Flexible Simulators for OBS Network Architectures

Oscar Pedrola[1], Sébastien Rumley[2], Miroslaw Klinkowski[1,3]
Davide Careglio[1], Christian Gaumier[2] and Josep Solé-Pareta[1]*
[1] *Technical University of Catalonia (UPC), Jordi Girona 3, 08034 Barcelona, Catalunya, Spain*
[2] *Swiss Federal Institute of Technology (EPFL), CH-1015 Lausanne, Switzerland*
[3] *National Institute of Telecommunications (NIT), 1 Szachowa Street, 04-894 Warsaw, Poland*
** Tel. (+34) 93 4016982, Fax. (+34) 93 4017055, e-mail: pareta@ac.upc.edu*

ABSTRACT
Since the OBS paradigm has become a potential candidate to cope with the needs of the future all optical networks, it has really caught the attention from both academia and industry worldwide. In this direction, OBS networks have been investigated under many different scenarios comprising numerous architectures and strategies. This heterogeneous context encourages the development of flexible simulation tools. These tools should permit both an easy integration of any possible new network protocol design and a rapid adaptation to different performance target goals. In this paper, we present two OBS network simulators, namely, a C-based simulator (ADOBS) and our novel Java-based simulator (JAVOBS). We compare their performances and we provide some exemplary results that point out remarkable flexibility that can be achieved with the JAVOBS simulator.
Keywords: optical burst switching (OBS), simulation tool, flexibility.

1. INTRODUCTION

To move towards IP-over-WDM architectures, various optical switching techniques have been under intensive research. Among them, three switching paradigms have appeared as potential candidates. Firstly, Optical Circuit Switching (OCS) [1] pursues a wavelength routed networking architecture with a whole wavelength as its finest granularity. However, it lacks both the flexibility and efficiency required to cope with the needs of current traffic patterns. Secondly, in the Optical Packet Switching (OPS) approach [2], each packet is sent into the network with its own header. This header is going to be either electronically or all-optically processed at each intermediate node while the packet is optically buffered. Although OPS may be seen as both the natural choice and conceptually ideal for the future all-optical networks, current optical technology is still immature and not able to overcome its exigencies. Finally, in order to provide optical switching for next-generation Internet traffic in a flexible yet feasible way, the Optical Burst Switching (OBS) paradigm was proposed in [3]. In an OBS network, a burst control packet (BCP) is sent out-of-band both to reserve all resources and to set up the path for its burst of data, which will be sent optically after an offset time in a cut through manner. In this way, OBS allows for an efficient use of resources without the need of optical buffering at any intermediate node. Although it can be seen as an intermediate step of the migration from OCS to OPS, OBS has emerged as a more competitive choice for the transmission of data traffic in the near future. In essence, OBS combines the best from both OCS and OPS while avoiding their shortcomings. Consequently, OBS has received an increasing amount of attention from the optical research community and has become, nowadays, a research field of its own.

OBS networks display a complex structure and the design of their constituent elements offers several degrees of freedom. So far, much of the research on OBS networks has been conducted through theoretical analysis. Undeniably, the analytical approach can provide valuable insights in reduced complexity scenarios but might scarcely cope with the multiple factors that hide behind a complete network schema. Simulation tools have become essentials to evaluate complex OBS network scenarios. Indeed, simulators solve many difficulties such as the need to build a real system, but more important, they allow for the reproducibility of results, which is the basis for scientific advance.

In this paper, we present both a C-based simulator (ADOBS) [4] and our novel Java-based simulator (JAVOBS). Considering how rapidly new strategies are engineered to improve the performance of OBS networks, it is our objective to demonstrate how versatile a simulation tool should be in order to be able to provide reliable results in a relatively fast yet straightforward way. Section 2 gives an insight of the wide variety of OBS schemes proposed so far. Section 3 gives a review on the OBS simulation tools presents in the literature. Section 4 presents and compares the ADOBS and JAVOBS simulators. Section 5 provides some exemplary results achieved by the JAVOBS simulator. We conclude this paper in Section 6.

