a higher load of transit traffic results in the loss of the locally assembled bursts in case the buer is full. Whereas, the transit bursts have no buering facility available. BTQs are further explained in channel scheduling in Section 2.2.

5) Scheduling: Scheduling in a joint node is a combination of scheduling in an edge node and the core node. The core node operates in all optical domain, at the arrival of a control packet, the scheduler module searches for the time slot to fit the burst on the output wavelength channel. If the wavelength is found possibly with the need of wavelength conversion), the output wavelength is reserved for arriving burst, starting from arrival time of the burst. Otherwise, if channel is not found, the burst is dropped due to the unavailability of optical buffers. At most, fiber delay line (FDL) components may be used but FDLs do not operate similar to the electronic buffers. As discussed in Section 4.1.4, the local burst data remains in the electronic domain during scheduling, therefore, rather than dropping the bursts, it is possible to buffer them until channel is available. The scheduler module searches for availability of channel, when it is found, the burst from the front of the queue is forwarded to the scheduler buffer, and channel is reserved for the considered burst after its offset time. Channel scheduling for the joint node is explained in detail in Section 2.2.

6) Buffers: In the joint node, there are three locations which require electronic buffers.

- Assembly queues
- Burst transmission queues
- Scheduler buffer

The assembly queues themselves are buffers, packets are delayed until burst completion according to the burst assembly algorithm implemented. Burst transmission queues are the queues for local bursts, and scheduler buffer is the queue where local bursts are delayed from their control packet generation to the actual transmission.

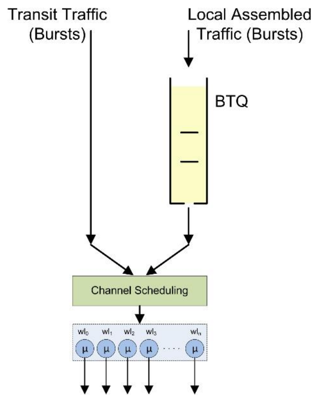

Fig. 3. Queueing diagram

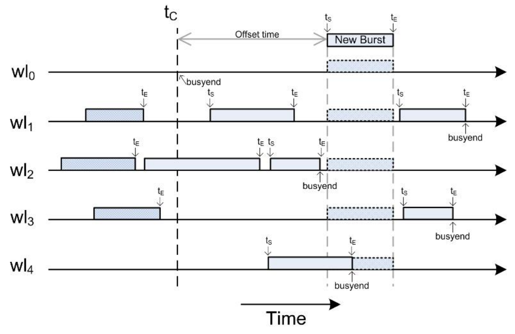

Fig. 4. Channel scheduling

7) Control Packet Processing and Generation: The control packet for the transit bursts is received after opto-electrical conversion, and forwarded to the routing module. Control packet generation module generates the control packet for the local assembled burst after the wavelength reservation by scheduling unit. This control packet contains necessary information of preceding burst, for example, burst arrival time, burst length, burst destination, selected path index and class of service. The primary function of the control packet is to reserve the resources in advance for the incoming burst.

2.2 Channel Scheduling

Channel scheduling is used to allocate suitable output wavelength channels to incoming bursts. Scheduling in an egdge node is different than scheduling in the core node. The edge node is capable of buffering bursts in electronic domain if there is no output wavelength channel available. Whereas, the core node operates in the optical domain, and if scheduling algorithm fails to find any output wavelength channel for the burst, the burst is simply dropped. Fiber delay lines are proposed to be used as buffers in the core node, but they work differently than queueing buffers. Channel scheduling in a joint node is different than both edge and core node. In a joint node, locally assembled traffic has to compete with transit traffic for output wavelengths. Similarly, the transit bursts coming from previous nodes have to compete for output resources with local traffic. The transit bursts are dealt as loss traffic and therefore, the burst is dropped if the scheduling algorithm remains unble to find a time slot for the burst to fit on output channel. While, the joint node is capable of buffering local assembled bursts. Figure 3 describes the queueing model of input traffic in a joint node. Arrivals of transit traffic go directly to wavelength channels, whereas, local bursts have buffers available.

A new burst scheduling algorithm composite edge core scheduling with void filling (CECS-VF) is proposed to schedule both local and transit bursts. This algorithm is based on LAUC-VF, but it also takes into account burst transmission queues where local assembled bursts are waiting.

