Packet Pacing in Short Buffer Optical Packet Switched Networks

Vijay Sivaraman*‡, Hossam Elgindy†, David Moreland‡, Diethelm Ostry‡
*School of Electrical Engineering and Telecommunications, University of New South Wales, Australia
†School of Computer Science and Engineering, University of New South Wales, Australia
‡ICT Centre, CSIRO, Australia

Abstract— **In the absence of a cost-effective technology for storing optical signals, emerging optical packet switched (OPS) networks are expected to have severely limited buffering capability. This paper investigates the resulting impact on end-to-end loss and throughput, and proposes that the optical edge switches "pace" packets into the OPS core to improve performance without adversely affecting end-to-end delays. In this context, our contributions are three-fold. We first evaluate the impact of short buffers on the performance of real-time and TCP traffic. This helps us identify short-time-scale burstiness as the major contributor to performance degradation, so we propose that the optical edge switches pace the transmission of packets into the OPS core while respecting their delay-constraints. Our second contribution develops algorithms of poly-logarithmic complexity that can perform optimal real-time pacing of high data rate traffic. Lastly, we show via simulations of a realistic network carrying real-time traffic that pacing can significantly reduce losses at the expense of a bounded increase in end-to-end delay. The loss-delay trade-off mechanism provided by pacing can help achieve desired OPS network performance.**

I. Introduction

The maturation of Wavelength Division Multiplexing (WDM) technology in recent years has made it possible to harness the enormous bandwidth potential of an optical fibre cost-effectively. As systems supporting hundreds of wavelengths per fibre with transmission rates of 10-40 Gbps per wavelength become available, electronic switching is increasingly challenged in scaling to match these transport capacities. All-optical switching [1] shows promise in meeting these challenges. To support data traffic efficiently, various optical sub-wavelength switching methods such as in [2], [3] have been proposed, of which optical packet switching (OPS) [4] is particularly attractive. Several experimental test-beds [5]–[9] have demonstrated the feasibility of OPS.

A fundamental concern in OPS networks is contention, which occurs at a switching node whenever two or more packets try to leave on the same output link, on the same wavelength, at the same time. In electronic store-and-forward switches, contention is resolved relatively easily using RAM (that can buffer as many as a million packets). By comparison, a state-of-the-art optical buffer available on an integrated opto-electronic chip can hold at most a few dozen optical packets [10]. Alternatively, spools of fibre can implement fibre delay lines (FDLs) [11] that provide optical buffering capability. Unfortunately, the high speed of light implies that even minimal buffering requires large fibre spools (1 km of fibre buffers light for only 5μsec; contrast this to conventional electronic routers that typically have buffering of 50-250 msec), and this makes the provision of more than a few tens of μsec of delay with FDLs unwieldy. Additionally, incorporating FDLs into a typical OPS switch design (such as the shared memory architecture [4], [12]) requires larger optical crossbars, which can add significantly to cost as the FDL buffers increase. At the present time it seems that OPS networks of the foreseeable future will have very short buffering capacity.

It should be noted that there have been other proposals for resolving contentions in the optical domain, such as wavelength conversion [13], [14], deflection routing [15], and combinational schemes [16]. However, these schemes also have limitations (cost of wavelength converters, packet reordering, etc.), and are not expected to significantly alleviate contentions in a cost-effective manner. Though we do not explicitly consider these schemes, our conclusions are generally applicable whenever contention resolution resources are sparse in the network.

Our objective in this paper is to investigate the impact of sparse OPS contention resolution resources (henceforth "short buffers") on the performance of real-time and TCP traffic, and to develop means of managing the performance degradation. Our contributions are three-fold. First, we demonstrate via traffic trace simulations and TCP experiments that in spite of the high capacity available in OPS networks, short buffers significantly impact performance when the traffic exhibits short-time-scale burstiness. To address this, we propose the "pacing" of packets at the optical edge, wherein the traffic is made less bursty at short-time-scales, while respecting delay constraints. Devising an optimal pacer that operates in real-time on arbitrary traffic at high data rates is challenging. Our second contribution therefore formulates algorithms that can achieve this in time poly-logarithmic in the number of queued packets, and points to approximations feasible for efficient hardware implementation. For our final contribution, we quantify via simulations the loss-delay tradeoffs of packet pacing, both for a single link and for an OPS network topology derived from an operational Australian network carrying realistic traffic streams. We propose that pacing at the optical edge can be instrumental in realising acceptable performance in emerging short-buffer OPS networks.

Parallel to our work, other researchers have presented arguments in favour of reducing network buffer sizes in general.

In [17] it was shown that contrary to the prevalent rule-of-thumb, router buffer size can be scaled down by the square root of the number of TCP flows sharing the bottleneck without loss in performance. Very recently, the authors in [18], [19] have argued that router buffers of size logarithmic in the TCP congestion window size suffice for high throughput provided each TCP sender paces packet injections from its window. Remarkably, this signifies that under certain conditions, as few as 10-20 packet buffers may suffice to achieve close to maximum TCP throughput performance, irrespective of the number of the TCP flows. While we have not undertaken a careful comparison with our work, it would seem their identification of the advantages of TCP pacing lends some support to our proposal of pacing traffic at the optical edge. There are however significant differences to our approaches – while their study considers only TCP traffic, our study aims to benefit simultaneous non-TCP real-time traffic as well. Another difference is that rather than pacing at the end-hosts, we focus on pacing at the edge of the optical network. The former has the advantage that it may require changes only in the end-host TCP implementation. Nevertheless, packet spacing may not be adequately preserved when traffic reaches the core network, particularly if there is a significant volume of bursty real-time traffic sharing links with the TCP traffic. Our approach puts the pacing as close to the short-buffer OPS network as possible, potentially delivering improved performance for all traffic. On the downside, our approach requires dedicated, possibly expensive, high-speed pacing engines.

The rest of this paper is organised as follows: section II illustrates the performance impact of short-buffers, discusses prior work addressing this issue, and outlines our approach of pacing packets. In section III we briefly describe the system setting and recall results on off-line optimal smoothing of video traffic relevant to our work. Section IV develops efficient algorithms for the real-time pacing of arbitrary time-constrained traffic, while section V quantifies via simulation the loss-delay trade-off achievable via pacing. The paper is concluded in section VI.

II. Short Buffers: Impact and Solutions

In this section we first illustrate the impact of short buffers on losses for synthetic trace traffic and throughput for real TCP flows. We then briefly review some prior approaches to improving this performance, and outline our approach to tackling this through packet pacing.

A. Short Buffers and Real-Time Traffic

A direct impact of short network buffers is an increase in packet losses. To illustrate with an example we consider a single link with a queue of finite and small capacity. The link rate is set at 10 Gbps, and packets have a constant size of 1250 bytes (this is consistent with earlier studies of slotted OPS systems). Fig. 1 shows the packet losses as a function of buffer size obtained from simulations of short range (Poisson) as well as long range dependent (LRD) input traffic at various system loads (the traffic model is detailed in section V). The

