Available online at www.sciencedirect.com
ScienceDirect
Optical Switching and Networking 4 (2007) 200–212
Optical Switching and Networking
www.elsevier.com/locate/osn

A new burst assembly scheme based on the average packet delay and its performance for TCP traffic

Konstantinos Christodoulopoulos*, Emmanouel Varvarigos, Kyriakos Vlachos

Computer Engineering and Informatics Department, University of Patras, GR26500, Rio, Greece
Research Academic Computer Technology Institute, GR26500, Rio, Greece

Received 31 July 2006; received in revised form 8 February 2007; accepted 7 May 2007
Available online 18 May 2007

Abstract

We propose and evaluate a new burst assembly algorithm based on the average delay of the packets comprising a burst. This method fixes the average delay of the packets belonging to an assembled burst to a desired value T_{AVE} that may be different for each forwarding equivalence class (FEC). We show that the proposed method significantly improves the *delay jitter* experienced by the packets during the burst assembly process, when compared to that of timer-based and burst length-based assembly policies. Minimizing packet delay jitter is important in a number of applications, such as real-audio and streaming-video applications. We also find that the improvement in the packet delay jitter yields a corresponding significant improvement in the performance of TCP, whose operation depends critically on the ability to obtain accurate estimates of the round-trip times (RTT).

Keywords: Optical burst switching (OBS); Burst assembly algorithm; Average delay; Delay jitter; TCP over OBS

1. Introduction

Switching in core optical telecommunications networks is currently performed using high-speed electronic or all-optical circuit switches. Switching with high-speed electronics requires optical-to-electronic (O/E) conversion of the data stream, making the switch a potential bottleneck of the network: efforts (including parallelization) for electronics to approach the optical speeds seem to meet practical limitations. Furthermore, the store-and-forward approach of packet-switching does not seem amenable to all-optical implementation because of the lack of practical optical random-access-memories to resolve contention. Circuit switching, on the other hand, is known to be inefficient for bursty traffic, involves a pretransmission delay for call setup and requires the aggregation of microflows into circuits, meaning that fine granularity and the control over individual microflows and their QoS are lost.

Optical burst switching (OBS) [1] has been introduced to combine both strengths of packet and circuit switching and considered a promising technology for next generation optical Internet. An OBS architecture 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. In general, each edge router maintains a separate (virtual) queue for each forwarding equivalence class (FEC) to hold

* Corresponding author at: Computer Engineering and Informatics Department, University of Patras, GR26500, Rio, Greece.
E-mail addresses: kchristodou@ceid.upatras.gr (K. Christodoulopoulos), manos@ceid.upatras.gr (E. Varvarigos), kvlachos@ceid.upatras.gr (K. Vlachos).

1573-4277/$ - see front matter © 2007 Elsevier B.V. All rights reserved.
doi:10.1016/j.osn.2007.05.001

the data packets that belong to that FEC until a burst is formed. (A FEC is defined from a source–destination pair and a given set of quality-of-service requirements).

The burst assembly process starts with the arrival of the first packet and continues until a predefined threshold is met. Two main burst assembly algorithms have been proposed in the literature: the *timer-based* method (T_{MAX}) [2], and the *burst length-based* method (B_{MAX}) [3]. In the timer-based method, a time counter is started any time a packet arrives belonging to a FEC whose queue is empty, and when the timer reaches the 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. In the second method, the burst length-based method, the bursts are thought of as containers of a fixed size B_{MAX} (in bytes), and as soon as the container is completely filled with data, the burst is created.

The timer-based method limits the delay of packets to a maximum value T_{MAX} but may generate undesirable burst lengths, 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 these drawbacks, hybrid (mixed time/length based) assembly algorithms were proposed [4], where bursts leave when either the time limit or the burst-size limit is reached, whichever happens first. Performance evaluation and comparison of these schemes in terms of burst length and interarrival time distribution as well as traffic autocorrelation studies (long range dependence), for various traffic profiles, have been carried out in [4,5]. In [6], the loss probability of the timer-based assembly algorithm when the buffer at the ingress node is limited was also studied.

Other burst assembly algorithms have also been proposed to provide QoS differentiation in OBS networks. [7] proposed assembling different classes of service (CoS) into the same burst, where packets are placed from head to tail with decreasing class. QoS is supported by performing burst segmentation to resolve contention and discarding the last part of the burst. In [8], a framework called assured horizon is presented. Assured horizon employs a timer-based algorithm that marks the bursts depending on their CoS and uses active dropping in the core to enable service differentiation. OBS network studies and evaluation of various QoS-enabling schemes have also been performed in [9–11].

There are two types of burst reservation protocols [12], tell-and-go and tell-and-wait. In tell-and-go protocols, the source transmits bursts without making any bandwidth reservations in advance. A control packet (called burst control header, or BCH) is usually sent out-of-band and leads the burst by a small offset time δ. If the reservation at a node is successful, the BCH packet (and the following burst) are forwarded to the next hop; otherwise, the burst is discarded, assuming there are no fibre delay lines (FDLs) to store it. In tell-and-wait protocols, on the other hand, a source that has a burst to transmit first tries to reserve the resources from source to destination by sending a short request message. If the requested bandwidth is successfully reserved on all the links along the path, an ACK is sent back to the source, which then sends out the burst; otherwise, a NAK is returned to release the previously reserved bandwidth, and initiate the retransmission of the request message by the source. An example of a tell-and-wait protocol is the efficient reservation virtual circuit (ERVC) protocol proposed in [13]. A number of one-way reservation schemes have been proposed for OBS, including the ready-to-go virtual circuit protocol [14], just-enough-time (JET) [1], Horizon [15] and just-in-time (JIT) [16].

Finally, the impact of burst aggregation process on TCP performance has been extensively studied in [17,18]. An adaptive burst assembly algorithm that takes into account the traffic situation and adapts the time threshold T_{max} accordingly was shown to improve TCP goodput and data loss rate [18]. The effect of the time threshold on short-lived TCP flows has been demonstrated in [19] and an optimum threshold value ($T_{\mathrm{MAX}} = \mathrm{RTT}/2$, where RTT is the round trip time between the TCP connection endpoints) has been proposed. A detailed analytical and simulation study of the effect of the variation of the burstification period for high and low speed sources in [20] has shown that under this traffic models the optimal value is $T_{\mathrm{MAX}} = 0.1 - 0.2\,\mathrm{RTT}$.

