A Novel Delay-Bounded Traffic Conditioner for Optical Edge Switches

Vijay Sivaraman, David Moreland and Diethelm Ostry
ICT Centre, CSIRO
PO Box 76, Epping, NSW 1710, Australia
Email: {Vijay.Sivaraman, David.Moreland, Diet.Ostry}@csiro.au

Abstract— **Optical Packet Switched (OPS) networks provide very limited contention resolution resources such as fibre delay lines (FDLs) and wavelength converters. Effective use of these resources in minimising contention losses within the all-optical core requires conditioning of traffic aggregates by the optical edge switches. Traditional rate-based shapers such as the leaky-bucket fail to provide acceptable delay performance for real-time traffic aggregates; this paper therefore explores novel conditioning mechanisms for OPS networks transporting traffic aggregates with time constraints. Using as a theoretical basis a known off-line optimum smoother for stored video traffic, we develop an on-line variable-rate conditioner that approximates the off-line optimum, and requires $O(1)$ amortised computation per packet arrival, making it amenable to efficient hardware implementation at the high data rates required by optical edge switches. We also demonstrate via simulation of short and long range dependent traffic that our conditioner allows losses in the optical core to be reduced by orders of magnitude at the expense of a bounded and relatively low increase in end-to-end delays. We believe that our conditioner can deliver significant performance benefits when employed at the edge of an all-optical network.**

Keywords: *Optical Packet Switching, traffic smoothing*

I. Introduction

Traffic burstiness is known to be detrimental to the performance of any network. The situation is particularly acute in optical packet switched (OPS) networks, where contention resolution resources such as fibre delay lines (FDLs) and wavelength converters are used sparingly due to cost and size limitations. Our previous work in [1], and similar studies in [2], have shown that the conditioning of traffic at the ingress to the optical network helps contain losses by allowing more effective utilisation of the sparse contention resolution resources. Conditioning, however, incurs a cost, namely an increase in end-to-end delay due to buffering at the conditioner. With traditional rate-based shapers such as GCRA or leaky-bucket, this delay performance is difficult to characterise, and requires assumptions about the traffic model. With long range dependent (LRD) traffic that exhibits burstiness at arbitrary time-scales, these shapers require a high setting for the shaping rate, which renders the conditioning ineffective in containing losses in the optical core.

Given the above inflexibility of rate-based shapers, we seek a conditioner wherein the rate may be variable, but the delay is bounded. The objective is, for arbitrary input traffic, to produce the smoothest output traffic that releases packets within their time constraints. Smoothing has been studied extensively in the context of video transmission. The authors in [3] consider off-line smoothing of stored video, and establish a theoretical reference by identifying the optimal smoothing strategy that minimises transmission rate variance subject to a given delay bound and server/client buffer sizes. This has led to several studies on dynamic smoothing of broadcast video streams [4], [5], [6] (where a few seconds of distribution delay is acceptable) as well as interactive video streams [7] (wherein only a few frames can be buffered at the smoother).

This paper explores the use of delay-constrained traffic conditioning in the context of optical networking. In contrast to video applications where one or a few streams are smoothed at end-hosts or video servers, optical networks require the smoothing of traffic aggregates at very high data rates. We therefore develop an online real-time conditioning algorithm that approximates the off-line optimum and has constant amortised computational complexity per packet arrival. This makes it amenable to efficient hardware implementation at high speeds. Unlike interactive video smoothing techniques, our conditioner does not predict or make assumptions about future arrivals, and is hence traffic model independent. Using simulations of short and long range dependent traffic, we show that our conditioner can reduce losses at a core OPS node by orders of magnitude, at the expense of a few milliseconds of additional delay. We believe our ideas can be very useful for future all-optical packet networks to provide acceptable loss performance without adversely compromising end-to-end delays.