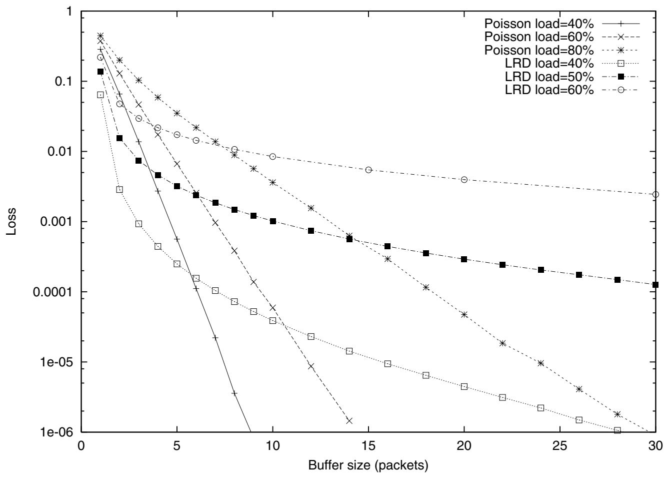

Fig. 1. Loss vs. buffer size at a finite-buffer switch

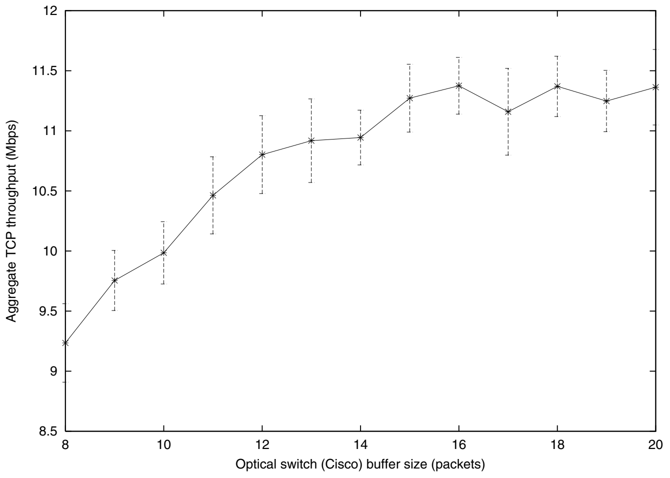

Fig. 2. OPS node (Cisco) buffer size vs. aggregate TCP throughput

plots illustrate that an OPS node with very limited buffering (say 10 to 20 packets) can experience significant losses even at low to moderate traffic loads, particularly with the LRD model which is more representative of real-world traffic. This loss performance may be unacceptable in a core network that supports real-time applications with stringent loss QoS requirements.

B. Short Buffers and TCP Traffic

The majority of Internet traffic today is TCP, and it may be argued that TCP performance is better measured in terms of throughput rather than loss, since TCP relies on loss for detecting and reacting to congestion. In [17] the authors consider buffering at bottleneck links and argue that it can be reduced considerably while maintaining high link utilisation. For our experiment the short-buffer link operates at low to moderate utilisation (as may be expected of high-speed OPS links), and is hence not the bottleneck link. Nevertheless, we show that the losses due to short buffers at these non-bottleneck links can still result in significant degradation of

TCP performance.

Our experiments were run between multiple PCs running Free-BSD at Sydney and Melbourne connected over the CeNTIE network [20], described in more detail in section V. A Cisco 7206 VXR in Sydney acts as an OPS switch, whose buffer size we vary. The Iperf tool was used to generate several (32 for this experiment) TCP flows traversing from Sydney to Melbourne, with a round-trip time of approximately 30 msec. We also use an IXIA traffic generator to create UDP traffic that multiplexes with the TCP flows and creates bottleneck links *prior* to entering the Cisco OPS node. This is because we are interested in the effect that short buffers at *non-bottleneck* high-speed OPS links have on TCP performance. We therefore ensure that the utilisation of the output link of the Cisco OPS node does not exceed 60%. We modify the buffer size on the Cisco OPS node and record the effect on the aggregate TCP throughput for all the flows, plotted in Fig. 2 (the readings for each run were collected for half an hour at one minute intervals, with the standard deviation also plotted in the figure). Our method for varying the buffer size is similar to [17], namely by employing egress link shaping and adapting the token bucket size; a limitation of this approach on the Cisco is that the buffer cannot be reduced to below 8 packets. Even at this buffering, we observe that the TCP throughput is 20% lower than what is achievable with buffering of 16 or more packets at the Cisco OPS node. This indicates that even though OPS networks have sufficient capacity so as not to bottleneck TCP flows, the extremely limited buffering at OPS nodes can significantly degrade TCP throughput.

C. Prior Approaches to Reducing OPS Losses

The problem of reducing packet loss in short buffer optical networks has been addressed by some earlier work. The approach in [21]–[23] is to treat this as a global scheduling problem, wherein packets are transmitted by the optical edge nodes at appropriate time instants that meet the packet's time-constraints while minimising (a weighted measure of) loss in the network. The general problem is shown to be NP-hard [23], and approximate off-line [22], [23] and on-line [21] algorithms are developed for restricted topologies. Though theoretically insightful, these methods require *global* network-wide co-ordinated scheduling amongst the nodes, which is not practically feasible in packet networks.

An observation that emerges from the simulations of real-time and experiments of TCP traffic is that when buffers are very short, losses result even when a few packets arrive back-to-back; in other words, short time-scale burstiness is a major contributor to losses. Typical routers today have sufficient electronic buffers to absorb such short time-scale burstiness, and longer time-scales rate fluctuations are protected against by means of rate-based shaping methods such as leaky-bucket or GCRA. Shaping, however, is unsuitable in the OPS context, since a low shaping rate leads to excessive delays, while larger shaping rates are ineffective in reducing short time-scale burstiness. This has been confirmed by studies in [24] and our own earlier work in [25]. What is therefore required

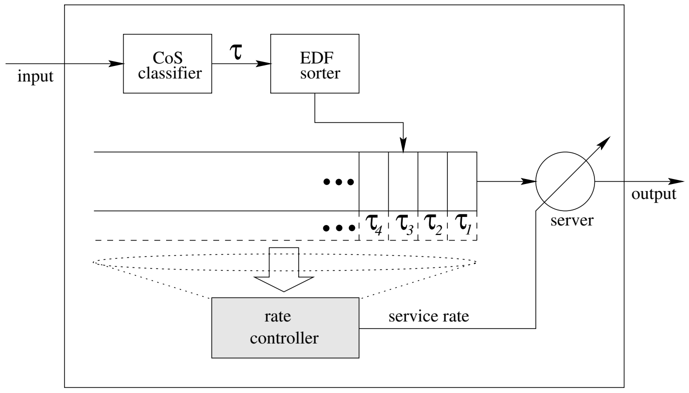

Fig. 3. Model of the pacer

is a means of smoothing traffic at short-time-scales without adversely impacting on end-to-end delays.

D. Packet Pacing

Pacing, also known as smoothing, has been studied before in the context of video traffic transmission. Unlike a shaper, which releases traffic at a given rate, a pacer accepts arbitrary traffic with given delay constraints, and releases traffic that is smoothest subject to the time-constraints of the traffic, where "smoothness" may be measured using the maximum rate, rate variance, etc. The delay tolerance of traffic passing through the pacer is crucial to the efficacy of the pacer – the longer the traffic can be held back at the pacer, the more the window of opportunity the pacer has to smooth traffic and reduce burstiness. A fundamental theoretical contribution in [26] identifies an *optimal* strategy for the off-line smoothing of stored video clips. This has led to several studies on dynamic smoothing of broadcast video streams [27]–[29] (where a few seconds of distribution delay is acceptable) as well as interactive video streams [30] (wherein only a few frames can be buffered at the smoother).

Apart from our preliminary exploration in [31], to the best of our knowledge there has not been a study of pacing in the context of short-buffer OPS networks. In this paper we investigate the applicability of pacing to OPS networks in greater detail, and specifically make three new contributions: (1) we provide additional motivation for pacing by demonstrating the impact of short-buffers at non-bottleneck links on TCP performance, (2) we develop new algorithms of provably low complexity that can perform the pacing in real-time for traffic with arbitrary delay constraints, and (3) we quantify via simulations of a realistic topology the loss-delay tradeoffs that packet pacing facilitates. In the next section we describe the architecture of the pacer and elaborate on the optimal off-line algorithm which provides the basis for our real-time pacing algorithms.

