A new window-based burst assembly scheme for TCP traffic over OBS

Kostas Ramantas[1], Kyriakos Vlachos[1], Óscar González de Dios[2] and Carla Raffaelli[3]

[1] Computer Engineering and Informatics Dept. & *Research Academic Computer Technology Institute*, University of Patras, GR26500, Rio, Greece
[2]*Telefónica I+D, Emilio Vargas 6, Madrid, Spain*
[3]*Dep. of Electronics, Computer Science and Systems, University of Bologna, Italy*
[*]*Corresponding author:* kvlachos@ceid.upatras.gr

Abstract: In this paper, the impact of burstification delay on the TCP traffic statistics is presented as well as a new assembly scheme that uses flow window size as the threshold criterion. It is shown that short assembly times are ideally suitable for sources with small congestion windows, allowing for a speed up in their transmission. In addition, large assembly times do not yield any throughput gain, despite the large number of segments per burst transmitted, but result in a low throughput variation, and thus a higher notion of fairness among the individual flows. To this end, in this paper, we propose a new burst assembly scheme that dynamically assigns flows to different assembly queues with different assembly timers, based on their instant window size. Results show that the proposed scheme with different timers provides a higher average throughput together with a smaller variance which is a good compromise for bandwidth dimensioning.

OCIS codes: (060.0060) Fiber optics and optical communications, (060.1155) All-optical networks, (060.4510) Optical communications .

References and links

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. A. Ge, F. Callegati and L.S. Tamil, "On optical burst switching and self-similar traffic", IEEE Communication Letters, vol 4, no. 3, pp. 98-100, 2000.
3. M. Düser and P. Bayvel, "Analysis of a dynamically wavelength-routed optical burst switched network architecture," IEEE/OSA J. Lightwave Technol., vol. 20, pp. 574–585, Apr. 2002.
4. V.M. Vokkarane, K. Haridoss, J. P. Jue, "Threshold-based burst assembly policies for QoS support in optical burst-switched networks", in Proc. of Optical Networking and Communication Conference (OptiComm), pp. 125-136, 2002.
5. X. Yu, Y. Chen, and C. Qiao, "Study of traffic statistics of assembled burst traffic in optical burst switched networks", in Proc. of Optical Networking and Communication Conference (OptiComm), pp. 149–159, 2002.
6. X. Cao, J. Li, Y. Chen, and C. Qiao, "Assembling TCP/IP packets in optical burst switched networks" in Proc. of IEEE GLOBECOM, vol. 3, pp. 2808–2812, 2002.
7. M. Casoni and M. Merani, "On the Performance of TCP over Optical Burst Switched Networks with Different QoS Classes", Lecture Notes in Computer Science, Volume 3375/2005, pp. 574-585, 2005.
8. S. Malik and U. Killat, "Impact of burst aggregation time on performance in optical burst switching networks", Elsevier J. of Optical Switching and Networking, vol. 2, no. 4, pp. 230-238, 2005.
9. 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.
10. M. Izal and J. Aracil, "On the Influence of Self-similarity on Optical Burst Switching Traffic", in Proc. of IEEE GLOBECOM, vol. 3, pp. 2308 – 2312, 2002.

11. 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.
12. X. Yu, C. Qiao and Y. Liu, "TCP implementations and false time out detection in OBS networks", in proceeding of INFOCOM 2004, vol. 2, pp. 774 – 784, 2004.
13. 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.

1. 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. Two distinct burst assembly algorithms have been proposed in the literature: the timer-based and the threshold-based. In the timer-based method, also denoted as T_{MAX} in the literature, [2,3] a time counter starts any time a packet arrives and, when the timer reaches a time threshold (T_{MAX}), a burst is created; the timer is then reset to 0 and it remains so until the next packet arrives at the queue. Hence, the ingress router periodically generates bursts every T_{MAX} time, independently of the yielding burst size. In the second scheme, [4], a threshold is used to determine the end of the assembly process. In most cases the threshold used is the burst length, denoted in the literature as B_{MAX}. In that case, bursts are thought as containers of a fixed size B_{MAX}, and as soon as the container is completely filled with data, the burst is transmitted. The timer-based method limits the delay of packets to a maximum value T_{MAX} but it may generate undesirable burst length; the burst-length based method generates bursts of equal size, but it may result in long delays when the traffic load is light. To address the deficiency associated with these assembly algorithms, hybrid (mixed time/threshold based) assembly algorithms were proposed [5], where bursts are created when either the time limit or the burst-size limit is reached, whichever happens first. Apart from the aforementioned assembly schemes, other more complex schemes have been also proposed, which are usually a combination of the timer -based, and the threshold-based methods [6].