2. OBS REVIEW

An OBS network is composed by two types of nodes, namely edge and core nodes. Edge nodes are in charge of assembling input packets coming from different sources (e.g. IP, Ethernet) into bursts. Then, for each burst, a separate BCP is sent well in advance, to reserve resources (e.g. bandwidth on a desired output channel) along the way from the ingress node to an egress node. Core nodes in OBS are responsible for switching individual bursts and for reading, processing, and updating burst control packets. The BCP carries, among other

978-1-4244-2626-3/08/$25.00 ©2008 IEEE

information, the offset time at the next hop, and the burst length. Finally, at the egress node, bursts are disassembled.

2.1 Burst Reservation Protocols

In order to transmit bursts over an OBS network, a resource reservation protocol must be put in place to ensure the allocation of resources and to properly configure the optical switch before the corresponding data burst arrives at the node. A wavelength-routed OBS reservation protocol was proposed in [5] as a two-way reservation scheme (i.e. a burst cannot be sent without the successful reception of an acknowledgement). Nevertheless, much of the research has been devoted to the one-way reservation scheme aiming to reduce the light-path setup time and consequently increase the resource utilization in OBS networks. The just-in-time (JIT) [6], Horizon [7] and just-enough-time (JET) [3] resource reservation protocols are the most well-known one-way reservation schemes. More recently, JIT+ [8] and E-JIT [9] protocols have also been proposed. The main difference between the one-way reservation schemes stems from the manner in which output wavelengths are reserved for bursts. These schemes include: (a) immediate reservation (JIT, E-JIT); (b) delayed reservation with void filling (JET); (c) delayed reservation without void filling (Horizon); (d) modified immediate reservation (JIT+).

A comparison of the JIT, JIT+, JET and Horizon protocols can be found in [8]. Delayed schemes produce a more efficient use of resources, especially when void filling is applied, and perform better in terms of burst loss probability. However, the sophisticated scheduling algorithms that they require increase the processing times of BCPs at intermediate nodes. Thus, in such scenario, the simplicity of JIT may balance its relative poor performance [8]. Indeed, in contrast to the other protocols, hardware implementations of the JIT signalling protocol have already been realized and published [10].

2.2 Burst Scheduling

When a core node receives a BCP, it must decide which channel to reserve to forward the corresponding data burst. The goal of a scheduling algorithm is to obtain the right switching configuration matrix for efficiently transferring input traffic to the desired output.

To date, several algorithms have been proposed to solve the wavelength scheduling problem in OBS networks. There are two categories of algorithms: (a) without void filling; (b) void filling. The idea behind algorithms belonging to group (a) is to find an available wavelength in a simple way. They are not aimed to maximize the use of resources but to generate low processing times. A simple scheduling algorithm, Horizon, which has also been called latest available unused channel (LAUC) [11], was proposed in [7]. Another example is the first fit unscheduled channel (FFUC) [12]. More advanced scheduling algorithms belong to group (b). These algorithms are designed both to provide efficient use of resources and to reduce blocking probabilities. However, void filling algorithms are more complex, hence difficult in implementation and time-consuming. Two void filling algorithms are: (1) latest available unused channel with void filling (LAUC-VF) [11]; (2) first fit unscheduled channel with void filling (FFUC-VF) [11]. More recently, the minimum starting void (Min-SV) and the minimum ending void (Min-EV) scheduling algorithms were presented in [13]. Min-SV and Min-EV algorithms improve significantly the processing time over LAUC-VF. However, Min-SV/EV algorithms involve time-consuming memory accesses. Therefore, both types of void filling algorithms are still considered too slow to provide a viable solution to the problem [14]. Table 1 summarizes the comparison between the algorithms based on the study in [15]. It uses the following notation: (w) number of wavelengths at each output port; (N_b) number of bursts currently scheduled on every data channel.

Table 1. Performance comparison of different scheduling algorithms.