III. System Model and Off-line Optimum

The packet pacer smoothes traffic entering the OPS network, and is therefore employed at the optical edge switches on their egress links connecting to the all-optical packet switching

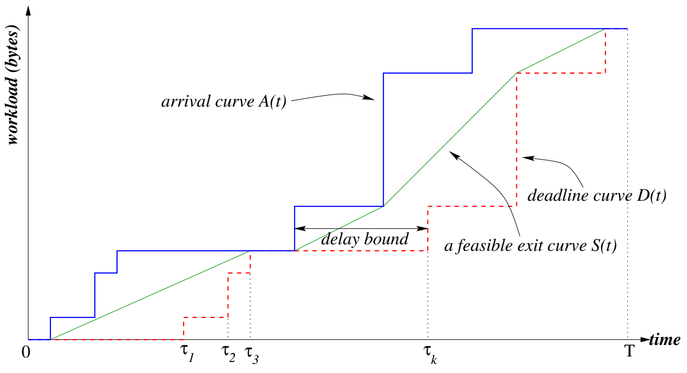

Fig. 4. Arrival, deadline, and exit curves for an example workload process

core. Note that the optical edge switches process packets electronically, and are therefore assumed to have ample buffers required to do the pacing. Once a packet enters the OPS core, is it processed all-optically by each OPS core switch, where buffering is limited. The idea of pacing is therefore to modify the traffic profile entering the OPS network so as to use the limited buffers more efficiently and reduce losses, but without adversely affecting end-to-end delay. As we mentioned in the previous section, rate-based shaping is unsuitable as it does not effectively resolve short-time-scale burstiness, while adversely affecting end-to-end delay. Our pacing method instead smoothes traffic, namely minimises output burstiness, subject to delay constraints. We will show in this paper that this approach is very effective in reducing short-time-scale burstiness, and hence OPS losses, while preserving end-to-end delay performance.

A generic architecture of our pacer is shown in Fig. 3. Incoming packets are classified (according to some criteria) and assigned a deadline by which they are to be released by the pacer. A special case we will consider later is when all packets have identical delay constraints, in which case the architecture can be simplified. The objective of the pacer is to produce the smoothest output traffic such that each packet is released by its deadline. It is natural for the pacer therefore to release packets in order of deadline, namely to implement Earliest Deadline First (EDF) service [32], [33], which has known optimality properties [34] and can be implemented efficiently [35]. However, the pacer, much like a traffic shaper, is non-work-conserving, and in trying to produce a smooth output, behaves as a variable rate server whose rate is modulated by the deadlines of the waiting packets. The challenge is in determining the rate modulation strategy that maximally smoothes the output (discussed in this section) and in implementing this scheme efficiently in real-time at high data rates (the subject of the next section).

Our pacing strategy derives from studies of video traffic smoothing, which we summarise next. Let $[0, T]$ denote the time interval during which the pacing system is considered, chosen such that the system is void of traffic at 0 and T. Denote by $A(t), 0 \leq t \leq T$ the *arrival curve*, namely the cumulative workload (say in units of bytes) arriving in $[0, t)$. Denote by $D(t), 0 \leq t \leq T$ the *deadline curve*, namely the cumulative workload that has to be served in $[0, t)$ so as not to violate any deadlines (thus any traffic with deadline earlier than t contributes to $D(t)$). Fig. 4 depicts an example $A(t)$ and $D(t)$ for the case where all arriving traffic has identical delay requirements. Note that by definition $D(t)$ can never lie above $A(t)$. Any service schedule implemented by the pacer can be represented by an *exit curve* $S(t), 0 \leq t \leq T$, corresponding to the cumulative traffic released by the pacer in $[0, t)$. A feasible exit curve, namely one which is causal and satisfies the delay constraint, must lie in the region bounded above by the arrival curve $A(t)$, and below by the deadline curve $D(t)$.

Amongst all feasible exit curves, the one which corresponds to the smoothest output traffic, measured by various metrics such as transmission rate variance, has been shown in [26] to be the *shortest path* between the origin $(0, 0)$ and the point $(T, D(T))$, as shown in Fig. 4. This curve always comprises a sequence of straight-line segments joining points on the arrival and deadline curves, each segment representing a period during which the service rate is a constant. Computation of this curve requires knowledge of the complete traffic arrival curve, which restricts the approach to off-line applications like the transmission of stored video files. For on-line video transmission applications such as news and sports broadcasts for which delays of seconds to minutes are tolerable, on-line algorithms can be derived from the above off-line optimal by maintaining a time window (i.e. delay buffer) to implement a lookahead capability (see for example [27]–[29]). There has also been some work in smoothing interactive video streams [30] wherein a few frames are buffered at the smoother.

To the best of our knowledge, there has been no prior study (other than our own in [31]) on the applicability of traffic smoothing or pacing to short-buffer OPS networks. Prior studies of video transmission have predominantly considered off-line or time-lagged on-line smoothing, whereas in the OPS context pacing has to be done in real-time for arbitrary input traffic. Further, unlike video applications where one or at most a few streams are smoothed at end-hosts or video servers, the OPS edge nodes have to smooth traffic aggregates at very high data rates. In the next section we develop algorithms that optimally pace high-rate traffic in real-time and are amenable to efficient implementation at OPS edge nodes.

IV. Efficient Real-Time Pacing

It is shown in [26] that an off-line pacer yields the smoothest output traffic satisfying the delay constraints if its service rate follows the shortest path lying between the arrival and deadline curves. In the on-line case, however, the packet arrival process is non-deterministic, and the arrival curve is not known beforehand. In the absence of any assumptions about future packet arrivals, our on-line algorithm determines the smoothest output for the packets *currently* in the pacer. Thus at time t, the arrival curve considered to the right of t is a horizontal line (since future arrivals are not known yet), and the shortest-path exit curve degenerates to the convex hull of the deadline curve [31]. Upon each packet arrival, the deadline

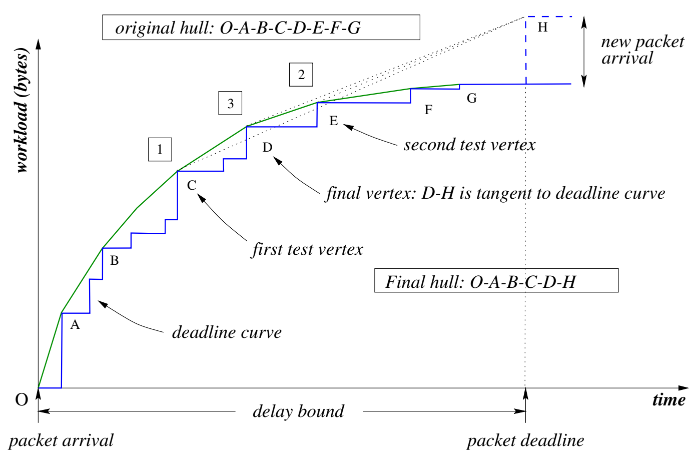

Fig. 5. Example showing single-class hull update

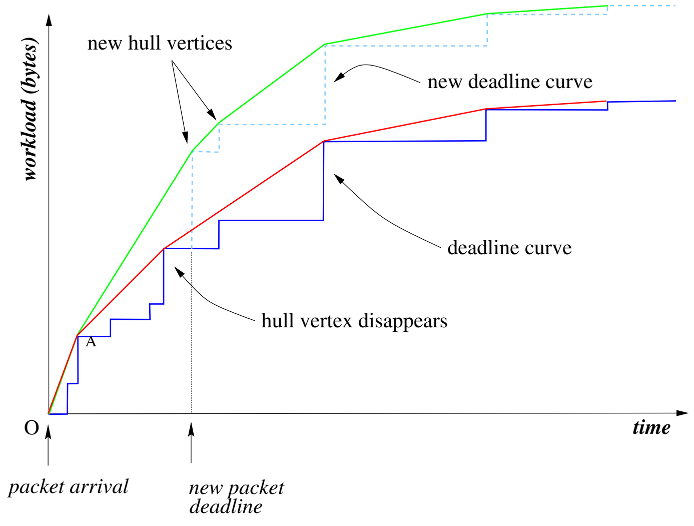

Fig. 7. Example illustrating that an arbitrary number of new hull points may appear when the arriving packet has an arbitrary deadline

curve is augmented, and this may require a recomputation of the convex hull which defines the optimal exit curve. This section develops algorithms for the efficient update of the convex hull of the deadline curve upon each packet arrival.

