Packet Pacing in Small Buffer Optical Packet Switched Networks

Vijay Sivaraman, Hossam Elgindy, David Moreland, and Diethelm Ostry

***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. To mitigate the performance degradation resulting from small buffers, this paper proposes that optical edge nodes “pace” the injection of traffic into the OPS core. Our contributions relating to pacing in OPS networks are three-fold: first, we develop real-time pacing algorithms of poly-logarithmic complexity that are feasible for practical implementation in emerging high-speed OPS networks. Second, we provide an analytical quantification of the benefits of pacing in reducing traffic burstiness and traffic loss at a link with very small buffers. Third, we show via simulations of realistic network topologies that pacing can significantly reduce network losses at the expense of a small and bounded increase in end-to-end delay for real-time traffic flows. We argue that the loss-delay trade-off mechanism provided by pacing can be instrumental in overcoming the performance hurdle arising from the scarcity of buffers in OPS networks.**

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 [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. Today’s electronic switches resolve this contention relatively easily by using electronic random-access memory (RAM) that can store over a million packets. By comparison, a state-of-the-art random-access optical buffer available on an integrated opto-electronic chip [10] can hold at most a few dozen packets. Alternatively, spools of fibre can implement fibre delay lines (FDLs) [11] that provide optical buffering capability. Unfortunately, the high speed of light means that a significant buffering capability would necessitate large fibre spools too unwieldy to be practical: 1 km of fibre stores 5μsec of optical data; by contrast a conventional router today typically buffers 50-250msec of electronic data. Moreover, incorporating FDLs into a typical OPS switch design requires larger optical crossbars, which can add significantly to cost as the FDL buffers increase. Wavelength conversion [12], [13] is another technique for resolving contentions in the optical domain, whereby a contending packet is converted to an unused wavelength on the same outgoing link. However, wavelength converters are expensive, and often limited in their conversion range. Other methods such as deflection routing [14] and combinational schemes [15] have also been investigated for alleviating contentions, but usually incur overheads such as packet reordering, complexity, etc. It therefore seems that OPS networks of the foreseeable future will have very limited contention resolution resources.

Our objective in this paper is to investigate the impact of limited contention resolution resources (henceforth “small buffers”) on the end-to-end performance of real-time traffic in an OPS network, and to develop means of managing the performance degradation. We begin by observing that not withstanding the high bandwidth available in OPS networks, small buffers at OPS nodes cause significant loss when the traffic exhibits short-time-scale burstiness. To alleviate this, we advocate that traffic characteristics be modified before injection into the OPS network, with the aim of using the limited contention resolution resources more effectively. The mechanism to achieve this, termed “pacing”, reduces the short-time-scale burstiness of arbitrary traffic, but without incurring significant delay penalties. Our contributions in this context are three-fold. We first develop algorithms that perform efficient real-time optimal pacing of high data rate traffic. Our algorithms vary in complexity between amortised constant time and poly-logarithmic time in the number of queued packets, and are shown to be amenable to efficient hardware implementation. Our second contribution develops a novel analytic framework to quantify the impact of pacing on a short or long range dependent traffic stream. It lets us estimate, for various pacer delay budgets, the burstiness of the output traffic stream and associated loss at a link with very small buffers. The analytical estimates match well with simulation and provide insight into the operation of the pacer. For our final contribution, we demonstrate via simulation of realistic OPS network topologies derived from operational networks

Vijay Sivaraman is with the School of Electrical Engineering and Telecommunications, University of New South Wales, Sydney, NSW 2052, Australia. Email: Vijay.Sivaraman@ieee.org. Ph: +61 2 9385 6577. Fax: +61 2 9385 5993.

Hossam Elgindy is with the School of Computer Science and Engineering, University of New South Wales, Sydney, NSW 2052, Australia. Email: elgindyh@cse.unsw.edu.au

David Moreland and Diethelm Ostry are with the CSIRO ICT Centre, PO Box 76, Epping, NSW 1710, Australia. Emails: {David.Moreland, Diet.Ostry}@csiro.au

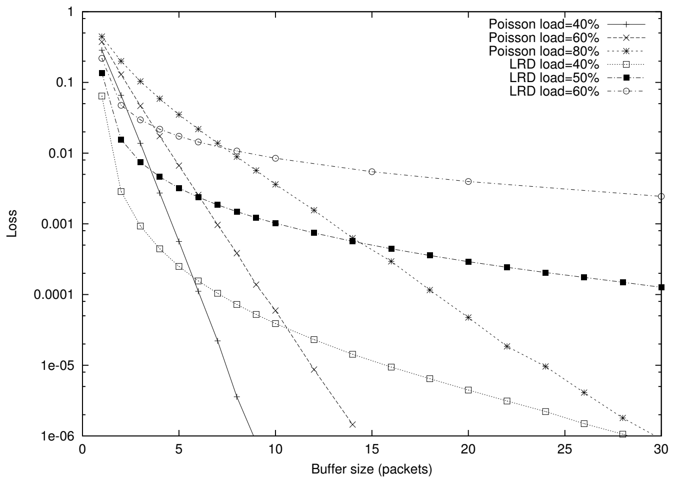


Fig. 1. Loss vs. buffer size at a finite-buffer switch

in Australia and the USA, that pacing can help achieve acceptable network loss performance at the cost of a small and controllable increase in flow end-to-end delays. We therefore propose pacing as an attractive mechanism for the realisation of cost-effective (i.e with small buffers) OPS networks that can provide the desired loss and delay performance.

The rest of this paper is organised as follows: section II illustrates the performance impact of small 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 from off-line smoothing techniques for video traffic relevant to our work, while section IV develops efficient algorithms for the real-time pacing of arbitrary time-constrained traffic. In section V we quantify via analysis the impact of pacing on traffic burstiness and loss for a single flow, after which the loss-delay trade-off achievable via pacing in realistic network scenarios is investigated via simulation in section VI. The paper is concluded in section VII, which also points to directions for future research.

II. Small Buffers: Impact and Solutions

In this section we first illustrate via simulation the impact of small buffers on losses for real-time traffic (the impact on TCP traffic when mixed with real-time traffic is studied in our subsequent work [16], [17]). We then briefly review some prior solutions to reducing the performance degradation, and outline our approach to tackling this through packet pacing.

A. Loss for Real-Time Traffic

A direct and obvious impact of small network buffers is an increase in packet losses. As 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 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 may be unacceptable in a core network supporting real-time applications with stringent loss requirements.

B. Prior Work

Some earlier works have proposed modifying traffic characteristics to improve performance in optical networks with limited buffers. The approach in [18]–[20] 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 [20], and approximate off-line [19], [20] and on-line [18] 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.

Traffic *shaping* has been widely used for controlling packet losses in electronic networks. ATM and IP networks have used rate-based shaping methods such as GCRA or leaky-bucket to protect the network against relatively longer-time-scale rate fluctuations, while short-time-scale burstiness is expected to be absorbed by router buffers. The requirements for controlled loss, delay and delay variation in ATM networks have stimulated extensive studies [21] of queueing behavior for a broad range of traffic models and traffic control and shaping policies. Low transmission delay requires small queue lengths, and it has been conjectured that actively spacing traffic might ensure negligible cell delay variation [21, pg. 121]; however, the provision of guaranteed delay constraints in networks limited to small buffers has not been investigated. When buffers at routers are small, we observed in simulations that short-time-scale burstiness seems to be the main cause of performance degradation. Rate-based shaping cannot protect against such losses, since a low shaping rate leads to excessive queueing delays at the shaper, while a large shaping rate is ineffective in reducing short-time-scale burstiness. This has been confirmed by studies in [22] and our earlier work in [23]. What is therefore required is a means of reducing the short-time-scale burstiness of the traffic without introducing excessive delays; such traffic "pacing" is the topic of this paper.

Parallel to our work, researchers at Stanford have developed techniques that make networks with very small buffers workable. Following on from their arguments in [24] showing that router buffer size need only be inversely proportional to the square-root of the number of TCP flows, they have recently shown in [25] that by making each TCP sender "pace" its packet injections into the network, a router buffer size of 10-20 packets suffices to realise near-maximum TCP throughput performance. Though some aspects of the model are still under discussion [26], [27], it would seem their identification of the advantages of TCP pacing bolsters our proposal to pace traffic at the optical edge. There are however significant differences to our approaches – while their study considers

TCP traffic, our current study primarily focuses on non-TCP real-time traffic (we have since extended our work [16], [17] to scenarios that have mixed TCP and real-time traffic). 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 small 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.