Scheduling Algorithm	Time complexity	Bandwidth utilization
FFUC	O($log\ w$)	Low
Horizon / LAUC	O(w)	Low
LAUC-VF	O($w\ log\ N_b$)	High
FFUC-VF	O($w\ log\ N_b$)	High
Min-SV/EV	O($log^2 N_b$)	High

3. OBS SIMULATION TOOLS

OBS networks are still in a phase where several options may have their own opportunity. Therefore, to evolve there is a strong need to mimic the behaviour of real OBS networks. That is precisely the task of simulation tools. Since OBS is a relatively young field, much of the studies that can be found in the literature use quite simple simulation models. For instance, the single node approach is used in [8] and [19]. In general, these simulation models were developed in purpose for a specific situation and are not suitable to study complete OBS scenarios. On the other hand, some well-known simulators such as the widely known ns-2, have their own extensions for OBS networks [16]. A comparison of some existent OBS simulator tools can be found in [17]. To our best knowledge, none of them was specifically developed for the study of OBS networks, and thus do not

provide support to the full set of OBS representations. Besides, given their divergence of perception of the OBS scenario it is not possible to compare their results [17].

In consequence, other tools exclusively aimed to analyse OBS networks have been proposed. For example, in [18] and [19], two new simulation models are presented. Both models exploit the object-oriented approach using either the C++ in the former case or the Java programming language in the latter. The common goal of these new models is to reach the flexibility degree that simulation of OBS networks requires. Following a modular construction process, a high degree of flexibility is exhibited. At the same time, the introduction of further developments is facilitated.

4. ADOBS AND JAVOBS SIMULATION TOOLS

The ADOBS simulator [4] is completely developed in C++. Since C++ is a low level programming language, the developer deals with concepts and operations strictly connected with computer hardware. Hence, speed and efficiency are achieved at the cost of complexity. Formerly, ADOBS served to study routing algorithms in OPS networks. Lately, it has been modified to become an ad-hoc event-driven simulator for OBS networks. ADOBS has been basically used to study the performance of the OBS network layer. On the other hand, the JAVOBS simulator is a Java-based application that has been exclusively built to simulate OBS networks on top of the JAVANCO framework [20]. It implements the event-driven model together with fixed-increment time progression [21]. Thus, we consider that JAVOBS implements a hybrid discrete event simulation model. By its nature, Java is an interpreted language. This means that user code is temporarily compiled into "Java byte code", and does not become executable code until the program is actually run. Consequently, C++ runtime performance is better than that of Java. Nevertheless, we selected Java for our simulator because of both the complexity of building a simulator from scratch using C++ and the fact that Java is a really flexible and dynamic language. In addition, Java is a garbage-collected language (i.e. memory handling procedures are automatically implemented).

Both simulators provide support to both the conventional OBS control architecture (C-OBS) and the emulated offset time control architecture (E-OBS) [4]. Table 2 presents some features of both simulators.

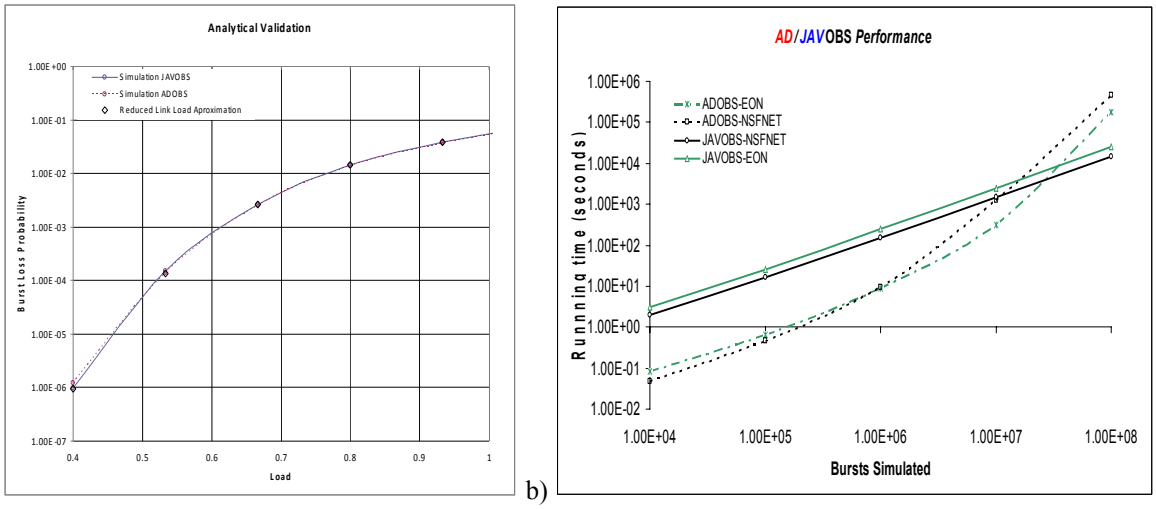
Table 2. ADOBS / JAVOBS features.

