An Adaptive Burst Assembly scheme for OBS-GRID networks

Nikos Korkakakis and Kyriakos Vlachos

Research Academic Computer Technology Institute and Computer Engineering and Informatics Department, University of Patras, GR26500, Rio, Patras (kvlachos@ceid.upatras.gr)

Abstract— This paper addresses the problem of burst assembly in OBS-GRID networks and particularly the TCP throughput maximization problem for large file sizes. For that purpose, a novel adaptive size-based approach is proposed, following some probabilistic methods and considering the effect of the burstification process in the overall transport system. The scheme has been evaluated in a high capacity GRID network. It was found that the proposed scheme results in shorter file transfer times, a significant higher TCP throughout, thus yielding a positive impact at the grid mechanics. The performance of the scheme is compared to that of a timer-based algorithm.

***Index Terms*—Transport Control Protocol, GRID, Optical Burst Switching.**

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 edge routers that are responsible for the creation of the bursts and a set of core routers that transparently forwards them to their destination.

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] 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, [3], 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 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 [4], 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 [5].

The performance of TCP over OBS networks has been studied in previous works [6]-[8] 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, [9], that challenges the window mechanism used by TCP protocol for congestion control. A useful insight on TCP traffic statistics is given in [10]. 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. However, short assembly time prohibit the fast expansion of the congestion window primarily because sources are allows to transmit only a few segments per round. Long assembly times, are more efficient especially for *fast* TCP flows [10], since they allow the transmission of multiple segments per burst. However, this throughput gain may be canceled by the large burstification delay.

In this paper, an adaptive size-based burst assembly algorithm is presented for the special case of an OBS-GRID network. Grid networks employ the transfer of large files between computing and storage elements and require fast completion times with low or minimal losses. To this end, it is important to adapt burstification process to minimize delays upon losses that inevitably lead to low TCP throughputs.

The rest of paper is organized as follows. Section II presents an outline of the proposed algorithm, while Section III its basic mathematic analysis. Section IV describes its event drive execution and finally Section V provide evaluation results over a hypothetical grid network.

II. ADAPTIVE TCP ALGORITHM OUTLINE

In the proposed algorithm (hereinafter called Adaptive TCP hereinafter), we have used some basic properties of the TCP dynamics to effectively adapt assembly time upon burst losses for large file transfers. The working model of the algorithm suggests that there is a way to efficiently count the active TCP window sizes by creating a bookkeeping mechanism in the burstifier's end. The idea of the algorithm is simple and very effective (as proved by the simulations); the burstification process is looking after the active TCP flows and their congestion window size in order to adapt the size of the burst that must be sent. The algorithm at the beginning of its execution assumes a small, but not negligible, burst size considering that most flows are in a slow-start phase. Such a

978-1-4244-1876-3/08/$25.00 ©2008 IEEE

value delays TCP transmission but is gradually eliminated due to the adaptiveness of the algorithm. At any stage or after every successfully sent burst, the algorithm is checking its log and calculates the new congestion windows of the TCP flows. In this way, it estimates the size of the data that each flow will send in the next period of time. Therefore the algorithm may calculate the size of the next burst to be transmitted as the sum of all the estimations from all flows.

Similarly, upon a burst loss, the algorithm approximates the sizes of the congestion windows of the affected TCP flows, in order to "re-evaluate" the network situation and adopt the burst size accordingly.

III. THE CALCULUS MECHANICS

The proposed scheme is based on a modification of the mixed B_{MAX} /T_{MAX} algorithm. The algorithm approximates packet arrivals in small time-offsets (T_{offset}) with a Poisson distribution of non- constant mean rate, $\lambda(x)$ and compares the actual collected data size with the approximated one. Equation 1 provides weighted probability sum that returns the expected burst size.

$$B_{size}(t) = \sum_{flow} C_{win}(flow) \cdot e^{-r(t)} \cdot r(t) \qquad \textbf{(eq.1)}$$

In this equation, $C_{win}(flow)$ and $r(t)$ stand for the approximations of the congestion window and the Poisson distribution. $r(t)$ function is obligatory for calculating the expected packet arrivals at the given time and is given by Eq.2.

$$r(t) = T_{offset} \cdot \int_{t-offset}^{t} \lambda(x)dx \qquad \textbf{(eq.2)}$$

$r(t)$ integrals can be quickly and precisely calculated by numerical methods such as integration by parts or Risch algorithm .