Algorithm II.1: BURSTARRIVAL(B_i)

```
if LocalBurst
  then
    Insert burst into BTQ
    $B_i \leftarrow$ Front burst of BTQ
    Find ch using LAUC-VF
    if Ch Found
      then
        Reserve ch
        Pop $B_i$ from BTQ
        $B_i$forwarded to Scheduler Buffer
        CALL CHAVAILABLE() at $t_E[B_i]$
        Update statistics                                   In
      else
        $B_i$ stays in BTQ
else if TransitBurst
  then
    Find ch using LAUC-VF
    if ChFound
      then
        Reserve ch
        CALL CHAVAILABLE() at $t_E[B_i]$
        Update statistics
      else
        Drop the burst $B_i$
        Update statistics
```

2.3 Composite Edge/Core Scheduling (CECS-VF)

The CECS-VF performs a composite scheduling of both local and transit bursts. This scheduling algorithm is different from general edge scheduler in the sense that it searches and reserves wavelength when there is the time to transmit burst control packet (BCP). EPMV-VF proposed in [ ], is an edge scheduler algorithm which reserves wavelengths in advance prior to BCP transmission time. And therefore, it keeps track of burst control packet time tcp, and transmits BCP when tcp occurs. This practice if implemented in composite scheduling technique will give priority to local traffic, and increase the loss rate of transit traffic. The offset time of local burst is already calculated in router module according to number of hops this burst have to travel, therefore, when the burst arrives, the scheduler finds for the available time slot for this burst to fit on the wavelength channel. The CECS-VF algorithm is explained in Algorithm II.1 and Algorithm II.2.

CECS-VF, whenever there is an arrival, either local burst or control packet for transit burst, the BURSTARRIVAL method is called. The burst received from local router module, is inserted into BTQ, then the reference of the burst at front of the queue is retrieved. The scheduler module searches for availability of output channel after offset time of burst from burst start time tS[Bi] to burst end time t_E[Bi] using LAUC-VF algorithm. As shown in the Figure 4, channel is searched for the time slot of burst size in time after current time plus pre calculated offset time of the burst. Five possibilities can occur, there can be no burst on the channel, burst can be scheduled in the void and contention may occur. In case of contention, the channel is supposed to be unavailable. If no channel is found for this time slot, the burst remains into BTQ. If a channel is found, it is reserved, BCP for the burst Bi is transmitted, and burst is moved from BTQ to scheduler buffer and statistics are updated. One more event is scheduled at the end time of scheduled burst t_E[B_i], i.e., CHANNEL AVAILABLE. This event tells the scheduler that there is void available at this moment and output wavelength channel is available for scheduling. When scheduling the local burst, all output wavelength channels are traversed for availability, and selected according to LAUC-VF.

Algorithm II.2: CHAVAILABLE( )

```
if BTQ ! = $NULL$
  then
    $B_i \leftarrow$ Front burst of BTQ
    Find ch using LAUC-VF
    if ChFound
      then
        Reserve ch
        Pop $B_i$ from BTQ
        $B_i$ forwarded to Scheduler Buffer
        CALL CHAVAILABLE() at $t_E[B_i]$
        Update statistics
      else
        $B_i$ stays in BTQ
```

The CHANNELAVAILABLE is called whenever there is availability of void on output wavelength channel. If there are bursts in the BTQ, this module performs same functions as the scheduling of local burst is performed.

3. Results and Discussion

3.1 Simulator

This section presents an overview of OBS simulator designed to facilitate the study of OBS networks. The simulator is built upon a general network simulator IKNSim was originally developed in 2004 [8], many protocols were added over the years to aid in the research work. The current version of the simulator is referred to as IBKSim. This simulator is capable of simulating tiny queueing models, single network nodes and able to asses the performance of bigger networks. Its modular structure provides high degree of flexibility and object oriented design approach facilitates extension on the development part. A detailed description about the simulator is beyond the scope of this document, and only fundamental concepts are covered here needed to simulate OBS networks. For the basic concept of IBKSim and the use of XML as configuration and logging language the readers are referred to [9]. The architecture of optical burst switched (OBS) networks has complex structure and components of OBS network have high degree of freedom in traffic pattern, arriving packet distribution, assembly techniques, service class, routing protocols, scheduling algorithms, buffering requirements, burst size, offset time settings and signaling protocols. So far, a vast literature on OBS is there and analytical modeling providing insights to individual components of OBS networks, but might scarcely cope with multiple factors that may arise in complete network scenarios [10]. Therefore, there coexists a need of simulation tools to evaluate the performance of complex network architectures such as OBS. The OBS module of IBKSim takes XML configuration file as input and generates logging file in the form of XML file. In the configuration file the user can describe the whole simulation scenario, such as, run time of simulation or number of specific events the simulation has to run, network, nodes, components within the nodes, links between nodes and internal links of components within a node. Figure 5. demonstrates a typical simulation flow of OBS network. After the start of simulation, packet sources connected to edge nodes start generating packets using packet size distribution and generate rate given in input configuration file. Packets are forwarded to burst assembler module which assembles packets into bursts according to given assembly scheme. Assembled bursts are forwarded to routing module, which looks up the routing table to forward the burst to relevant scheduler or the sink in case current node is destination for the burst. Scheduler module searches for the output wavelength, if found, the burst is forwarded to the routing module of next node. Simulation lasts until the input simulation time or the number of time the specific event is occurred. OBS is mainly composed of edge router and core router, which further contain many components having number of internal connections. Both edge and core router, and main components are discussed below.