A. Single Delay Class

We first consider the case where all packets entering the pacer have identical delay constraints. This simplies the hull update algorithm since each packet arrival augments the deadline curve at the end. This is illustrated with an example in Fig. 5. Starting with the original convex hull O-A-B-C-D-E-F-G, a new packet arrival at time 0 adds a new point H to the deadline curve. From H, we find a tangent to the original convex hull by doing a binary search on the hull segment slopes. Operation $\boxed{1}$ examines the mid-point C of the hull, realises that H-C lies below the original hull, and so moves right. In operation $\boxed{2}$ the mid-point E of the right half is examined, and it is found that EH lies above the original hull, so the algorithm moves left, till it reaches point D in operation $\boxed{3}$ that gives the desired tangent and final hull O-A-B-C-D-H.

1) *Determine size and deadline of newly arrived packet and create new vertex u.*
2) *Search in the* AVL *tree for the tangent point r from vertex u to the convex hull.*
3) *Delete all vertices with time larger than that of r, insert vertex u, and rebalance the* AVL *tree.*

Fig. 6. Single-class hull update algorithm

Fig. 6 depicts the update algorithm more formally. Recalling that the deadline curve is a piecewise-linear curve, where the start/end times of its individual segments correspond to arrival of new packets as illustrated in Fig. 5, we can represent it as a planar polygonal line whose vertices $v_0, v_1, \ldots v_n$ are in increasing order with respect to both axes. The vertices are stored in a height-balanced search tree T structure, the *AVL* tree for example, with the value of time used as the search key. Along with each vertex we also store pointers to its predecessor and successor on the boundary of its convex hull.

In step 1 of the algorithm, the size of the incoming packet is determined, along with its deadline, and a new vertex u is created. The arrival of the new packet, which causes the deadline curve to be amended, results in appending a new vertex to the *end* of the hull. The new vertex may cause it to lose convexity, since the newly added segment may have slope larger than that of the preceding hull edge. Step 2 therefore searches the tree T for the unique vertex r such that slope of the segment connecting r and u is smaller than that of the preceding hull edge but larger than that of the following hull edge. The search process is a binary search of the *AVL* tree, as described by Preparata [36, procedure TANGENT]. At this stage the hull representation is restructured in step 3 by removing all vertices with value of time larger than that of r, inserting the new vertex u, and rebalancing the tree T, again as described in [36].

The complexity of the above convex hull update operation that need to be performed upon the arrival of each packet has $O(\log n)$ cost, where n is the number of queued packets [36]. Once the pacer has released packets corresponding to a segment of the hull, the segment needs to be deleted. The complexity of deleting a vertex of the convex hull and restructuring the tree T also has $O(\log n)$ cost, where n is the number of queued packets, as shown in [36].

B. Multiple Delay Classes

We now consider the general case where arriving packets may have arbitrary delay constraints. Handling packets with different delay times is complicated by the fact that the arrival of a new packet causes significant changes to the deadline curve. Recalling that the deadline curve is a piecewise-linear curve, where the start/end times of its individual segments correspond to deadlines of packets already in the system, we represent it as a planar polygonal line whose vertices

$v_0, v_1, \ldots v_n$ form a sequence in increasing order with respect to both axes. The arrival of the new packet with a deadline between two existing vertices, say v_i and v_{i+1}, changes the deadline curve through the insertion of a new vertex u between them in the sequence and raising each of the vertices in the sub-sequence $v_{i+1}, v_{i+2}, \ldots, v_n$ by a value corresponding to the size of the new packet. As a result some vertices of the deadline curve, which were not part of the hull prior to arrival of the new packet, may appear as convex hull vertices as illustrated in Fig. 7. The number of such new points can be as high as the number of packets in the system, and the process of re-computing the convex hull is not as simple as searching a binary tree as we did in the single-class case above.

The idea behind the algorithm is that for an incoming packet with arbitrary deadline, the original deadline curve is split into two parts, corresponding to the left and right of the new arrival's deadline. The convex hulls for each of the parts is independently computed, after the deadline curve to the right has been shifted up to account for the new packet arrival. The two hulls are then merged back to get the complete convex hull. The goal is to perform this process as efficiently as possible.

1) *Determine size and deadline of newly arrived packet and create new vertex u.*
2) *Insert u into the 2-3 tree and divide it into trees T_L and T_R. T_L holds keys $\leq$ that of u, and T_R the remaining. Store size of new packet in root of T_R.*
3) *Merge T_L and T_R into a single tree.*

Fig. 8. Multi-class hull update algorithm

Fig. 8 depicts our algorithm for determining the convex hull upon each packet arrival. Our vertices are stored in the leaves of a search tree *T* structure which is capable of supporting concatenable-queue operations, such as the 2-3 tree [37, sections 4.12], with the value of time used as the search key. Each internal node of *T* stores the convex hull of its leaves in a secondary tree structure that is also capable of supporting concatenable-queue operations. A linear size of the tree *T* and all its secondary structures is achieved by storing a vertex in the convex hull of an internal node only if it is not stored in any of its ancestor nodes [38].

In step 1 of the algorithm, the size of the incoming packet is determined, along with its deadline, and a new vertex u is created. The arrival of the new packet, which causes the deadline curve to be altered, triggers re-computation of the convex hull. Step 2 therefore searches the tree *T* along the root-to-leaf path and inserts the new vertex u as a new leaf according to its deadline value. The tree *T* is then *divided* about u so that all the leaves to the left of u and u itself are in one 2-3 tree T_L and all the leaves to the right of u are in a second 2-3 tree T_R. The division is a recursive process detailed in [37, section 4.12]. For each internal node visited during the search process that will be deleted in the divide process, we use the convex hull stored in its secondary structure to compute a complete hull for each of its children, as described in [38]. At the end of the divide process, values of all vertices in T_R need to be incremented by the size of the incoming packet (i.e. shift the deadline curve up); this is achieved by storing the size of the new packet in the root of T_R. In step 3, the two trees T_L and T_R are again concatenated into a single tree, which yields the final convex hull of the new deadline curve. The complexity of the entire convex hull update operation that need to be performed upon each packet arrival is $O(\log^2 n)$, where n is the number of queued packets [38].

As in the single-class case, a hull segment needs to be deleted once packets corresponding to the segment have been released. The complexity of deleting a vertex of the convex hull and restructuring the tree *T* again has an $O(\log^2 n)$ cost, where n is the number of queued packets, as shown in [38].

V. Performance Study

Having shown that pacing can be implemented efficiently at high speeds, we now show that pacing can be very effective in improving loss performance, at the expense of a small and bounded increase in end-to-end delays. In this section we use simulations to show the performance improvement at a single switch with short-buffers and in an OPS network based on a real topology.

The traffic model we use for our simulations is based on long range dependent (LRD) generators derived from Norros' self-similar traffic model [39]. This model combines a constant mean arrival rate with fractional Gaussian noise (fGn) characterised by zero mean, variance σ^2 and Hurst parameter $H \in [1/2, 1)$. We use our filtering method [40], related to the FFT-based methods described in [41], [42], to generate, for a chosen H, a sequence $\{x_i\}$ of normalised fGn (zero mean and unit variance). A discretisation interval Δt is chosen, and each x_i then denotes the amount of traffic, in addition to the constant rate stream, that arrives in the i-th interval. Specifically, the traffic y_i (in bits) arriving in the i-th interval of length Δt seconds is computed using:

$$y_i = max\{0, \rho_c \Delta t + s x_i\}$$

where ρ_c denotes in bits-per-second the constant rate stream, and s is a scaling factor that determines the instantaneous burstiness. This truncated Gaussian nature of y_i implies that the resulting traffic rate $\rho = E[y_i]/\Delta t$ is different from ρ_c. Specifically, they satisfy the relation:

$$\frac{\rho \Delta t}{s} = \frac{1}{\sqrt{2\pi}} e^{-(\rho_c \Delta t/s)^2/2} + \frac{\rho_c \Delta t}{s} Q(-\rho_c \Delta t/s)$$

where $Q(X) = P\{x > X\}$ denotes the complementary cumulative density function of the standard Gaussian random variable x.