When a packet arrives, the algorithm recalculates the packet arrival rate, $\lambda(x)$ using the least square algorithm per TCP flow. Thus we add in every packet arrival a small fixed length calculation O(1). This is because each approximation that use the least square method needs three operation; namely one square calculation, one addition and one division.

In order to prove, that Equation 1 is a weighted Poisson probability sum, we use in our analysis general statistical properties and the Poisson distributions properties. The probability for exactly one TCP segment arrival is $P(1,\lambda') = \lambda' \cdot e^{-\lambda'}$, where λ' denotes the approximated arrival rate, $\lambda(x)$, calculated as described earlier. The probability for exactly two segments to arrive from two different sources is $P_1(1,\lambda_1') = \lambda_1' \cdot e^{-\lambda_2'}$, if $P_1(1,\lambda_1') \le P_2(1,\lambda_2')$ or $P_2(1,\lambda_2') = \lambda_2' \cdot e^{-\lambda_2'}$ elsewhere. The probability for exactly n segments to arrive from n different sources is $\min_n(\lambda_n' \cdot e^{-\lambda_n'})$.

From the definition of Poisson distribution we have;

$$\lim_{n\to\infty} \Pr(X = s) = \lim_{n\to\infty} \binom{n}{s} p^s (1-p)^{n-s} =$$

$$= \lim_{n\to\infty} \frac{n!}{(n-s)!s!} \left(\frac{\lambda}{n}\right)^s \left(1-\frac{\lambda}{n}\right)^{n-s} =$$

$$= \lim_{n\to\infty} \left(\frac{n}{n}\right)\left(\frac{n-1}{n}\right)\left(\frac{n-2}{n}\right)\cdots\left(\frac{n-k+1}{n}\right)\left(\frac{\lambda^s}{s!}\right)\left(1-\frac{\lambda}{n}\right)^n \left(1-\frac{\lambda}{n}\right)^{-s}$$

where $p=\lambda/n$. As n approaches ∞, then $\frac{n!}{(n-s)!s!}$ approaches one, while $\left(1-\frac{\lambda}{n}\right)^n \cdot \left(1-\frac{\lambda}{n}\right)^{-k}$ approaches $e^{-\lambda}$. In the expanded form of the above equation the only object that is not affected by n approaching ∞ is $\left(\frac{\lambda^s}{s!}\right)$.

Based on the above analysis, it is easily understood that Eq. 1 is the sum of two terms; one showing the Poisson probability and its weight. This weight is used to provide identification and classification to the incoming aggregated traffic using the general statistical property. Thus, some important information concerning the characteristics of the tcp flow such as congestion window (others may be applicable but not in this schema) can be obtained.

To this end, after n TCP segments arrival, the expected burst size is given by Equation 1, which returns the expected size of the aggregated data. The simulations suggest that this probability is over 98,4% before sending out the burst. $\boldsymbol{C_{win}}$ represents bytes, and help towards assessing the expected size of a burst given the TCP congestion window values.