The rest of the paper is organised as follows. Section II specifies the problem, and reviews the off-line optimality result from earlier work. In section III we propose a practical real-time algorithm which approximates the off-line optimum, and demonstrate its feasibility for implementation in practical packet switches. Section IV demonstrates via simulation the effectiveness of our conditioning method in reducing burstiness, and consequently losses at an optical core switch. Section V summarises our contributions, and points out directions for future study.

II. Problem Specification

The structure of the traffic conditioner we consider is shown diagrammatically in figure 1. It consists of a FIFO queue, a variable-rate server, and a rate controller which dynamically adjusts the server rate based on the deadline times τ_i of packets currently in the queue. The deadlines specify the times by which each packet must have been placed on the output link. Packets are assumed to enter the conditioner according to an arbitrary arrival process and are released onto the output link by the server.

The limited buffering capability of contemporary optical switch designs makes packet losses in an OPS core network sensitive to burstiness in the traffic. Traffic burstiness can be reduced by a rate-based (e.g. leaky-bucket) shaping mechanism, but this results in a traffic-dependent delay which may be unacceptably large. In this work, therefore, we impose a constraint

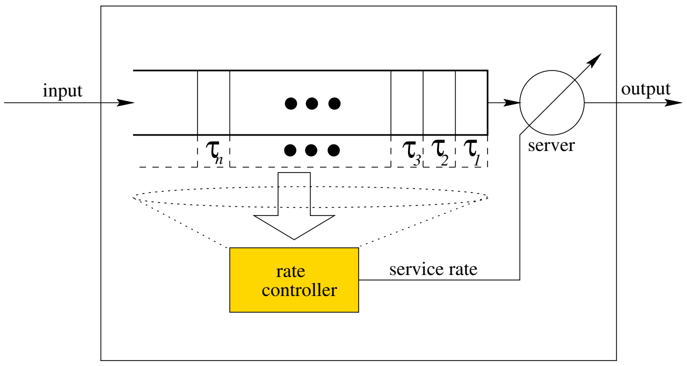

Fig. 1. Traffic conditioner structure

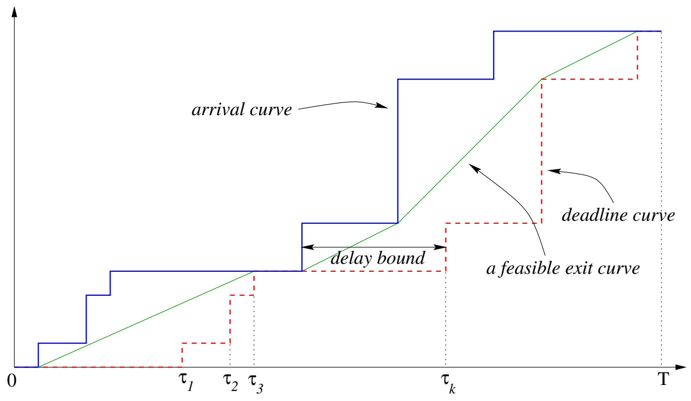

Fig. 2. Arrival, deadline, and exit curves for an example workload process

that the delay experienced by a packet be no greater than some prescribed bound d, and the goal of the rate controller in figure 1 is then to establish and maintain a service schedule which gives the smoothest (i.e. least bursty) output traffic flow, while ensuring that no packet exceeds its delay bound.

We briefly review previous work directed towards determining this service schedule. Consider the conditioner of figure 1 fed by an arbitrary input traffic stream. Suppose the system starts at time 0 and stops at time T. Denote by $A(t), 0 \le t \le T$ the *arrival curve*, namely the cumulative workload that arrives in the interval $[0,t)$. The system starts empty, thus $A(0) = 0$. Recall that traffic arriving at the conditioner has to be released within the delay bound d; hence an arriving workload unit (i.e. packet) at time t has deadline $t+d$. Denote by $D(t), 0 \le t \le T$ the *deadline curve*, namely the cumulative workload that has to be served in $[0,t)$ so as not to violate any deadlines. We set $D(0) = A(0)$ and $D(T) = A(T)$. The curves $A(t)$ and $D(t)$ are depicted in figure 2, and it can be observed that $D(t)$ is a right-shifted version of $A(t)$ with a horizontal distance equal to the delay bound d. Any service schedule can be represented by a non-decreasing curve $S(t), 0 \le t \le T$ describing a particular cumulative exit curve. A feasible exit curve, that is, one which is causal and satisfies the delay constraint, must lie in the region bounded by the arrival curve $A(t)$, and the deadline curve $D(t)$.