In this paper, we propose and evaluate a new burst assembly algorithm based on the average delay of the packets comprising a burst. More specifically, when a packet belonging to a given FEC arrives at the corresponding queue, a *Running Average Delay* of that queue is updated. When the average delay of the assembled packets in the queue reaches a threshold value T_{AVE}, a burst is created. This method fixes the average delay of the packets belonging to a given FEC to the desired value. Using the average delay as the assembly criterion, we show that the packet *delay jitter* in the assembled bursts is significantly improved compared to that of the timer-based and the length-based policies. Packet delay jitter is important in a number of applications, and especially in real-audio and streaming-video applications. We also find that packet delay jitter is important for TCP throughput. Since the performance of the TCP congestion control scheme

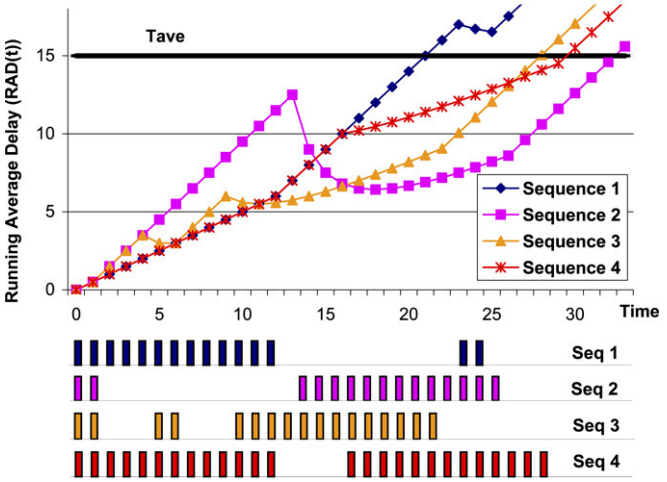

Fig. 1. Running average delay of packets in the queue (RAD(t)) for different packet sequences.

depends critically on the ability to obtain accurate estimates of the round-trip-times (RTT), minimizing the delay jitter introduced by the burstification process improves significantly TCP performance, as our results indicate.

The remainder of the paper is organized as follows: Section 2 describes the average-delay-based burst assembly algorithm that we introduce. Section 3 computes the average delay exhibited by the packets for the T_{AVE} and the T_{MAX} assembly schemes. Section 4 investigates the effects of the assembly scheme used on the packet delay distribution and the delay jitter, and compares the performance of the proposed scheme to that of the timer-based and the length-based algorithms. Finally, Section 5 evaluates TCP performance and the way TCP timeouts and TCP throughput depend on the applied burst assembly scheme and threshold value.

2. The average-delay-based burst assembly scheme

Each edge router maintains a separate queue for each Forwarding Equivalence Class (FEC). For each FEC, a parameter T_{AVE} is defined, corresponding to (an upper limit on) the desired average delay of the packets that belong to that FEC when going through the burst assembly process.

In the proposed burst assembly scheme, a running average delay (RAD) estimator starts being computed whenever a packet arrives at an empty queue. More specifically, the running average delay of the packets in the assembly queue at time t is defined as:

$$\mathrm{RAD}(t) = \frac{T_1(t) + T_2(t) + \cdots T_{n(t)}(t)}{n(t)} = \frac{\sum_{i=1}^{n(t)} T_i(t)}{n(t)}, \quad (1)$$

where $T_i(t) = t - S_i$ is the current (up to time t) delay of the ith packet at the queue, S_i is its arrival time, and $n(t)$ is the number of packets at the queue at time t. In the proposed algorithm, RAD(t) is computed continuously, and when its value becomes equal to the threshold value T_{AVE}, a burst is created. Therefore, burst assembly ends at the first time BAT (called *burst aggregation time*) at which RAD(BAT) $= T_{\mathrm{AVE}}$. When the burst is created, the RAD(t) of the FEC is reset to 0 and remains equal to 0 until the next packet arrival at the queue.

The running average delay is a function of time, and its value at time $t + \delta t$ can be computed from its value at time t as:

$$\mathrm{RAD}(t + \delta t) = \frac{n(t) \cdot (\mathrm{RAD}\,(t) + \delta t)}{n(t + \delta t)}. \quad (2)$$

When no packets arrive at the queue in the time interval of duration δt (that is, when $n(t) = n(t + \delta t)$), we have $\mathrm{RAD}(t + \delta t) = \mathrm{RAD}\,(t) + \delta t$ and RAD(t) increases proportionally to time with a slope equal to 1, reaching the threshold value T_{AVE} relatively soon. During periods at which many packets arrive, RAD(t) increases at a considerably slower rate or even decreases, and reaches the threshold value T_{AVE} at a later time.

Fig. 1 depicts the way RAD(t) varies with the arrival of packets for four sequences of packets. For simplicity we have assumed that all packets have equal length and arrive at discrete time instances, called slots. The proposed burst assembly algorithm monitors the average delay of the packets and stops assembling the bursts when RAD(t) reaches the predefined threshold T_{AVE}. If the proposed burst assembly algorithm with, say, threshold value $T_{\mathrm{AVE}} = 15$ was applied to the example of Fig. 1, the Burst Aggregation Times would

be 21, 32.5, 28, and 29.5 for sequences 1, 2, 3 and 4 respectively. In sequence 1, there is large number of packets that arrive soon after the first packet arrives at the empty queue, in sequence 2 a small number of packets arrive soon and many packets arrive much later, while sequence 3 is an intermediate case. We observe that in the case of sequence 1 ("front-loaded" arrivals) the burst is transmitted earlier than in the case of sequence 2 ("back-loaded" arrivals). Sequences 1 and 4 are identical up to a time, after which sequence 1 has much fewer arrivals than sequence 4. We can observe that the proposed assembly algorithm decides to send the burst much earlier in the case of the sequence 1 than in the case of the sequence 4. Intuitively, this seems like what we would like a burst assembly algorithm to do: wait while packets arrivals occur and transmit the burst when packet arrivals become scarce. Finally, if the packet arrival process experiences a large gap, the burst departure is expedited. Note that in the case of the timer-based algorithm ($T_{\max}$) the burst would leave at the same time for all sequences of packets depicted in Fig. 1.

Since RAD(t) increases proportionally to time with slope 1 when no packets arrive at the queue, the assembly process can be performed with the following timer-based algorithm that does not require continuous-real time monitoring of RAD(t), but updates RAD(t) only at discrete time instances, whenever new packets arrive at the assembly unit.

Event: A packet arrive at queue Q at time CT (Current Time)

If (the queue is empty)
RAD = 0;
N = 1;
PAT = CT;
Start the Assembly_Timer = Tave;
Else
RAD = $\frac{n\cdot(\mathrm{RAD}+(\mathrm{CT}-\mathrm{PAT}))}{n+1}$;
N = N + 1;
PAT = CT;
Update the Assembly_Timer = (Tave– RAD);

Event: Assembly_Timer Timeout

Assemble the burst, send out the control packet and schedule the data burst to be sent out after the offset time;
Stop the Assembly_Timer;

In the above algorithm, N is the current number of packets in the queue and PAT is the arrival time of the previous packet.