C. 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. Here “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 [28] 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 [29]–[31] (where a few seconds of distribution delay is acceptable) as well as interactive video streams [32] (wherein only a few frames can be buffered at the smoother).

To the best of our knowledge, there has not been a study (apart from our own [33], [34]) on the use of traffic pacing techniques for alleviating contentions in OPS networks with very small buffering resources. Our paper makes three new contributions in this regard: (1) we develop new algorithms of provably low complexity that can perform efficient pacing of arbitrary traffic in real-time, (2) we develop a novel analytical framework for estimating burstiness and loss of a paced traffic stream, and (3) we quantify via simulation of realistic topologies 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 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

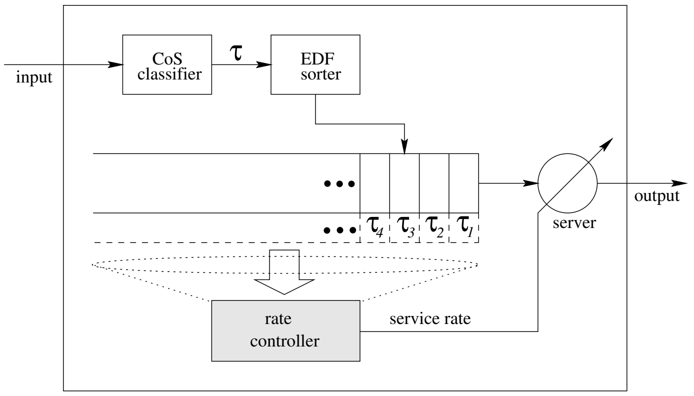


Fig. 2. Model of the pacer

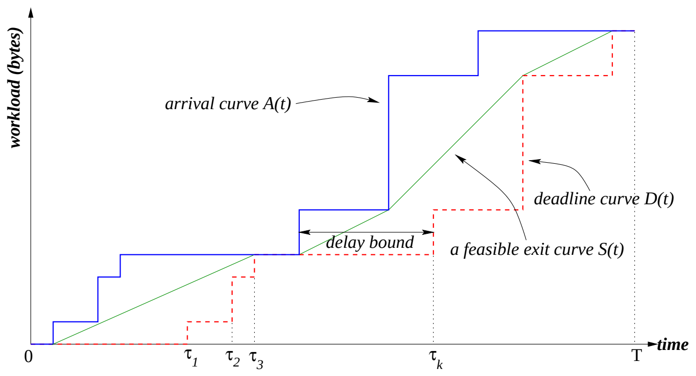


Fig. 3. Arrival, deadline, and exit curves for an example workload process

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, that is, it minimises output burstiness, subject to delay constraints. We 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. 2. 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 [35], [36], which has known optimality properties [37] and can be implemented efficiently [38]. 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 (this section) and in implementing this scheme efficiently in real-time at high data rates (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 devoid 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. 3 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 [28] to be the *shortest path* between the origin $(0, 0)$ and the point $(T, D(T))$, as shown in Fig. 3. 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 optimum by maintaining a time window (i.e. delay buffer) to implement a lookahead capability (see for example [29]–[31]). There has also been some work in smoothing interactive video streams [32] wherein a few frames are buffered at the smoother.

Unlike the video transmission context, smoothing or pacing in OPS networks will have to operate under much more demanding conditions. Current mechanims for smoothing consider one or a few video streams at end-hosts or video server; by contrast OPS edge nodes will have to perform the pacing on large traffic aggregates at extremely high data rates. The time-constraints for computing the optimal pacing patterns are also much more stringent – unlike video smoothing where a few frames (tens to hundreds of milliseconds) of delay is acceptable, OPS edge nodes will have buffer traffic for shorter time lest the buffering requirement becomes prohibitively expensive (at 10 Gbps, 1 msec of buffering needs 10 Mbits of RAM). Our next section therefore develops algorithms that are amenable to efficient implementation at OPS edge nodes.

IV. Efficient Real-Time Pacing

It is shown in [28] 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

```
     // determine length and deadline of new packet $p$
1.   $L$ = length($p$); $T$ = currtime; $T_p = T + d$
     // append new hull piece
2.   h = new hullPiece
3.   h.startT = ((hullList.empty()?) T : hullList.tail().endT);
4.   h.endT = $T_p$;
5.   h.slope = $L$/($T_p$-h.startT)
6.   hullList.append(h)
     // scan backwards to restore hull convexity
7.   h = hullList.tail()
8.   while ((hPrev=h.prev)$\neq$NULL $\wedge$ hPrev.slope $\leq$ h.slope)
9.       h.slope = [h.slope $*$ (h.endT $-$ h.startT)
                  + hPrev.slope $*$
                  (hPrev.endT $-$ max(T, hPrev.startT))]
               / (h.endT $-$ max(T, hPrev.startT))
10.      hullList.delete(hPrev)
11.  end while // the hull is now convex
```

Fig. 4. On-line algorithm for hull update upon packet arrival

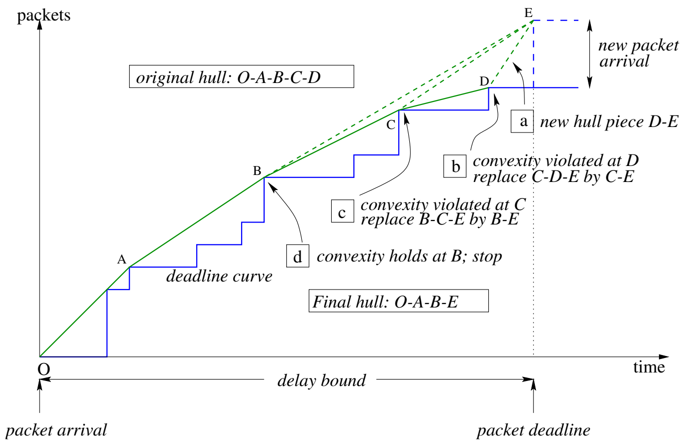


Fig. 5. Example showing single-class hull update in amortised $O(1)$ time

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 [33]. Upon each packet arrival, the 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 – Constant Amortised Cost Algorithm

We first consider the case where all packets entering the pacer have identical delay constraints. This simplifies the hull update algorithm since each packet arrival augments the deadline curve at the end. Our first algorithm computes the convex hull in $O(1)$ amortised time per packet arrival.

Fig. 4 depicts this update algorithm performed upon each packet arrival, and Fig. 5 illustrates the operations with an example. Recalling that the convex hull is piecewise-linear, we store it as a doubly linked list, where each element of the list corresponds to a linear segment whose start/end times and slope are maintained. In step 1 of the algorithm, the length of the incoming packet is determined, along with its deadline. The arrival of this new packet causes the deadline curve to be amended, which results in a new segment being appended to the hull. Steps 2-6 therefore create a new linear segment with the appropriate slope and append it to the end of the hull (shown by operation $\boxed{a}$ in Fig. 5). The new piece may cause the hull to lose convexity, since the newly added piece may have slope larger than its preceding piece(s). Steps 7-11 therefore scan the hull backwards and restore convexity. If a hull piece has slope larger than its preceding piece, the two can be combined into a single piece which joins the end-points of the two pieces (as depicted by operations $\boxed{b}$ and $\boxed{c}$ in Fig. 5). The backward scan repeatedly fuses hull pieces until the slope of the last piece is smaller than the preceding piece (operation $\boxed{d}$ in Fig. 5). At this stage the hull is convex and the backward scan can stop, resulting in the new hull.

Claim 1: The algorithm of Fig. 4 has constant *amortised* computation cost per packet arrival.

Proof: Our proof method follows the technique outlined for amortised analysis in [39] that assigns a dollar cost to each unit of computation. We start with the invariant that every point on the hull has a \$1 deposit associated with it. Upon packet arrival, steps 1-6 are constant time operations, consuming \$1 paid by the arriving packet. Further, an additional \$1 is deposited at the end point of the newly added hull segment. The loop in steps 7-11 walks backwards through the hull checking for convexity at each hull point. Each check is a constant time operation, and is paid for by the \$1 deposited at the hull point. If convexity fails, the hull point is removed, fusing two hull pieces into one. If convexity holds, the arriving packet deposits \$1 at that hull point, and the algorithm terminates. Thus each arriving packet has paid a constant \$3 in computation cost, and at termination of processing, a \$1 deposit is still available at each hull point, maintaining the invariant. This completes the proof. ■

In spite of a constant *amortised* cost per packet arrival, a packet arrival in the *worst-case* may cause all hull points to be scanned (steps 7-11) in order to restore convexity. We therefore develop in the next section an algorithm that has worst-case complexity logarithmic in the number of packets queued at the pacer.

