TCP traffic analysis for timer-based burstifiers in OBS networks

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

[1]Computer Engineering and Informatics Dept., and Research Academic Computer Technology Institute, University of Patras, Rio, Greece (email: kvlachos@ceid.upatras.gr)

[2]Telefónica I+D, Emilio Vargas 6, Madrid, Spain

[3]Dep. of Electronics, Computer Science and Systems, University of Bologna, Italy

Abstract. The purpose of this paper is to assess the impact of timer-based burst assembly algorithms for TCP traffic. We present an analysis for short, medium and long assembly times and investigate segment and flow distribution over the assembled bursts. Further, we also analyze their impact on the congestion window evolution and on the effective throughput achieved. It has been found out that short assembly times are ideally suitable for sources with small congestion windows, allowing for a speed up, while large assembly times yield a lower throughput variation among the individual assembled flows. For long assembly times, the transfer of more segments from the same source is trading off the increase of the burstification delay but no throughput gain is obtained. However, large assembly times smooth out individual flow performance and provide a significant lower variation of throughput. To this end, in this paper, we propose a new adaptive burst assembly algorithm that dynamically assigns flows to different burstifiers based on their instant window size.

Keywords: Optical burst switching, transport control protocol, burst assembly

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 t*hreshold-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 arrival at the queue. Hence, the ingress router generates periodically 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.

I. Tomkos et al. (Eds.): ONDM 2007, LNCS 4534, pp 176-185, 2007.

The timer-based method limits the delay of packets to a maximum value T_{MAX} but may generate undesirable burst length, while the burst-length based method generates bursts of equal size, but 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].

The performance of TCP over OBS networks has been studied in previous works [7]-[9] where it has been observed that the burst assembly process at the edge nodes has a significant impact on the end-to-end performance of TCP, mainly because it introduces an unpredictable delay, [10], that challenges the window mechanism used by TCP protocol for congestion control. TCP performance is also challenged by the burst lost ratio that results in multiple segment losses for multiple sources. A useful insight on TCP traffic statistics is given in [11],[12]. In particular, it was found that 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. Long assembly times, are more efficient especially for *fast* TCP flows [11], since they allow the transmission of multiple segments from the same flow per burst. However, this throughput gain may be canceled by the large burstification delay.

In this paper, we present a thorough analysis of TCP traffic over OBS networks. We first analyze how segments and flows are distributed over the assembled bursts for various assembly times and further analyze their effect in the number of transmitted and lost bursts per flow. It is shown that the short assembly times result in a significant increase in the number of bursts needed for a transfer completion, while for larger assembly times, this is relative constant. Furthermore, it is investigated how congestion window increases and what its effect is in average throughput. We argue in this paper that the characterization of a flow as slow, medium or fast depends on its instant congestion window size and we show that a mix burst assembly timer, where burstification delay varies with the size of the congestion window yields a higher throughput together with a smaller variance. For the performance evaluation and traffic analysis, we have developed a comprehensive and detailed TCP over OBS simulator based on ns-2 tool, capable of simulating ~hundreds of active TCP sources per edge node.

The rest of the paper is organized as follows. Section 2 provides an overview of TCP variants and most suitable assembly schemes, whiles Section 3 presents a flow/segment analysis of TCP traffic over OBS. Section 4 discusses the effect of burstification delay on the congestion window expansion and the yielding throughput, and finally Section 5 presents the performance of a new assembly scheme based on the flow congestion window size.

2. Overview of TCP variants and aggregation schemes for OBS networks

There are a number of TCP versions such as Tahoe, Vegas, Reno, New Reno and SACK, combined with a number of different burst assembly strategies. The most interesting are the three last ones. 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 an exponential increase, 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 window is now increasing only by one segment after all segments have been acknowledged.