3. Average packet delay of the assembly algorithms

We denote by the random variable (r.v.) D_i the delay of the ith packet arrival. We also denote by the r.v. BAT_j the burst aggregation time of burst j. We then have $D_i = \mathrm{BAT}_j - S_i$. The average delay of the packets included in burst j is

$$\frac{\sum_{i=1}^{N(\mathrm{BAT}_j)} \mathrm{BAT}_j - S_i}{N(\mathrm{BAT}_j)} = \mathrm{RAD}(\mathrm{BAT}_j) = T_{\mathrm{AVE}}$$

where $N(\mathrm{BAT}_j)$ is the number of packets included in burst j.

The average packet delay for the proposed algorithm is clearly $E[D_i] = T_{\mathrm{AVE}}$, since by the construction of the proposed average-delay-based assembly scheme, all the assembled bursts exhibit the same average packet delay (T_{AVE}) which is also the overall average packet delay for that FEC.

Consider now the timer-based burst assembly scheme [2], which places an upper bound T_{MAX} on the maximum delay exhibited by all the packets. We assume that data units (requests) belonging to a given FEC arrive at an assembly queue according to a Poisson process with rate λ, and each data unit i consists of m_i packets. The m_i are assumed to be independent and identically distributed random variables that follow some arbitrary distribution. In Appendix A we show that the average delay suffered by the packets when the timer-based algorithm with parameter T_{MAX} is used can be calculated to be

$$\begin{aligned} E[D_i]^{T_{\mathrm{MAX}}} &= E[T_{\mathrm{MAX}} - S_i] \\ &= \frac{T_{\mathrm{MAX}}}{2} + \frac{1 - \mathrm{e}^{-\lambda\cdot T_{\mathrm{MAX}}}}{2\cdot\lambda}. \end{aligned} \tag{3}$$

In other words, in order for the timer-based assembly algorithm to result in a given desired average packet delay T_{AVE}, we should choose its parameter T_{MAX} so that

$$T_{\mathrm{AVE}} = \frac{T_{\mathrm{MAX}}}{2} + \frac{1 - \mathrm{e}^{-\lambda\cdot T_{MAX}}}{2\cdot\lambda}, \tag{4}$$

assuming the arrival process is a compound Poisson.

Note that the choice of the parameter T_{MAX} depends on the arrival rate λ, which has to be known and is independent of the distribution of m_i (for example, for λ small, we have to choose $T_{\mathrm{MAX}} = T_{\mathrm{AVE}}$, while for λ large we should choose $T_{\mathrm{MAX}} = 2T_{\mathrm{AVE}}$). Even if we assume that the traffic arrival model described above is a good approximation of the actual traffic and the traffic load is known, so that by choosing the parameter

T_{MAX} according to Eq. (4), we can make the average packet delay of the timer-based scheme equal to the desired value T_{AVE}, the *variance* of the packet delay of the T_{MAX} scheme will still be larger than the packet delay variance of our proposed scheme. Intuitively, the reason this happens is that the proposed average-delay-based scheme enforces the average packet delay to be equal to T_{AVE} where the averaging is taken over all packets belonging to a given burst, while the timer-based scheme can only enforce, in the best case, the long-term average packet delay to be equal to T_{AVE} (through the appropriate choice of T_{MAX}). The performance results obtained in the following section quantify the advantage of the proposed scheme with respect to the packet delay variance, while the results in Section 5 show the significance of the improved delay jitter for the case of TCP traffic.

4. Performance evaluation results

We have extended the ns-2 platform [22] by programming the average-delay-based burst assembly scheme and conducted simulation experiments to compare its performance to that of the timer-based and the burst-length-based methods. For the network simulations we have used modules found in [17].

4.1. Traffic generating source

We modeled a Poisson–Pareto traffic generating source that directly feeds a burst assembly queue. The traffic generator has a rate r bit/s and produces superpackets ("DATA ON" periods) whose size follows the Pareto distribution with shape parameter a, and mean size $1/\mu$. When the generator produces a superpacket whose size is greater than *packetsize* Bytes, the superpacket is fractured and delivered as a sequence of packets of *packetsize*. Superpackets arrive according to a Poisson process with rate λ superpackets/s.

Unless explicitly stated otherwise, the following values were used in the simulation experiments: $\alpha = 1.2, r = 1$ Gbps, *superpacket_size* = 60 KB (thus *mean_data_on_time* = 0.48 ms), *packetsize* = 1500 bytes, and load $p = 0.3$.

4.2. Packet delay distribution

Each packet, depending on its arrival time and its burst assembly completion time, remains in the assembly queue for a different amount of time. Fig. 2 shows the probability density function of the packet delay distribution for various values of the offered load, for the proposed average-delay-based assembly scheme with parameter T_{AVE}, (abbreviated "T_{AVE} scheme"), and for the timer-based and length-based assembly schemes with parameters T_{MAX} and B_{MAX} (abbreviated "T_{MAX} scheme" and "B_{MAX} scheme", respectively). For fairness in the performance comparisons, the T_{MAX} and B_{MAX} threshold values in the latter schemes were chosen so that the packets would have an overall average delay equal to the parameter T_{AVE} of the average-delay-based scheme (computed from Eq. (3) for the timer-based scheme, and experimentally for the length-based scheme). In the results shown in Fig. 2(a) and (b), the T_{AVE} parameter was set to 3.2 ms and 20 ms, respectively. From Fig. 2, we can see that for the T_{AVE} scheme, the packet delay distribution resembles a symmetrical, bell-shaped, Gaussian-like distribution. The variance of this bell-shaped distribution increases as the traffic load increases and eventually the distribution becomes uniform when the load approaches $p = 1$. On the other hand, in the T_{MAX} scheme, the packet delay distribution resembles a uniform distribution, with a sizeable peak appearing at the applied T_{MAX} value (due to the initiation of the assembly process upon reception of a packet at an empty queue). Furthermore, the B_{MAX} algorithm has a peak at low values of delay since its completion is triggered by a "final" packet arrival that always gets the smallest delay. However, its distribution is wider than the others since there is no time constraint of any form. The performance differences between the schemes tend to diminish when the traffic load increases.

Fig. 3 illustrates the average number of assembled bursts per superpacket generated under the three burst-assembly algorithms considered. Labels in Fig. 3 (as well as in the following graphs) display the corresponding values of the B_{MAX}, T_{MAX} and T_{AVE} parameters. The average number of generated bursts is plotted against the measured average delay of the packets. As expected, when we can tolerate a larger average packet delay for the assembly process, fewer and larger bursts will be generated. For a given value of the average delay experienced by the packets, we see that the number of bursts generated by the B_{MAX} scheme is considerably larger than the number of bursts generated by the T_{MAX} scheme and the T_{AVE} scheme (the latter schemes are rather identical in this respect). Note that the smaller the number of bursts generated by a burst assembly scheme (for a given set of QoS parameters; here the QoS parameter is the average delay), the better the burst assembly scheme, since few, larger bursts imply smaller processing requirements at the nodes of the OBS network.