For this work we set the Hurst parameter at $H = 0.85$ and the discretisation interval $\Delta t = 1.0\mu s$. The scaling factor s is chosen to satisfy $\rho_c \Delta t/s = 1.0$, which corresponds to moderate burstiness (around 16% of the samples are truncated), and ρ_c is then adjusted to give the desired mean traffic rate. The fluid traffic is then packetised into fixed-length packets (of size 1250 bytes) before being fed to the simulations.

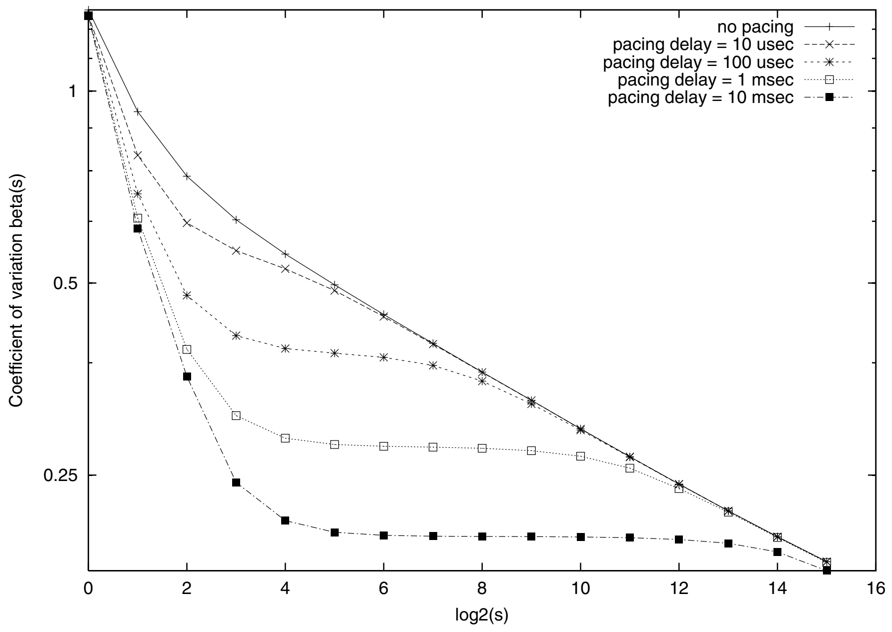

Fig. 9. Burstiness $\beta(s)$ vs. time-scale s with pacing for LRD traffic

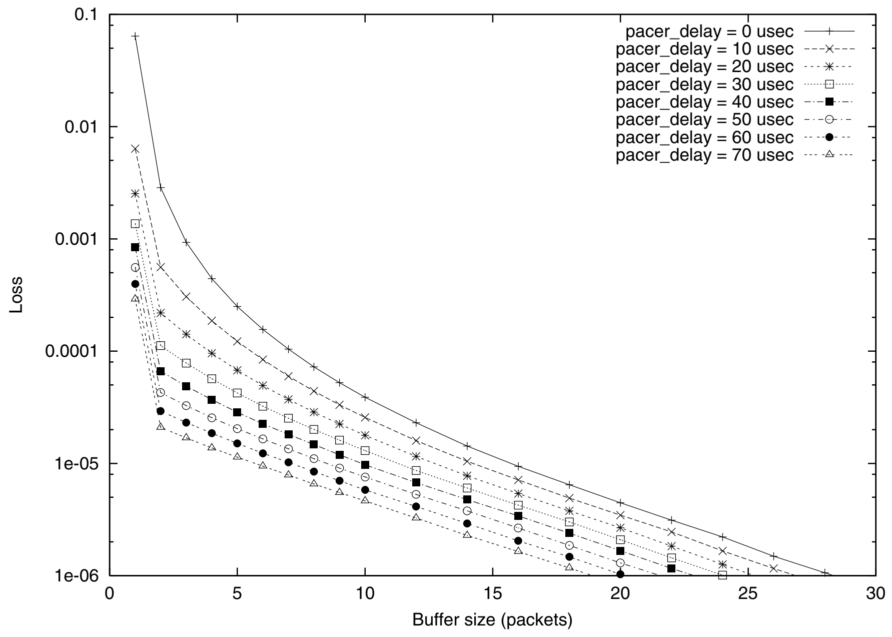

Fig. 10. Loss vs. buffer size with pacing for LRD traffic

A. Single Node

In this scenario we consider again the single link with short buffer studied in section II-A, and introduce our pacer between the traffic source and the link queue. We first study the effect of pacing on burstiness, quantified by the coefficient of variation $\beta(s)$ of the traffic volume measured over time interval of length s. Log-log plots of $\beta(s)$ versus s are routinely used to indicate self-similarity of traffic traces and to show the influence of the Hurst parameter H. Fig. 9 plots burstiness on log scale for LRD input traffic at 40% loading. The slope of $-(1-H) = -0.15$ validates the Hurst parameter setting of 0.85 (the different slope at very short time-scales is explained by the discretization of the ideal fluid model required to generate packets). The figure also shows burstiness of traffic paced using our method described in the previous section for pacing delay budgets of 1μsec, 10μsec, 100μsec, 1msec, and 10msec. As expected, at short time-scales pacing dramatically reduces burstiness, and this benefit increases with the pacing delay budget. In fact, at short time-scales, the burstiness of paced LRD traffic seems to fall at slope 0.5, indicative of short range dependent behaviour. However, as time-scales increase, the burstiness levels off, till it converges back to that of the input LRD traffic, since pacing ceases to be effective at time-scales larger than the delay "window" available to the pacer. This plot indicates that pacing is quite effective in reducing short time-scale burstiness but does not impact the long time-scale characteristics of the traffic.

In Fig. 10 we plot the total losses at the single-link of finite and small buffer capacity. Various per-packet delay bounds are considered at the pacer, measured in μsec (recall that 1μsec also corresponds to the transmission time of our 1250 byte packet at the link rate of 10 Gbps). A pacer wherein all packets have delay budget 0 produces output identical to its input, and the resulting loss curve is therefore identical to the corresponding curve in Fig. 1. As the delay budget at the pacer is increased, the pacer becomes more effective in eliminating the bunching of packets being sent to the small-buffer node, and this helps reduce losses. In this example, for buffer size of 10, an extra 70μsec in end-to-end delay introduced by the pacer helps reduce loss by more than an order of magnitude – a considerable cost-benefit advantage. The losses can be reduced even further if the traffic can tolerate larger delays at the pacer. However, if the buffer size were to get much larger, paced traffic would not have a significant advantage over unpaced traffic, since losses from large buffers are a result of longer time-scale burstiness which is not alleviated by pacing.

B. *CeNTIE Network*

We now quantify the impact of pacing via simulations of an OPS network with topology derived from a real-world network, transporting realistic traffic flows carrying LRD traffic. Fig. 11 shows part of the CeNTIE network, a trans-continental Australian research network [20] with MANs in Sydney, Canberra, Melbourne, and Perth. It includes end-user research groups from the health, education, film post-production, and finance industries. We simulate a subset of the CeNTIE network, with logical topology and fibre lengths shown in Fig. 12. There are 4 core switches in the chosen topology – two in Sydney, at CSIRO-Marsfield and the University of Technology, Sydney (UTS), and one each in Melbourne and Perth. There are four edge switches connected to the Marsfield core switch - one each at CSIRO-Marsfield, CSIRO-Riverside, MacQuarie University (MQU), and the Royal North Shore Hospital (RNSH). The UTS core switch is connected to three edge switches – the Conservatorium of Music (Con), University of New South Wales (UNSW), and Nepean Hospital. At Melbourne, there is an edge switch at the University of Melbourne (UMel), while Perth has two edge switches, at the Australian Resource Research Centre (CSIRO-AARC) and the University of Western Australia (UWA). All the above mentioned sites are currently live on the CeNTIE network.