IV. EVENT DRIVEN EXECUTION

Based upon the above mathematical schema, the algorithm's execution idea is quite simple. While packets are arriving at the edge router, the burstifier each time calculates the expected incoming data size and then decides whether to send or not the data burst. The algorithm uses an event driven mechanism that can provide an easy hardware implementation and a significant performance boost. The algorithm's events driven input mechanism are based on the following events:

- TCP segment arrival (the algorithm may look after some other input e.g. IP datagram arrival etc).
- Congestion Window of flow n changed (in hardware simulation it is probably easier to create a pooling mechanism to check for such alterations)
- Burst report which consists of three different events: burst discarded from the core routers, burst discarded from the destination router and burst successfully sent

A simple running example of the algorithm is the following:

1) Event::PacketReceived.Trigger: Decision making method based on **eq.1**:
   a. Burst is transmitted (if current Burst Size is equal or larger than the expected).
   b. Burst is not transmitted (if current Burst Size is smaller than the expected).
2) After a Burst is transmitted, a receipt is expected to arrive on whether the burst reached its destination or not. However

the receipt by itself is not important. Important is to have enough information (from Event::PacketReceived.Trigger method) on the packet arrival rate and the size of the aggregated data at each of the edge routers burstifier mechanism:

a. Event::BurstDroppedCore.Trigger (may point out at some extent that the path is overloaded or some indication of network fault. It may be important to reroute or find another destination)
b. Event::BurstDroppedEdge.Trigger (may point out at some extent that the node accessed is overloaded and it may be important to change our destination if that is possible. In grid network is important to be able to reroute and reassign resources.
c. Event::BurstSuccPassedCore.Trigger (Not in use in this algorithm. May provide information of the location of the fault)
d. Event::BurstSuccSentEdge.Trigger (Message that everything is successful and the burst is successfully sent).

V. THE SIMULATION SCENARIO

The adaptive TCP algorithm has been evaluated using ns-2 over the NSF network topology that has been modified to support an OBS-GRID network. It is therefore assumed that it consists of six computing and two storage elements located at the edges of the network, while an additional edge node, processes and parses user requests. Figure 1 displays the experimental setup simulated. It is assumed that all storage and computing element exchange data sets based on some user requests. We have limited our study on evaluating the effectiveness of these transfers, when TCP transportation mechanism is used, for large datasets (e.g. 30-150 MB files), assuming that all traffic is ftp like. Each optical link employs two wavelength channels, each at 10Gbps. File transfer arrival is modeled either with a Gaussian distribution with $\sigma^2 = 2, \mu = 3$, or with a constant arrival rate of 6 requests per second, while the access rate from the source machine to the OBS edge router is 100Mbps.

We have compared the adaptive TCP algorithm with the timer-based algorithm (Tmax) taking into consideration the yielding flow duration, number of active connections and bandwidth utilization for the same input traffic (data set exchanges).

In order to provide a more accurate comparison scheme Tmax time threshold was set equal to the mean time of the adaptive TCP burstification time (~7,2ms). Other constant values usually selected in the literature (2ms, 5ms, 10ms) provided inferior results for Tmax and are not provided here.

The main effect of adaptive TCP burstification scheme is shown in Figure 2, which displays the file transfer time using the cumulative density function (CDF). It can be seen that the end-to-end completion time is significant smaller when using the adaptive scheme. In particular, the 80% of the flows complete their transfer within 1.4sec when using the adaptive while in 1.7sec when using Tmax algorithm. This was primarily due to the dynamic variation of the assembly time, which in turns optimizes the burstification delay based on the flows' congestion window. In other words, assembly time was

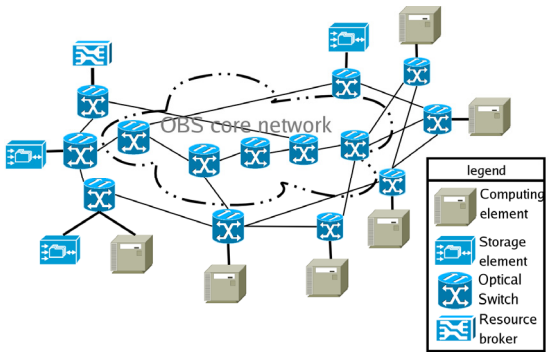

Figure 1: Hypothetical Grid over NSF network topology

decreasing when flows' windows were decreasing upon segment losses. In [11], it was shown that short assembly times offer a high advantage to flows with smaller congestion windows, while large assembly times was proved to unnecessarily delaying segment transmission. To this end, upon a burst loss, many flows will suffer from segment losses and thus will halve their windows. In such case, burstification delay must be adapted to the new traffic situation, since less data are expected to arrive in the next assembly cycles.