The pacer releases packets based on the computed exit curve in a fairly straightforward manner: a process accumulates "credits" or "tokens" (much like a leaky-bucket) at an instantaneous rate stipulated by the slope of the piece (corresponding to the current time) in the exit curve. A packet from the head of the pacer queue is released as soon as sufficient credits (corresponding to the packet size) are available, such credits being deducted when the packet departs the pacer. The released packet is eligible for transmission on the output link. The link scheduler selects from amongst eligible packets for transmission, and is assumed to be FIFO in this work.

B. Single Delay Class – Logarithmic Cost Algorithm

Our $O(\log n)$ worst-case complexity algorithm for optimal exit curve computation is illustrated with an example in Fig. 6. 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.

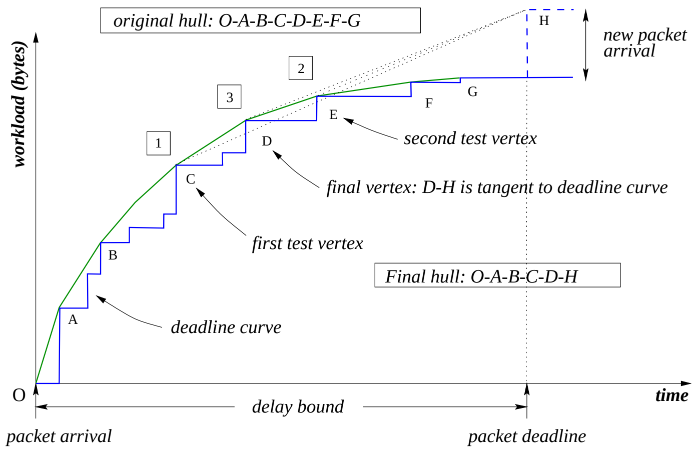


Fig. 6. Example showing single-class hull update

```
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. 7. Single-class hull update algorithm

Fig. 7 depicts the update algorithm in pseudocode. Recalling that the deadline curve is piecewise-linear, and the start/end times of the individual segments correspond to arrival of new packets as illustrated in Fig. 6, 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 [40, procedure TANGENT]. At

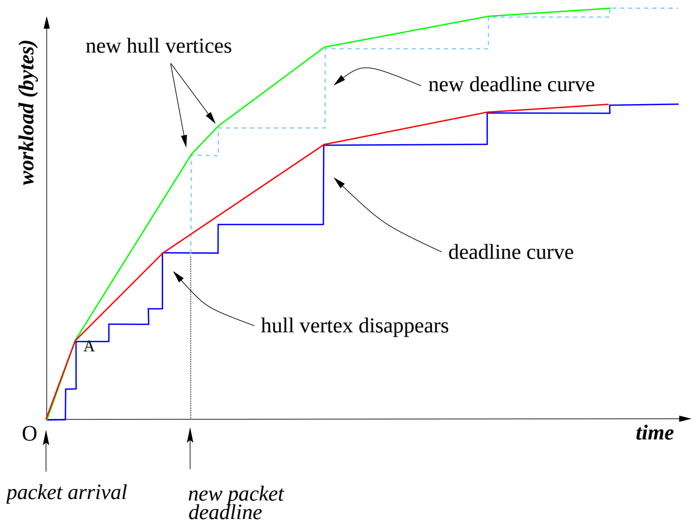


Fig. 8. Example illustrating that an arbitrary number of new hull points may appear when the arriving packet has an arbitrary deadline

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 [40].

The complexity of the above convex hull update operation that needs to be performed upon the arrival of each packet has $O(\log n)$ cost, where n is the number of queued packets [40]. 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 [40].

C. General Poly-Logarithmic Cost Algorithm

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. 8. 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. 9. Multi-class hull update algorithm

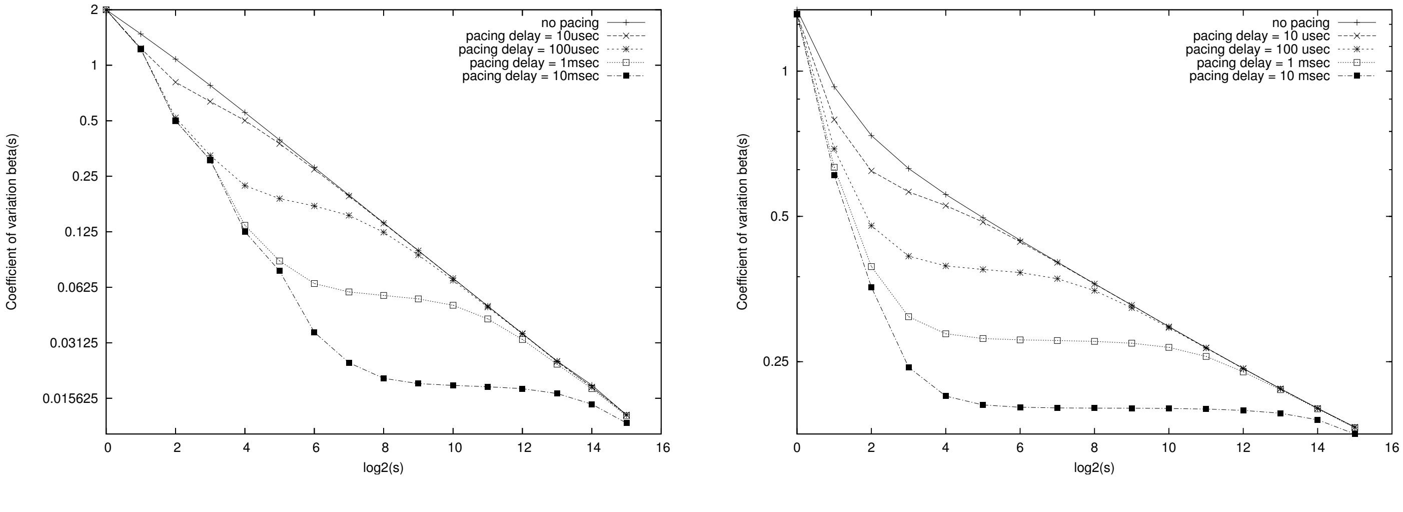
Fig. 9 depicts our algorithm for determining the convex hull upon each packet arrival. Our vertices are stored in the leaves of a search tree T structure which is capable of supporting concatenable-queue operations, such as the 2-3 tree [41, sections 4.12], with the value of time used as the search key. 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 [42].

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 [41, 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 [42]. 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 [42].

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 [42].

V. Performance Evaluation for a Single Flow

Having addressed the *feasibility* of pacing at high data rates, we demonstrate the *utility* of pacing in OPS systems with small



(a) Poisson traffic

(b) LRD traffic

Fig. 10. Burstiness vs. time-scale for various pacing delays from simulation of Poisson and LRD traffic

buffers. This section evaluates via analysis and simulation the impact of pacing on traffic burstiness and loss for a single flow, while the next section evaluates via simulation loss performance for several flows in realistic network topologies.

A. Traffic Models

We apply our pacing technique to Poisson and long range dependent (LRD) traffic models (both of which were introduced in section II-A); the Poisson model is selected for its simplicity and ease of illustration of the central ideas, while the LRD model is chosen since it is believed to be more reflective of traffic in real networks. Our LRD traffic generators are derived from Norros' self-similar traffic model [43]. 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 novel filtering method [44] to generate long sequences $\{x_i\}$ of normalised fGn (zero mean and unit variance) for a chosen H. A discretisation interval Δt is chosen, and the traffic y_i (in bits) arriving in the i-th interval is obtained by scaling and shifting the normalised fGn samples as follows:

$$y_i = max\{0, c + \sigma x_i\} \tag{1}$$

where c denotes the traffic quantity (in bits) obtained in the interval from the constant rate stream, and σ is a scaling factor that determines the instantaneous burstiness. The y_i is truncated at zero (to prevent arrival of negative quantity of traffic in that interval), causing the mean $E[y_i]$ to differ from c. Letting $m = E[y_i]$ denote the mean traffic rate (i.e. bits per interval), we take expectation of both sides of eq. (1) to obtain:

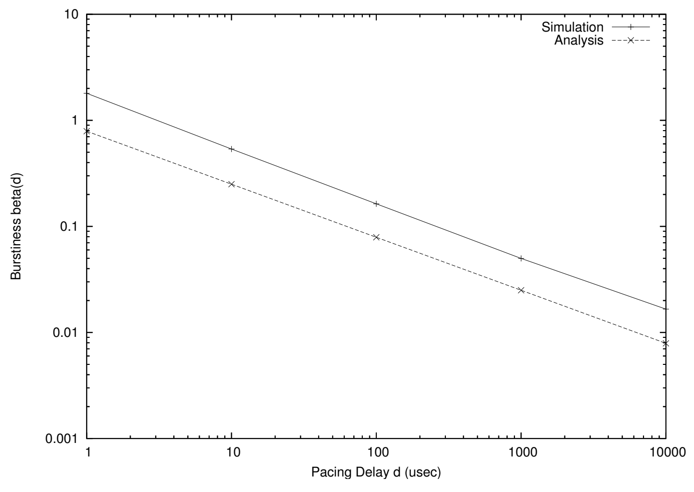
$$\frac{m}{\sigma} = \frac{1}{\sqrt{2\pi}} e^{-(c/\sigma)^2/2} + \frac{c}{\sigma} Q(-c/\sigma) \tag{2}$$

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 σ was chosen to satisfy $c/\sigma = 1$, which corresponds to truncations for around 16% of the samples, and c is then adjusted to give the desired mean traffic rate m. The fluid traffic is then packetised for use in simulation. For simplicity, in this section we assume packet sizes are fixed at 1250 bytes, such that a packet can be transmitted in a slot of size 1μsec on a 10Gbps optical link; our packet and slot sizes are consistent with studies of OPS networks in the literature [45], and are also commensurate with current optical crossbar technology such as [46]. The fixed packet-size assumption will be relaxed in the next section that considers several traffic flows in realistic network topologies.

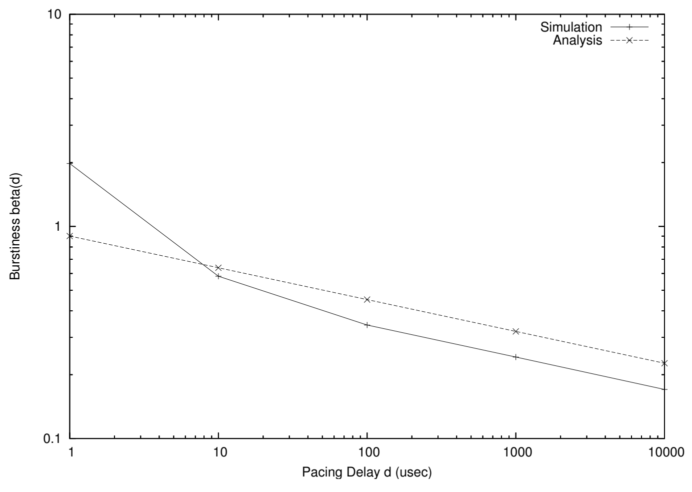
B. Impact of Pacing on Traffic Burstiness

We now study using simulation and analysis how pacing changes the burstiness of a traffic stream at various time-scales. Burstiness at time-sale s is quantified by $\beta(s)$, the coefficient of variation (i.e. ratio of standard deviation to mean) of traffic volume measured over time intervals of size s. Log-log plots of $\beta(s)$ versus s are routinely used in the literature to depict traffic burstiness over various time-scales as an indicator of self-similarity of traffic traces and to show the influence of the Hurst parameter H. Our simulations in this section fix the link rate at 10 Gbps and packet sizes at 1250 bytes (such that each packet requires exactly $1\mu s$ for transmission).