We simulate the CeNTIE network as if it were an OPS network. Numerous architectural options exist for OPS networks – slotted versus unslotted, space-switching versus broadcast-and-select fabrics, feed-forward versus feed-back FDLs, etc.

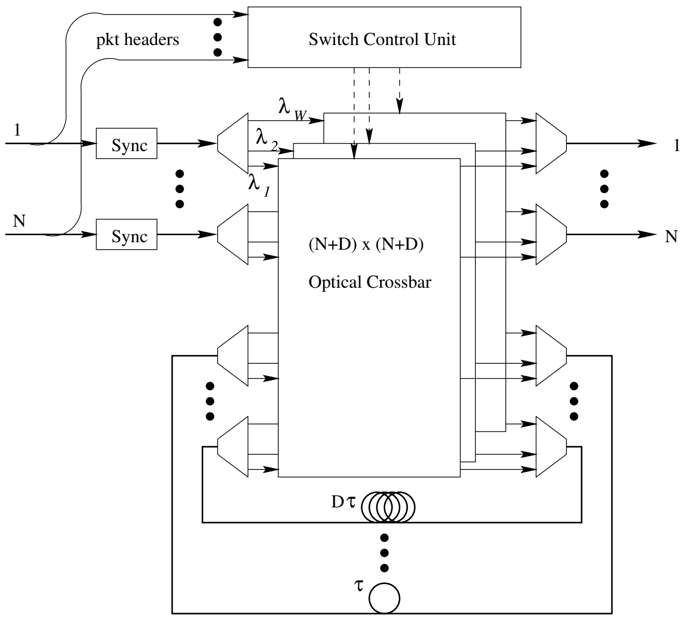

Fig. 13. OPS node architecture

Since short packet buffering capability is a problem common to all the above, we illustrate its effect in a relatively simple architecture, namely a slotted and synchronous system [6] transporting fixed length packets. Fig. 13 depicts the shared-memory core optical packet switch [4], [12] architecture used in this work. On each of the input fibres, an optical splitter splits a small amount of power from the incoming packets and sends it to the control unit. The control unit extracts timing information (used for configuring the synchronisation stages) and packet header information (used for determining the packet route and configuring the optical crossbar accordingly). Input signals are synchronised to align packets to slot boundaries, and demultiplexed into the component wavelengths. We assume that wavelength converters are not employed, and each wavelength traverses its own switching plane. Output port contentions are resolved using a set of D FDL buffers of increasing length that provide delays of $1, 2, \ldots, D$ slots across all wavelengths.

Though time-slotted OPS systems require complex hardware to perform synchronization, the scheduling algorithms are significantly simpler than for asynchronous systems transporting variable length packets [43]. We choose our slot size to be 1μsec, which, at bandwidth per wavelength of 10Gbps, carries an optical packet of size 1250 bytes. This slot size is commensurate with studies in the literature [44], and is also consistent with current optical crossbar technology such as [45].

For the simulations, we selected eight traffic flows typical of the usage of the CeNTIE network. These flows are depicted in Fig. 12, and Table I shows their characteristics including the type of traffic and the number of core-links traversed. The diversity in hop-lengths and end-to-end propagation delays give a representative sampling of traffic flows in the CeNTIE

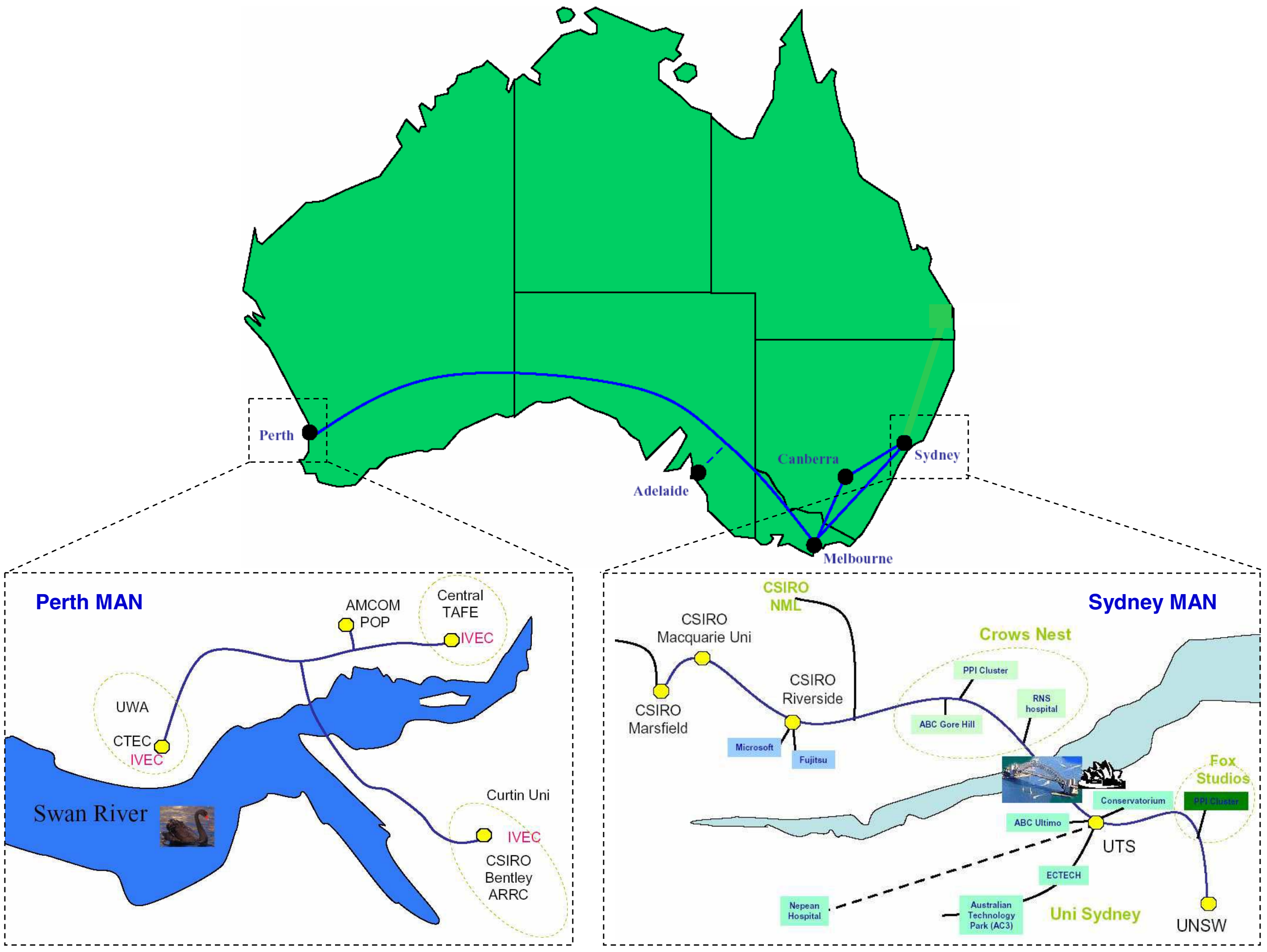

Fig. 11. CeNTIE Network

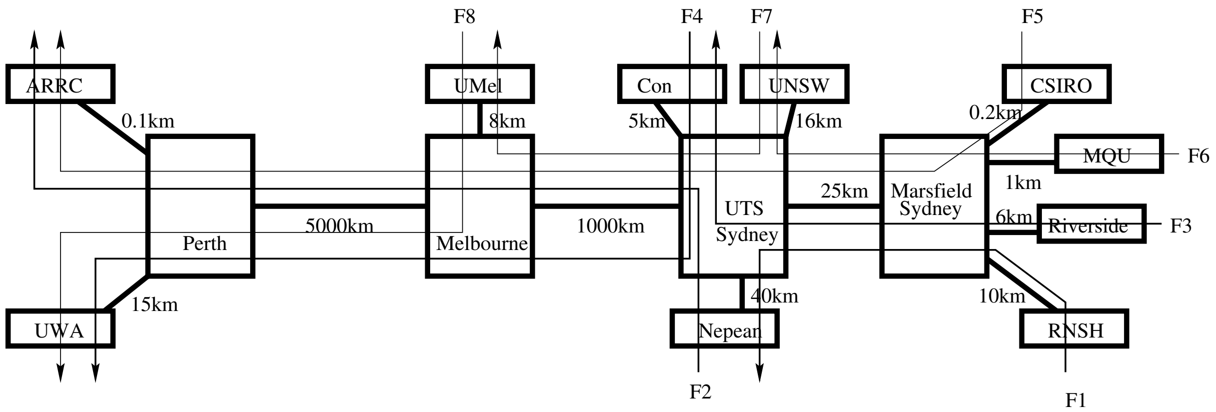

Fig. 12. CeNTIE Simulation Topology and Flows

network. Flows $F1$-$F4$ carry time-critical traffic, and so we choose to not subject them to pacing. Flows $F5$-$F8$ carrying intranet and university traffic are not considered to be very time-critical, and are hence subject to pacing at the OPS edge.