	ADOBS	JAVOBS
OBS Protocols	JET, Horizon	JET, JIT, Horizon, E-JIT, JIT+
Scheduling Algorithms	LAUC, FFUC, FFUC-VF, LAUC-VF	LAUC, FFUC, FFUC-VF, LAUC-VF
OBS Architectures	C-OBS, E-OBS	C-OBS, E-OBS
Model Building	Predefined input file	Graphical, script or xml input file.
Programming Language	C++	Java

In order to give credibility to results obtained, both simulators have been analytically validated. The analytical results are based on a reduced link load model for OBS networks presented in [22]. Figure 1.a presents the results obtained by both simulators. We used a network topology called SIMPLE [4] with 6 nodes and 8 links and the shortest-path routing algorithm. Each node is an edge node generating 25.6 Erlangs (0.8, when normalised to the link capacity) and each link has 32 wavelengths at 10 Gbit/s. Bursts have exponential distributed arrival time and length (mean 1 Mb). In obtaining the simulation results, we have estimated 99% confidence intervals. Since the confidence intervals found are very narrow, we do not plot them in order to improve readability. As can be seen, both simulators match the analytical results. Hence, in this case, we consider both simulators validated.

A test measuring the running time of both simulators has also been performed. Simulations were run according to the number of bursts generated and prompted more than one hundred hours of simulation (on an Intel Core 2 Quad 2.4 GHz desktop computer). In this case we consider two network topologies: (1) NSFNET (US network); (2) EON (a pan-European network defined in European COST 266 action) with 15 and 28 nodes, and 23 and 39 links, respectively. JET signalling and LAUC-VF scheduling are used. The mean length of bursts generated is 40 kB for this experiment. Figure 1.b presents the results obtained.

As expected, ADOBS performs better at low values due to the fact that uses C++ programming. However, it starts to change at 1 million bursts. From this point on, the ADOBS curves exhibit an exponential increase which finally creates gaps of up to 96 hours between both simulators. These results may be due to a non-optimized implementation of the ADOBS simulator. Although JAVOBS is outperformed in short simulations, we observe a constant growth of the running times for all time scales which exhibits its robustness. Thus, in this case, become clear the advantages of using a garbage-collected language.



a) b)

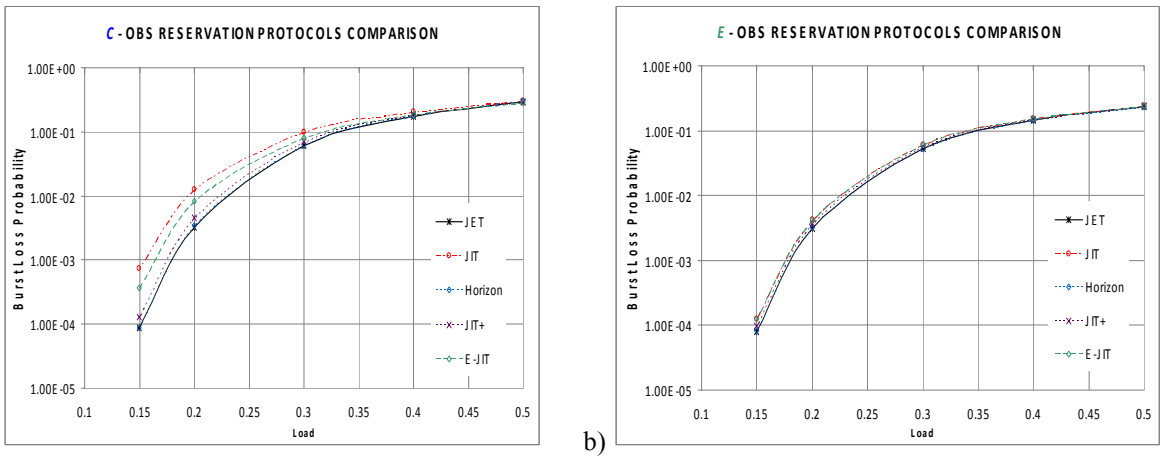
Figure 1. a) ADOBS / JAVOBS analytical validation. b) ADOBS / JAVOBS runtime test.

