New Assembly Techniques for Optical Burst Switched Networks Based on Traffic Prediction

Angeliki Sideri and Emmanouel A. Varvarigos

Computer Engineering and Informatics Dept, University of Patras, Greece [⋆]

Abstract. We propose new burst assembly techniques that aim at reducing the average delay experienced by the packets during the burstification process in optical burst switched (OBS) networks, for a given average size of the bursts produced. These techniques use a linear prediction filter to estimate the number of packet arrivals at the ingress node in the following interval, and launch a new burst into the network when a certain criterion, which is different for each proposed scheme, is met. Reducing the packet burstification delay, for a given average burst size, is essential for real-time applications; correspondingly, increasing the average burst size for a given packet burstification delay is important for reducing the number of bursts injected into the network and the associated overhead imposed on the core nodes. We evaluate the performance of the proposed schemes and show that two of them outperform the previously proposed timer based, length based and average delay-based burst aggregation schemes in terms of the average packet burstification delay for a given average burst size.

1 Introduction

Optical Burst Switching (OBS) [1] aims at combining the strengths of packet and circuit switching, and is considered a promising technology for implementing the next generation optical Internet, required to cope with the rapid growth of Internet traffic and the increased deployment of new services (e.g., VoIP telephony, video on demand, grid computing, digital repositories). In OBS, bursts consisting of an integer number of variable size packets are switched through the optical network. The OBS network consists of a set of optical backbone nodes, responsible for the forwarding of the bursts, and a set of edge nodes, known as ingress and egress nodes, responsible for burst assembly and disassembly. When a burst is formed at an ingress node, a control packet is sent out through the backbone network to reserve the required resources, followed after a short offset time interval by the burst.

An important issue for the design of OBS networks is the burst assembly strategy used at the edge nodes. The burst assembly process starts with the arrival of the first packet and continues until a predefined threshold is met. The

[⋆] We would like to achknowledge the support of the Phosphorus and the e- Photon/ONE+ IST projects.

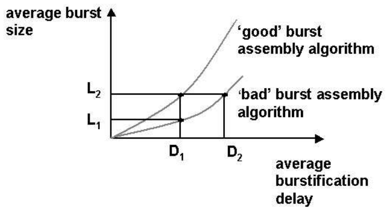


Fig. 1. Performance of a "good" and a "bad" burst assembly algorithm

burst aggregation policy influences the traffic characteristics in the network, but also the end-to-end performance. Reducing the packet burstification delay (so as to reduce total delay), and increasing the burst size (so as to reduce the number of bursts and the associated processing overhead at the core nodes) are two main performance objectives of the burst aggregation strategy. These objectives, however, contradict each other, since increasing the burst size also increases burstification delay (see Fig.1). A burst assembly algorithm, therefore, should be judged based on how well it performs with respect to one of these two performance metrics of interest, for a given value of the other performance metric. Given a burst assembly algorithm, choosing the desired balance between the burstification delay and the burst size then depends on the QoS requirements of the users, and the processing and buffering capabilities of the backbone nodes.

A number of burst assembly schemes have appeared in the literature, including the time-based algorithm (abbreviated T_{MAX} algorithm) and the length-based algorithm (abbreviated L_{MAX} algorithm) [3],[4], [5]. In the time-based algorithm, a time counter is started any time a packet arrives at an empty ingress queue, and the burst is completed when the timer reaches the threshold T_{MAX}; the timer is then reset to 0 and it remains so until the next packet arrival at the queue. Even though the time-based algorithm succeeds in limiting the average burstification delay, by limiting the maximum delay a packet can remain in the queue, it may generate very small bursts. In the length-based method, the burst is released into the network when its size reaches a threshold L_{MAX} (in fixed size packets, bytes, etc). This method produces bursts of a desired length, but may result in large burstification pdelays, especially when the traffic generation rate is low.

Hybrid schemes [2] have also been proposed, where a burst is completed when either the time limit T_{MAX} or the length limit L_{MAX} is reached, which ever happens first. An average delay-based algorithm (abbreviated T_{AVE} algorithm) was also introduced in [6], which aims at controlling the average burstification delay by letting out the bursts, the moment the average delay of the packets that comprise it reaches a threshold T_{AVE}. This method guarantees a desired average burstification delay, and also tends to minimize packet delay jitter. None of the burst assembly algorithms proposed so far, however, achieve an optimal tradeoff between the average burstification delay and the average burst size in the way described above.