In TCP Reno, there are two kinds of losses identified; the *Time Out* (TO) and *Triple Duplicate* (TD) loss. In the *Triple Duplicate* (TD) case, the sender receives three duplicates ACKs, that acknowledge a new segment, but not the one with the highest sequence number. In that case TCP Reno enters the Fast Retransmit phase, and starts 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 resets its window back to one segment size. TCP New 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, where the non-contiguous set of data that have been received are stored. To this end, the sender is aware of the lost packets 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's performance (e.g., throughput) depends heavily 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. Segment and Flow distribution

In this section, a segment and flow analysis of TCP traffic is presented when OBS is used as the underlying transport technology. Work presented hereinafter concerns TCP-SACK variant and timer-based assembly schemes that although it is the most promising combination, very few works exist on providing an in-depth analysis of how segments, flows and their parameters vary with burstification delay. We have developed a dedicated TCP-over-OBS simulator using ns-2 platform capable of

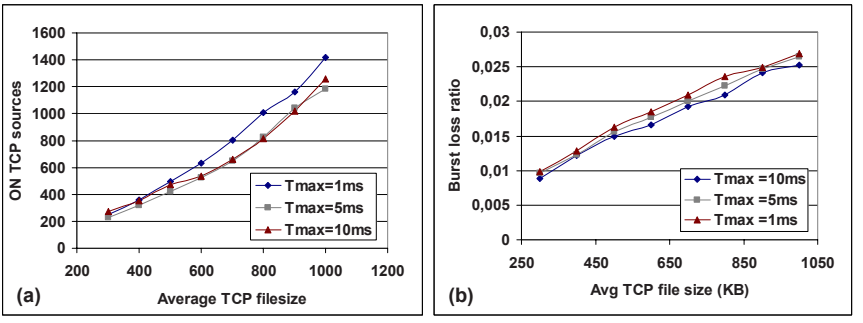

Fig. 1. (a) Number of simultaneous active TCP sources at each edge node versus average file size. (b) Corresponding burst loss ratio.

simulating ~hundreds of active sources per edge node. Such a scenario is close to reality, but requires significant amount of memory and CPU resources. We have modified the raw ns-2 code to efficiently manipulate TCP flows and available CPU resources. The experiments were carried out on the NSF network topology, with 8 edge and 6 core nodes, where each link was employing a single wavelength at 10Gbps. Access rate was set to 100Mbps, equal for all sources. TCP arrivals were modeled with an exponential process with a mean of λ=50 flows/sec, while flow file size was modeled with a Pareto distribution of p load and a minimum ON size of 40KByte. Using this set of metrics, it was possible to vary the TCP arrival rate and/or the mean file size, and obtain measurements for a different number of active sources. Fig. 1(a) displays the number of active sources at each edge node versus the average file size for three different assembly timers namely 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, which took place after 200sec of simulation time.

From Fig. 1(a), it can be seen that the number of active sources in the case of 5 and 10*msec* timer is close, while in the case of 1*msec*, 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 burst losses occur and thus we argue that this corresponds to a real, instant picture of the network under study. An important issue noticed is that the simulating scenario was entering the steady state much earlier in the case of 5 and 10*msec* timers, than in the case of 1*msec*.

In what follows we have selected a mean file size of 700KB, which corresponds to a burst loss ratio of 2% and 600 or 800 active source respectively for 5, 10 and 1*msec* timers. Using these as reference, we have measured two basic statistics; the distribution of segments and the distribution of flows over the assembled bursts. Fig. 2 displays the cumulative density function (CDF) of (a) the number of segments and (b) 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 conjugate probability of Fig. 1(b), Fig. 2(a) and Fig. 2(b).

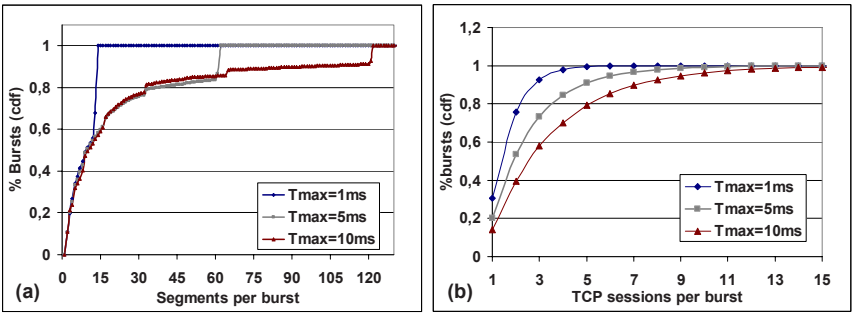

Fig. 2. (a) Segment and (b) flow distribution over the transmitted bursts for 1, 5 and 10*msec* burstifiers.