1) Simulation Results: Fig. 10 shows for Poisson and LRD traffic the burstiness $\beta(s)$ versus time-scale s (in μsec) on log-log scale observed in simulation for pacing delay d of 0 (i.e. no pacing), 10μsec, 100μsec, 1msec, and 10msec. We first note that the unpaced Poisson stream (at 8 Gbps or 80% load) in Fig. 10(a) and the unpaced LRD stream (at 4 Gbps or 40% load) both exhibit straight lines with respective slopes -0.5 and -0.15; this validates the expected slope $-(1 - H)$ for Hurst parmeter settings $H = 0.5$ and $H = 0.85$ of the short and long range dependent traffic respectively (the different



(a) Poisson Traffic



(b) LRD Traffic

Fig. 11. Traffic burstiness $\beta(d)$ versus pacer delay d from analysis and simulation of Poisson and LRD traffic

slope at very short time-scales for LRD traffic arises from the packetisation of the ideal fluid model).

We next note that pacing reduces burstiness only within a range of time-scales, explained as follows. At ***very short*** time-scales (e.g. $s = 2^0 = 1\mu$sec in our example), burstiness is invariant to pacing due to the discrete nature of packet release. Specifically, if $p_i(\Delta t)$ denotes the probability that an interval of size Δt has i packets, then for $i \geq 2$: $\lim_{\Delta t \to 0} p_i(\Delta t) = o(\Delta t)$ is vanishingly small when the traffic model does not permit batch arrivals. The burstiness $\beta(\Delta t)$ at very short time-scales therefore depends only on $p_0(\Delta t)$ and $p_1(\Delta t)$, which in turn are determined by the mean traffic rate and are invariant to pacing. At ***very long*** time-scales (beyond the delay budget d of the pacer), burstiness is again invariant to pacing. This is because the pacer does not hold any packet back beyond its deadline, and so pacing cannot alter the characteristics of the traffic at time scales that are much longer than the pacing delay budget.

Pacing is most effective in the range of time-scales within the pacer's delay budget. Barring the very-short time-scale region (that is dominiated by the discrete nature of the packet pacing process), we observe that for both Poisson and LRD traffic, and for any fixed pacer delay budget d, the burstiness of the paced traffic remains nearly constant at $\beta(d)$ with time-scale, till it converges with the burstiness of the input traffic. This demonstrates the efficacy of pacing in reducing short time-scale burstiness, without altering longer time-scale traffic characteristics. The burstiness $\beta(d)$ achieved by pacing is estimated analytically next.

2) Analytical Estimate: The analysis of the pacer is non-trivial, since it operates as a variable-rate server whose rate is modulated by the deadlines of the packets in queue; classical queueing thoery does not handle such servers. Our approach uses a fluid approximation of the pacer *output* (input to the pacer can be fluid or discrete packets), and attempts to estimate the pacer's fluid service rate $R(t)$ at time t. To simplify the analysis we assume that all packets arriving at the pacer have identical delay constraint d.

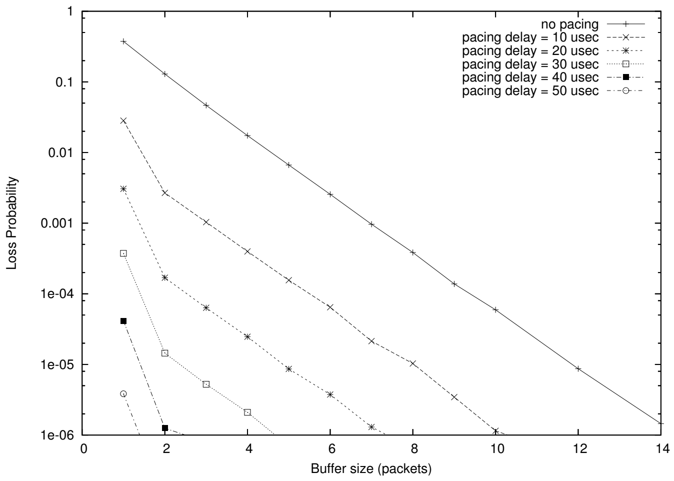
Recall from the previous section that at time t, $R(t)$ equals the slope of the convex hull of the deadline curve. Though estimating this slope precisely is complex, we can approximate it by considering the shape of the deadline curve in the region surrounding time t. The pacer rate $R(t)$ at time t is influenced by the shape of the deadline curve in the preceding interval $[t-d, t)$ as well as the succeeding interval time $[t, t+d)$, since the pacer can "look-ahead" no more than its delay budget d. The deadline curve prior to time $t-d$ does not influence its rate at time t, nor does the deadline curve beyond time $t+d$ since packets contributing to that region of the deadline curve have not arrived by time t. In the interval $[t-d, t+d)$ of interest for the deadline curve, the total traffic volume that has to be released by the pacer equals the traffic volume $A[t-2d, t)$ arriving in interval $[t-2d, t)$, since such traffic has deadlines within this region. Noting that the pacer has $2d$ units of time to release this traffic, and that the optimal pacer will release traffic at the smoothest possible rate within this interval, we can approximate the pacer rate by

$$R(t) \approx A[t-2d, t)/2d \tag{3}$$

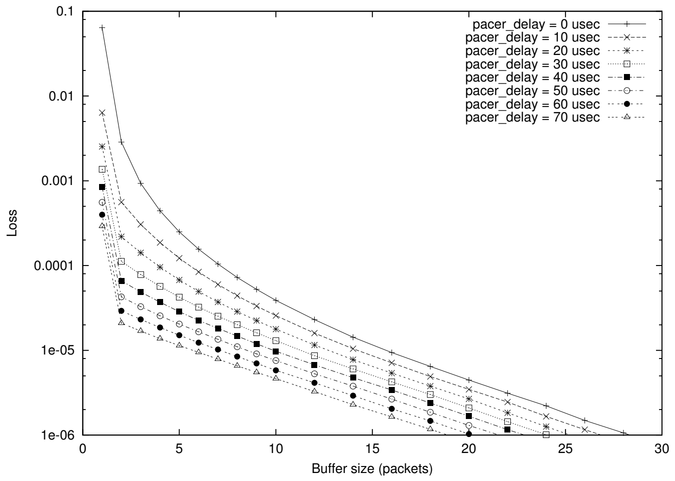
For any stationary arrival process the quantity $A[t-2d, t)$ is independent of t and depends only on d. Using known expressions for the traffic arrival volumes under the Poisson and LRD models, we now estimate the burstiness of the (fluid) traffic egressing the pacer.

For **Poisson** arrivals at rate λ (normalised to units of packets per packet-transmission-time), the distribution of the number of arriving packets in interval $[t-2d, t)$ is given by $P[A[t-2d, t) = k] = e^{-2\lambda d}(2\lambda d)^k/k!$. This lets us determine the mean, standard deviation, and their ratio the burstiness β, of the pacer output rate $R(t)$ estimated in (3):

$$\beta(d) \approx 1/\sqrt{2\lambda d} \tag{4}$$



(a) Poisson traffic



(b) LRD traffic

Fig. 12. Loss versus buffer size at various pacing delays from simulation of Poisson and LRD traffic

This analytical estimate of burstiness of the fluid pacer traffic is plotted against pacer delay d in Fig. 11(a) for Poisson traffic at 80% load, and compared to the burstiness of the paced traffic at time-scale commensurate with the pacing delay d observed in simulation. Though the plot shows that analysis predictions of burstiness to be lower than simulation (fluid approximation do typically show lower burstiness than the corresponding discrete packet system), the slopes of the two curves are in excellent agreement, validating the model as an accurate predictor of the impact of pacing on traffic burstiness.

