A combined TCP aware scheduling and assembly scheme for OBS networks

(Invited Paper)

Kostas Ramantas and Kyriakos Vlachos, *Member IEEE*

Computer Engineering and Informatics Department & Research Academic Computer Technology Institute, University of Patras, Rio, Greece (Tel: +30 2610 996990, email: kvlachos@ceid.upatras.gr)

Abstract— **The efficient transmission of TCP traffic over OBS networks is a challenging problem, due to the high sensitivity of TCP congestion control mechanism to losses. In this work, we propose solutions to this problem, through the burst assembly and scheduling mechanism of OBS. We investigate and propose algorithms to achieve performance enhancements for TCP traffic by taking into account TCP state information. We prove our claims with references to well-established TCP models, and validate the proposed algorithms through simulations.**

Index Terms— **TCP scheduling, burst Scheduling, burst assembly, optical burst Switching, preemptions, class of service.**

I. INTRODUCTION

Optical burst switching (OBS) [1] has been introduced to combine both strengths of packet and circuit switching and is the most promising technology for next generation optical Internet. An OBS network consists of a set of optical core routers, with edge routers at its edges that are responsible for the burst assembly/disassembly function. In OBS networks, an optical burst is constructed at the network edge, from an integer number of variable size packets. Various burst assembly and burst scheduling algorithms have been proposed in the bibliography, but the efficient transmission of TCP over OBS networks remains an open problem, since the (relatively) high packet loss ratio experienced in OBS networks is incompatible with TCP congestion control mechanism. Performance of TCP over OBS networks has been studied in previous works, [2],[3], where it has been observed that the burst losses have significant impact on the TCP end-to-end performance. In particular, TCP transmission over OBS networks suffers from the high number of segments which are lost, upon a single burst drop. This typically results in many sources timing out and which will subsequently enter a slow start phase. Such an event, will significantly delay their transfers.

The assembly process also affects TCP end-to-end performance, by introducing an unpredictable delay that challenges the window mechanism used by TCP protocol for congestion control. Short assembly times reduce the total end-to-end delay associated with the round trip-time, and thus they are more appropriate for slow flows [4]. Long assembly times, are more efficient especially for fast TCP flows since they allow the transmission of multiple segments per burst and achieve a high DFL gain [3]. However, this throughput gain may be compensated by the large increased delay. Useful insights on TCP traffic statistics is given in,0,[5], while in [6] the estimated burst loss probability is combined with the TCP sending rate, used as the input load over a single link in the network.

Scheduling algorithms are vital for sustaining TCP throughput. OBS, as a real-time system suffers from a very bad worst case performance [7], as QoS provisioning in one-way signaling protocols like JET is a challenging problem, given the lack of buffering at the core. Bursts cannot be stored in intermediate nodes, so contention resolution options are limited, and priority inversions among high-priority and low-priority bursts are difficult to tackle. This is because an online OBS scheduling algorithms (like, LAUC-VF) have no information about future bursts, and thus inevitably makes non-optimal decisions regarding channel selection and burst dropping. For example, if the control packet, of a low priority burst arrives at an OBS node before the control packet of a higher priority overlapping burst, LAUC-VF has no means to give priority to the latter [8].

In this work, we investigate ways to achieve performance enhancements for TCP traffic, in terms of throughput maximization and fairness, by taking into account intrinsic details of TCP protocol in the burst assembly and scheduling process. We design TCP-friendly burst assembly and scheduling protocols and test, whether these can achieve a significant performance advantage. In particular, we propose a new multi-class TCP-aware preemptive scheduling scheme that supports strict priorities and we perform an in-depth study through mathematical modeling and simulation. For their evaluation, we have performed some real-world experiments using Network Simulator (ns-2) environment with synthetic traffic representing typical internet use, combining long-lived TCP connections (ftp transfers) and short-lived connections (mostly http traffic or small file transfers).