The sharp increases in Fig. 2(a) are due to the finite access rate that does not allow more segments to be sent within the burstification time delay. To this end, in the case of 1*msec* assembly time up to 13 segments can be loaded onto a single burst, independent of the flow window size. If TCP sources have more segments to send, these are transmitted with a next burst. To this end, it can be easily derived that the congestion window of a high number of sources is incompatible larger than the 1*msec* assembly time. For larger assembly times, this sharp increase is shifted to higher numbers of segments.

Furthermore from Fig. 2(b) and in the case of 1*msec* timer, it can be seen, that the 80% of the transmitted bursts employ segments from only 2 different sources, while for 5 and 10*msec*, this increases to 4 and 6 respectively. To this end, it is clear that large aggregation times may result 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. The latter is the side effect when upon a burst loss a higher percentage of sources (see Fig. 2(b) will face a multiple segment loss. Thus, more sources will halve their window and then either time-out and enter a slow-start phase or try to recover from the loss and linearly increase their window. However, at which point the network will balance is still unknown but it is reflected in the segment and flow distribution, shown in Fig. 2 and the number of active sources, shown in Fig. 1.

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

From Fig. 3, it is clear that small assembly times result in a significant increase in the number of bursts needed to a complete a transfer, while the lost bursts per flow vary much less. In particular, in the case of 1*msec* timer, 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 corresponding lost bursts are 5 and 4. Of course the number of bursts needed to complete a transfer depends heavily on the

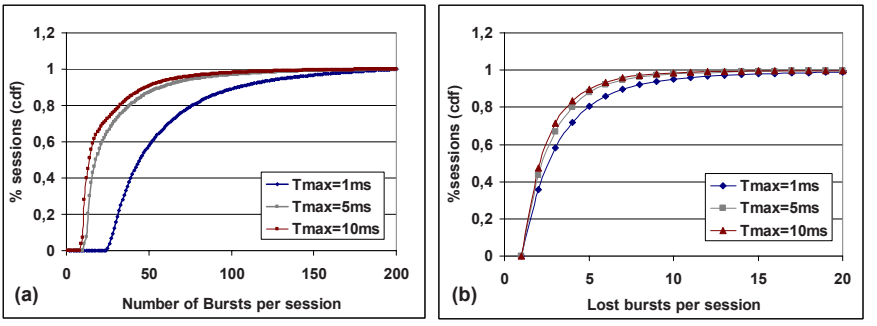

Fig. 3. Distribution of (a) number of bursts needed to complete a TCP session and (b) number of lost bursts per session for 1, 5 and 10*msec* burstifiers for a single source-destination pair.

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.

From Fig. 4(a), it can be seen that the number of transmitted bursts increase almost linearly with flow size, while 1*msec* curve diverges rapidly. In particular, for the maximum flow size of 2MB, 3.5 times more bursts are needed in the case of 1*msec* timer. This was however expected, since the corresponding 80% of that bursts were carrying less than 13 segments from only 2 flows at most. However, this does not result to smaller throughputs or higher completion times, as shown in the next section. The behavior of the number of lost bursts per flow size is similar (see Fig. 4(b)) but the difference between the curves diverges at a smaller rate.

Based on the above analysis, we may conclude that small assembly times increase the network overhead but potentially constrain individual flow performance due to limited number of segments per flow per burst transmitted. The latter may result to longer file transfer times that in turns leads to more flows remaining active. However, on average burstification delay is by default smaller and thus it is expected that short assembly time to result in higher throughputs as discussed in the next section. Further, they may also result to less *Time-Out* events since it is unlikely a single flow to employ its complete window onto a single (lost) burst. On the other hand, large assembly times service more flows at a time, carrying more segments from each individual flow. This smoothes out any traffic instabilities in the sense that individual flow throughput is absorbed and diluted but however impose such a burstification