Amongst all feasible exit curves, the one which corresponds to the smoothest output traffic flow has been shown [3] to be the shortest path between the origin and the point $(T, A(T))$, as shown in figure 2. 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. Nevertheless the optimum exit curve identified by this approach, (computed a posteriori in the case of real-time flows) can provide a useful benchmark against which to compare practical on-line algorithms.

The problem of determining good service schedules in on-line applications has been studied in the context of transmitting those video streams in which delays of seconds to minutes are tolerable, for example in some news and sports broadcasts. The framework provided by the optimum off-line schedule suggests on-line algorithms which seek optimal schedules within a time window maintained by introducing a delay buffer to implement a lookahead capability (see for example [6]).

In this paper, however, we are concerned with the specific goal of conditioning aggregated traffic at the ingress of optical networks to compensate for severely limited buffer resources in the network core switches. Traffic burstiness can cause losses by overloading these resources, and our previous work [1] suggests that such losses can be significantly reduced by smoothing the traffic. The next section develops algorithms for real-time conditioning of traffic within specified delay bounds.

III. A Practical On-Line Traffic Conditioner

The off-line optimum identified above is useful in contexts where the workload arrival is known in advance. At OPS edge nodes, however, the packet arrival process is non-deterministic, and the arrival curve is not known beforehand. We propose an algorithm which is implementable in real-time, and approximates the off-line algorithm above. At any instant of time, our on-line algorithm maintains the optimal (i.e. least bursty) exit curve for the packets *currently* in the system, without accounting for future arrivals. 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. Upon each packet arrival, the deadline curve is augmented, and this may require a recomputation of the convex hull which defines the exit curve.

Figure 3 depicts the update algorithm performed upon each packet arrival, and figure 4 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 $\boxed{\text{a}}$ in figure 4). The new piece may cause the hull to lose it 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 op-

```
    // determine length and deadline of newly arrived 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. 3. On-line algorithm for hull update upon packet arrival

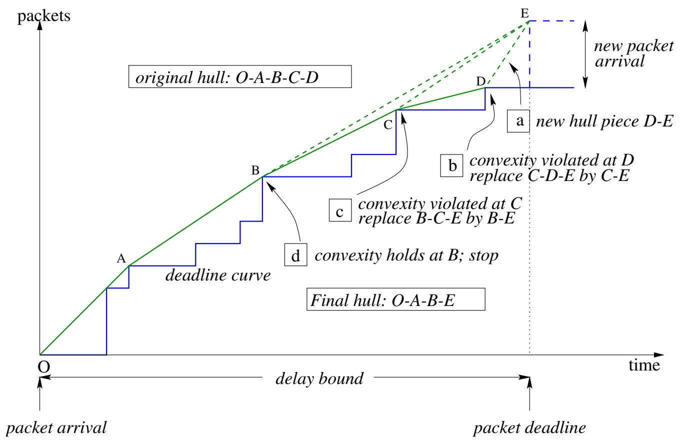

Fig. 4. Illustration of hull update upon packet arrival

erations $\boxed{b}$ and $\boxed{c}$ in figure 4). The backward scan repeatedly fuses hull pieces until the slope of the last piece is smaller than the preceding piece (operation $\boxed{d}$ in figure 4). At this stage the hull is convex and the backward scan can stop, resulting in the new convex hull.

We claim that the above algorithm, which needs to be performed upon every packet arrival, has only $O(1)$ *amortised* cost.

Claim 1: The algorithm of figure 3 has constant amortised computation cost per packet arrival.

Proof: Our proof method follows a technique outlined for amortised analysis in [8] 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