In the present work we propose and evaluate several novel burst aggregation schemes that use traffic prediction to maximize the average length of the bursts produced for a given average burstification delay. Prediction of traffic characteristics has previously been examined in [7], [8] and [9]. In [7], specifically, it is demonstrated that despite the long-range dependence of Internet traffic, which would lead us to expect that we must look deep into the "past" for a precise estimation, a prediction filter of small order is sufficient for good performance. We use traffic prediction in order to estimate the number of packets that will arrive in the assembly queue in the near future and determine whether it would be beneficial for the burst assembly process to wait for these packets, or the burst should be sent immediately.

The performance measure we use to compare the algorithms proposed is the Average Packet Burstification Delay to Burst Size ratio (DBR) defined as

$$DBR = \frac{Average\ Packet\ Burstification\ Delay}{Average\ Burst\ Size}. \tag{1}$$

We find that two of the proposed schemes improve burstification efficiency, by reducing the average burstification delay by up to 33% for a given size of the bursts produced, compared to previously proposed schemes.

The remainder of the paper is organized as follows. In Section 2 we discuss the algorithms proposed. In Section 3, we examine the performance of the proposed schemes and compare it to that of the timer-based, length-based and average-delay-based algorithms. We also investigate the impact the various parameters involved have on performance.

2 The proposed burst assembly schemes

We assume that the time axis is divided into time frames of equal duration τ (see Fig. 2). During a frame, an edge OBS node assembles the IP packets arriving with the same destination address and the same QoS requirements (such packets are said to belong to the same Forwarding Equivalence Class or FEC) into a burst. At the end of each frame, a decision is taken about whether the burst should be sent out immediately and the assembly of a new burst should start, or the edge node should wait for another frame in order to include more packets in the current burst. This decision is taken by using a linear prediction filter to estimate the expected number $\hat{N}(n+1)$ of packet arrivals in the following frame $n+1$, and checking if a specific criterion (different for each algorithm proposed) is fulfilled. This criterion tries to quantify if the increase in the burst length expected by waiting for an extra frame is significant enough to warrant the extra delay that will be incurred.

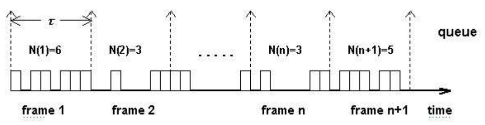


Fig. 2. Time frame structure. At the end of each frame n the algorithm decides if it should send out the burst immediately, or it should wait for another frame.

2.1 Fixed Additional Packets Threshold Algorithm (N_{MIN} algorithm)

In this proposed scheme, we define a lower bound N_{MIN} on the number of future arrivals above which we decide to wait for an extra frame before assembling the burst. At the end of frame n, the estimate $\hat{N}(n+1)$ produced by the linear predictor is compared to the threshold N_{MIN}, and if it is smaller than that, the burst leaves the queue immediately, otherwise it waits for another frame to be completed, at the end of which the same procedure is repeated. Therefore, the burst is sent out at the end of the n-th frame if and only if

$$\hat{N}(n+1) < N_{MIN}.$$

2.2 Proportional Additional Packets Threshold Algorithm (αL algorithm)

In this scheme, instead of using a fixed threshold value N_{MIN}, a fraction of the current length of the burst is used as the threshold. If α is the multiplicative parameter, the burst is completed at the end of the n-th time frame, if and only if

$$\hat{N}(n+1) < \alpha \cdot L(n),$$

where $L(n)$ is the burst length at the end of the n-th frame, and $\hat{N}(n+1)$ is the predictor's estimate for the number of packet arrivals expected during the following frame $(n+1)$.

2.3 Average Delay Threshold Algorithm (T_A algorithm)

This method tries to improve on the average-delay-based algorithm proposed in [6], which computes a running average of the packet burstification delay and lets out the burst, the moment the average delay of the packets that comprise it reaches a threshold T_{AVE}. The algorithm in [6] has two drawbacks: a) computing the running average introduces considerable processing overhead, and b) bursts may not be sent out at the optimal time, since the running average is non-monotonic in time and could drop in the future due to new packet arrivals. The T_A algorithm addresses these drawbacks using traffic prediction. At the end of each frame, it estimates the average burstification delay we expect to have at the end of the following frame, and launches the burst if this estimate exceeds some threshold value T_A.