Performance of TCP over OBS networks has been studied in previous works [7-9] 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 entering a slow start phase that will significantly delay their transfers. The assembly process also affects their end-to-end performance, by introducing an unpredictable delay, [10], that challenges the window mechanism used by TCP protocol for congestion control. Short assembly times yield a higher throughput to TCP sources primarily because they reduce the total end-to-end delay associated with the round trip-time delay. However, short assembly times prohibit the fast increase of the congestion window since sources are allowed to transmit only a few segments per burst. Long assembly times, are more efficient especially for *fast* TCP flows [11], since they allow the transmission of multiple segments per burst. However, this throughput gain may be compensated by the large burstification delay. Useful insights on TCP traffic statistics is given in [11,12], while in [13] the estimated burst loss probability is combined with the TCP sending rate, used as the input load over a single link in the network.

In this paper, we present a TCP traffic analysis for different burstification delays. We first analyze how segments and flows are distributed over the assembled bursts for various assembly timers and further analyze their effect in the number of transmitted and lost bursts per flow. Further, window behaviour and its impact on average and variance of the yielding

throughput is investigated. We argue in this paper that the characterization of a flow as slow, medium or fast depends on its instant window size and we show that a dynamic multi-queue assembly scheme with different timers, provides a higher average throughput together with a smaller variance.

The rest of the paper is organized as follows. Section 2 presents an overview of the different TCP variants, while Section 3 a detailed TCP over OBS traffic analysis. Section 4 discusses the effect of burstification delay in the congestion window evolution and the yielding throughput, while finally Section 5 presents the performance of a new assembly scheme based on the flow's congestion window size.

2. Transport Control Protocol variants

There are a number of TCP flavours such as Tahoe, Vegas, Reno, New Reno and SACK, combined with a number of different burst assembly strategies. The last three protocols are the most interesting. The main differences among them are the algorithms that they employ when congestion is detected. TCP Reno refers to TCP with *Slow Start*, *Congestion Avoidance*, *Fast Retransmit* and *Fast Recovery* algorithms. When Reno starts, it enters the Slow Start phase first with a congestion window of one segment size and then exponentially increase it, upon the acknowledgement of all the packets transmitted. When the window reaches a certain threshold of *w*, it enters, the Congestion Avoidance phase, according to which the windows is now increasing only by one segment after all segments have been acknowledged.

In TCP Reno, two different methods are applied to identify losses; the *Time Out* (TO) and *Triple Duplicate* (TD) loss. In the *Triple Duplicate* (TD) case, the sender receives three duplicates ACKs that acknowledge the last segment before the first lost one. In that case TCP Reno enters the Fast Retransmit phase, and start transmitting the lost segments. For every successful transmission of these segments, the sender halves its congestion window and receives a TD ACK message for the next lost segment in the burst. In Reno, the maximum number of recoverable segment losses in a congestion window without timeout is limited to one or two segments in most cases. In the case of a *Time Out* (TO) loss case, no ACK is received in a certain time period, denoted by the expiration of a timer. In that case TCP Reno enters the Slow Start phase, and reduces its window back to one segment size. New TCP Reno is a slight modification according to which the sender retransmits one lost segment per round-trip-time upon receiving a partial ACK message, without waiting for a TD ACK and without halving its window until all lost segments are successfully acknowledged. On the other hand, SACK (Selective Acknowledgment) TCP implements a different ACK message, which carries the non-contiguous set of received packet sequence numbers. To this end, the sender is aware of the lost packets and which are transmitted altogether. In that case the congestion window is halved, before linearly increasing again. Detailed SACK performance in OBS networks is clearly superior, as shown in [11], since all the segments that were employed in a burst that was dropped can be identified and subsequently retransmitted at the same round.