For the Norros **LRD** traffic model, recall that y_i in (1) denotes the amount of traffic arriving in the i-th discretization interval of size Δt. The average traffic arrival rate over time interval $2d$ is then deduced by generating blocks of $m = 2d/\Delta t$ aggregated samples as defined in [47, pg 18], where the rate in the j-th block is given by:

$$Y^{(m)}(j) = \frac{1}{m} \sum_{i=m(j-1)+1}^{mj} y_i \tag{5}$$

The asymptotic mean and variance of the aggregated block samples $Y^{(m)}$ can be determined using the expressions in [47, pg 20], which allow us to estimate, for the setting $c/\sigma = 1$, the rate burstiness of $R(t)$ in (3) as:

$$\beta(d) \approx (2d)^{-(1-H)} \tag{6}$$

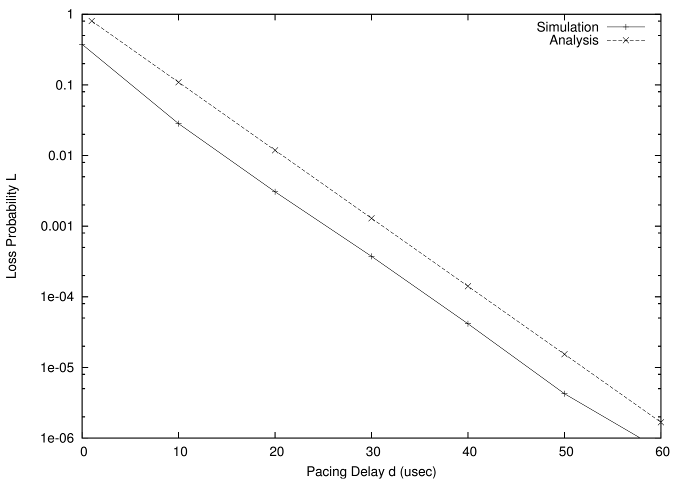
Fig. 11(b) shows the analytical estimate of the burstiness of paced LRD traffic (at 40% load) as a function of pacing delay budget, and compares it to simulation. The discrepancy between analysis and simulation is attributed to the asymptotic nature of the estimates of moments of the rate process. Nevertheless, the slope of the burstiness curve from analysis shows excellent agreement with simulation, thereby validating the model and demonstrating the effectiveness of pacing in reducing burstiness.

C. Impact of Pacing on Packet Loss

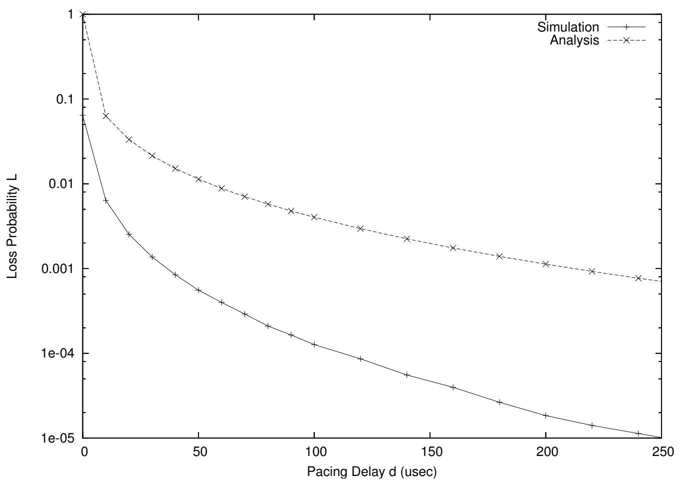
Having quantified the impact of pacing on the burstiness of Poisson and LRD traffic, we now feed the paced traffic into a constant rate server (OPS link) and observe how losses are affected by pacing. For analytical and conceptual simplicity we consider only a single flow with fixed size packets in this section; realistic network topologies with multiple flows and variable packet sizes are considered in the subsequent section.

1) Simulation Results: In our simulation we feed the paced traffic stream into an OPS link with fixed capacity and small buffers, and observe the packet loss probability as a function of buffer size for various pacing delays. Fig. 12(a) plots for Poisson traffic (at 60% link load) the observed loss (on log scale) as a function of buffer size (in packets) for pacing delays of 0 (corresponding to no pacing), 10, 20, 30, 40, and 50 μsec (recall that 1μsec corresponds to the transmission time of a packet). Pacing is seen to be extremely effective in reducing loss: at the cost of a few tens of μsec of increase in end-to-end delay, the packet loss probability can be reduced by multiple orders of magnitude, which is a very attractive cost-benefit trade-off. Fig. 12(b) plots the losses for an LRD traffic stream (at 40% load) when pacing delay of 0 to 70μsec is employed. Once again pacing is seen to be effective: for example, a 70μsec pacing delay (which contributes to a very small increase in end-to-end delay), reduces losses at the link by more than an order of magnitude.

2) Analytical Estimate: The modification of traffic characteristics by the pacer makes it very challenging to estimate the packet loss probability of the paced traffic stream when fed into a finite capacity queue. We estimate loss under a fluid bufferless approximation, which has been used very successfully in the past for analysing loss at core nodes such as ATM multiplexers [48] and generalised processor sharing (GPS) schedulers [49]. Specifically, the core OPS link is assumed to be bufferless, and losses happen whenever the input (fluid) rate exceeds the service rate. The bufferless approximation



(a) Poisson traffic



(b) LRD traffic

Fig. 13. Loss probability versus pacing delay from analysis (fluid model) and simulation (buffer size of one packet) for Poisson and LRD traffic

is particularly reasonable for our study since the OPS core nodes are expected to have very small buffering capability. The packet loss probability L can then be approximated by the probability that the pacer service rate $R(t)$, which we have estimated in (3), exceeds the core link capacity (which we normalise to unity for convenience). This probability is then estimated using the Chernoff bound:

$$L \approx P[R(t) > 1] = P[e^{uR(t)} > e^u] < \frac{E[e^{uR(t)}]}{e^u} \quad \forall u > 0 \tag{7}$$

For **Poisson** traffic, we can compute $E[e^{uR(t)}]$ as follows:

$$\begin{aligned} E[e^{uR(t)}] &= \sum_{k=0}^{\infty} e^{uk/2d} P[A[t-2d,t) = k/2d] && (8) \\ &= \sum_{k=0}^{\infty} e^{uk/2d} e^{-2\lambda d} (2\lambda d)^k / k! && (9) \\ &= e^{-2\lambda d} \sum_{k=0}^{\infty} (e^{u/2d} 2\lambda d)^k / k! && (10) \\ &= e^{-2\lambda d} e^{2\lambda d e^{u/2d}} = e^{2\lambda d(e^{u/2d}-1)} && (11) \end{aligned}$$

Subsituting this in (7) gives us the upper bound on loss $L \leq e^{2\lambda d(e^{u/2d}-1)}/e^u \quad \forall u > 0$. To obtain the tightest bound, we minimise over u; this yields $u = 2d \ln 1/\lambda$, which when substituted back into the loss estimate gives us the tightest upper bound:

$$L \leq (\lambda e^{1-\lambda})^{2d} \tag{12}$$

Fig. 13(a) shows the analytical estimate of loss (as a function of pacing delay) and compares it to observations from simulation (the buffer size in simulation is set to one packet). The analytical bound is seen to match the simulation very well in slope, making it a good predictor of packet loss when a paced Poisson traffic stream is fed into a queue of small buffer capacity.

For **LRD** traffic based on the Norros model, $E[e^{uR(t)}]$ where the rate $R(t) \approx A[t-2d,t)/2d$ is obtained from the underlying aggregated process (5) using expressions in [47]:

$$E[e^{uR(t)}] = e^{cu + \sigma^2 u^2 / 2(2d)^{2-2H}} \tag{13}$$

Substituting this in (7) gives us the loss estimate $L \leq e^{cu+\sigma^2 u^2/2(2d)^{2-2H}}/e^u$. This loss estimate is minimised when $u = (1-c)(2d)^{2-2H}/\sigma^2$, which when substituted back in the Chernoff loss bound yields:

$$L \leq e^{\frac{1}{2}\frac{(1-c)^2}{\sigma^2}(2d)^{2-2H}} \tag{14}$$