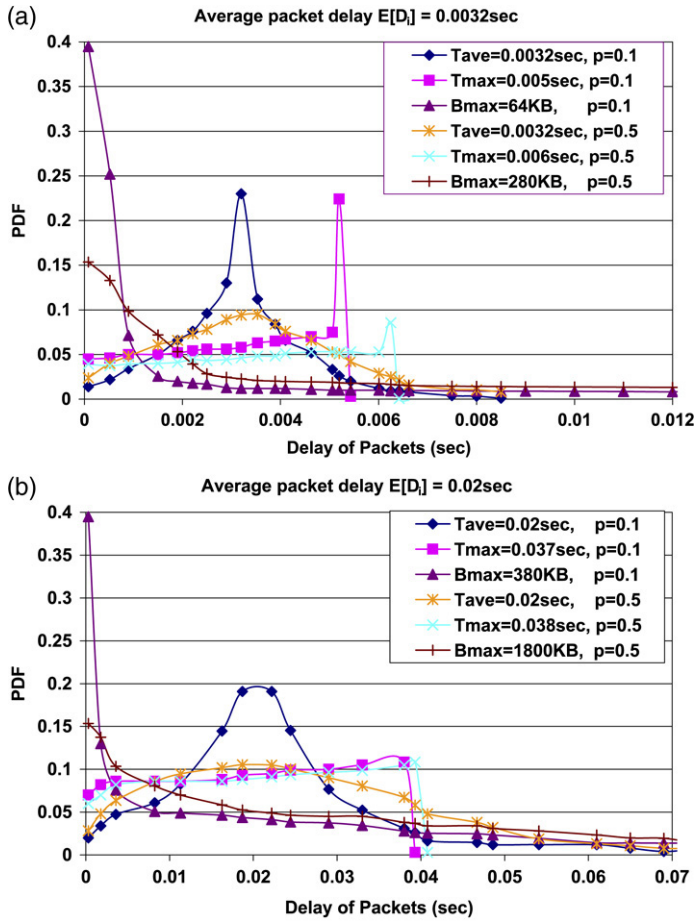

Fig. 2. Packet delay distribution for load $p = 0.1$ and 0.5 when T_{AVE}, T_{MAX} and B_{MAX} assembly schemes are applied to obtain an average packet delay (a) $E[D_i] = 3.2$ ms, (b) $E[D_i] = 20$ ms.

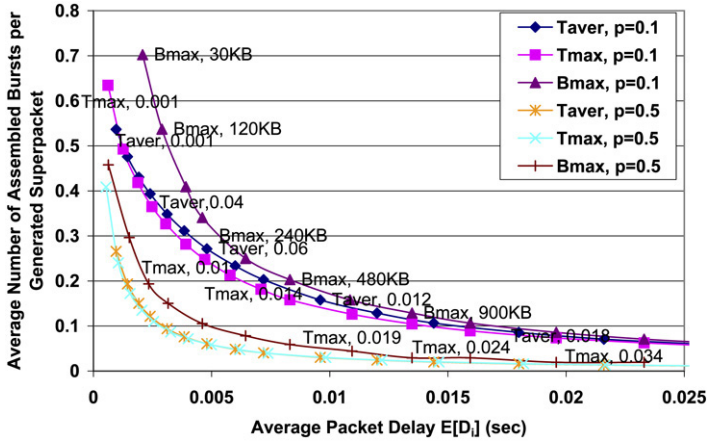

Fig. 3. Average number of assembled bursts per generated superpacket for the T_{AVE}, T_{MAX} and B_{MAX} assembly schemes.

Fig. 4(a) shows the variance of the packet delay (*delay jitter*) for the three burst assembly algorithms evaluated, while Fig. 4(b) the coefficient of variation (ratio of the standard deviation over the average packet delay).

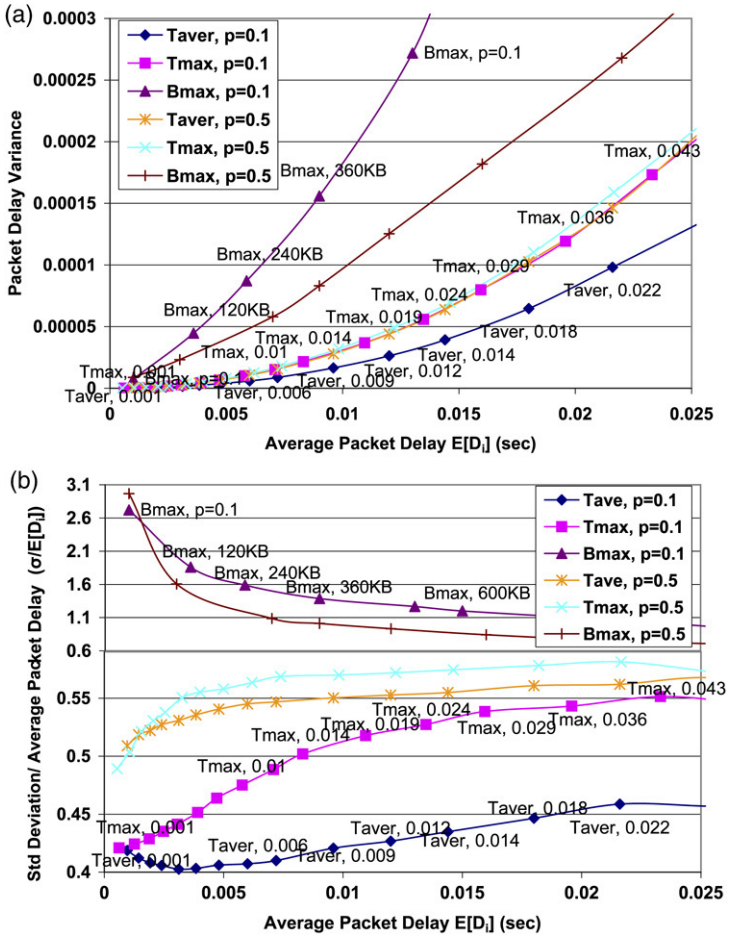

Fig. 4. (a) Packet delay variances for T_{AVE}, T_{MAX} and B_{MAX} assembly schemes and (b) coefficient of variation (standard deviation over $E[D_i]$).