TCP performance (e.g., throughput) heavily depends on burst assembly time due to the extra delay enforced (denoted as burstification delay). Therefore TCP mechanism adjusts its window mechanism upon a burst transmission or reception and thus timer-based assembly schemes may perform better than size-based algorithms. For the timer (T_{MAX}) threshold there could exist an optimal value that maximizes throughput performance in a TCP over OBS network [11]. In [4], it has been shown that optimal performance can also be achieved with an optimal burst length algorithm, while in [6], it is shown that a dynamic assembly algorithm that adjusts the threshold values (e.g., time, burst-length or both) according to traffic statistics can achieve an even better performance.

3. Burstification effect on segment and flow distribution

In this section, a thorough analysis of TCP traffic is presented when OBS is used as the underlying transport technology. Although TCP-SACK implementation and timer-based assembly schemes are the most promising, very few previous works providing an in-depth analysis of how segments, flows and their parameters vary when aggregation time is varied. We have developed a dedicated TCP-SACK over OBS simulator using ns-2 simulator and made modifications in the raw code to efficient manipulate TCP sources and available CPU resources, thus being capable of simulating hundreds of active sources. The experiments were carried out on the NSF network topology, with 8 edge and 6 core nodes whereas each link was employing a single wavelength at 10Gbps. It is assumed that edge nodes have sufficient resources to store bursts and losses occur only in the core. Access rate was set to 100Mbps. TCP arrivals were modeled with a Poisson arrival process with a λ=50 flows/sec rate and an exponential inter-arrival time of $1/\mu$=10msec, while TCP file size was modeled with a Pareto distribution process of p load and a min ON size of 40KByte. Using this set of metrics, it was possible to vary the TCP arrival rate and/or the mean file size, to obtain measurements for a different number of simultaneous active flows. Fig. 1(a) display the number of active flows versus the average file size selected for three different assembly timers namely for 1, 5 and 10*msec*, while Fig. 1(b) displays the corresponding burst loss ratio. Results shown hereinafter correspond to the steady state (constant number of active sources) of the experiments after 20sec of execution time.

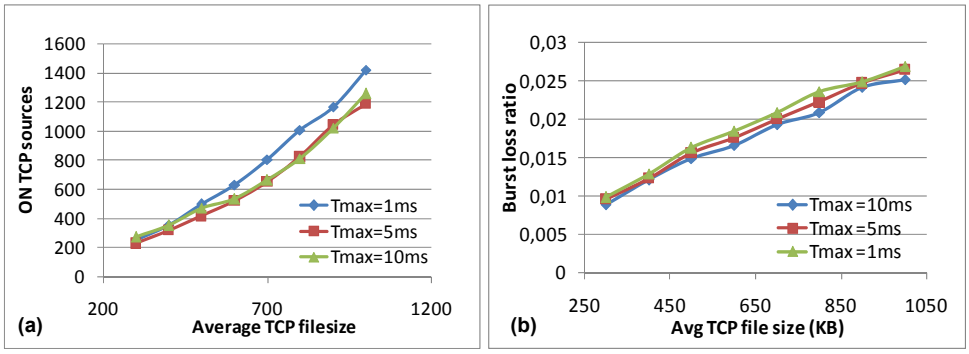

Fig. 1. (a) Number of simultaneous active TCP sources at each edge node versus average file size and (b) the corresponding burst loss ratio.

From Fig. 1(a), it can be seen that the number of active sources in the case of 5 and 10msec timers is similar, while in the case of 1msec, it is 200 more for all flow sizes above 600KB. It must be noted here that the number of active sources measured is a dynamic parameter of the simulating experiment that can vary when additional burst losses occur and thus this parameter corresponds to a real, instant figure of the network under study. An important issue noticed is that more than 30% of the active sources were in a slow start phase, while the simulating scenario was entering the steady state much earlier in the case of 5 and 10msec, than in the case of 1msec timer. In what follows, we have selected a mean file size of 700KB, which correspond to 600 and 800 active sources respectively for 5,10*msec* and 1*msec* timers in the network and a burst loss ratio of 2%. Using these as reference, we have measured two basic statistics; the distribution of segments and the distribution of TCP flows over the assembled bursts for various assembly times.