The Average Packet Delay $D(n)$ of the packets in the burst assembly queue at the end of frame n is defined as

$$D(n) = \frac{\sum_{i=1}^{L(n)} T_i(n)}{L(n)}, \tag{2}$$

where $L(n)$ is the burst size at the end of frame n, $T_i(n) = n \cdot \tau - t_i$ is the delay of the i-th packet from the moment it enters the queue until the end of n-th frame, τ is the duration of the frame, and t_i is the arrival time of i-th packet.

Alternatively and more easily, we can compute $D(n)$ using the recursion

$$D(n) = \frac{L(n-1)\cdot D(n-1)+L(n-1)\cdot\tau+\sum\limits_{i=1}^{N(n)} T_i(n)}{L(n-1)+N(n)} \ , \tag{3}$$

where $N(n)$ is the number of packet arrivals during frame n. If a burst was sent out at the end of the $(n-1)$-th frame then we take $L(n-1) = 0$ in Eq. 3.

To obtain an estimate $\hat{D}(n+1)$ of the Average Packet Delay at the end of frame $n+1$ we assume that the $\hat{N}(n+1)$ packets estimated by the predictor to arrive by the end of frame $n+1$ will have an average delay of $\tau/2$. Using Eq. 3, the estimated Average Packet Delay $\hat{D}(n+1)$ at the end of frame $n+1$ is

$$\hat{D}(n+1) = \frac{L(n)\cdot D(n)+\tau\cdot L(n)+\hat{N}(n+1)\cdot\frac{\tau}{2}}{L(n)+\hat{N}(n+1)} \ . \tag{4}$$

A burst is completed at the end of the n-th frame if and only if

$$\hat{D}(n+1) > T_A,$$

where T_A is the predefined threshold value.

2.4 Average Delay to Burst Size Ratio Improvement Prediction Algorithm (L_{MIN} algorithm)

The proposed Average Delay to Burst Size Ratio Improvement algorithm (abbreviated L_{MIN} algorithm) uses traffic prediction to compute an estimate $D\hat{B}R(n+1)$ of DBR at the end of frame $n+1$, and decides that the burst is completed, if this estimate is worse than the current $DBR(n)$. The average burstification delay to burst size ratio $DBR(n)$ at the end of frame n is defined as

$$DBR(n) = \frac{D(n)}{L(n)} = \frac{\sum\limits_{i=1}^{L(n)} T_i(n)}{L^2(n)} \ . \tag{5}$$

Alternatively, and more easily, $DBR(n)$ can be found recursively as

$$DBR(n) = \frac{\frac{L(n-1)\cdot D(n-1)+L(n-1)\cdot \tau + \sum_{i=1}^{N(n)} T_i(n)}{}}{(L(n-1)+N(n))^2} \quad . \tag{6}$$

The Estimated Average Packet Burstification Delay to Burst Size ratio $D\hat{B}R(n+1)$ at the end of frame $(n+1)$ can be found as

$$D\hat{B}R(n+1) = \frac{L(n)\cdot D(n)+L(n)\cdot \tau + \hat{N}(n+1)\cdot \frac{\tau}{2}}{(L(n)+\hat{N}(n+1))^2} \quad . \tag{7}$$

The algorithm decides that a burst is completed and should be sent out at the end of frame n if and only if

$$D\hat{B}R(n+1) < DBR(n) \ AND \ L(n) > L_{MIN} .$$

During the first frames following a burst assembly completion, there is a great likelihood that the right term of the preceding inequality will be quite large, making it difficult to fulfill. The threshold L_{MIN} is used as a lower bound on the length of the bursts, and also makes the algorithm parametric (as with all the previous algorithms examined) so that the desired tradeoff between the average burst size and the average packet burstification delay can be obtained.