```
    // for each slot
1.  $T$ = curTime; $L$ = length(q.head())
2.  credit = credit + hullList.head().slope
3.  if (credit $\geq L$)
4.      dequeue and release packet
5.      credit = credit - $L$
6.  end if
    // remove expired hull piece
7.  if (hullList.head().endT $\leq T$
8.  delete hullList.head()
9.  end if
```

Fig. 5. Service in each slot based on computed hull

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. In those cases, practical implementations may choose from various heuristics, such as limiting the backward scan to at most k hull points, or restoring convexity every k-th packet arrival. The study of these heuristics is deferred to future work.

The conditioner releases packets according to the computed exit curve. This operation is straightforward, and is depicted for a slotted OPS system in figure 5. Step 1 determines the current slot time and the length of the packet at the head of the queue. Steps 2-6 update the credit count based on the service rate in the hull, and release the packet if sufficient credits are available, while steps 7-9 remove the hull piece once it has been used and pertains to the past. This sequence has $O(1)$ complexity per slot. We note that the hull computation algorithm does not constrain the hull slope by the link rate. Since service rate cannot exceed link capacity, this can violate the delay bound d for some packets. Though an implementation can choose to drop such packets, we do not, for simplicity. Moreover, this allows us to set the smoothing delay budget $d = 0$, which degenerates to the unsmoothed case for comparison.

IV. Simulation Study

In this section we study the effectiveness of our proposed edge conditioner in reducing burstiness of short and long range dependent input traffic, and the corresponding impact on loss and delay in a simple OPS network. The system is time-slotted, with notional slot size of 1μsec, consistent with earlier works [9] and current optical crossbar technology [10]. Each link