5. JAVOBS IMPLEMENTATIONS / RESULTS

The objective of this paper is to present the flexibility of the JAVOBS simulation tool. To do this, in this section, we present three different case-studies: (1) Performance comparison of reservation protocols under both the C-OBS and the E-OBS control architectures supported in JAVOBS; (2) Comparison of the Horizon and the Constant Time Burst Resequencing (CTBR) [14] schedulers under the single node topology; (3) Analysis of the network topology flexibility using different degrees of meshed-rings.

5.1 Comparison of the E-OBS and C-OBS control architectures

Considering that fiber delay lines (FDLs) buffers are not used, it has been proved in [23] that the best worst-case performance of an online best-effort scheduling algorithm is achieved when all bursts have the same offset time and the same length. One of the benefits of E-OBS comes from the fact that offset times are introduced at each core node by means of additional fiber delay coils inserted in the data path at the input port of the node, thus, E-OBS does not experience offset variation inside the network. In such scenario, scheduling algorithms do not need to implement any void filling technique. Therefore, in an E-OBS network, JIT and Horizon reservation mechanisms seem to be the most appropriate ones due to its low complexity compared to JET. Indeed, the over-provisioning of resources that characterizes JIT is substantially reduced using E-OBS due to smaller offset times. Figure 3 presents the comparison between both architectures under the different signalling protocols obtained by JAVOBS. We consider the EON network topology and a mean burst length of 40 kB. The processing and switching times are estimated according to [8] and [4]. We observe that using E-OBS, the performance of the 5 different signalling protocols is quite similar, thus, the possibility of reducing the network complexity by using low complexity techniques such as JIT is not unfounded. On the contrary, in C-OBS becomes clearer the advantage of using complex reservation mechanisms due to the variable offsets.



a) b)

Figure 3. Burst loss probability vs. load, in a) conventional and b) emulated offset time OBS.

5.2 Constant Time Burst Resequencing (CTBR) implementation

Since OBS has ultra high speed requirements, the bandwidth efficient scheduling algorithms proposed so far are not considered a viable solution to the problem due to its large processing times. Recently, in [14], a hardware implementation of an optimal wavelength scheduler that can produce burst schedules in a time complexity of O(1) was presented. The idea consists of producing schedules by bursts arrivals rather than BCPs arrivals. The optimal wavelength scheduler consists of two components: (a) the CTBR block; (b) the horizon scheduler. It is important to notice that the driving force behind this technique is the simplicity of horizon and its ability to operate at high speed.

In order to test, once more, the flexibility of our simulator, we have developed a set of classes to study the CTBR mechanism. To perform the simulation, we have used the parameters specified in [14] with the aim of comparing the results obtained. Since the topology utilised for the simulation is not mentioned, we assumed the single node implementation. The performance of both the Horizon and CTBR scheduler is compared. The offset times of all bursts are generated according to a lognormal distribution with mean 100 μs. Figure 4 shows the results obtained. We observe a clear match with the results presented. The burst loss probability of the horizon scheduler increases when the ratio between the offset time standard deviation and the burst length increases. On the other hand, in the CTBR scheduler, the curves remain flat regardless of the ratio variation.