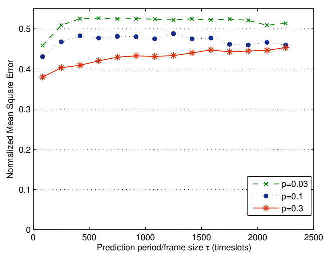
3 Performance Analysis and Simulation Results

We used the Matlab environment to simulate the burst aggregation process at an ingress queue, in order to evaluate the performance of the proposed schemes and compare it to that of previously proposed schemes. We also quantified the impact the parameters N_{MIN}, α, T_A and L_{MIN} have on performance, and the effect of the frame size τ and the order h of the linear predictor.

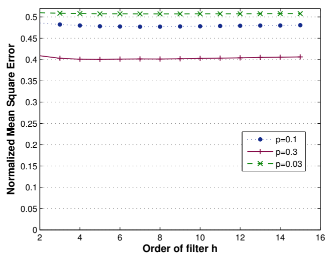
It is worth noting that each of the proposed schemes corresponds to a different Burst Size versus Packet Burstification Delay curve (the reader is referred to Fig. 1), while the choice of the parameters N_{MIN}, α, T_A and L_{MIN} (or of the parameters T_{AVE}, T_{MAX}, L_{MAX} in previously proposed schemes) determines the exact points on each curve the burst assembly process is operating at.

3.1 Traffic Generating Source

In our experiments, the arrivals at the ingress queue were obtained from an Exponential-Pareto traffic generating source of rate r *bits/sec*. The traffic source generates superpackets (they can also be viewed as busy periods) with exponentially distributed interarrival times of mean $1/\lambda$ seconds. The size of each superpacket follows the Pareto distribution with shape parameter β. If a superpacket has size greater than l bytes, which is taken to be the size of the packets used in the network, it is split and sent as a sequence of packets of size l. The



(a)



(b)

Fig. 3. LMS performance for various values of: (a) the prediction period τ, (b) the length of the predictor h.

time units used for displaying our results are measured in packet slots, where 1 slot $= l/r$ (the transmission time of a packet).

The values of the parameters we used in our experiments are the following: β=1.2, r=1Gbps, l=1500 bytes, $1/\mu$=60KB, and $1/\lambda$=1.6 msec or 4.8msec, yielding corresponding load utilization factors p=0.1 and p=0.3. The parameter β determines the Hurst parameter $H = (3 - \beta)/2$, which takes values in the interval $[0.5, 1)$ and defines the burstiness of the traffic. The closer the value of H is to 1, the more bursty the traffic can be characterized.

3.2 Linear Predictor LMS

The *Least Mean Square Error (LMS)* predictor, described in [10] and also used for traffic prediction in [8] and [9], has been chosen as the linear predictor in our burst assembly schemes. It is simple, fast and effective without great computational cost. The estimate $\hat{N}(n+1)$ of the number of packet arrivals during the $(n+1)$-th frame is generated according to the relationship

$$\hat{N}(n+1) = \sum_{i=1}^{h} w_i \cdot N(n-i+1) \enspace ,$$

where $w_i, i \in 1, \ldots, N$, are the filter coefficients and h is the length of the filter. The error $e(n)$ of the n-th frame is calculated as $e(n) = N(n) - \hat{N}(n)$, and the coefficients of the filter are updated at each iteration according to

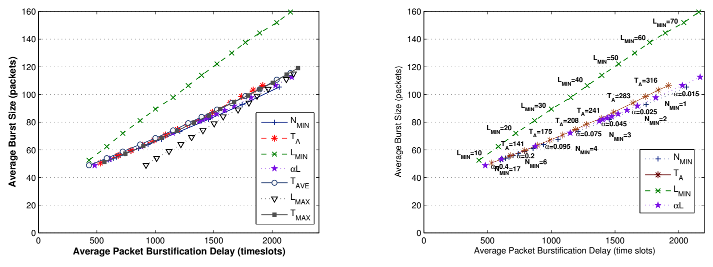
$$w_i(n+1) = w_i(n) + \delta \cdot e(n) \cdot N(n-i+1) \enspace ,$$

where δ is the step size.