The rest of the paper is organized as follows: Section II presents a new congestion window-based burst assembly schemes, and provides insight on the interaction between TCP congestion control, OBS scheduling and burst assembly algorithms. Section III presents a multi class TCP-aware preemptive scheduling algorithm, which provides QoS guarantees to TCP flows according to their throughput characteristics. Its efficiency is evaluated experimentally, through ns-2 simulations. Finally Section IV concludes the paper.

Digital Object Identifier: 10.4108/ICST.BROADNETS2009.7250
http://dx.doi.org/10.4108/ICST.BROADNETS2009.7250

II. Congestion window based assembly schemes

Based on previous research work on TCP over OBS [3], it is clear that fixed timer-based burstifiers are not appropriate, since they do not provide maximum performance for all TCP flows, but only optimal performance for individual flows with similar characteristics (i.e. file size, size of congestion window, etc.). The performance of a TCP flow over OBS networks depends not only on burst drop probability, but also on the assembly algorithm used, the flow access rate and the TCP implementation. Steady state performance of TCP flows depends on congestion window evolution, while TCP send rate is bounded by $\frac{\overline{cwnd}}{RTT}$. TCP congestion window evolution on OBS networks is affected by burst drop probability p and Delay First Loss (DFL) gain. Due to the strong correlation of TCP segment losses, in practice the average drop probability of a single segment is smaller than burst drop probability, and ranges between p (for slow flows) to $(1-(1-p)^2)$ for fast flows 0. Thus, the period between segment losses is increased, which in turn enables TCP to reach a larger sending window.

A. Congestion-window based burst assembly algorithm

DFL gain, as described in [2] increases proportionally to the square root of the number of segments per burst, so large assembly timers will generally lead to increased correlation gains. On the other hand, RTT increases linearly with the assembly timer, as RTT = $RTT_0 + 2T_{MAX}$. Thus, there exists a trade-off between maximizing DFL gain and minimizing delay penalty (denoted by RTT) for a specific TCP flow, based on its congestion window and there is an optimal assembly timer that maximizes its TCP throughput. On these grounds, multi-class congestion window based assembly scheme was proposed in [4], which dynamically assigns TCP flows to burst assembly classes with different timers, according to their TCP congestion window, as denoted by Eq. (1).

$$T_{assembly\ time} = \begin{Bmatrix} 1msec & if\ window < B \\ 5msec & if\ B < window < C \\ 10msec & if\ window > B \end{Bmatrix} \qquad \textbf{Eq. (1)}$$

According to Eq. (1), TCP flows are dynamically assigned to classes on each burstification cycle, based on the (static) congestion window limits of C and B (in terms of segments). This scheme, albeit maximizing throughput of TCP flows for a given burst loss ratio, it also increases contention, since using multiple burtifiers per destination increases the number of bursts transmitted. In this work, we extend and revise the abovementioned burst assembly scheme, with an aim to reduce the contention induced by multiple burstifiers and focus on fairness issues as well as TCP throughput maximization. Our first step towards this direction is to serialize bursts from different classes. Bursts from different traffic classes of the same burstifier, are transmitted in an interleaved fashion (see Figure 1), and thus assembly timers have to be integer multiples of each other. At the same time, buffering in the electronic domain is employed at the edge, to prevent contention among different traffic classes at the same edge node (i.e. the burstifier will never send two bursts that will contend with each other at the edge).

In what follows, we will investigate how it is possible to achieve further performance enhancements for TCP traffic (in terms of throughput maximization and fairness) if we take into account some intrinsic implementation, independent details of TCP (like TCP window state).

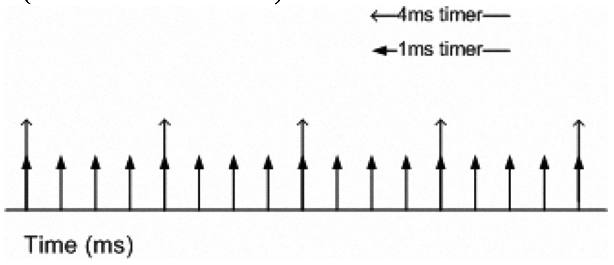

Figure 1: Interleaved transmission of bursts in a multi-class burstifier

In this work, we used only two classes per burstifier, which provide a good compromise between maximizing throughput and keeping contention at acceptable levels. We used one low-latency traffic class, which features a short timer to help the congestion window to evolve fast. The other traffic class has a large assembly timer for flows that have a large enough congestion window, to support a large DFL gain. However, the basic design goal of the proposed burst assembly scheme is to limit the number of segments in the class with the short timer, to avoid contention to the "*normal*" priority class, which still contains the majority of TCP traffic. In the next section, we provide a solution to the problem of which segments should be assigned to the low-latency class, using some well-established TCP protocol analytic models.