For LRD traffic with $c = \sigma = 0.4$ packets (i.e. 40% load), Fig. 13(b) shows the analytical estimate of loss L as a function of pacing delay d, and compares it to simulation. We first note that there is a significant vertical discrepancy between our analysis and simulation. This is a known problem in the asymptotic performance studies of LRD systems, which predict the slope accurately but not the absolute values. Refinements such as the Bahadur-Rao theorem [50, sec 9.4.5],[51], [52] can be used to determine the multiplicative factors that improve accuracy, but are outside the scope of this paper.

Barring the vertical displacement, analysis shows the losses decay with pacer delay at a rate comparable to that observed in simulation. We expect their asymptotic slopes to converge; however, observing this in simulations is difficult since losses become too low to measure with sufficient confidence as pacer delays increase. Increasing loss rates by increasing traffic load is infeasible since that will not leave sufficient gaps between packets to make smoothing effective.

VI. Simulation Study of Networks

The previous section considered the impact of pacing on burstiness and loss at a single node. In this section we study its impact in more complex OPS networks with very limited contention resolution capabilities. It should be borne in mind that the efficacy of pacing in a network setting will be tempered by two considerations: First, unlike the single-link scenario considered earlier in which all traffic was smoothed by the

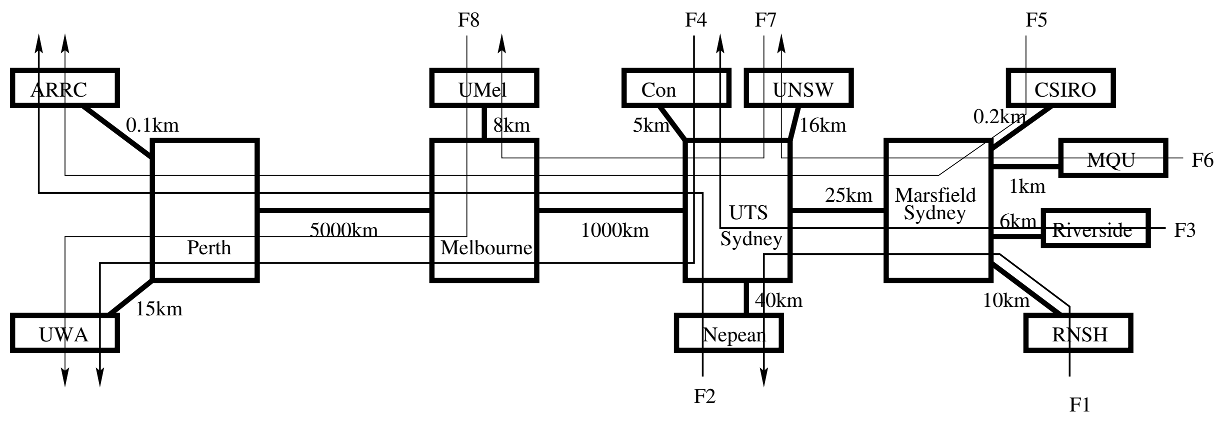


Fig. 14. CeNTIE Simulation Topology and Flows

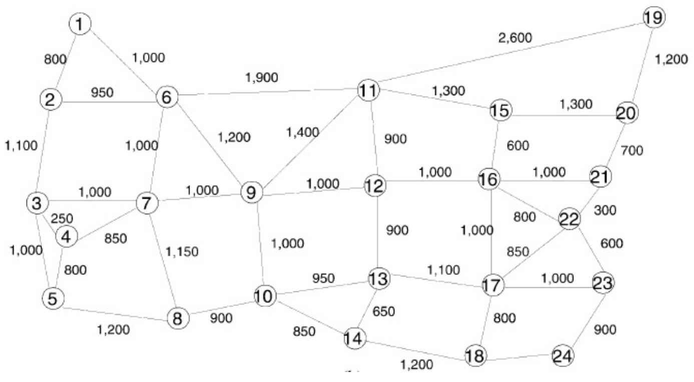


Fig. 15. NSFNet Simulation Topology [15]

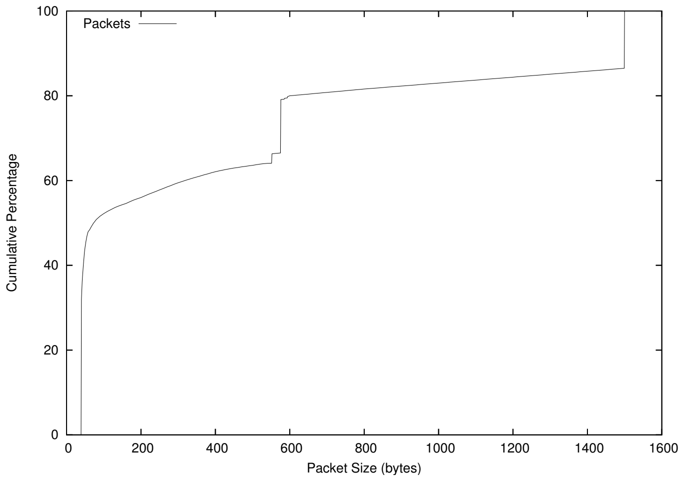


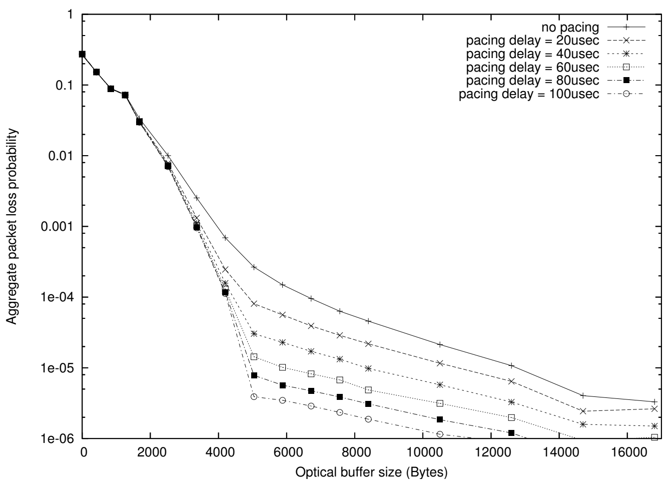
Fig. 16. Packet Size Distribution from CAIDA [54]

one pacer, traffic will now be paced locally and independently by each ingress point of the OPS core – though globally sub-optimal, this is the only feasible practical option. Second, the traffic smoothing at the edges will to some extent be negated by the multiplexing of streams at core nodes which will increase burstiness downstream. We are currently addressing the latter issue by developing scheduling schemes for OPS core nodes that best preserve the packet spacing introduced by the edge pacers [53], however this work is beyond the scope of the current paper.

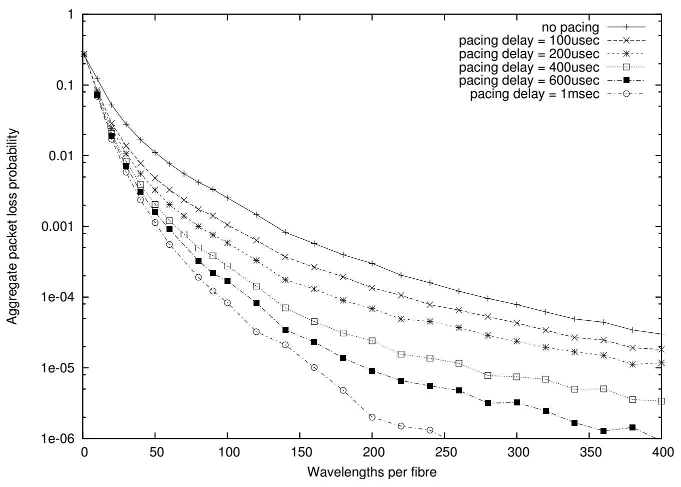
Our simulation study of pacing in a multi-hop OPS network uses two network topologies: the CeNTIE network [55] which is a trans-Australian research network chosen because our group built and operated it, and the NSFNet backbone in the USA which has a richer topology and has been used in prior optical networking research studies e.g. [15]. For both topologies we quantify the core packet loss as a function of contention resolution resources (optical buffers and wavelength converters) for various end-to-end delay penalties (incurred via pacing at the edge). The network models simulated operate in an unslotted aysnchronous fashion [56], and transport variable size IP packets. We assume a packet size distribution similar to the one observed by CAIDA [54] from a capture of over 87 million packets in the year 2000 at the NASA Ames Internet Exchange (AIX). The cumulative distribution of packet size is shown in Fig. 16, with distinct steps at 40 (the minimum packet size for TCP), 1500 (the maximum Ethernet payload size), as well as at 532 and 576 from TCP implementations that don't use path MTU discovery. The mean packet length is 420 bytes, and our experiments in this section will for convenience configure buffer size at the switches to be multiples of 420 bytes. The buffering at the OPS core nodes is assumed to be First-Come-First-Served, as shown feasible by emerging all-optical random access storage technology [10].

Our first set of experiments are performed on the CeNTIE topology, which has metropolitan networks in Sydney, Canberra, Melbourne, and Perth, and includes end-user research groups from the health, education, film post-production, and finance industries. We simulate the core of the CeNTIE network as an OPS network, with logical topology and fibre lengths shown in Fig. 14. The figure also shows the eight trafic flows we chose to simulate that are representative of the usage of the CeNTIE network. Each flows generates LRD traffic based on the model described in section V-A, with an average load of 60% on each of the core links.

We first consider contention resolution using optical buffers without any wavelength conversion. Fig. 17(a) plots the aggregate network losses versus optical buffer size at each OPS node. Each run simulated over 40 million packets, and confidence intervals were small enough not to be shown in the plots. The buffer size in bytes is set at integeral multiples (0, 1, 2, 3, 4, 6, 8, 10, 12, 14, 16, 18, 20, 25, 30, 35, and 40)

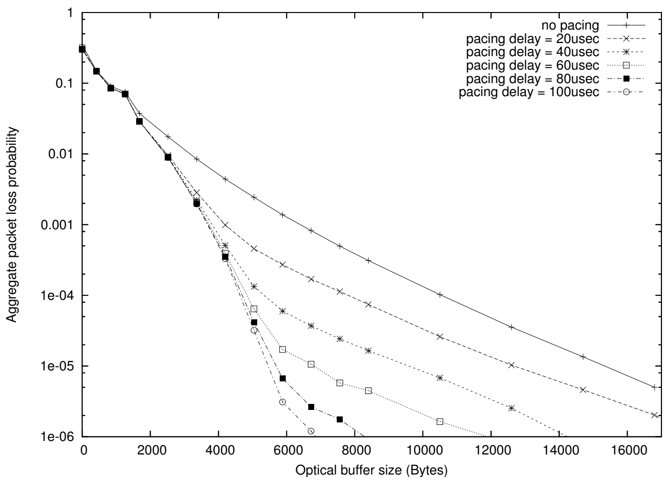


(a) Loss versus Buffer Size

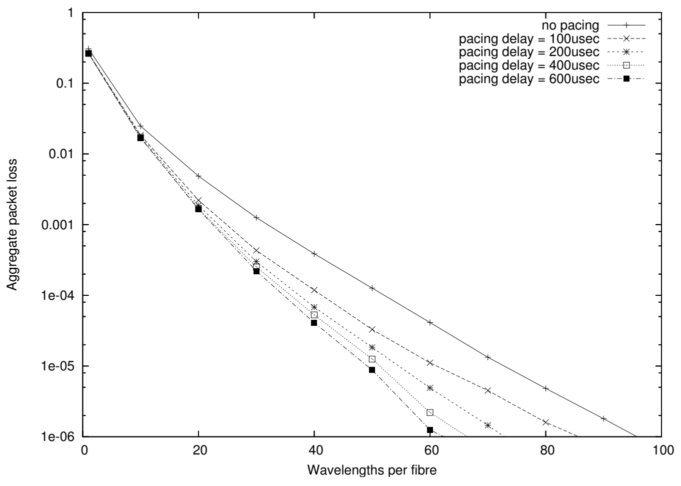


(b) Loss versus Number of Wavelengths

Fig. 17. CeNTIE network: Packet loss for various pacing delay budgets



(a) Loss versus Buffer Size



(b) Loss versus Number of Wavelengths

Fig. 18. NSFNet: Packet loss for various pacing delay budgets

of the average packet size of 420 bytes. The various curves in the figure correspond to different pacing delay bounds at the optical edge nodes. 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 system depends only on how many input lines have a packet destined for the same output line at any given time, and this is invariant to pacing. Pacing helps reduce burstiness over a period of time, but does not alter the average rate of packets.