3.3 Predictor Performance

The accuracy of the estimations produced by the LMS predictor can be assessed by the *Normalized Mean Square Error (NMSE)* parameter, defined as:

$$NMSE = \frac{MSE}{P_{INPUT}} = \frac{\sum_{n=1} e^2(n)}{\sum_{n=1} N^2(n)} \enspace .$$



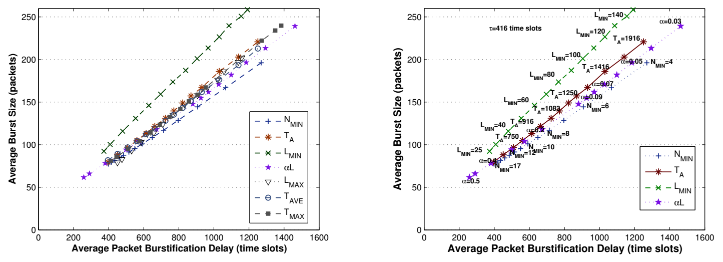
(a) (b)

Fig. 4. Performance of the proposed algorithms for traffic load $p = 0.03$: (a)comparison of the proposed schemes with previously proposed algorithms, (b) parameters applied on the proposed algorithms

The first set of results presented examines the dependence of the performance of the LMS predictor on the frame duration τ, the order of the prediction filter h, and the traffic load p. Fig. 3(a) shows the way *NMSE* varies with the frame duration τ for bursty traffic (H=0.9). Note that short frame durations result in smaller values of *NMSE*, which can be justified by the fact that for bursty traffic, the characteristics of its behaviour remain static for only shorts periods of time. For light traffic, the predictor's performance is worse than it is for heavy traffic. This can also be seen in Fig.3(b), which illustrates the impact the order of the filter has on *NMSE*. This figure also indicates that there is very little improvement when the order of the filter is increased beyond a certain value. This is in agreement with the results in [7], where it was argued that the performance of linear predictors for internet traffic is dominated by short-term correlations, and we don't have to "look deep" into the history of traffic arrivals to obtain a valid estimation. A small order of filter is, therefore, preferable, since it also results in smaller computation overhead. As the frame size τ increases, the *NMSE* remains steady after a certain value ($\tau > 0.005sec$) when the traffic is light ($p = 0.1$, $p = 0.03$), while it worsens slightly for heavier traffic ($p = 0.3$).

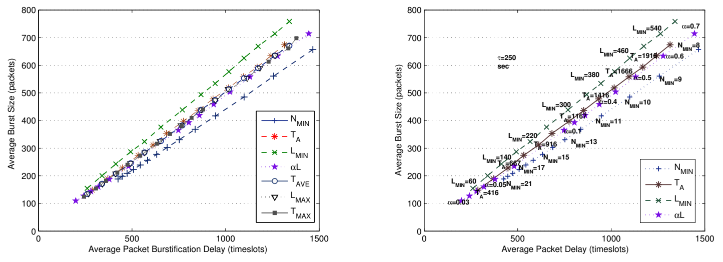
3.4 Comparison Between the Burst Assembly Schemes

In this section we compare the performance of the proposed schemes to that of the previously proposed $T_{AVE}, T_{MAX}, L_{MAX}$ burst assembly schemes. The results reported here were obtained for bursty traffic ($H = 0.9$) and varying load utilization p. The length h of the LMS predictor was set to 4, while the frame size τ varied depending on the traffic load. The parameters of all the schemes have been chosen so as to produce average burstification delays that lie in the same range so that the resulting burst sizes can be compared. Time delays are measured in slots. Figures 4(a), 5(a) and 6(a) illustrate the average burst size produced versus the average packet burstification delay when the traffic load utilization factor is $p = 0.03$, 0.1 and 0.3, respectively. The labels



(a) (b)

Fig. 5. Performance of the proposed algorithms for traffic load $p = 0.1$: (a)comparison of the proposed schemes with previously proposed algorithms, (b) parameters applied on the proposed algorithms.



(a) (b)

Fig. 6. Performance of the proposed algorithms for traffic load $p = 0.3$: (a) comparison of the proposed schemes with previously proposed algorithms, (b) parameters applied on the proposed algorithms.

in Figs. 4(b), 5(b) and 6(b) display the details on the values of the parameters N_{MIN}, T_A, L_{MIN} and α that give the corresponding results.