operates at 10 Gbps per wavelength, and optical packets have fixed length of 1250 bytes such that they fit exactly in one slot. Our generic core switch is assumed to have a shared-memory architecture [11] as shown in figure 6. No wavelength converters are employed, so each wavelength traverses its own switching plane. Output port contentions are resolved using FDL buffers of capacity D, which denotes a set of FDLs of increasing length that provide delays of $1, 2, \ldots, D$ slots for all wavelengths. We simulate the simple network topology shown in figure 7, consisting of eight edge nodes, each equipped with a conditioner, feeding traffic into one core node having one output link. All our simulations assume a single wavelength, and load each input link

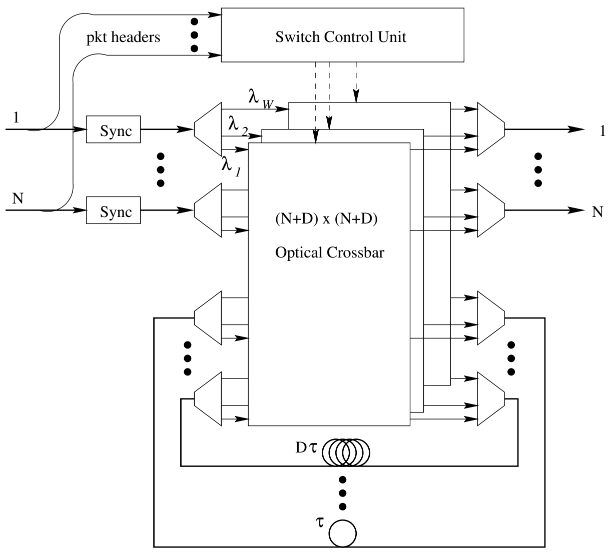

Fig. 6. Generic core switch architecture

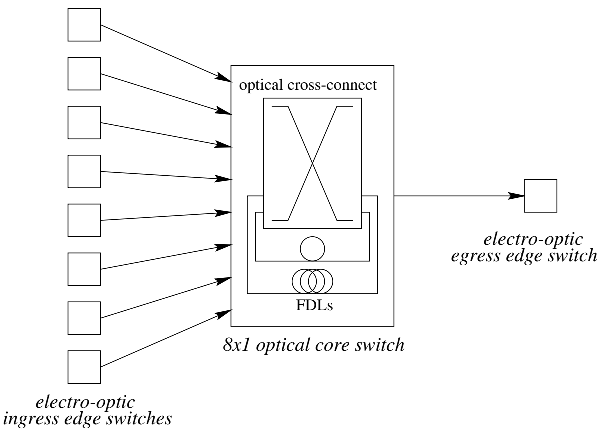

Fig. 7. Simulation topology

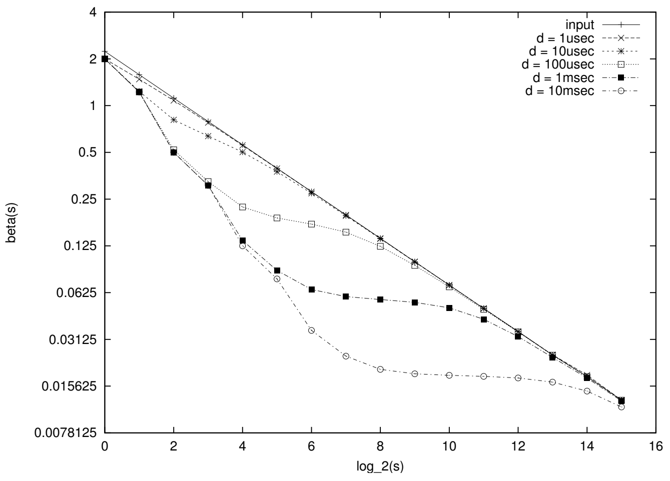

Fig. 8. Burstiness $\beta(s)$ vs. time-scale $\log_2(s)$ for Poisson traffic

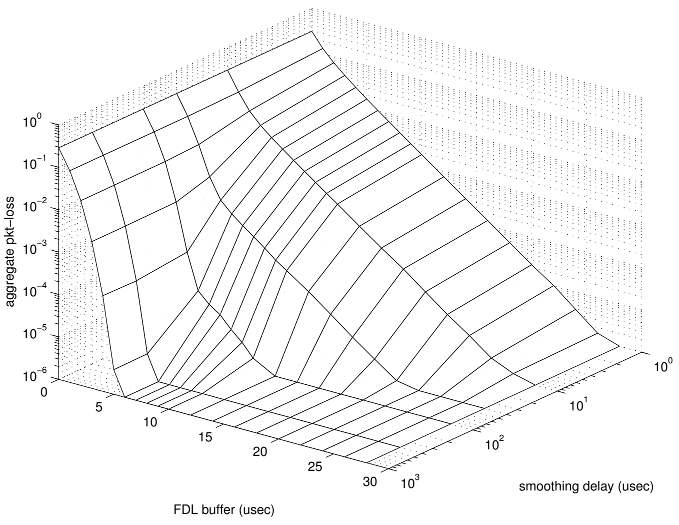

Fig. 9. Loss vs. edge smoothing delay and core FDL buffers for Poisson traffic

to 10% of link capacity; the output link is thus loaded at 80%. Each simulation result corresponds to a run of at least 60 million packets. Our burstiness measure $\beta(s)$ corresponds to the coefficient of variation of the traffic volume over time intervals 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.

Our first simulation considers short-range dependent Poisson input traffic, whereby fixed-size optical packets at the edge nodes have exponential inter-arrival times. Figure 8 plots burstiness $\beta(s)$ versus s on log scale for the input traffic stream, and shows a straight line with slope $-(1-H) = -0.5$ confirming $H = 0.5$ indicative of short-range dependent traffic. The figure also plots the burstiness of the traffic released by the conditioner, for smoothing delay bounds d of 1μsec, 10μsec, 100μsec, 1msec, and 10msec. The