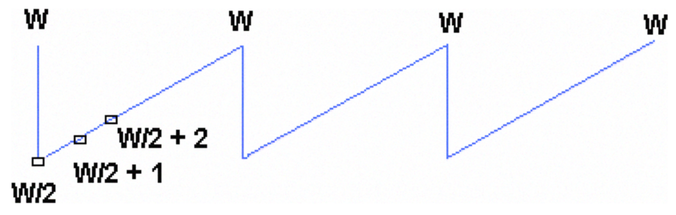

Figure2: TCP Congestion window evolution at the steady state

B. Theoretical Analysis

In this section, using some well-established TCP models [10] we provide a solution to the problem of choosing a minimum number of TCP segments to be assigned to the low-latency class, for maximizing TCP throughput, without inducing too much contention to the "bulk" TCP traffic class. First, we present the principle of operation for the TCP protocol, focusing on TCP steady state (see Figure 2). A TCP flow starts its data transmissions at an *initial slow start* phase, followed by a *steady-state* phase. TCP tries to capture the maximum available bandwidth during the initial slow start phase, by doubling the TCP window (and the sending rate as a result) on every RTT. At the end of the slow start phase, which is indicated by the first packet loss, TCP goes into the steady-state. Steady-state of a TCP flow comprises of a series of timeout periods (TO), each of which is composed of a *slow start* phase and multiple triple duplicate periods (TDP). During the slow start phase the TCP window increases linearly (i.e. it is incremented by one segment per RTT) and it is divided by two per every packet loss.

The idiosyncrasies of TCP protocol and especially its conservative approach to avoid congestion (dividing the sending rate by two after just one packet loss) make it impossible for the TCP protocol to sustain a large congestion window, unless the packet loss ratio is unrealistically small.

Table I: Steady State TCP window for various packet loss probabilities.

Packet_loss	tcp_window
10^-2	8
10^-3	25
10^-4	80
10^-5	252
10^-6	800
10^-7	2530
10^-8	8000
10^-9	25298
10^-10	80000

According to the TCP steady state model [10] for a given packet loss ratio, TCP steady state window converges to W_o for packet drop rate of p, see Table I for some numerical examples.

$$W_0 = \sqrt{1.5} \Big/ \sqrt{p} \qquad \textbf{Eq. (2)}$$

In OBS networks, where burst losses happen due to contention, even at light load conditions, packet loss cannot be arbitrarily small (typically it is in the range of 10^{-4} to 10^{-2}). Thus, TCP flows cannot sustain large congestion windows. In fact, congestion windows larger than W_0 are not sustainable, while smaller ones are limiting the throughput potential of the flows. Thus, to achieve TCP throughput maximization, we should minimize the time needed for the flows to reach W_0 and then allow them to oscillate between $W_0/2$and W_0. At the point, where a TCP window exceeds W_0 (and provided that the burst loss probability is relatively small), then we can safely assume that it will be large enough to better be served by a larger assembly timer, which will provide a higher DFL gain. Thus, in the proposed scheme, when the window of TCP flows that are in the slow start phase, exceed W_0, then these flows are assigned to the "normal" assembly class, which features a larger assembly timer than that of the low-latency class. Further, they will only return to the low-latency class after a possible time-out event. In this way, the low-latency class has a relatively light load (our experiments showed that typically less than 10% of flows fall in this category at the steady state). Consequently, the risk of increasing contention to the TCP flows that are at the congestion avoidance phase is effectively avoided.