From Figs. 4(a), 5(a) and 6(a), we can see that the L_{MAX} algorithm exhibits (as expected) the worst performance for light load ($p = 0.003$), while its performance becomes relatively better for heavier load ($p = 0.1$ and 0.3). The opposite is true for the N_{MIN} algorithm, which behaves relatively worse for heavy traffic ($p = 0.1$ and 0.3), while its performance improves for light traffic ($p = 0.003$). For a given traffic load, the N_{MIN} algorithm shows worse relative performance when the parameter N_{MIN} is set at low values so as to produce large bursts (this is because for small values of N_{MIN}, the algorithm is rather intolerable to estimation errors). The αL algorithm always gives better results than the N_{MIN} algorithm, but does not succeed in outperforming some of the other algorithms considered. For a given traffic load, its relative standing compared to the other algorithms does not change with the choice of the parameter α (small values of α produce longer bursts as it can be seen in Figs. 4(b), 5(b) and 6 (b)). Among the burst assembly algorithms already proposed in the literature (that is, the T_{MAX}, L_{MAX}, T_{AVE} algorithms), the T_{AVE} algorithm gives the best performance. The proposed T_A algorithm outperforms the T_{AVE} algorithm, even though the improvement is rather small, as shown in Fig. 6(a). The improvement is more pronounced when the T_A algorithm generates longer bursts and when the traffic load is heavier.

The best performance is consistently demonstrated by the L_{MIN} algorithm, which achieves a 33% improvement over the T_A algorithm (the second best) for light traffic load ($p = 0.003$ and 0.1) and a 8% improvement for heavier traffic load ($p = 0.3$). The L_{MIN} algorithm can be considered a variation of the L_{MAX} algorithm, enhanced with the ability to predict the time periods where the value of *DBR* is expected to improve because of a large number of future packet arrivals. Note that in most of the figures, the curve that corresponds to the L_{MIN} algorithm is parallel to and above that of the L_{MAX} algorithm.

To conclude, the L_{MIN} algorithm seems to be the algorithm of choice when the average burstification delay (for a given burst size) or the average burst size (for a given burstification delay) is the criterion of interest. One should note, however, that the T_{AVE} and the T_A algorithms may be preferable when the delay jitter [6] is the main consideration (both of these algorithms also give a satisfactory average burstification delay to average burst size ratio).

References

1. T. Battestilli and H. Perros,"An Introduction to Optical Burst switching",*IEEE Optical Communications*, August 2003.
2. X. Yu, Y. Chen, and C. Qiao, "Study of traffic statistics of assembled burst traffic in optical burst switched networks," Proc. Opticomm, 2002, pp. 149-159.
3. V. Vokkarane, K. Haridoss, J. Jue, "Threshold-Based Burst Assembly Policies for QoS Support in Optical Burst-Switched Networks", Proc. Opticomm, pp. 125-136, 2003.
4. A. Ge, F. Callegati and L. Tamil,"On Optical Burst Switching and Self-Similar Traffic", *IEEE Communications Letters*, Vol. 4, No. 3. March 2000.
5. X. Cao, J. Li, Y. Chen and C. Qiao,"Assembling TCP/IP Packets in Optical Burst", Proc. IEEE Globecom, 2002.
6. K. Christodoulopoulos, E. Varvarigos and K. Vlachos, "A new Burst Assembly Scheme based on the Average Packet Delay and its Performance for TCP Traffic", under 2nd revision, Optical Switching and Networking, Elsevier, 2006.
7. Sven A. M. Östring, H. Sirisena,"The Influence of Long-range Dependence on Traffic Prediction", Proc. ICC, 2001.
8. D. Morató, J. Aracil, L. A. Díez, M. Izal, E. Magaña, "On linear prediction of Internet traffic for packet and burst switching networks", Proc. 10th Int. Conf. Computer Communications Networks, 2001, pp. 138-143.
9. J. Liu, N. Ansari, T. Ott, "FRR for Latency reduction and QoS Provisioning in OBS Networks", *IEEE JSAC*, Vol. 21, No. 7, Sept. 2003.
10. J.R. Treicher, C.R. Johnson Jr. and M. G. Larimore, *Theory and Design of Adaptive Filters*, New York: Wiley 1987.