From these graphs we can deduce that the improvement in the delay jitter obtained when the T_{AVE} scheme is used is significant. The B_{MAX} scheme has the worst delay jitter over all schemes, even though its performance improves as the load increases. We can calculate the improvement in the variance obtained by using the T_{AVE} scheme instead of the T_{MAX} scheme by:

$$\frac{\sigma^2_{T_{\mathrm{MAX}}}(p) - \sigma^2_{T_{\mathrm{AVE}}}(p)}{\sigma^2_{T_{\mathrm{MAX}}}(p)},$$

where $\sigma^2(p)$ is the variance of packet delay of the T_{AVE} or T_{MAX} scheme for traffic load p. The T_{AVE} scheme outperforms the T_{MAX} scheme in terms of delay variance by more than 35% for load $p = 0.1$ and more than 5% for load $p = 0.5$. For heavy traffic loads, the delay variances of both algorithms eventually converge since heavy packet generation rates lead to uniform packet delay distributions and diminish the difference between the schemes.

Fig. 5 shows the effect of the burstiness of the generated traffic, quantified through the generator's Pareto shape parameter a, on the packet delay jitter. In particular, Fig. 5(a) displays the packet delay variance for the burst assembly schemes considered, while Fig. 5(b) shows the coefficient of variation. From Fig. 5, it can be seen that as traffic gets more bursty (small values of a), the variance in the delay incurred by the packets during the burst assembly process increases for the B_{MAX} scheme, remains constant for T_{MAX} scheme and decreases for the T_{AVE} scheme. The delay jitter performances of all the schemes converge as the traffic becomes smooth ($a \geq 1.5$) and diverge as traffic burstiness increases ($a < 1.5$). Therefore, we conclude that the benefits obtained by using the T_{AVE} scheme are more significant when the traffic is bursty.

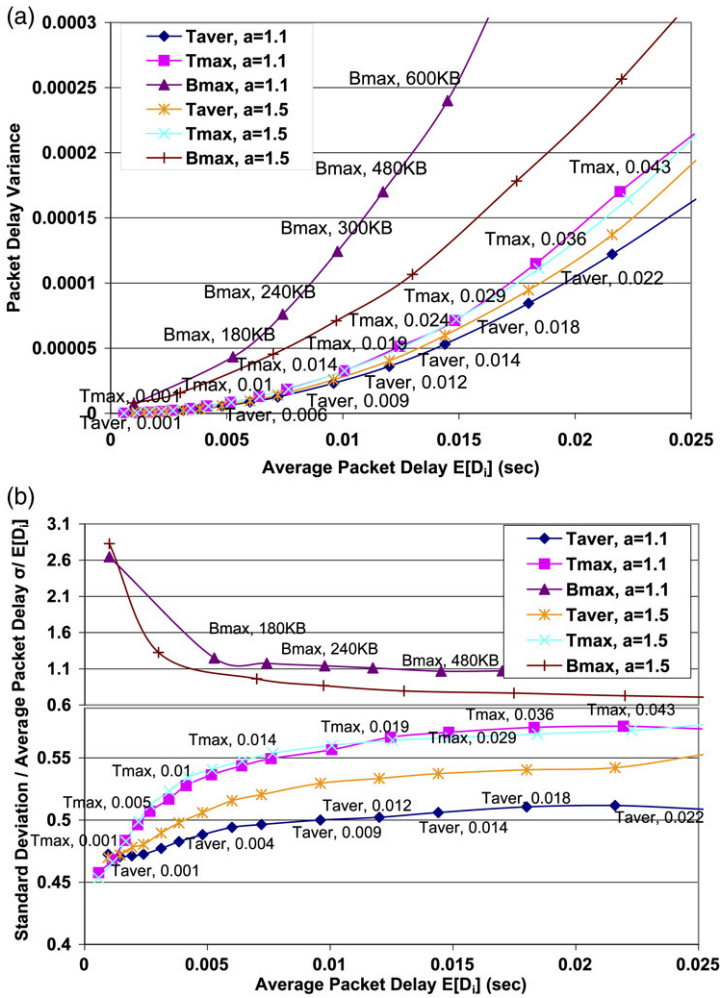

Fig. 5. Effect of burstiness through the generated superpacket size Pareto shape parameter a. (a) graphs the packet delay variance while (b) the coefficient of variation.

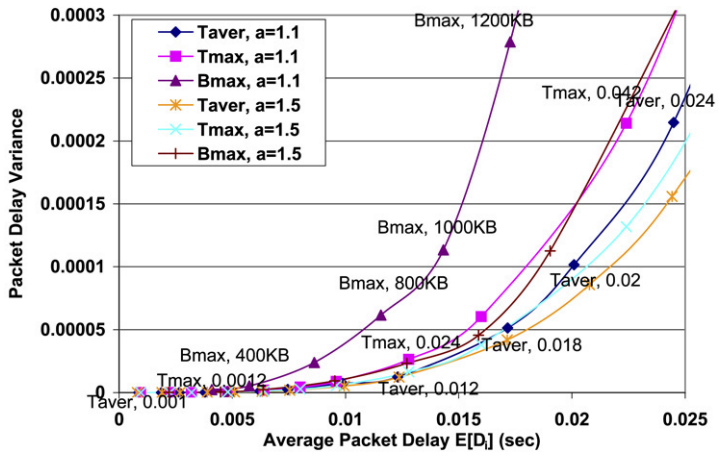

Fig. 6. Effect of burstiness through the interarrival times Pareto shape parameter a.

Finally, we also examined the performance of the burst assembly algorithms when the arrival times of the superpackets (which are themselves bursts) are not Poisson but correlated across different time scales. More specifically, we have assumed that the size of a superpacket is a random variable (r.v.) that follows a Pareto distribution with mean size $B = 60$K and shape parameter $a = 1.2$, while superpackets' interarrival times follow, again, a Pareto distribution with mean size $= 0.016$ (to obtain load $= 0.3$) and two different shape parameter values ($a = 1.1$ and $a = 1.5$). From Fig. 6, we can observe that as interarrival times get more bursty (moving from larger values of a to smaller), the variance in the delay incurred by the packets during the burst assembly process increases for all the assembly schemes. The performance of T_{AVE} assembly scheme remains better in all examined cases, a fact that further illustrates that the proposed algorithm can improve the delay variance of the assembled packets.

5. TCP performance

The performance of TCP over OBS networks has been studied in previous works [17–20] 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 that challenges the window mechanism used by TCP protocol for congestion control.

More specifically, TCP implementations attempt to predict future round trip times (RTT) by sampling the behavior of packets sent over a connection and averaging these measurements into a "smoothed" round-trip time estimate. To this end, every time an ACK packet is received, the source estimates the mean deviation, which is the average absolute difference between the samples and the RTT mean value, and calculates a new timeout based on these estimations [21].

Although the increase of RTT in TCP performance is widely considered in the corresponding literature, the increase of the packet delay variance which is also introduced in the assembly process is often overlooked. The burstification process affects the accuracy of the RTT estimates of TCP, since the delay jitter introduced by it adds "noise" that may result in inaccurate TCP timeout estimations. If the TCP retransmission timeout is chosen too small, a slow packet will be misinterpreted as being lost leading to the activation of the congestion avoidance mechanism. The congestion avoidance procedure decreases the TCP window size when it perceives a packet loss, and retransmits the packet, resulting in a deterioration of the TCP throughput performance.