Finally, since in an OBS network burst drop probability is not equivalent to packet loss probability, a good approximation of steady state window W_0 is derived from Eq. 3, [2]. This formula takes into account DFL gain experienced from TCP flows in OBS networks, though parameter S (where S is the ratio of segments/burst).

$$W_0 = \sqrt{3 * S} \Big/ \sqrt{2p} \qquad \textbf{Eq. (3)}$$

III. MULTI CLASS PREEMPTIVE TCP-AWARE SCHEDULING

In the previous sections it has been stressed, how critical the slow start phase is for the evolution and sustainment of a high TCP throughput. A single packet loss at the very early stages of the slow start phase will abruptly end the exponential evolution of TCP congestion window, driving the TCP flow to the congestion avoidance phase. This causes fairness issues, as some "*unlucky*" TCP flows with losses at an early round, will have a much worse performance, requiring significantly more time to reach steady state window. Therefore and in order to ensure fairness among all the active flows that compete for bandwidth in a burst, it is vital to also provide prioritized delivery (QoS) to the low-latency class.

Therefore, in order to satisfy both design goals, i.e. ensuring fairness and maximizing throughput, we propose the use of a TCP-aware scheduling algorithm, which can provide QoS guarantees to a low low-latency class, reserved for TCP segments that have been marked as critical for maximizing throughput and ensuring fairness among the active TCP flows. However, QoS provisioning in OBS networks is hard, since due to the lack of buffering at the core, priority inversions among high-priority and low-priority bursts are difficult to handle. One very effective technique proposed in the bibliography for strict priority provisioning in OBS networks is preemptive scheduling. Preemption is a well known technique that adds flexibility to the burst scheduling process, allowing the re-arrangement of already scheduled bursts, thus making QoS differentiation possible. In the next section, we present the principal of operation of PLAUC-VF scheduling algorithm, [9], which is a preemptive variation of LAUC-VF.

A. Strict priority provisioning with PLAUC-VF

In this section, we describe the principle of operation of PLAUC-VF, a preemptive scheduling algorithm based on LAUC-VF. PLAUC-VF stores burst length, class of service, as well as a unique burst identifier for all scheduled bursts. This information is typically required for preemptive algorithms to base preemption decisions, and make the reconfiguration of OBS nodes possible. The PLAUC-VF variation used in this paper has the ability to keep track of and preempt the two most recent scheduled bursts. Thus, the channel selection phase remains identical to the one used in LAUC-VF, ensuring that on average the time complexity remains low. If the channel selection fails, then the more computationally demanding preemption phase follows. When a new reservation request arrives, the preemption-capable scheduling unit follows these steps (see Figure 3 for an algorithm illustration):

1. First, it scans all channels for an idle period to schedule the burst.

2. If voids are found in more than one channel, the one that minimizes the remaining idle period is chosen, as in LAUC-VF.
3. If no voids are found, we conclude that there is at least one overlapping burst in each channel. Then, the scheduling algorithm iterates over overlapping bursts, and decides whether one of them has to be preempted, in order to free resources for the newly arrived burst. The decision is based on the priority class that the burst belongs to. As an example, if we assume that in Figure 3, Burst-0 belongs to the "*normal*" priority class, while the reservation request belongs to the high priority class, then Burst 0 will be preempted and the new burst would be scheduled in its place.