Though our work does not preclude the use of wavelength converters, our simulations assume that they are not available. Consequently, traffic from different wavelengths are independent and do not interact, and it suffices to simulate a single wavelength. Also, our traffic flows are unidirectional; the study of closed-loop traffic is left for the future. All links operate at 10Gbps. Each flow generates LRD traffic at a mean rate of 1.5 Gbps. Each of the three core links carries 4 flows, and is thus loaded at 6 Gbps or 60% of link capacity. We believe such a loading scenario is realistic.

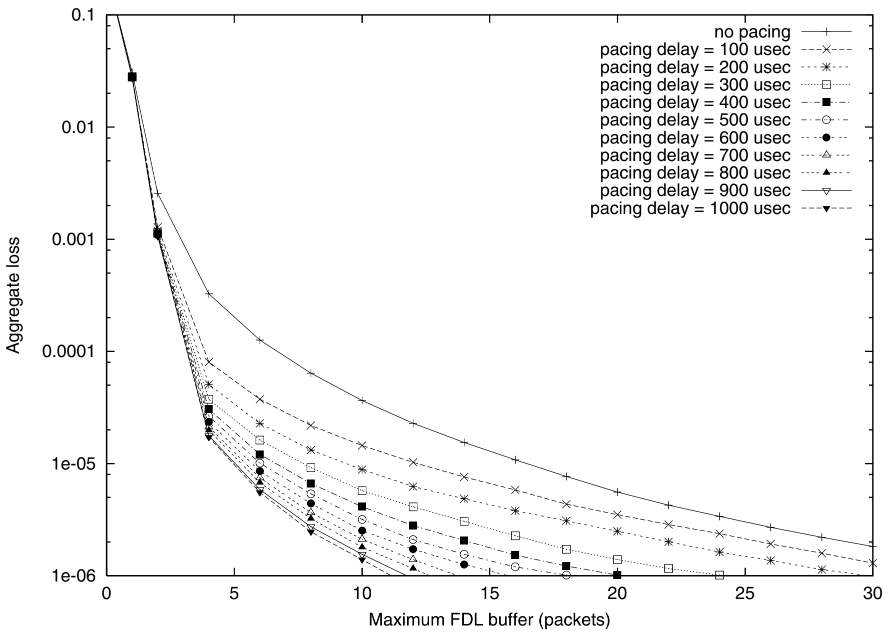

Fig. 14. Loss vs. maximum FDL buffer for the CeNTIE network

TABLE I
CeNTIE Network Flows Simulated

flow#	src → dest	traffic type	hops
$F1$	RNSH → Nepean	medical	1
$F2$	Nepean → ARRC	management	2
$F3$	Riverside → Conservatorium	functions	1
$F4$	Conservatorium → UWA	music collab	2
$F5$	CSIRO-Marsfield → ARRC	intranet	3
$F6$	MQU → UNSW	university	1
$F7$	UNSW → UMel	university	1
$F8$	UMel → UWA	university	1

Fig. 14 plots the aggregate network losses against the maximum FDL buffer size at each OPS node for various delay constraints at the pacer for packets of the non-time-critical flows $F5$-$F8$. Note first that in the absence of any buffers at the OPS nodes, pacing has no effect on loss rates. Though counter-intuitive, this is because loss in a bufferless slotted system depends only on how many input lines have a packet destined for the same output line in a particular slot, and this is invariant to pacing. In a slotted system, pacing helps reduce burstiness in terms of variation in the number of packets carried in *groups* of slots, it can not change the statistics at the granularity of a single slot.

For maximum FDL buffer size in the range of 10-20 packets, it is seen that pacing the non-time-critical flows by introducing a few hundred μsec of delay reduces losses in the OPS network by more than an order of magnitude. For a cross-continental network with multiple tens of msec in propagation delay, an additional delay of less than a msec is negligible, but allows substantial improvement in loss performance. This ability to trade-off a slight increase in end-to-end delay for a substantial reduction in loss, and the ability to explicitly control this trade-off, could make pacing a very useful mechanism in short buffer OPS networks.

VI. Conclusions

Emerging optical packet switched (OPS) networks will likely have very limited buffering capability, possibly no more than a few tens of packets. This can adversely impact end-to-end performance, causing high losses for real-time traffic and reduced throughput for TCP flows. Since short-time-scale burstiness is the major contributor to the performance degradation, we proposed the "pacing" of traffic at the optical edge prior to injection into the OPS core. Pacing reduces the traffic burstiness for a bounded and controllable penalty in end-to-end delay. We developed algorithms of poly-logarithmic complexity that can efficiently implement optimal real-time pacing of traffic aggregates with arbitrary delay requirements. Through simulation of a realistic network, we showed that pacing can reduce losses in short-buffer OPS networks by more than an order of magnitude, at the expense of a small increase in end-to-end delay. This ability to trade-off delay for loss points to a feasible way of realising acceptable performance from short-buffer OPS networks.

Our future work targets a deeper study of TCP performance over short-buffer networks, particularly when it multiplexes with real-time traffic. It would also be interesting to undertake a comparison of our approach of pacing traffic at the optical edge to the approach in [18], [19] of pacing TCP at end hosts.

VII. Acknowledgements

The authors would like to thank Craig Russell of CSIRO, together with Mohit Mundhra and Amit Kumar of the Indian

Institute of Technology for their help in setting up the TCP experiments in this paper.

References

[1] R. Ramaswami and K. N. Sivarajan, *Optical Networks, A Practical Perspective*, 2nd ed. Morgan Kaufmann, 2002.

[2] C. Qiao and M. Yoo, "Optical burst switching (OBS) – A new paradigm for an optical Internet," *J. of High Speed Networks*, vol. 8, no. 1, pp. 69–84, 1999.

[3] I. Chlamtac, V. Elek, A. Fumagalli, and C. Szabo, "Scalable WDM access network architecture based on photonic slot routing," *IEEE/ACM Trans. Networking*, vol. 7, no. 1, pp. 1–9, Feb 1999.

[4] S. Yao, S. Dixit, and B. Mukherjee, "Advances in photonic packet switching: an overview," *IEEE Comm. Magazine*, vol. 38, no. 2, pp. 84–94, Feb 2000.

[5] A. Carena *et al.*, "OPERA: An optical packet experimental routing architecture with label swapping capability," *J. Lightwave Tech.*, vol. 16, no. 12, pp. 2135–2145, Dec 1998.

[6] C. Guillemot *et al.*, "Transparent optical packet switching: The European ACTS KEOPS project approach," *J. Lightwave Tech.*, vol. 16, no. 12, pp. 2117–2134, Dec 1998.

[7] D. Hunter *et al.*, "WASPNET: A wavelength switched packet network," *IEEE Comm. Magazine*, vol. 37, no. 3, pp. 120–129, Mar 1999.