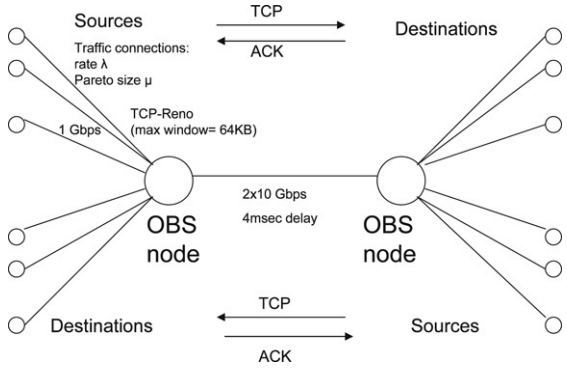

Fig. 7. Network scenario.

In order to evaluate the effect of the burst assembly schemes under investigation on TCP performance, we have connected two Poisson–Pareto traffic sources at two directly connected OBS nodes, as shown in Fig. 7. Source access delay was set equal to 10 μs and thus the RTT of the considered network was 8.020 ms $\simeq$ 8 ms. The connection between the OBS nodes is assumed to be lossless, in order to focus on the effect of the burst assembly process and the associated delay. Each superpacket generated is assumed to be a new request for transmitting information with size equal to the superpacket's size. This traffic model resembles web traffic as found in [19]. Web page sizes follow a Pareto distribution with shape parameter $a = 1.2$ and mean size 60 KB, and page requests follow a Poisson arrival process with rate λ. The experiments were performed for 50 000 connection requests. We have considered HTTP-1.1 at the session level, TCP-Reno at the transport layer, and assumed that web-pages are transmitted through persistent connections. TCP Data and ACK packets are forwarded to the OBS nodes where burst assembly is performed (and thus are multiplexed in the same queue). Just Enough Time (JET) [1] was implemented as the connection establishment protocol for the transmission of the assembled burst from one node to another.

Fig. 8 shows the average number of TCP timeouts per connection. The increase of the RTT – alone – is not able to produce TCP timeouts (without other constrains such as limited buffers or burst losses), since it only affects the delay performance of TCP without interfering with the congestion avoidance mechanism. Therefore all these timeouts were caused by false packet loss detections by the transport layer. As expected, when

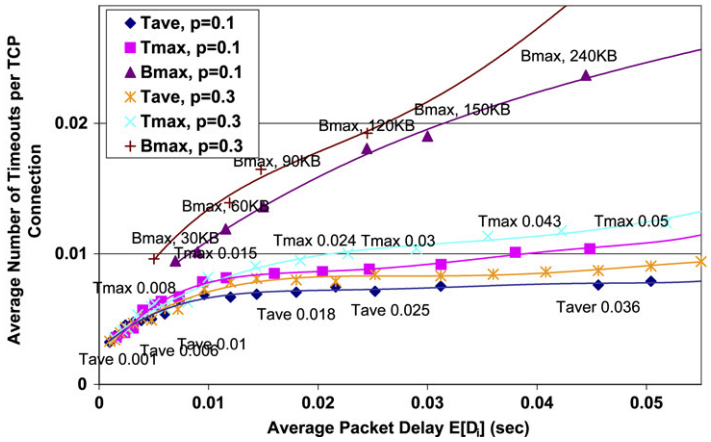

Fig. 8. Average number of timeouts per connection versus the average packet delay for load $p = 0.1$ and $p = 0.3$.

the values of the parameters T_{AVE}, T_{MAX}, and B_{MAX} are chosen to be small, the assembled bursts consist of a small number of packets that exhibit small delay jitter, and thus the performances of the examined schemes are similar. However, as the allowed $E[D_i]$ increases the T_{AVE} assembly scheme performance remains almost constant due to the small delay jitter it introduces, while the performance of the other assembly schemes deteriorates.

Fig. 9(a) shows the average number of assembled bursts per TCP connection versus the average delay allowed for the assembly process. In all the simulation experiments the traffic sources generated the same number of connection requests, and since the mean request size was kept fixed at 60 KB, the transmitted information was the same in all cases. The number of assembled bursts for small allowed average packet delay (i.e. small threshold values for the parameters T_{AVE}, T_{MAX}, and B_{MAX}) is large, resulting in high control plane overhead (OBS setup packet transmissions) [17]. Fig. 9(b) shows the corresponding average burst size. It can be seen that for a given average delay, the T_{AVE} scheme assembles fewer and larger bursts, meaning that it introduces smaller processing overhead at the nodes of the OBS network.

Fig. 10 illustrates the TCP throughput for the three assembly schemes considered, for traffic loads $p = 0.1$ and $p = 0.3$. It is interesting to see that when the average delay is small (i.e. when the values of the parameters T_{AVE}, T_{MAX}, and B_{MAX} are chosen to be small), TCP performance is not satisfactory. The generated bursts contain a small number of TCP packets and thus if we consider a TCP flow in which the TCP window has "opened", breaking the packet transmissions to more than one bursts would hinder TCP performance. In this case the network propagation delays (large number of transmitted bursts) dominate the assembly delays and burden the TCP windows mechanism. As the average delay increases (equivalently, as the parameters used in the burst assembly schemes increase), larger bursts are transmitted, and TCP achieves higher rates. In this case we observe a balance between the network propagation delays and the assembly delays. Thus, TCP throughput steadily increases as the average packet delay allowed for the burst assembly process increases, and reaches a constant rate for average packet delay values higher than 10 ms. However, for significantly higher values of the average packet delay ($\geq$40 ms), TCP throughput starts to decrease again. If we consider a TCP source which has transmitted all the packets that correspond to its window, a large assemble parameter would delay these packets at the ingress node. In this case the large assembly periods dominate the network propagation delays and thus negatively affect the overall roundtrip time and the throughput that can be achieved. Under all cases and especially when the average packet delay allowed for the burst assembly process is large, the proposed T_{AVE} scheme outperforms the other burst assembly schemes considered. As far as the optimal choice for the parameter T_{MAX} of the timer-based scheme is concerned our experiments indicate that the throughput performance is better when medium timeout values are used. In our approach, the values that optimize T_{MAX} performance are slightly larger than those argued in [19,20], while high throughput is also maintained for a larger period. The aforementioned differences are the combined result of the lossless connection between the two OBS nodes and the assumed traffic model, since a large number of short-lived TCP connections are continuously active.