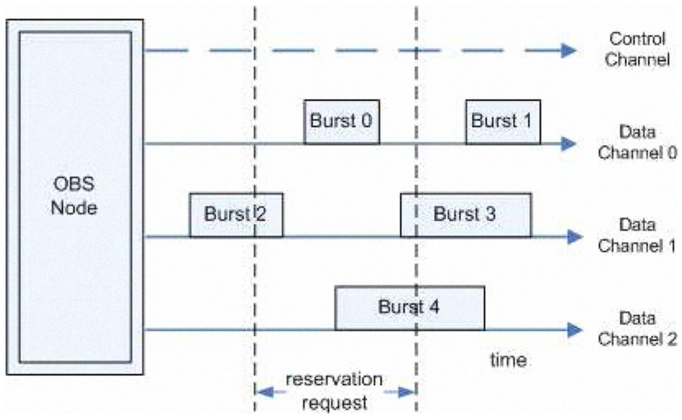

Figure 3: Illustration of PLAUC-VF algorithm

B. Experimental Setup

The preemptive scheduling algorithm was evaluated through experimentation, in the Network Simulator (ns-2) environment. The experiments were carried out on the NSF network topology, with 8 edge and 6 core nodes whereas each link was employing two wavelengths at 10Gbps. It is assumed that edge nodes have sufficient resources to store bursts and losses occur only in the core. Access rate of TCP agents to the OBS edge was set to 100Mbps. A realistic scenario of typical internet traffic was modeled, including both long lived TCP connections, that were transmitting data throughout the simulation cycle and short-lived TCP connections, which were only active for the duration of a single file transfer. Short-lived TCP arrivals (file transfers) were modeled with an exponential arrival process with λ=50 flows/sec rate at each edge and random destinations. File transfer size was modeled with a Pareto distribution process with a minimum size of 2 Mbytes. Using this set of metrics, it was possible to vary the TCP arrival rate and/or the mean file size, to obtain measurements with a different number of simultaneous active flows.

In what follows, we consider a buffer-less OBS network employing full wavelength conversion, where burst reservations are performed using JET signaling protocol. The modeled OBS network supports two classes of service. One high priority / low latency class for the transmission of "*critical*" TCP segments (as defined in previous section) and one normal class for the bulk of TCP traffic. The burst assembly process is performed in the OBS network's ingress nodes, using a timer-based aggregation algorithm as described in Section II. Packets from different classes arriving at the ingress node are assigned to distinct burst assembly queues, according to their congestion window and TCP state and are never mixed in the same burst. Since TCP protocol does not offer an option to access state information such as congestion window and TCP state, which are necessary for the proposed burst assembly and scheduling schemes, the ns-2 TCP protocol implementation had to be modified. Our approach was to modify TCPAgent class, so as to write cwnd and ssthresh values (as described in [5]) to the TCP packet headers. Of course this approach requires modifications to the TCP stack and is not viable for real-world adoption. However it is possible to estimate this information indirectly, by analyzing TCP traffic patterns. The cwnd and ssthresh fields are processed by the burstifier, for classifying packets to the normal or low latency class.

In the experiments carried out, the steady state congestion window was computed using Eq. (3). This formula uses as input the burst loss ratio and number of segments per burst (depending on the access rate and the segment size of TCP flows), to compute the steady state TCP window. For the current network setup, and given that the input load is not varied, the steady state window size was computed once, and kept constant to the value of 120. In real world applications, an appropriate feedback system is envisioned, which will be able to dynamically make approximations of steady state window of each TCP flow individually.

C. Numerical Results

In this section, we will provide numerical results obtained through simulation, regarding the performance of the proposed TCP-aware scheduling scheme. Figure 4 shows the performance of PLAUC-VF (in terms of packet loss ratio), for two priority classes at high load conditions. We can see that PLAUC-VF can effectively provide class separation, ensuring a relatively low packet loss ratio for the high priority class. It must be noted here, that preemptions in high load conditions can in fact increase the burst loss ratio of the low priority class and potentially cause starvation. However, this is not the case in the scenario modeled here. Since only some of the TCP segments are marked as critical for enhancing TCP throughput, only these are allowed to enter the high priority class (it was found to account for less than 10% of the total TCP traffic). In fact, it was measured that the effect of preemptions to normal priority class were measured to be negligible.