Joint Edge-Core Components: The joint edge-core node (JECN), assembles the data bursts performing the function of edge router, and forwards transit bursts to the next nodes while performing the function of core router. Components in the simulation node of joint node are shown in 2. Packets arrived from different packet sources are assembled into bursts which are forwarded to the edge scheduler after routing decision by routing module. Transit bursts are received by routing module as well, which are also forwarded to scheduler module. The scheduler module of joint node differentiates between locally assembled bursts and transit burst, buffers the local bursts in BTQs, whereas, transit bursts are scheduled without buffering. Forwarded bursts are sent to the routing module of the next node, which after routing decision, forwarded to the channel scheduler.

3.2 Discussion

In this section, the behavior of ECJN is analyzed in terms of mixing of local and transit traffic using simulation environment described in Chapter 3. It is worth mentioning that higher rate of transit traffic causes increase in waiting time of local bursts in the burst transmission queues (BTQs), and vice versa, higher rate of local traffic causes higher burst loss of transit traffic.

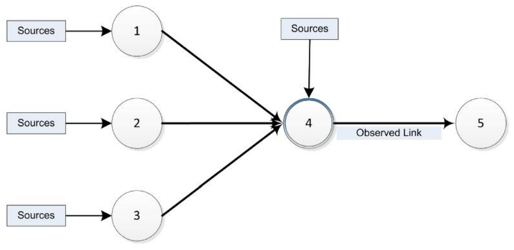

Fig. 5. sim Network

Simulation Setup: Simulations of edge-core joint node are performed on a 5-node sample topology presented in Figure 5. In this topology, node 1 ,2, 3 are ingress edge nodes and node 4 is a joint node. All of the four nodes generate bursts which are destined to egress node 5. The local bursts assembled at joint node 4 compete for output wavelengths with transit bursts arriving from previous nodes. Figure 5: 5-node topology, with node 4 as a joint node. Packet arrivals to the nodes are Poisson with exponentially generated mean packet length of 40 kbits. Bursts are assembled using hybrid based burst assembly mechanism with time threshold of 100 micro second and size threshold of 1 Mbits. CECSVF explained in previously is used for wavelength scheduling. Burst transmission queues (BTQs) can buffer upto 1000 bursts. Following assumptions are taken:

- The number of wavelengths on each link is 4 with 10 Gbits capacity.
- Full wavelength conversion is available at node 4.
- No re-attempt is performed when a connection is blocked.

Calculation of offered network load and blocking probabilities are calculated using standard procedure. Total simulation time is divided into 20 intervals plus a warm-up or transient period and 95 % confidence interval is used to determine the accuracy of the output. 0:1 million bursts are generated for each simulation interval. Figure 6 plots the waiting time of local bursts as a function of offered transit load while the local load is fixed at 0:2, 0:3 and 0:5. It is observed that with the increase in transit load, the waiting time of the local bursts increases. Similar behavior is shown in Figure 7, which plots the BTQ size for local bursts. Figure 8 draws the mean blocking probability of transit bursts as a function of local load. It can be observed that with the increase in local traffic load, transit traffic is affected badly in terms of burst blocking probability. The results show that if both local and transit traffic has equal priority, both will effect the traffic of other. The usage of wavelengths can be restricted for both local and transit traffic to achieve a bargain between mean waiting time of local traffic and the loss rate of transit traffic. Mixing of local and transit traffic is studied in to control the output channel sharing by extending the basic mixed loss-delay queueing models and are solved by Markov chain techniques.

4. Conclusion

This paper introduced the requirement of an edgecore joint node (ECJN) in optical burst switched (OBS) networks. Architecture of ECJN is described in detail with functionalities of each module. A new scheduling algorithm, composite edge-core scheduling with void filling (CECS-VF) for multiplexing both local and transit traffic is proposed and the buffer requirements for the ECJN is discussed. The effect of high transit load on local assembled bursts, and effect of high local load on transit traffic is analyzed through simulations.