Fig. 2 displays (a) the probability distribution function of the number of segments and (b) the cumulative distribution function of the number of different TCP flows per transmitted burst. The results shown correspond to all the bursts transmitted in the network, for all source-destination pairs. Fig. 2 provides a useful insight of how many TCP sources will experience a

segment loss from a single burst loss as well as how many segments will be lost. The exact number of lost segments per source is the combined probability of Fig. 1(b), Fig. 2(a) and Fig. 2(b). From Fig. 2(b) and in the case of 1*msec* assembly time, it can be seen, that 80% of the transmitted bursts employ segments from only 2 different sources, while for 5 and 10*msec* timers, this increases to 4 and 6 segments respectively. It is clear that large aggregation times contribute in the transfer of a higher number of sources and segments per burst that in turns may lead to smaller completion times, fewer bursts generated but with the trade off that more sources will be potentially affected upon a burst loss. However, at which point the network will balance is still unknown but is reflected in the number of active sources, shown in Fig. 1, and segment and flow distribution.

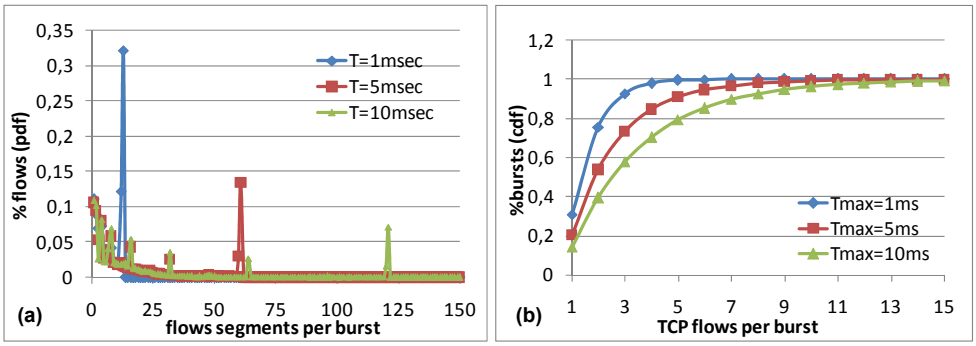

Fig. 2. (a) Segment and (b) flow distribution per burst transmitted for 1, 5 and 10*msec*, assembly timers

The probability distribution of segments per burst (Fig. 2a) exhibits sharp increases at the end of the corresponding curves. These sharp increases are due to the finite access rate in combination with the specific burstification time. For example, for 1*msec* assembly time, no more than 13 segments from the same flow could be added in the same burst, due to the limited 100Mbps access rate. For 5 and 10*msec*, the corresponding maximum numbers of segments were 61 and 121. It is worth noting, that the number of bursts carrying that maximum number of segments is decreasing with the increase of the assembly time. For example, in the case of 1*msec* timer, the number of bursts carrying 13segments from the same source is 33%, and which decreases to 13% and 7% for 5 and 10*msec* timers. This can be attributed to the different window sizes of the active sources. In particular, it was found out that sources with a large window size were capable of adding more segments in the transmitted bursts as a result of their larger pool of unacknowledged segments. Thus, when using short assembly times, a high number of sources had a window size larger than 13 segments, and therefore the high percentage of bursts (~33%). On the other hand, in large assembly times only a few flows had large enough windows to transmit more segments. For example, when employing a 10*msec* timer, only the flows with a window size larger than 121 segments could send that maximum number of segments. To this end, we may conclude that segment and flow distribution depends on both the burstification delay and the flow window size and large timers result in the transfer of a higher number of segments per burst but only for flows that have unacknowledged segments to transmit. Otherwise, they unnecessarily delay burst transmission, resulting in poor throughputs, especially for flows that are in a slow start phase.