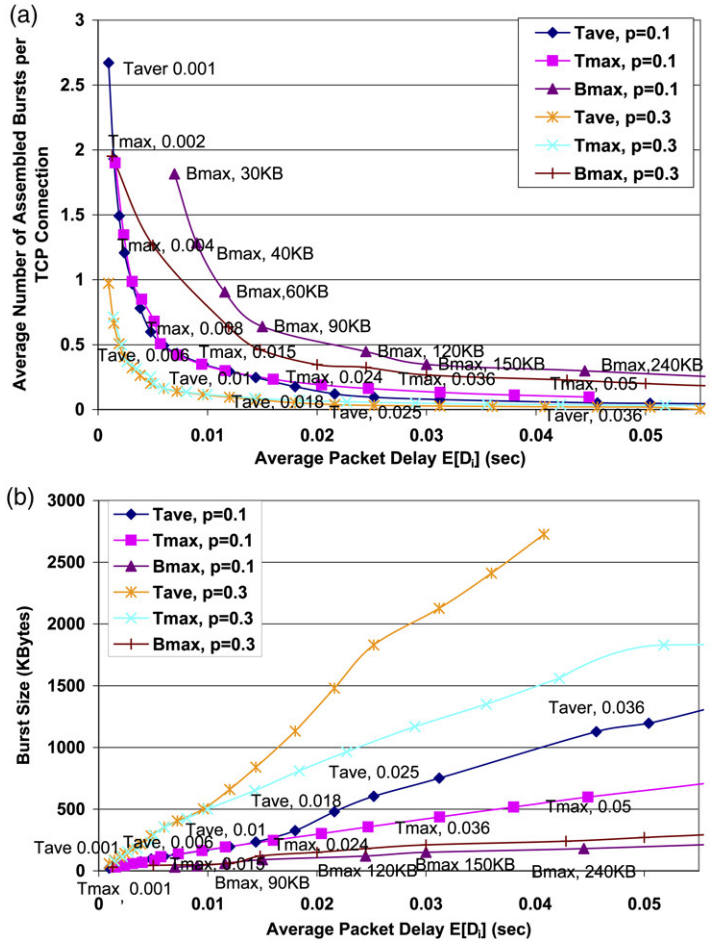

Fig. 9. (a) Average number of assembled bursts per TCP connection request and (b) the corresponding average burst sizes.

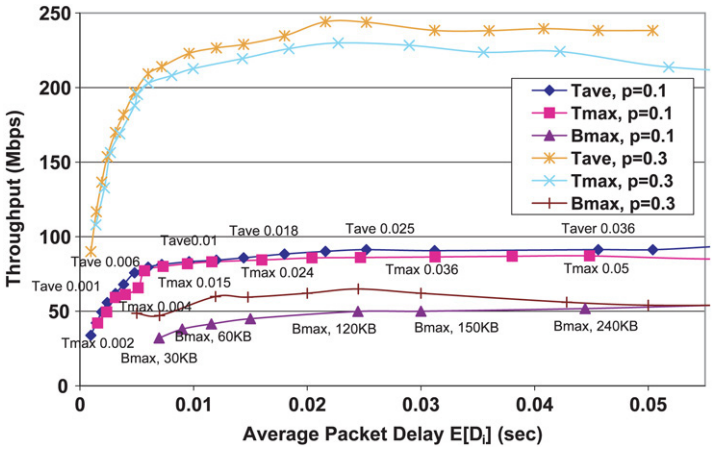

Fig. 10. Throughput versus the average packet delay.

It is worth noting that in these experiments the chosen propagation delay = 4 ms (correspond to a 800 km connection) is comparable to the average delay suffered by the packets. The aggregation parameters were chosen realistically with respect to the corresponding literature in which burst aggregation

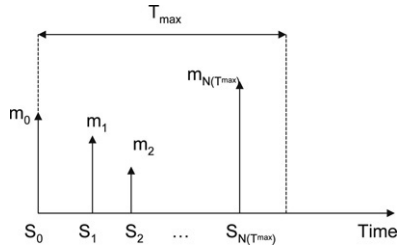

Fig. 11. Calculating the average packet delay for the T_{MAX} timer based assembly algorithm.

times are usually taken between a few to a few tens of milliseconds (1–50 ms). The effect of our algorithm is higher in a network in which propagation delays are comparable to aggregation times. We have actually executed some experiments in smaller and larger networks and conclude that in a larger network the effects of the burst assembly processes decrease, irrespective of the used algorithm. However, the positive effects of our algorithm remain in most realistic cases.

6. Conclusions

We proposed and evaluated a new burst assembly algorithm based on the average delay of the packets comprising a burst. The algorithm computes the running average delay of the packets in the burst being formed, and when it reaches a threshold T_{AVE}, the burst is created. The proposed scheme guarantees that the average delay incurred by the packets during the assembly process is equal to the desired value T_{AVE}.

We have shown that using the average delay as the assembly criterion significantly improves the delay jitter of the assembled packets compared to the timer-based (T_{MAX}) and the burst length-based (B_{MAX}) schemes. We have also shown that the improvement in the packet delay jitter yields a significant improvement in the operation of TCP, whose performance depends critically on the ability to obtain accurate estimates of the round-trip times between the connection endpoints (RTT).

Acknowledgements

This work has been supported by European Commission through projects IST-LASAGNE and the Greek General Secretariat for Research and Technology—GSRT via the PENED project.

Appendix A

A.1. Calculation of the average packet delay for the timer-based assembly scheme with parameter T_{MAX}

We assume that transmission requests arrive at the assembly queue that corresponds to a given forwarding equivalent class (FEC) according to a Poisson process with rate λ requests/s. Request i consists of a random number m_i of packets, each of constant size PS. The m_i's are assumed to be a family of independent and identical distributed random variables that follow an arbitrary probabilistic distribution, and are independent of the arriving Poisson process. This stochastic process is commonly referred to as a *compound poisson process* [24].

Fig. 11 illustrates the effect of applying the timer-based assembly algorithm with parameter T_{MAX} when traffic arrives according to the compound Poisson process described above. The delay of a packet arriving in request i is the random variable (r.v.)

$$D_i = T_{\mathrm{MAX}} - S_i, \quad i = 1, 2, \ldots, N(T_{\mathrm{MAX}}),$$

where $N(T_{\mathrm{MAX}})$ is the number of requests that arrive during the burst assembly period T_{MAX}. The probability density function of $N(T_{\mathrm{MAX}})$ is:

$$\mathrm{PDF}\,(N(T_{\mathrm{MAX}}), n) = \mathrm{e}^{-\lambda \cdot T_{\mathrm{MAX}}} \frac{(\lambda \cdot T_{\mathrm{MAX}})^{n-1}}{n-1}, \quad [23]$$

where we have used the fact that a burst assembly period is always triggered by an arriving request. We can compute the average packet delay for the T_{MAX} scheme as

$$E\,[D_i]^{T_{\mathrm{MAX}}} = E\,[T_{\mathrm{MAX}} - S_i] = T_{\mathrm{MAX}} - E\,[S_i]\,.$$