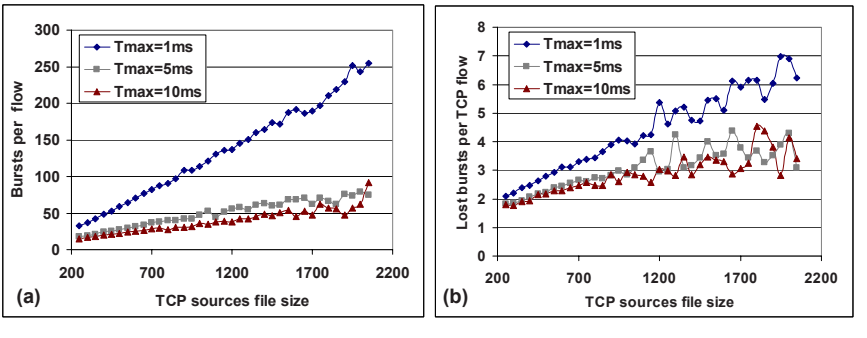

Fig. 4. Number of bursts needed to complete a TCP session and (b) number of lost burst per session versus file size for 1, 5 and 10*msec* burstifiers for a single source-destination pair**.**

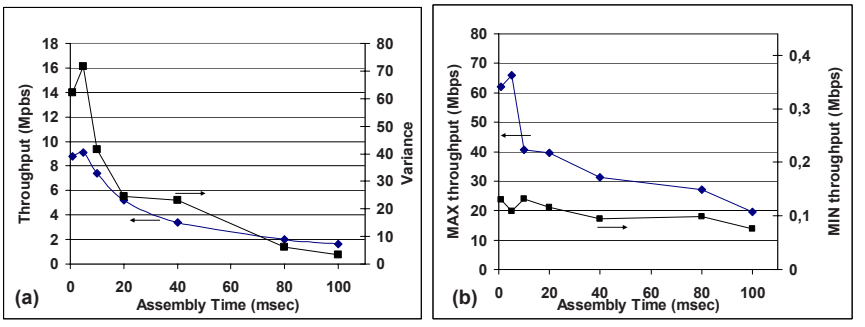

Fig. 5. (a) Average and variance of throughput of all flows of a single source-destination pair versus burst aggregation time. (b) Corresponding maximum and minimum values measured.

4. Efficient Throughput and Congestion Window expansion

In order to qualitatively investigate the performance of the different assembly timers, we have measured the average throughput achieved along with its variance, its maximum and minimum value. In general, TCP performance depends on the TCP variant used in combination with the number of segments lost and further depends on the flow access rate. A TCP source is characterized as *slow*, *medium* or *fast*, depending on its access rate. From [11], it can be inferred that a flow with a slow access rate (*slow flow*) must be accompanied with a small assembly time while flows with a large access bandwidth (*fast flows*) with relatively larger assembly timers. However, in most cases, access rate for all flows at the edge node is the same and service is differentiated by other means.

Fig. 5(a) displays the average throughput and variance for all flows of a single source-destination for different burstifiers, while Fig. 5(b) shows its maximum and minimum value. From Fig. 5(a), it can be seen that there is no throughput gain for large assembly times albeit *delay first loss* (DFL) gain is maximized [11]. This is also clear by Fig. 5(b), where maximum throughputs higher than >60Mbps were measured only for 1 and 5*msec* timers. As a result, flow transfer time increases with assembly time and specifically, it was found to increase from 1.25*sec* for 1*msec* timer to *1.6*sec and 6,7*sec* for 10*msec* and 100*msec* respectively. However, performance (throughput) variance drops also fast with assembly time (see Fig. 5a) and this is because individual flow performance is diluted via the long burstification delay process. To this end, we may argue that large assembly times can provide a higher notion of fairness among the aggregated flows and smooth out any performance instabilities.

In order to further analyze this, we have measured the evolution of the congestion window of three lossless flows transmitted over the same source-destination route and with a similar file size. Fig. 6(a) displays in detail the rising edge of their congestion window, while Fig. 6(b) displays the sequence number (modulo 132) of the transmitted segments. The results shown correspond to 1*msec* (left column*),* 5*msec* (middle column) and 10*msec* (right column) burstifiers.