This was also clear from the number of active flows during the simulation as well as the bandwidth utilization profile in the network. Figure 3 and Figure 4 display the corresponding results. From Figure 3, it can be seen that the average number of simultaneous active flows, (after a sharp increase in the beginning of the simulation experiment), is 75 and 45 respectively for the Tmax and adaptive TCP algorithm. It is therefore clear that file transfer using the Tmax scheme is longer resulting in a higher overhead for the network. Figure 4 shows that capacity is 80% utilized in the case of the adaptive TCP scheme and only 40% in the case of the Tmax scheme. To this end, it is clear how efficient is the adaptive TCP algorithm and how much capacity is wasted, when using a fixed timer-based to transfer large data sets. The results shown in Figure 4 are normalized to the maximum throughput achieved. In the simulation experiment, a capacity usage of 100% corresponded to 6Gbps bandwidth.

The number of retransmissions over the tcp plane is also a significant indicator regarding the effectiveness of our scheme. Figure 5 displays the number of TCP flows that retransmit at least one (or more) segments per logging epoch (every epoch has about 0.0025 sec duration). As shown in Figure 5, the number of flows that retransmit segments is 9 in the case of Tmax scheme for all the simulation cycle, while the proposed scheme exhibit a high number during the initialization but only a few in the sequence. In particular, more than 30 flows retransmit segments in the first 3sec (epoch 349), and only 1 or 2 during the rest simulation cycles.

To this end, and based on the above results, we may argue that the proposed scheme is more suitable for grid application involving large file transfers. It yields a higher usage of available resources with shorter file transfer times. The proposed adaptive TCP algorithm reacts to burst losses rapidly by adapting assembly time to smaller values, before gradually increasing them again, upon the successful segment transmission. In contrast fixed, constant burst assembly times

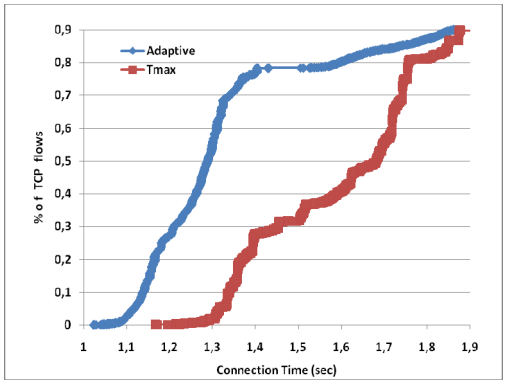

Figure 2: Cumulative density function of TCP transport times

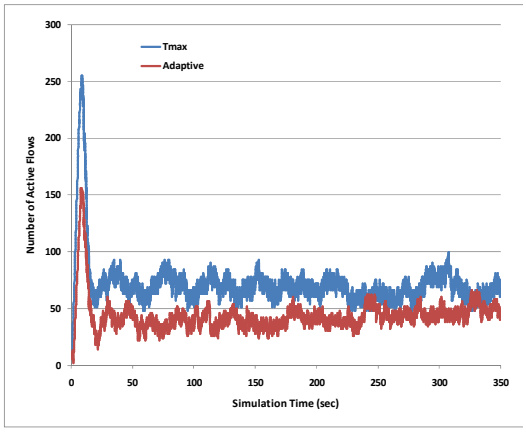

Figure 3: Active TCP connections over simulation time.