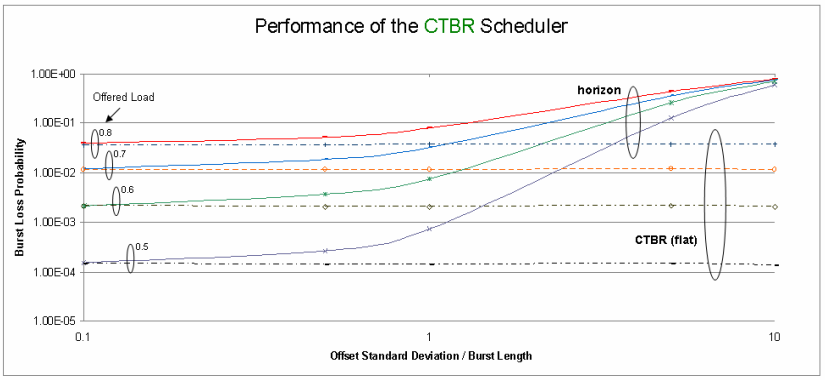


Figure 4. Performance of the CTBR scheduler

5.3 Flexible topology simulations

Eventually, the flexibility of JAVOBS has been tested regarding the network topology side. The study consists of a set of simulations over a variable ring topology. Indeed, when designing a network there is a trade-off between the costs of the deployment of resources and the performance achieved. In order to reach an optimal solution to the problem (if a solution exists), it is very helpful to have a tool permitting "*what-if*" studies. JAVOBS also allows topological modifications in a straightforward way. To prove it, we present a very simple case in an 8 node ring topology. The study consists of a simulation that starts with 8 links and 32 wavelengths per link and ends with 28 links (full-mesh) and 9 wavelengths per link. At each step, keeping intact the last topology, new links are added. In order to keep constant the network capacity, the number of wavelengths per link is recomputed at each step. Figure 5 shows the results obtained together with the topologies generated. A shortest-path routing algorithm has been used. The arrival rate λ of BCPs is such that $\lambda/\mu = 51.2$ for all scenarios. As expected using the shortest-path routing algorithm, the blocking probability is clearly reduced as more direct links between each source-destination pair become available.

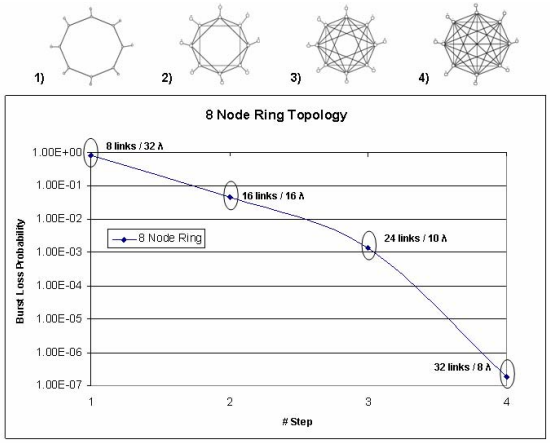


Figure 5. Ring topology study.

6. CONCLUSIONS

We have presented our novel Java-based simulation tool JAVOBS, which has been exclusively developed for the study of OBS networks. We have also given a recent overview of the existent simulation tools for OBS networks. We have verified that comparisons between simulators were impossible due to their heterogeneity. The ADOBS and JAVOBS simulators have been described, validated and compared. Finally, we have shown the flexibility of our simulator through a series of experiments that exhibit its performance. From the results of these experiments, it is concluded that: (1) as OBS networks are still undergoing intense research and development, its study requires simulation tools that facilitate the introduction of enhancements and new techniques, (2) as long as the simulation model is valid, flexible simulation tools such as JAVOBS can save time and computational resources.

ACKNOWLEDGEMENTS

The work described in this paper was carried out with the support of the CATARO-project (TEC2005-08051-C03-01) funded by the Spanish Ministry of Education and Science (MEC) and within the COST Action 291, supported by Swiss National Science Foundation.