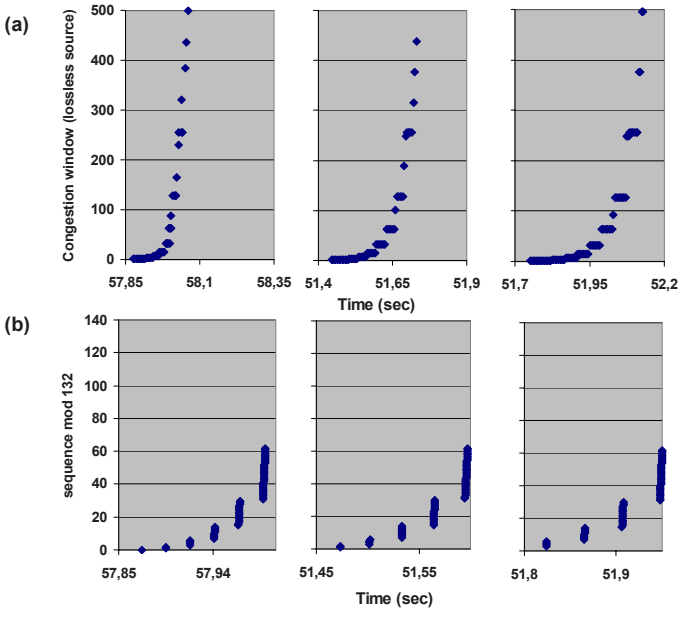

Fig. 6. (a) Congestion window evolution and (b) Sequence number of the transmitted segments under identical timescales of three different, lossless flows with similar files sizes for 1*msec* (left column), 5*msec* (middle column) and 10*msec* (right column) assembly times.

From Fig. 6(a), it can be seen that the congestion window rises faster for 1*msec* timer rather than for 5 or 10*msec*. In this particular case, a 500segment window is reached within 0,171*sec*, while for 5 and 10*msec* timers in 0,290 and 0,38sec. The highest speed up gain is noticed for a window increase from 1 to 100segments. This is the result of a higher number of burst transmissions over the same time period. In particular, in the case of 1*msec*, 6 bursts are transmitted within 0,15*sec* (see counts of segment sequence number in Fig. 6(b), left column), while only 5 and 4 in the rest.

A significant finding is that the maximum instant throughput measured for all three burstifiers was 92Mbps, and which was achieved when window size was 128segments wide for all cases. However this was obtained much faster in the case of 1*msec*, and thus the average yielding throughput was much higher. In particular, average throughput was found to be 23Mbps whereas only 17 and 14Mbps in the cases of 5 and 10*msec* timers. However, it must be clearly noted that the yielding, instant throughput was equal for all three different timers and we may argue that the resulting average throughput actually depends on the flow size and the burst per flow losses.

A second important finding is that after having reached the maximum throughput and before flow completion, instant throughput was constant to its maximum value (92Mbps) only in the case of the 10*msec* burstifier. In the rest and especially for 1*msec* burstifier, this was instantly varying from 50Mbps to 90Mbps. This instability was primarily due to fact that segment distribution over the assembled bursts was not constant due to the short burstification delay. On the contrary, this was varying

significantly and thus a steady segment per burst transmission rate could not be obtained albeit the higher number of burst created.

It is therefore clear that measuring averaging throughputs is not indicative for TCP performance. Further, a fixed assembly time does not provide maximum performance but only optimal performance for individual flows with similar characteristics (i.e. flow size, loss ratio, etc.). Large assembly times offer an advantage of carrying more segments but only when flows can send more segments, otherwise they constrain throughput performance. In order to truly enhance TCP performance, the instant congestion window is a metric to be considered for determining the optimum assembly time. For example, short assembly times should be applied to sources with a relatively small congestion windows (<100segments), while larger timers to sources with larger windows. In the next section, we analyze such a scheme and investigate its performance gains.

5. Congestion Window-based Burst Assembly Scheme

In this section, we propose a new adaptive burst assembly scheme that assigns different burstification delays to flows based on their congestion window size. In particular, we define burst assembly time *(T_{BAT})* as follows:

$$T_{BAT} = \begin{cases} 1\,msec & if\ 1 \leq congestion\ \ window < B\ segments \\ 5\,msec & if\ B \leq congestion\ \ window < C\ segments \\ 10\,msec & if\ C \leq congestion\ \ window \end{cases}$$

In our scheme, we propose the characterization of a flow as *slow* or *medium,* when its instant congestion window is less than *B* or *C* segments wide and as a *fast* flow when it is even higher. Thus, *slow* TCP flows with a congestion window of less than *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* timer. Similarly, when their congestion windows reach the *C* segments 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 that will suffer from a segment loss will be downgraded to *medium* flows or even to *slow* flows if they time out.