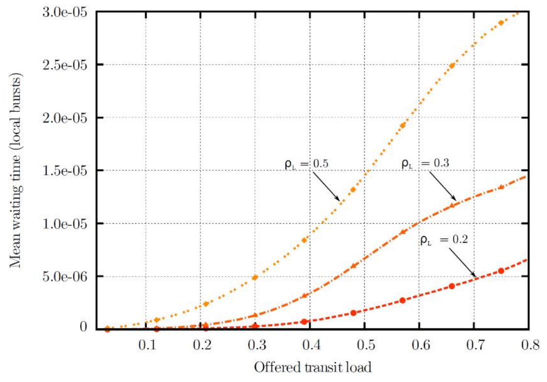

Fig. 6. Waiting time of local bursts as a function of offered transit load

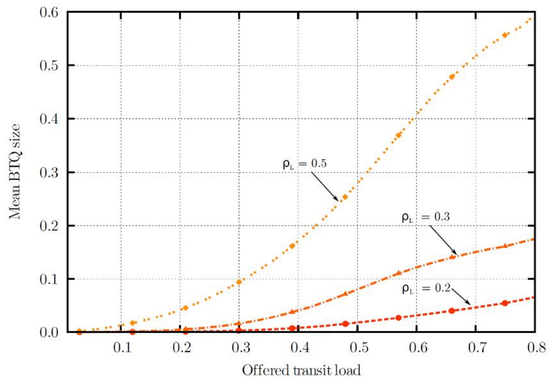

Fig. 7. Mean BTQ size as a function of offered transit load

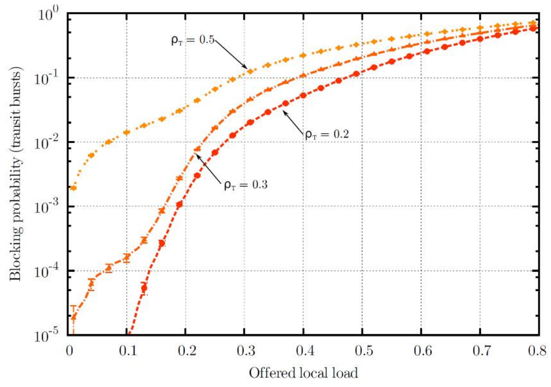

Fig. 8. Mean blocking probability of transit bursts as a function of offered local load

5. References

[1] A. Barradas and C. Medeiros. Edge-node deployed routing strategies for load balancing in optical burst switched networks. ETRI Journal, 31(1):31–41, 2009.

[2] A. Barradas and M. Medeiros. Pre-planned optical burst switched routing strategies considering the streamline effect. Photonic Network Communications, 19:161–169, 2010.

[3] R. Bikram, N. Charbonneau, and V. Vokkarane. Coordinated multi-layer loss recovery in TCP over optical burst-switched (OBS) networks. Proc. IEEE International Conference on Communications ICC 2010.

[4] F. Callegati, W. Cerroni, and C. Rafaelli. Impact of optical packet loss and reordering on TCP performance. Proc. IEEE GLOBECOM 2006.

[5] S. Gunreben. An optical burst reordering model for timebased and random selection assembly strategies. Performance Evaluation, 68(3):237–255, 2011.

[6] S. Lee, I. Hwang, and H. Park. A new burst scheduling algorithm for edge/core combined optical burst switched networks. Lecture Notes in Computer Science, NETWORKING 2006.

[7] C. Yuan, Z. Zhang, Z. Li, Y. He, and A. Xu. A unified study of burst assembly in optical burst switched networks. Photonic Network Communications, 21:228–237, 2011.

[8] J.H. Schuringa. Mobile Routing Agents deployed in Resilient Packet Networks. PhD thesis, Vienna University of Technology, Vienna, Austria, 2004.

[9] Lukas Wallentin, Marco Happenhofer, Christoph Egger, and Joachim Fabini. XML meets Simulation: Concepts and Architecture of the IBKSim Network Simulator. Eurosim Simulation News Europe, vol. 1, pp. 16{20, 2010.

[10] Oscar Pedrola, Miroslaw Klinkowski, Davide Careglio, Josep Sol-Pareta, Sbastien Rumley, and Christian Gaumier. JAVOBS: A Flexible Simulator for OBS Network Architectures. Journal of Networks, vol. 5, no. 2, pp. 256{264, Feb 2010.

8/8/2012