[8] D. Wonglumson *et al.*, "HORNET: A packet switched WDM network: Optical packet transmission and recovery," *IEEE Photonics Tech. Letters*, vol. 11, no. 12, pp. 1692–1694, Dec 1999.

[9] L. Dittmann *et al.*, "The European IST project DAVID: A viable approach toward optical packet switching," *IEEE J. Selected Areas in Comm.*, vol. 21, no. 7, pp. 1026–1040, Sep 2003.

[10] H. Park, E. F. Burmeister, S. Bjorlin, and J. E. Bowers, "40-Gb/s optical buffer design and simulations," in *Numerical Simulation of Optoelectronic Devices (NUSOD)*, Santa Barbara, CA, Aug 2004.

[11] D. Hunter, M. Chia, and I. Andonovic, "Buffering in optical packet switches," *J. Lightwave Tech.*, vol. 16, no. 12, pp. 2081–2094, Dec 1998.

[12] M. J. Karol, "A shared-memory optical packet (ATM) switch," in *Proc. 6th IEEE Wksp. Local and Metro Area Networks*, 1993, pp. 205–211.

[13] S. L. Danielsen, P. B. Hansen, and K. E. Stubkjaer, "Wavelength conversion in optical packet switching," *J. Lightwave Tech.*, vol. 16, no. 12, pp. 2095–2108, Dec 1998.

[14] V. Eramo and M. Listani, "Packet loss in a bufferless optical WDM switch employing shared tunable wavelength converters," *J. Lightwave Tech.*, vol. 18, no. 12, pp. 1818–1833, Dec 2000.

[15] F. Forghierri, A. Bononi, and P. R. Prucnal, "Analysis and comparison of hot-potato and single-buffer deflection routing in very high bit rate optical mesh networks," *IEEE Trans. Commun.*, vol. 43, no. 1, pp. 88–98, Jan 1995.

[16] S. Yao, B. Mukherjee, S. J. B. Yoo, and S. Dixit, "A unified study of contention-resolution schemes in optical packet-switched networks," *J. Lightwave Tech.*, vol. 21, no. 3, pp. 672–683, Mar 2003.

[17] G. Appenzeller, I. Keslassy, and N. McKeown, "Sizing router buffers," in *Proceedings of SIGCOMM 2004*, Portland, OR, Aug-Sep 2004.

[18] M. Enachescu, Y. Ganjali, A. Goel, N. McKeown, and T. Roughgarden, "Part III: Routers with very small buffers," *ACM SIGCOMM Computer Communications Review*, vol. 35, no. 3, pp. 83–90, July 2005.

[19] ——, "Routers with very small buffers," in *to appear in the Proceedings of IEEE Infocom*, Barcelona, Spain, Apr 2006.

[20] T. Percival, "An introduction to CeNTIE," Presentation, *http://www.centie.org/docs/CeNTIE-web-intro.ppt*.

[21] J. Naor, A. Rosen, and G. Scalosub, "Online time-constrained scheduling in linear networks," in *Proceedings of INFOCOM 2005*, Miami, FL, Mar 2005.

[22] M. Adler, S. Khanna, R. Rajaraman, and A. Rosen, "Time-constrained scheduling of weighted packets on trees and meshes," *Algorithmica*, vol. 36, no. 2, pp. 123–152, 2003.

[23] M. Adler, A. L. Rosenberg, R. K. Sitaram, and W. Unger, "Scheduling time-constrained communication in linear networks," *Theoretical Comp. Sc.*, vol. 35, no. 6, pp. 599–623, 2002.

[24] H. Elbiaze and T. Atmaca, "Traffic management in multi-service optical network," in *Proceedings of IEEE ICN 2001*, Colmar, France, Jul 2001.

[25] V. Sivaraman, D. Moreland, and D. Ostry, "Ingress traffic conditioning in slotted optical packet switched networks," in *ATNAC 2004*, Sydney, Australia, Dec 2004.

[26] J. D. Salehi, Z.-L. Zhang, J. Kurose, and D. Towsley, "Supporting stored video: Reducing rate variability and end-to-end resource requirements through optimal smoothing," *IEEE/ACM Transactions on Networking*, vol. 6, no. 4, pp. 397–410, August 1998.

[27] S. Sen, J. Rexford, J. Dey, J. Kurose, and D. Towsley, "Online smoothing of variable-bit-rate streaming video," *IEEE Trans. Multimedia*, vol. 2, no. 1, pp. 37–48, Mar 2000.

[28] R. Chang, M. Chen, J. Ho, and M. Ko, "An effective and efficient traffic-smoothing scheme for delivery of online VBR media streams," in *IEEE Infocom*, New York, NY, Mar 1999, pp. 447–454.

[29] G. Cao, W. Feng, and M. Singhal, "Online variable-bit-rate video traffic smoothing," *Computer Communication*, vol. 26, no. 7, pp. 639–651, 2003.

[30] Y. Mansour, B. Patt-Shamir, and O. Lapid, "Optimal smoothing schedules for real-time streams," in *ACM Symposium on Principles of Distributed Computing*, Portland, OR, 2000, pp. 21–29.

[31] V. Sivaraman, D. Moreland, and D. Ostry, "A novel delay-bounded traffic conditioner for optical edge switches," in *IEEE HPSR 2005*, Hong Kong, May 2005.

[32] D. Ferrari and D. Verma, "A scheme for real-time channel establishment in wide-area networks," *IEEE Jrnl. Selected Areas in Comm.*, vol. 8, no. 3, pp. 368–379, Apr 1990.

[33] D. Verma, H. Zhang, and D. Ferrari, "Guaranteeing delay jitter bounds in packet switching networks," in *Proc. TRICOMM*, Chapel Hill, NC, Apr 1991, pp. 35–46.

[34] L. Georgiadis, R. Guérin, and A. Parekh, "Optimal multiplexing on a single link: delay and buffer requirements," *IEEE Trans. Inf. Theory*, vol. 43, no. 5, pp. 1518–1535, Sep 1997.

[35] V. Sivaraman, "End-to-end delay service in high speed networks using earliest deadline first scheduling," Ph.D. dissertation, University of California, Los Angeles, March 2000.

[36] F. Preparata, "An optimal real-time algorithm for planar convex hull," *Communications of the ACM*, vol. 22, pp. 402–405, 1979.

[37] A. Aho, J. E. Hopcroft, and J. D. Ullman, *The Design and Analysis of Computer Algorithms*. Addison-Wesley Publishing Company, 1975.

[38] M. H. Overmars and J. van Leeuwan, "Maintenance of configuration in the plane," *J. Computer and System Sciences*, vol. 23, pp. 166–204, 1981.

[39] I. Norros, "On the use of fractional brownian motion in the theory of connectionless traffic," *IEEE J. Selected Areas in Comm.*, vol. 13, no. 6, pp. 953–962, Aug 1995.

[40] D. Ostry, "Synthesis of accurate fractional Gaussian noise by filtering," *to appear in IEEE Trans. Information Theory*, 2006.

[41] A. T. A. Wood and G. Chan, "Simulation of stationary Gaussian processes in $[0,1]^d$," *J. Computational and Graphical Statistics*, vol. 3, no. 4, pp. 409–432, Dec 1994.

[42] C. R. Dietrich and G. N. Newsam, "Fast and exact simulation of stationary Gaussian processes through circulant embedding of the covariance matrix," *SIAM J. Scientific Computing*, vol. 18, no. 4, pp. 1088–1107, July 1997.

[43] L. Tančevski *et al.*, "Optical routing of asynchronous, variable length packets," *IEEE J. Selected Areas in Comm.*, vol. 18, no. 10, pp. 2084–2093, Oct 2000.

[44] D. Careglio, J. Pareta, and S. Spadaro, "Optical slot dimensioning in IP/MPLS over OPS networks," in *Proc. WOAN 2003.*, Zagreb, Croatia, Jun 2003.

[45] T. McDermott and T. Brewer, "Large-scale IP router using a high-speed optical switch element," *J. Optical Networking*, vol. 2, no. 7, pp. 229–240, Jul 2003.