We have also measured the actual number of bursts (transmitted and lost) per flow, which are necessary to complete a transfer. Fig. 3(a) and (b) shows the corresponding distribution functions for a single (randomly selected) source-destination pair, independently of the flow size. The selection of single source-destination pair allows for the fair evaluation of bursts transmitted over the same path, with the same round-trip-time delay and blocking ratio. From

Fig. 3, it is clear that small assembly times result in a significant increase in the number of bursts needed to complete a transfer, while the lost bursts per flow vary much less. In particular, in the case of 1*msec*, 80% of the flows need on average up to 80 bursts to complete their transfer, while only 40 and 34 are needed in the case of 5 and 10*msec* respectively. The difference in the lost bursts is smaller and particularly 5 bursts per flow are lost, when using an 1*msec* timer as opposed to 4 bursts in the case of 5 and 10*msec*. Of course, the number of bursts needed to a complete a transfer depends heavily on the amount of data to be transferred. Therefore, in Fig. 4, we have further analyzed the transmitted and lost bursts per flow with respect to the corresponding flow size.

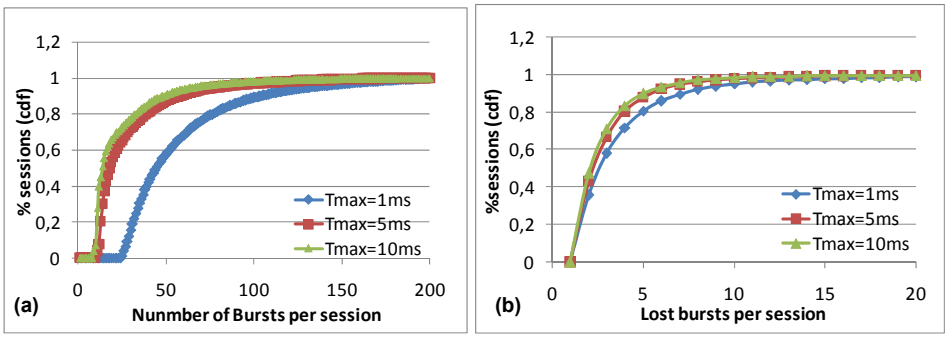

Fig. 3. (a) Distribution of the number of bursts needed to complete a TCP transfer and (b) the number of lost burst per flow for 1, 5 and 10*msec* assembly timers. Results shown correspond to a single source-destination pair.

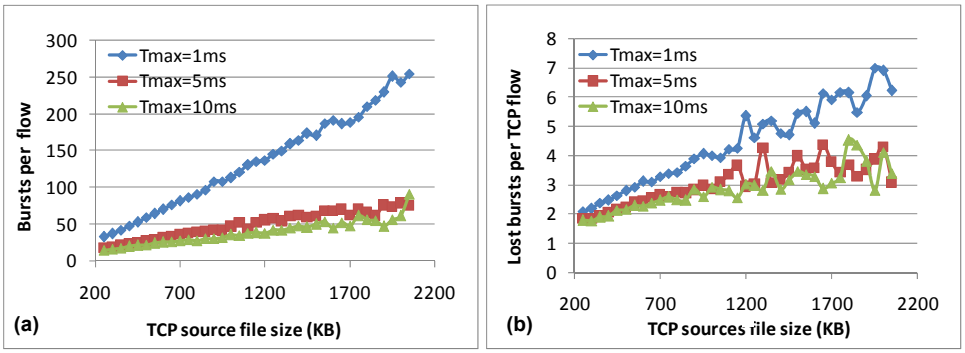

Fig. 4. (a) Number of bursts needed to complete a flow transfer and (b) number of lost burst per flow for 1, 5 and 10*msec* assembly timers, versus the TCP file size. Results shown correspond to a single source-destination pair.