The implementation of the proposed assembly scheme requires three different queues per destination, one for each type of flow, as well as the communication of the window size to the burstifier. Albeit the latter require a modification of the TCP mechanism, we have implemented the scheme in ns-2 platform to particularly measure the yielding throughput and variance for various *B*, *C* values. TABLE 1 summarizes our findings, where the performance of the simple cases of 1, 5 and 10*msec* burstifiers is also denoted for comparison. The cases with no *B* value means that the intermediate transition to 5*msec* assembly is omitted. From TABLE 1, it can be seen that best performance is obtained for *B, C* values of 32 and 100 segments. In that case, average throughput is increased to 11Mbps, while variance is dropped significantly to 50. This combination merges the optimum operation points of all three cases providing fast transmission times for *slow* sources that are in an early slow-start phase, *medium* rates for flows with up to 100 segments window size and slow transmission rates, (large assembly delays) for the rest.

TABLE 1. Average throughput and variance for various combinations of the proposed congestion-window based, burst assembly scheme

B	C	Average Throughput (Mbps)	Variance
100	200	8,4	60
32	100	11,2	50
-	32	9,1	55
-	200	7,7	59
1msec		8,8	62
5*msec*		9,0	71
10*msec*		7,4	41

In the rest cases, performance either approximates the performance of 1*msec* timer, when B is too high ($B = 100$, $C = 200$) or 10*msec*, when C is too high ($B = C = 200$). However, best performance is obtained upon the optimum combination of both values. It must be noted here however, that these depend on the actual flow and segment distribution and thus can be different for different arrival rates and burst loss ratios.

6. Conclusions

In this paper, an analysis of TCP traffic over OBS networks was presented for various timer-based burstifiers. It was found that short assembly times provide a higher average throughput but result to 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 throughputs. However, it was found that the yielding instant throughputs were the same in all cases and large assembly times are beneficial only when there are pending segments to be sent. Otherwise they delay burst transmission unnecessarily. In order to truly enhance individual TCP performance, we proposed a mixed timer-based algorithm that assigns flows to different timer-based burstifiers based on their instant window size. It was found that short assembly times provides a significant performance speed up when windows sizes are less than 100segments, and thus a mixed algorithm can combine the advantages of both a fast throughput increase along with a low performance variation among the individual flows.

Acknowledgments. This work has been supported by EC through the NoE E-Photon/ONe+ project via the Joint Project on *Optical Burst Switching* (JP-B).

References

[1] C. Qiao and M. Yoo, J. High Speed Networks, vol. 8, no. 1, pp. 69–84, 1999.
[2] F. Callegati and L. Tamil, IEEE Commun. Lett, Vol 4, pp. 98-100, Mar. 2000.
[3] M. Düser and P. Bayvel, J. Lightwave Technol., vol. 20, pp. 574–585, Apr. 2002.
[4] V. Vokkarane, K. Haridoss, and J.P. Jue, in Proceeding of Opticomm, pages 125-136, 2002.
[5] X. Yu, Y. Chen, and C. Qiao, in Proc. Opticomm, 2002, pp. 149–159.
[6] X. Cao, J. Li, Y. Chen, and C. Qiao, in Proc. IEEE GLOBECOM, vol. 3, pp. 2808–2812, Nov. 2002,.
[7] M. Casoni, E. Luppi and M. Merani, in Proceedings of Workshop on Optical Burst Switching.
[8] S. Malik and U. Killat, in Proc. of ONDM, 2005.
[9] A. Detti and M. Listanti, in Proc. IEEE, INFOCOM 2002.
[10] M. Izal and J. Aracil, IEEE Globecom 2002, Taipei, Taiwan, November 2002.
[11] Xiang Yu et al. J. of Lightwave Technology, vol. 22, no. 12, pp. 2722 – 2738, Dec. 2004.
[12] Óscar González de Dios, Ignacio de Miguel, Víctor López, in Proceedings of ONDM 2005.