REFERENCES

[1] I. Chlamtac, A. Ganz, G. Karmi: Lightpath communications: An approach to high-bandwidth optical wans, *IEEE Trans. Commun.*, vol. 40, no. 7, pp. 1171-1182, July 1992.
[2] D. Chiaroni: A novel photonic architecture for high capacity atm switch applications, in *Proc. PS 1995*, Salt Lake City, UT, April 1995.
[3] C. Qiao, M. Yoo: Optical burst switching (OBS)-A new paradigm for an optical Internet, *J. High Speed Networks*, vol. 8, no. 1, pp. 69-84, Jan. 1999.
[4] M. Klinkowski: Offset time-emulated architecture for optical burst switching - modelling and performance evaluation, *Phd's Thesis,* DAC department, Technical University of Catalonia (UPC), Feb. 2008.
[5] M. Duser, P. Bayvel: Analysis of a dynamically wavelength-routed, optical burst switched network architecture, *IEEE/OSA J. Lightwave Technol*, vol. 20, no. 4, pp. 574-585, April 2002.
[6] J. Y. Wei, R. I. McFarland: Just-in-time signaling for WDM optical burst switching networks, *J. Lightwave Technol.*, vol. 18, no. 12, pp. 2019-2037, Dec. 2000.
[7] J. S. Turner: Terabit burst switching, *J. High Speed Networks*, vol. 8, no. 1, pp. 3-16, Jan. 1999.
[8] J. Teng, G. N. Rouskas: A detailed analysis and performance comparison of wavelength reservation schemes for optical burst switched networks, *Photonic Network Commun.*, vol. 9, pp. 311-335, May 2005.
[9] J. J. P. C. Rodrigues, *et al.*: Enhanced Just-in-Time: A new resource reservation protocol for optical burst switching networks, in *Proc ISCC 2007*, Aveiro, Portugal, July 2007.
[10] I. Baldine, M. Cassada, A. Bragg, G. Karmous-Edwards, D. Stevenson: Just-in-time optical burst switching implementation in the ATDnet all-optical networking testbed, in *Proc. Globecom 2003*, San Francisco, CA, Dec. 2003.
[11] Y. Xiong, M. Vandenhoute, H. Cankaya: Control architecture in optical burst switched WDM networks, *IEEE J. Select. Areas Commun.*, vol.18, pp. 1838-51, Oct 2000.
[12] K. Dozer, C. Gauger, J. Spath, S. Bodamer: Evaluation of reservation mechanisms for optical burst switching, *AEU Internat. J. Electron. Commun*, vol. 55, no. 1, Jan. 2001.
[13] J. Xu, C. Qiao, J. Li, G. Xu: Efficient channel scheduling in optical burst switched networks, in *Proc. IEEE Infocom 2003*, vol. 3, pp. 2268–2278, San Francisco, CA, March 2003.
[14] Y. Chen, J. S. Turner, P.-F. Mo: Optimal burst scheduling in optical burst switched networks, *J. Lightwave Technol.*, vol. 25, no. 8, August 2007.
[15] V. M. Vokkarane, J.P. Jue: Segmentation-based nonpreemptive channel scheduling algorithms for optical burst-switched networks, *J. Lightwave Technol.*, vol. 23, no. 10, Oct. 2005.
[16] Optical Internet Research Center. OIRC OBS-ns simulator: http://wine.icu.ac.kr/~obsns/index.php, accessed at October, 2007.
[17] V. Soares, *et al.*: OBS simulation tools: A comparative study, in *Proc. ICC 2008*, Beijing, China, May 2008.
[18] M.Casoni, *et al.*: M_OBS_SIM: A powerful modular optical burst switched (OBS) network simulator, *Simulation Modelling Practice and Theory*, Elsevier Journal, available on line.
[19] J. Rodrigues, *et al.*: Object-oriented modelling and simulation of optical burst switching networks, in *Proc. GTC Workshops*, collocated with Globecom 2004, Dallas, TX, Nov. 2004.
[20] S. Rumley, C. Gaumier, et al.: Software tools and methods for research and education on optical network, in *COST Action 291 Final Report*, June 2008.
[21] C. Phillips: A review of high performance simulation tools and modeling concepts, *Recent Advances in Modeling and Simulation Tools for Communication Networks and Services*, pp. 29-48, Springer, 2008.
[22] Z. Rosberg, H. L. Vu, M. Zukerman, J. White: Blocking probabilities of optical burst switching networks based on reduced load fixed point approximations, in *Proc. IEEE Infocom 2003*, San Francisco, CA, March 2003.
[23] J. Li, *et al.*: Maximizing throughput for optical burst switching networks, in *Proc. Infocom 2004*, Hong Kong, March 2004.