As the optical buffer size increases to say 10-20 packets (4200-8400 bytes), edge pacing with a fairly small end-to-end delay penalty (less than hundred μsec) reduces losses in the OPS core by nearly two orders of magnitude. For a cross-continental network with tens of milliseconds in propagation delay, an additional few hundred microseconds of delay is negligible, while the loss performance improves substantially. This ability to trade-off some end-to-end delay for a substantial reduction in loss, and the ability to explicitly control the trade-off, suggest that pacing a very useful mechanism for achieving a desired performance in OPS networks with small buffers.

We now consider contention resolution using wavelength converters rather than optical buffers. The converters are assumed to have a full range of conversion, i.e., can convert any incoming wavelength to any outgoing wavelength. There are no optical storage buffers, and hence “small buffers” in this context refers to the limited number of wavelengths available for resolving contentions. Although in a real system each wavelength would operate at 10 Gbps or even higher, we maintain the total fibre capacity at 10 Gbps as in the previous experiment, and split this capacity equally amongst the wavelengths on the fibre. Maintaining the same fibre capacity allows us to use the same input traffic as in the previous scenario (keeping link loads at the same level), and allows a direct comparison to the results from the previous scenario. Fig. 17(b) plots the aggregate network losses against the number of wavelengths on each OPS link. As before, the various curves correspond to the various delay constraints at the edge node pacers. Note again that in a single wavelength system, i.e. in the absence of any contention resolution mechanims, pacing does not impact loss rates. As the number of wavelengths increases, pacing helps reduce loss rates: for example, when each link carries 100 wavelengths, pacing at the edges with a delay penalty of 1 msec reduces losses by nearly two orders of magnitude, demonstrating again the efficacy of pacing when contention resolution resources are sparse.

Our second set of experiments simulate the NSFNet topology, with 24 core OPS nodes interconnected as shown in Fig. 15. Each core node is connected to four edge nodes (not shown in the figure), and the network thus has 96 edge nodes. Each edge node generates traffic to one other randomly chosen edge node, giving us a total of 96 flows in the network. Each flow is routed along the shortest path (computed using the shown fibre lengths) from origin to destination. All flows generate Poisson traffic (we did not have sufficiently long traces of LRD traffic to generate sufficient packets for this large topology), and the maximum core link load in the network is maintained between 64% and 80% of link capacity.

Fig. 18(a) shows the aggregate packet loss probability in the network as a function of buffer size (in bytes) when optical buffering is employed for contention resolution. Each curve in the figure correponds to a different pacing delay. Once again pacing is seen to be very effective in reducing core loss for a very small penalty in end-to-end delay: for 10-20 packets (4200-8400 bytes) of buffering, a pacing delay of 100μs reduced loss by as much as three orders of magnitude, which would be a very attractive loss-delay trade-off in a core OPS network. Fig. 18(b) plots loss as a function of number of wavelengths per fibre when wavelength converters (with full conversion capability) are used for contention resolution. The benefits of pacing are again clear: for say 60 wavelengths per fibre, pacing delays of less than a millisecond are able to reduce loss by more than one order of magnitude.

VII. Conclusions

Emerging optical packet switched (OPS) networks will likely have very limited contention resolution resources usually implemented in the form of packet buffers or wavelength converters. This can cause high packet losses and adversely impact end-to-end performance. We identify short-time-scale burstiness as the major contributor to the performance degradation, and proposed to mitigate the problem by "pacing" traffic at the optical edge prior to injection into the OPS core. Pacing dramatically reduces 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. We developed a novel analytical model that accurately quantified the impact of pacing on traffic burstiness and loss at a core node with small buffers. We showed via simulation of realistic OPS network topologies that pacing can reduce losses by orders of magnitude, at the expense of a small and bounded increase in end-to-end delay. This ability to trade-off delay for loss makes pacing a very attractive way of realising acceptable performance from OPS networks with small buffers.

Our future work targets a deeper study of TCP performance, particularly when mixed with real-time traffic [16], [17]. We also intend to compare our traffic pacing at the optical edge to pacing TCP traffic at end-hosts [25].

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] 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.

[13] 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.

[14] 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.

[15] 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.

[16] A. Vishwanath and V. Sivaraman, "Routers with Very Small Buffers: Anomalous Loss Performance for Mixed Real-Time and TCP Traffic," in *IEEE IWQoS*, Enschede, Netherlands, Jun 2008.

[17] A. Vishwanath, V. Sivaraman, and G. N. Rouskas, "Are Bigger Optical Buffers Necessarily Better?" in *IEEE Infocom Student Workshop*, Phoenix, AZ, Apr 2008.

[18] J. Naor, A. Rosen, and G. Scalosub, "Online Time-Constrained Scheduling in Linear Networks," in *Proceedings of INFOCOM 2005*, Miami, FL, Mar 2005.

[19] 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.

[20] 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.

[21] J. Roberts, U. Mocci, and J. Virtamo, *Broadband Network Teletraffic: Performance Evaluation and Design of Broadband Multiservice Networks: Final Report of Action COST 242*. Springer, 1996.