The average value of the arrival times, where the averaging is taken over all packets (and not over all requests) is given by

$$\begin{aligned} E[S_i] &= E[E[S_i \mid N(T_{\mathrm{MAX}}) = n]] \\ &= E\left[E\left[\frac{m_1 \cdot 0 + \sum_{i=2}^{n} m_i \cdot \frac{T_{\mathrm{MAX}}}{2}}{\sum_{i=1}^{n} m_i}\right]\right] \\ &= \frac{T_{\mathrm{MAX}}}{2} \cdot E\left[1 - E\left[\frac{m_1}{\sum_{i=1}^{n} m_i}\right]\right] \\ &= \frac{T_{\mathrm{MAX}}}{2} - \frac{T_{\mathrm{MAX}}}{2} E\left[E\left[\frac{m_1}{\sum_{i=1}^{n} m_i}\right]\right] \end{aligned}$$

where we have used the fact that $S_1 = 0$ (the burst assembly period starts upon the arrival of request 1), and all other S_i ($i > 1$) are uniformly distributed in $[0, T_{\mathrm{MAX}}]$, due to the Poisson character of the

requests [24]:

$$E\left[\frac{m_1}{\sum_{i=1}^{n} m_i}\right] = E\left[\frac{m_2}{\sum_{i=1}^{n} m_i}\right] = \cdots = E\left[\frac{m_n}{\sum_{i=1}^{n} m_i}\right]$$

$$\Rightarrow E\left[\frac{m_1}{\sum_{i=1}^{n} m_i}\right] + E\left[\frac{m_2}{\sum_{i=1}^{n} m_i}\right] \cdots + E\left[\frac{m_n}{\sum_{i=1}^{n} m_i}\right]$$

$$= 1 \Rightarrow E\left[\frac{m_1}{\sum_{i=1}^{n} m_i}\right] = \frac{1}{n}$$

and thus

$$E\left[E\left[\frac{m_1}{\sum_{i=1}^{n} m_i}\right]\right] = E\left[\frac{1}{N(T_{\mathrm{MAX}})}\right]$$

$$= \sum_{n=1}^{\infty} \frac{1}{n} \cdot \mathrm{PDF}\left(N(T_{\mathrm{MAX}}), n\right)$$

$$= \frac{1 - \mathrm{e}^{-\lambda \cdot T_{\mathrm{MAX}}}}{\lambda \cdot T_{\mathrm{MAX}}}.$$

Concluding, the average packet delay introduced by the timer-based burst assembly scheme with parameter T_{MAX} is

$$E\left[D_i\right]^{T_{\mathrm{MAX}}} = T_{\mathrm{MAX}} - \frac{T_{\mathrm{MAX}}}{2}\left(1 - E\left[\frac{1}{N(T_{\mathrm{MAX}})}\right]\right)$$

$$= \frac{T_{\mathrm{MAX}}}{2} + \frac{1 - \mathrm{e}^{-\lambda \cdot T_{\mathrm{MAX}}}}{2 \cdot \lambda}.$$

References

[1] C. Qiao, M. Yoo, Optical burst switching (OBS) — A new paradigm for an optical internet, Journal of High Speed Networks 8 (1) (1999) 69–84.

[2] A. Ge, F. Callegati, L. Tamil, On optical burst switching and selfsimilar traffic, IEEE Communications Letters 4 (2000) 98–100.

[3] V. Vokkarane, K. Haridoss, J.P. Jue, Threshold-based burst assembly policies for QoS support in optical burst-switched networks, in: Proc. of Opticomm, 2002, pp. 125–136.

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

[5] M. Izal, J. Aracil, On the influence of self-similarity on optical burst switching traffic, in: Proc. IEEE Globecom, vol. 3, 2002, pp. 2308–2312.

[6] M. Düser, P. Bayvel, Analysis of a dynamically wavelength-routed optical burst switched network architecture, Journal of Lightwave Technology 20 (2002) 574–585.

[7] V. Vokkarane, Q. Zhang, J.P. Jue, B. Chen, Generalized burst assembly and scheduling techniques for QoS support, in: Proc. of GLOBECOM, vol. 3, 2002, pp. 2747–2751.

[8] K. Dolzer, Assured Horizon — An efficient framework for service differentiation in optical burst switched networks, in: Proc. OptiComm, 2002.

[9] K. Dolzer, C. Gauger, On burst assembly in optical burst switching networks — A performance evaluation of just-enough-time, in: Proc. of the 17th International Teletraffic Congress, ITC 17, 2001, pp. 149–160.

[10] C. Gauger, K. Dolzer, J. Späth, S. Bodamer, Service differentiation in optical burst switching networks, Beiträge zur 2. ITG Fachtagung Photonische Netze, Dresden, 2001, pp. 124–132.

[11] M. Casoni, E. Luppi, M. Merani, Impact of assembly algorithms on end-to-end performance in optical burst switched networks with different QoS classes, in: Proc. IEEE/SPIE 3rd Workshop on Optical Burst Switching, 2004.

[12] Yang Chen, Chunming Qiao, Xiang Yu, Optical Burst Switching (OBS): A new area in optical networking research, IEEE Network Magazine 18 (3) (2004) 16–23.

[13] E.A. Varvarigos, V. Sharma, An efficient reservation connection control protocol for gigabit networks, Computer Networks and ISDN Systems 30 (12) (1998) 1135–1156.

[14] E.A. Varvarigos, V. Sharma, The ready-to-go virtual circuit protocol: A loss-free protocol for multigigabit networks using FIFO buffers, IEEE/ACM Transactions on Networking 5 (5) (1997) 705–718.

[15] J.S. Turner, Terabit burst switching, Journal of High Speed Networks 8 (1) (1999) 3–16.

[16] I. Baldine, G.N. Rouskas, H.G. Perros, D. Stevenson, JumpStart: A just-in-time signaling architecture for WDM burst-switched networks, IEEE Communications 40 (2) (2002) 82–89.

[17] S. Gowda, R.K. Shenai, K.M. Sivalingam, H.C. Cankaya, Performance evaluation of TCP over optical burst-switched (OBS) WDM networks, in: Proc. of ICC, vol. 2, 2003, pp. 1433–1437.

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

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

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

[21] V. Jacobson, Modified TCP congestion avoidance algorithm, April 30, 1990, End2end-interest mailing list.

[22] The Network Simulator – ns-2. http://www.isi.udu/nsnam/ns.

[23] M. de Vega Rodrigo, J. Gota, An analytical study of optical burst switching aggregation strategies, in: Proc. IEEE/SPIE 3rd Workshop on Optical Burst Switching, 2004.

[24] Sheldom M. Ross, Stochastic Processes, second ed., John Willey & Sons, 1996.