In what follows, average throughput is used as a performance metric for TCP traffic, since packet loss ratio is not uniform in all priority classes due to preemptions, and so it is not indicative of TCP steady state performance. In the proposed scheme we favor the forwarding of certain TCP segments that are marked as critical for maximizing throughput and achieving fairness. Figure 5 shows the average TCP throughput for a large sample of TCP flows, for 6 different assembly schemes, by keeping the input load constant. A selection of 3 possible assembly timers was chosen for the low-latency (high priority) traffic, that is 1ms, 2ms and 4ms timers and a selection of 2 timers for the normal priority class, that is 4ms and 8ms. Their combination provides in total 6 combinations, while the combination of 4ms/4ms actually corresponds to one class burstifier. On our network setup, optimal performance was achieved with the selection of 2ms timer for the high priority class, and 4ms timer for the normal priority class, yielding an average throughput of 50Mbps. A similar performance was achieved with the 2ms/8ms combination yielding an average throughput of 46Mbps. The 1ms timer was found to generate a large number of small bursts, increasing contention, and thus the 1ms/4ms system exhibited a lower average throughput of 40Mbps.

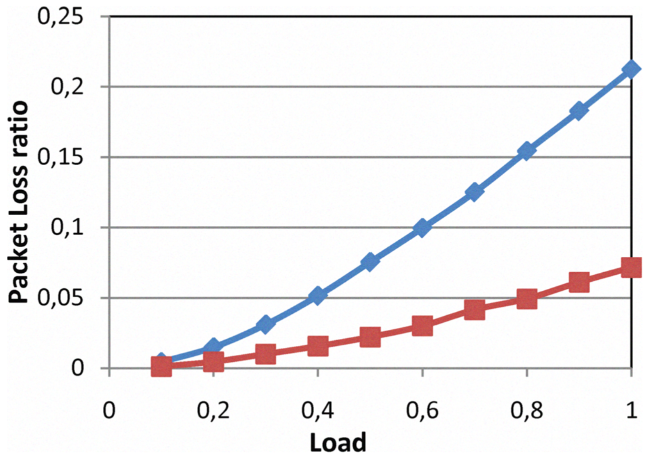

Figure 4: Packet loss ratio of two priority classes (PLAUC-VF) Low priority (CL1) and high priority (CL2).

It must be noted here that all the multi-class assembly schemes clearly outperformed the single assembly class scheme, which averaged at 35Mbps. This is attributed to the fact that the small assembly timer allows short lived TCP connections to quickly develop a large congestion window, while being at the slow start phase. When their TCP window becomes large, this short timer unnecessarily delays segment transmission, clipping DFL gain.

With respect to fairness and how bandwidth is distributed to TCP flows, Table II, shows the standard deviation of TCP throughput measured at different assembly schemes. It can be seen that the 2ms/4ms combination of timers also ensures the highest degree of fairness.

Table II: Standard deviation of TCP throughput for different multi class assembly schemes.

Timer 1(ms)	Timer 2(ms)	stdv
1	4	17
2	4	15
2	8	17
4	4	18
8	8	18

In order to further elaborate on our findings, we have measured the time that a TCP source needs for its window to reach the steady state value. Figure 6 shows the window evolution for different timers. With the selection of 1ms timer, the time it takes for the TCP window to reach W_0 =120 segments is 90ms, while for the 4ms and 8ms timer this time increases to 110ms and 133ms respectively.

IV. Conclusions

In this work, we proposed a new TCP-aware burst assembly and burst scheduling scheme. The combing scheme utilizes a high priority / low latency class, to be used by TCP packets that are marked as critical for TCP throughput evolution. We sustained our claims through simulation experiments and with references to a well established TCP model. It was found that the combined scheme can achieve significant performance advantages, enhancing average TCP throughput and sustaining fairness among all active flows.

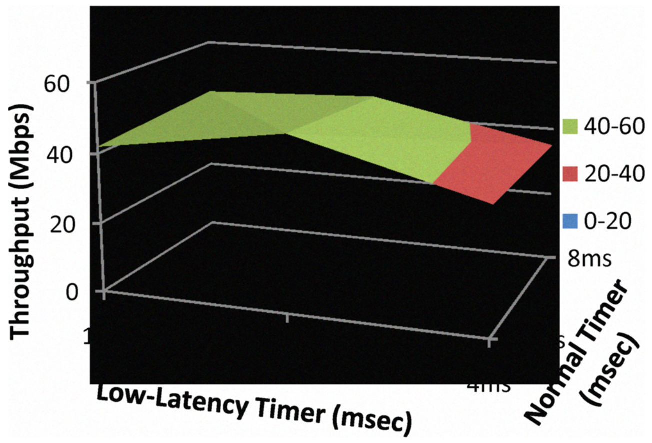

Figure 5: TCP throughput (in Mbps) for different assembly timer combinations.