[22] H. Elbiaze and T. Atmaca, "Traffic Management in Multi-Service Optical Network," in *Proceedings of IEEE ICN 2001*, Colmar, France, Jul 2001.

[23] V. Sivaraman, D. Moreland, and D. Ostry, "Ingress Traffic Conditioning in Slotted Optical Packet Switched Networks," in *ATNAC 2004*, Sydney, Australia, Dec 2004.

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

[25] M. Enachescu, Y. Ganjali, A. Goel, N. McKeown, and T. Roughgarden, "Routers with Very Small Buffers," in *Proceedings of IEEE Infocom*, Barcelona, Spain, Apr 2006.

[26] A. Dhamdhere and C. Dovrolis, "Open Issues in Router Buffer Sizing," *ACM SIGCOMM Computer Communications Review*, vol. 36, no. 1, pp. 87–92, Jan 2006.

[27] Y. Ganjali and N. McKeown, "Update on Buffer Sizing in Internet Routers," *ACM SIGCOMM Computer Communications Review*, vol. 36, no. 5, pp. 67–70, Oct 2006.

[28] 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.

[29] 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.

[30] 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.

[31] G. Cao, W. Feng, and M. Singhal, "Online Variable-Bit-Rate Video Traffic Smoothing," *Computer Communication*, vol. 26, no. 7, pp. 639–651, 2003.

[32] 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.

[33] 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.

[34] V. Sivaraman, H. ElGindy, D. Moreland, and D. Ostry, "Packet Pacing in Short Buffer Optical Packet Switched Networks," in *Proceedings of IEEE Infocom*, Barcelona, Spain, Apr 2006.

[35] 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.

[36] 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.

[37] 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.

[38] 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.

[39] T. H. Cormen, C. E. Leiserson, and R. L. Rivest, *Introduction to Algorithms*. McGraw-Hill, 1990.

[40] F. Preparata, "An Optimal Real-Time Algorithm for Planar Convex Hull," *Communications of the ACM*, vol. 22, pp. 402–405, 1979.

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

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

[43] 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.

[44] D. Ostry, "Synthesis of Accurate Fractional Gaussian Noise by Filtering," *IEEE Trans. Information Theory*, vol. 52, no. 4, pp. 1609–1623, Apr 2006.

[45] D. Careglio, J. Pareta, and S. Spadaro, "Optical Slot Dimensioning in IP/MPLS over OPS Networks," in *Proc. WOAN 2003.*, Zagreb, Croatia, Jun 2003.

[46] 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.

[47] K. Park and W. Willinger, *Self-Similar Network Traffic and Performance Evaluation*. Wiley, 2000.

[48] A. Elwalid, D. Mitra, and R. H. Wentworth, "A New Approach for Allocating Buffers and Bandwidth to Heterogeneous, Regulated Traffic in an ATM Node," *IEEE Journal on Selected Areas in Communications*, vol. 13, no. 6, pp. 1115–1127, August 1995.

[49] A. I. Elwalid and D. Mitra, "Design of Generalized Processor Sharing Schedulers which Statistically Multiplex Heterogeneous QoS Classes," in *Proceedings of INFOCOM '99*, New York, NY, March 1999.

[50] J. Walrand and P. Varaiya, *High-Performance Communication Networks*. San Francisco: Morgan Kaufman, 2000.

[51] Z. Fan and P. Mars, "Accurate Approximation of Cell Loss Probability for Self-Similar Traffic in ATM Networks," *IEE Electronics Letters*, vol. 32, no. 19, pp. 1749–51, Sep 1996.

[52] R. R. Bahadur and R. R. Rao, "On Deviations of the Sample Mean," *Ann. Math. Statist.*, pp. 1015–1027, 1960.

[53] Z. Rosberg and D. Ostry, "Fractional Wavelength OCS Based on the Golden Ratio," in *ICTON*, Athens, Greece, Jun 2008.

[54] "CAIDA packet length distributions." [Online]. Available: http://www.caida.org/analysis/AIX/plen_hist/

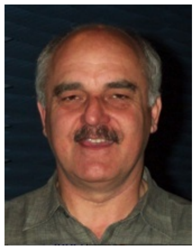
[55] T. Percival, "An Introduction to CeNTIE," Presentation, *http://www.centie.org/docs/CeNTIE-web-intro.ppt*.

[56] 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.

Vijay Sivaraman (M '94) received his B. Tech. from the Indian Institute of Technology in Delhi, India, in 1994, his M.S. from North Carolina State University in 1996, and his Ph.D. from the University of California at Los Angeles in 2000. He has worked at Bell-Labs and a silicon valley start-up manufacturing optical switch-routers. He is now a senior lecturer at the University of New South Wales in Sydney, Australia, and works part-time at the CSIRO. His research interests include Optical Networking, packet switching and routing, network design, and QoS.

Hossam Elgindy got his M.Sc. and Ph.D. in Computer Science from McGill University in 1985. He has been Assitant Professor at the University of Pennsylvania and McGill University, and senior lecturer at the University of Newcastle. Since 1999 he has been an Associate Professor at the University of New South Wales. He specialises in algorithms, particularly for combinatorial, geometric and multi-dimensional data. More recently he has been researching reconfigurable algorithms for adaptive computing systems and programmable logic devices.

David Moreland graduated in 1978 from Liverpool John Moores University, UK, and got his PhD in electro-optical metrology from the same University in 1987. He was a senior lecturer at Coventry University, UK, and then worked from 1990 to 2001 as a senior systems designer for Marconi and Fujitsu. He has since been a principal research scientist at the CSIRO ICT Centre. His research interests include optical packet/burst switching, and integrated service architectures for networking, storage, and trust.



Diethelm Ostry received the B.Sc. degree in Physics from the Australian National University, Canberra, in 1968 and the M.Comp.Sc. degree from the University of Newcastle, Australia, in 1996. He is currently a research scientist in the Networking Technologies Laboratory of the CSIRO ICT Centre in Sydney, Australia. His research interests include wireless communication, network traffic modeling, and security in wireless sensor networks.