From Fig. 4(a), it can be seen that the number of transmitted bursts increases almost linearly with flow size, while 1*msec* curve diverge rapidly. In particular, for the maximum flow size of 2MB, 3.5 times more bursts are needed in the case of 1*msec*, while the number of lost bursts (see Fig. 4(b)) differs less. This was however expected since the corresponding 80% of the transmitted bursts were carrying less than 13 segments from only 2 flows at most (see Fig. 2). It is therefore clear that small assembly times significantly increase network overhead, constraining the transmission of multiple-segments from multiple flows per burst. This results to longer file transfer times that in turns lead to more flows remaining active. However, small aggregation times may result in higher throughputs for individual flows since burstification delay is by default smaller. On the other hand, large assembly times service more flows at a time, carrying more segments from each individual flow. However, the burstification delay that they impose result by default in lower throughputs. To conclude, it is not yet clear if small

aggregation times result in a higher throughput or eventually in a worse performance, or if large assembly times are capable of canceling the large burstification delays imposed. In the next section, we investigate how throughput varies for an individual flow and how congestion window evolves for lossless flows.

4. TCP Throughput and Congestion Window analysis

In order to qualitatively investigate the performance of assembly timers, we have measured the average throughput achieved along with its variance, maximum and minimum value. In [11], it has been shown that the TCP throughput decreases with the increase of burstification delay, but this occurs for very large timers. However, measuring average throughputs does not reveal the real performance of individual flows. To this end, in Fig. 5a, we have measured the average and the variance of throughput, while in Fig. 5b its maximum and minimum value versus assembly time for a specific source–destination pair.

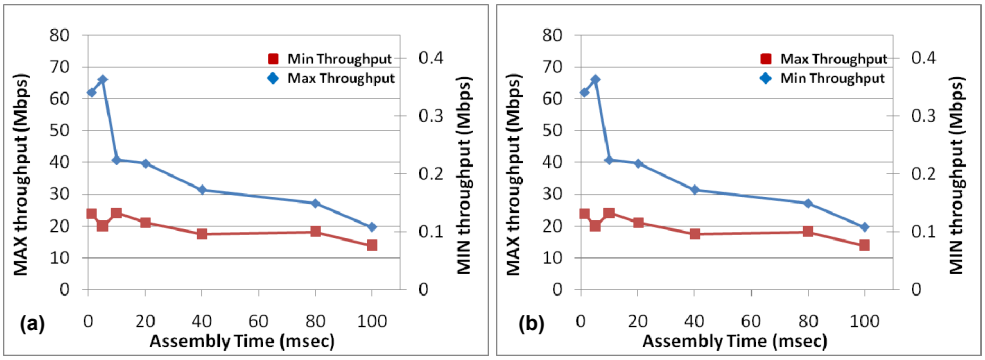

Fig. 5. (a) Average throughput and variance of all flows of a single source-destination pair and (b) maximum and minimum throughputs measured versus burst aggregation time.

From Fig. 5(a), it can be seen that both average and variance of throughput decreases rapidly with the increase of assembly time. Throughput variance decreases primarily because large assembly times smooth out and dilute individual flow performances. This can be derived from Fig. 5(b) as well, where the maximum throughput decreases but the minimum changes slightly. To this end, large assembly times yield a lower throughput, with no gain from multiple segment transmission, but however can provide a higher notion of fairness among the individual flows in the sense that all flows reach the same throughput value and thus fairly share network bandwidth.

To further elaborate on this, we have selected three lossless flows with similar characteristics, like file size, routing path and starting time and measured the evolution of their window. Fig. 6(a) displays in detail the rising edge of their window, while Fig. 6(b) displays the sequence number (modulo 132) of the segments transmitted for an assembly time of 1*msec* (left column*),* 5*msec* (middle column) and 10*msec* (right column). From Fig. 6(a), it can be seen that the flow window rises faster for 1*msec* rather than for 5 or 10*msec*. In particular, in the case of 1*msec*, a 600segment window is reached within 0,194sec, while for 5*msec* and 10*msec* in 0,350 and 0,405sec. The highest speed up gain is noticed for a window increase from 1 to 100segments. Above this value, the rate at which all windows expand converge. To this end, segment transmission takes place faster, and in particular 6 bursts (carrying segments from these sources) are transmitted within 0,15*sec* in the case of 1*msec* assembly time (see counts of segment sequence number in Fig. 6b), while 5 and 4 in the case of 5 and 10*msec*.