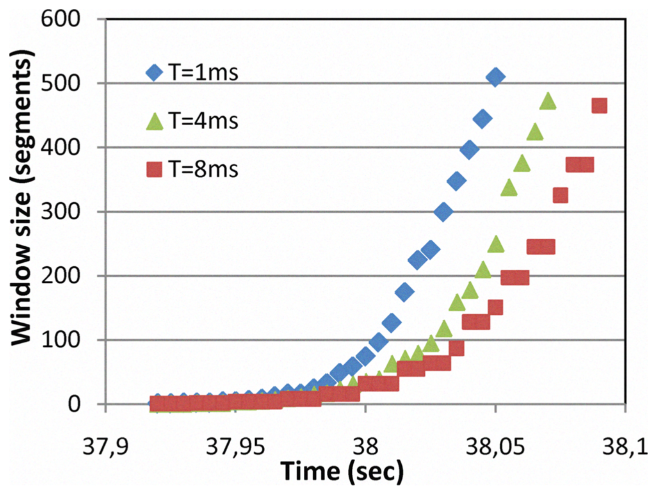

Figure 6: TCP window evolution for different T_{MAX} values.

V. Acknowledgements

The work described in this paper was carried out with the support of the BONE-project ("Building the Future Optical Network in Europe"), a Network of Excellence funded by the European Commission through the 7th ICT-Framework Programme

VI. References

[1] C. Qiao and M. Yoo, “Optical burst switching (OBS)-A new paradigm for an optical internet,” J. High Speed Networks, vol. 8, no. 1, pp. 69–84, 1999.

[2] X Yu, C Qiao, Y Liu, D Towsley, "Performance evaluation of TCP implementations in OBS networks", Technical Report 2003-13, CSE Dept., SUNY, Buffalo, 2003

[3] X. Yu, J. Li, X. Cao, Y. Chen and C. Qiao; “Traffic statistics and performance evaluation in optical burst switched networks”, IEEE/OSA Journal of Lightwave Technology, vol. 22, no. 12, pp. 2722 – 2738, Dec. 2004.

[4] K. Ramantas, K. Vlachos, Ó. G. de Dios, J. Aracil and C. Raffaelli, “Window-based burst assembly scheme for TCP Traffic over OBS”, OSA J. Optical Networks, vol. 7, no.. 5, pp. 487-495, May 2008.
A. Detti and M. Listanti, "Impact of segments aggregation on TCP Reno flows in optical burst switching networks", in Proc. of IEEE INFOCOM, vol. 3, pp. 1803 – 1812, 2002.

[5] Handley et al, "TCP Congestion Window Validation", RFC 2861, June 2000.

[6] C. Cameron, H. Le Vu, J. Choi, S. Bilgrami, M. Zukerman, and M. Kang, “TCP over OBS - fixed-point load and loss”, Optics Express, vol. 13, no. 23, pp. 9167-9174, 2005.

[7] J Li, C Qiao, J Xu, D Xu , “Maximizing throughput for optical burst switching networks”, IEEE/ACM Transactions on Networking (TON), 2007.

[8] Y. Xoing, M. Vandenhoute, and C. Cankaya. Control architecture in optical burst-switched WDM networks. IEEE J. on selected areas in communications, 18:1838–1851, Oct. 2000.

[9] K Ramantas, K Vlachos, "A preemptive scheduling scheme for flexible QoS provisioning in OBS networks", in proceed. of Broadnets 2009.

[10] J. Padhye, V. Firoiu, D. Towsley, J. Kurose, Modeling TCP throughput: a simple model and its empirical validation, in: Proc. ACM SIGCOMM, 1998.

Digital Object Identifier: 10.4108/ICST.BROADNETS2009.7250
http://dx.doi.org/10.4108/ICST.BROADNETS2009.7250