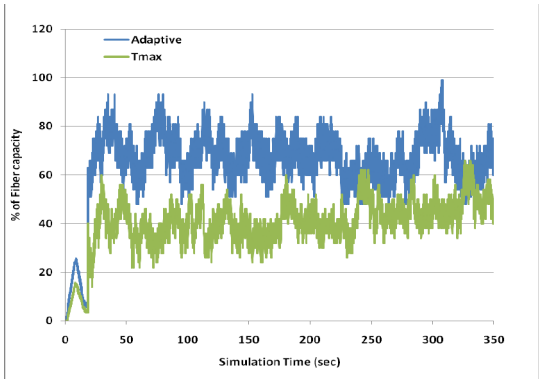

Figure 4: Bandwidth utilization Results

result to a poor bandwidth usage, longer transfer times that possibly cannot be tolerated by higher layer grid applications.

VI. Conclusions

In this paper, we have presented a new adaptive assembly algorithm suitable for OBS grid networks that involve large file transfers. The transmission of large files over an OBS networks result in a poor performance primarily due to the many burst losses that will occur and the tcp behavior due to these losses. The proposed algorithm approximates packet arrivals in small time-offsets with a Poisson distribution of non- constant mean rate, and thus approximated the next burst

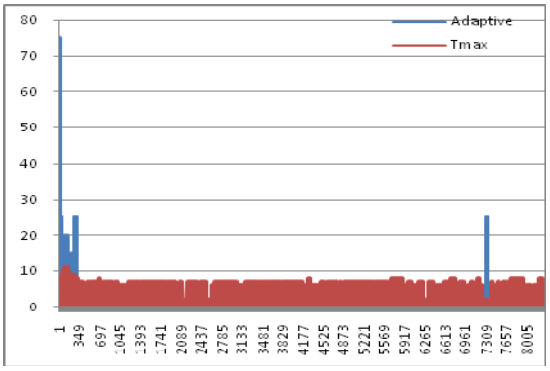

Figure 5: Number of flows that retransmit one or more TCP segments over logging epoch (*every epoch has about 0.0025 sec duration*)

size. To this end, the scheme is capable of adapting to varying packet arrival rates that are caused by the increase/decrease of the flows' windows. Simulation results have shown that the proposed scheme results in faster file transfers with a higher consumption of bandwidth resources.

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 and IST-PHOSPHORUS project.

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] F. Callegati and L. Tamil, "On optical burst switching and self-similar traffic", IEEE Commun. Lett, Vol 4, pp. 98-100, Mar. 2000.

[3] V. Vokkarane, K. Haridoss, and J.P. Jue, "Threshold-based burst assembly policies for QoS support in optical burst-switched networks". In Proceeding of Opticomm, pages 125-136, 2002.

[4] X. Yu, Y. Chen, and C. Qiao, "Study of traffic statistics of assembled burst traffic in optical burst switched networks," in Proc. Opticomm, 2002, pp. 149–159.

[5] Cao, X., Y. Chen, J. Li, and Qiao, C. (2002). IEEE Globecom 2002, 3, 2808 – 2812.

[6] X. Cao, J. Li, Y. Chen, and C. Qiao, "Assembling TCP/IP packets in optical burst switched networks" in Proc. IEEE GLOBECOM, vol. 3, Nov. 2002, pp. 2808–2812.

[7] S. Malik and U. Killat, "Impact of burst aggregation time on performance in optical burst switching networks", in Proc. Optical Network Design and Modelling (ONDM-2005), 2005.

[8] A. Detti and M. Listanti, "Impact of segments aggregation on TCP Reno flows in optical burst switching networks", in Proc. IEEE, INFOCOM 2002.

[9] M. Izal and J. Aracil, "On the Influence of Self-similarity on Optical Burst Switching Traffic", Proceedings, IEEE Globecom 2002, Taipei, Taiwan, November 2002.

[10] Xiang Yu et al. "Traffic statistics and performance evaluation in optical burst switched networks", Journal of Lightwave Technology, vol. 22, no. 12, pp. 2722 – 2738, Dec. 2004.

[11] K. Ramantas, K. Vlachos, Ó. González de Dios and C. Raffaelli, "TCP traffic analysis for timer-based burstifiers in OBS networks" in proceeding of ONDM 2007.