A significant finding is that the maximum, instant throughput of these flows was found to bet the same (~65Mbps) independently of the assembly timer. However that maximum was

obtained much faster in the case of 1*msec* timer, and thus the higher average throughput. In particular, average throughput was measured to be 23Mbps against only 17 and 14Mpbs for 5 and 10*msec* assembly timers. A second important finding is that this maximum throughput was constant only for the case of 10*msec* timer. In the rest, and especially for 1*msec* assembly timer, it was varying by more than 50%. This instability was due to the fact that short assembly times cannot sustain a constant segment-to-burst ratio and thus cannot yield a constant throughput. In contrast, large assembly times can absorb such traffic instabilities due to the longer burstification delay time and thus the lower variance shown in Fig. 5a.

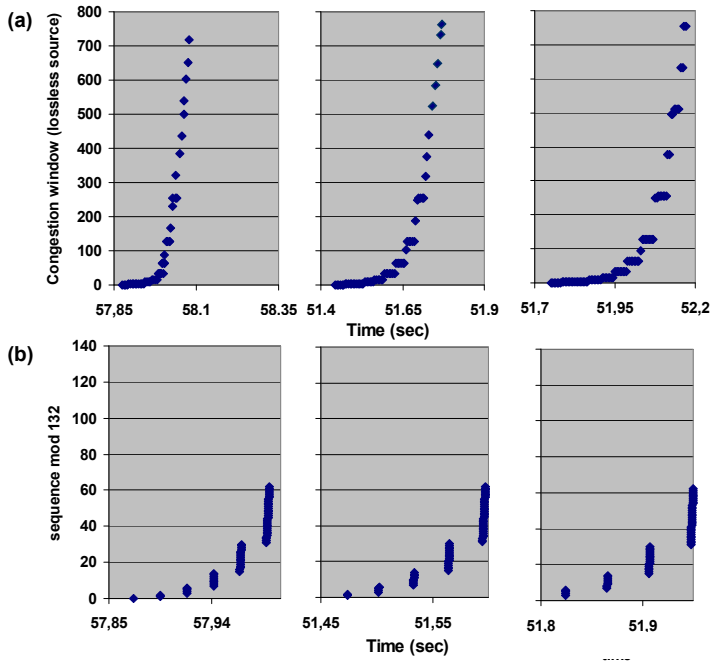

Fig. 6. (a) Congestion window and (b) Sequence number of the transmitted segments under identical timescales and flow sizes for 1*msec* (left column), 5*msec* (middle column) and 10*msec* (right column) assembly time.

5. Window-based Burst Assembly Scheme

Based on the above analysis, it is clear that fixed timer-based burstifiers are not appropriate, since they do not provide maximum performance but only optimal performance for individual flows with similar characteristics (i.e. file size, size of window, blocking, etc.). Further, measuring averaging throughputs is not indicative of TCP performance, and conceals true flow behavior in the network. In order to truly enhance TCP performance, the instant window size is a metric to be considered for determining the optimum assembly time. Further, static use of fixed timer is not enough and a dynamic process is preferred that assigns flows to different assembly queues with different assembly timers. In order to evaluate such a scheme, we have implemented a new assembly scheme with three different queues per source–destination pair. Each queue had a different assembly timer, and incoming packets were assigned to these, based on their instant window size as follows:

$$T_{assembly\ time} = \begin{cases} 1m\sec & if\ CW < B \\ 5m\sec & if\ B < CW < C \\ 10m\sec & if\ CW > C \end{cases} \qquad \textbf{Eq. (1)}$$

, where B, C is the flow window size in segments. A flow is characterized as *slow* when its instant window size is less than *B* segments, as *medium*, when it is less than *C* segments and as a *fast* flow when it is even higher. Thus, all *slow* TCP flows with a congestion window of less that *B* segments are aggregated together under 1*msec* delay. When their congestion windows reach the limit of *B* segments, the flows are upgraded to *medium* rate flows and their segments are assembled under a 5*msec* delay. Similarly, when their congestion windows reach the *C* segment limit, these flows are upgraded again to *fast flows* and their assembly time is increased to 10*msec*. In this way, each flow is treated separately and thus upon a burst loss only the flows experiencing losses will be downgraded to *medium* flows or even to slow flows if they eventually time out. The implementation of this assembly scheme requires the communication of the window size to the burstifier, which is not standard in the current TCP implementations. In any case, we have implemented such a scheme using ns-2, in order to investigate its throughput gains and measure the yielding average and variance of the throughput for various *B* and *C* combinations.

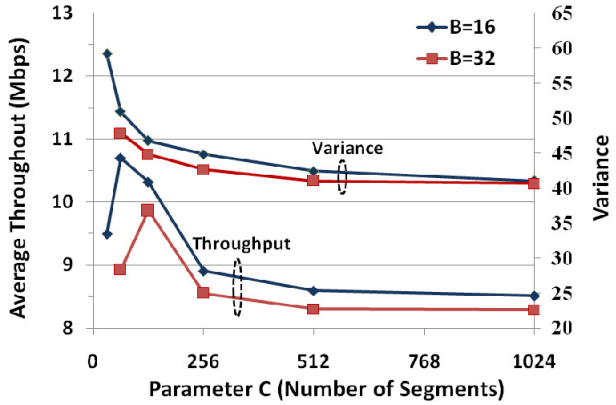

Fig. 7. Average and variance of throughput of all flows aggregated at a single node and heading for the same destination for various combinations of B and C values.

Fig. 7 displays the average throughput versus parameter *C,* for two different *B* values; namely for *B=16* and *32*segments. It can be seen that the highest gains in throughout are noted when *C* parameter is fourfold *B* value. For example the "*B=16*" curve exhibits a maximum of 10.7Mbps for "*C=64*", while the "*B=32*" curve (corresponding throughput 9.86Mbps) for "*C=128"*. For even higher values of *C* (>256), the performance of the system resembles the case of having two only queues. This is because TCP windows do not always reach such a large size, as a result of burst losses. With respect to the variance of the throughput, it is continuously decreasing with the increase of *C* parameter as expected, since overall performance starts being dominated by the next queue that employs a higher assembly timer. However it worth noting, that for the particular case of "*B=16*" and "*C=64*" that exhibited the highest throughput, variation has been significantly decreased to only 51, as opposed to 70, in the case of a single queue burstifier with 5*msec* time. Thus, we may argue that this *B,C* combination merges best the performance advantages of long and short assembly times and can support an average throughput of 10.3Mbps with a variance as low as 51. It must be noted here, that overall performance of the assembly scheme is dominated by that queue/timer

having the largest segment range, that is B and $|C - B|$, for the first and the second queue. For example, for higher B values (>32), the system performance approximates that of a single queue with 1*msec* burstifier, while for B values smaller than 16, the case of a 5*msec* timer.

The abovementioned B,C, were selected having in mind that a speed up in TCP transmission in needed when window size is relative small. In general, B,C parameters depend on the network (i.e. blocking ratio) and the traffic statistics (i.e. flow arrival rate, file size etc) and can be different for different network setups. In any case, there will always be a set of parameters that optimizes performance, since the TCP congestion algorithm (window mechanism) operates independently.

6. Conclusions

In this paper, analysis of TCP traffic over OBS networks was presented for multi-queue burstifiers based on different timer values. It was found that short assembly times provide a higher average throughput but result in a significant performance variation of the individual aggregated flows. On the other hand, large assembly times are capable of smoothing out performance differences but eventually lead to poor performance. In addition, it was shown that the yielding instant throughput can be the same, independently of the burstification delay imposed, leading to the conclusion that measuring average throughput is not indicative for the individual flows. To this end, we have proposed a new burst assembly scheme, employing more than one queue per destination with different timers, where flows are dynamically assigned based on their instant window size. It was found that such a 3-queue burstifier is able to increase TCP performance in a fair way for all the aggregated flows.

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.