burstiness at time-scales of one slot (i.e. $\log_2 s = 1$) is invariant to conditioning, but as the time-scale increases, smoothing significantly lowers burstiness in comparison to the input traffic. As expected, the larger d is, the smoother the traffic ouput by the conditioner. At time-scales larger than d, the burstiness converges back to that of the input stream, since smoothing ceases to be effective beyond the delay "window" d available to the conditioner. Since OPS networks are expected to have small buffers, a relatively small d at the edge conditioners should effectively lower burstiness at short time-scales, helping reduce losses. Figure 9 plots the loss at the core OPS switch, as a function of core FDL buffer capacity and edge conditioning delay budget. Note first that if the OPS core has no buffering, losses are invariant to smoothing, as predicted by theory [1]. Also note that losses fall as FDL buffer capacity increases, and also as the smoothing delay increases, showing that our edge conditioning can substitute for core FDL buffering in reducing losses in the OPS network. To achieve loss rates of around 10^{-4} in this example, one can choose to either equip the core with 16μsec of buffering, or alternatively equip the core with just 4μsec of buffering while employing our conditioner at the edge nodes that introduce an additional delay bounded by 1msec. Since each μsec of FDL buffering translates to around 200 metres of fibre coil, the cost benefit of employing our edge conditioners can be significant.

Our second scenario feeds each edge node with long range dependent (LRD) traffic which has in recent years been shown to be more representative of real traffic in data networks [12].

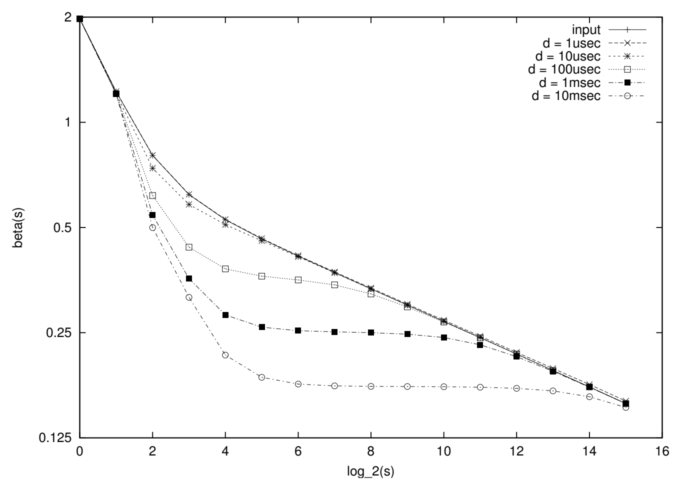

Fig. 10. Burstiness $\beta(s)$ vs. time-scale $\log_2(s)$ for LRD traffic

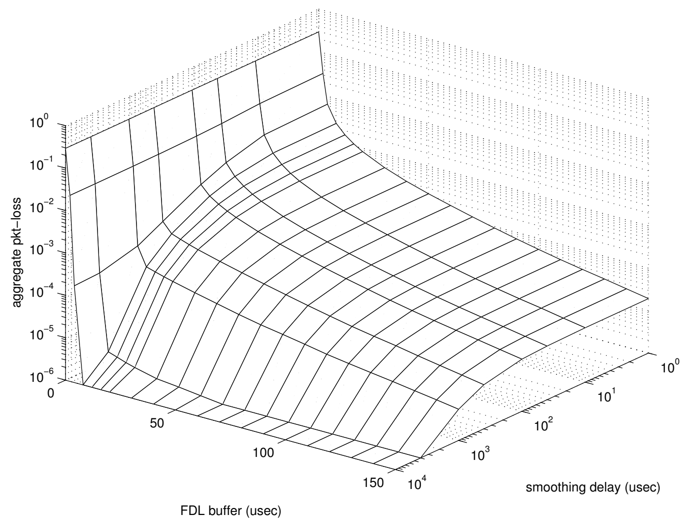

Fig. 11. Loss vs. edge smoothing delay and core FDL buffers for LRD traffic

We generate LRD traffic using Norros' self-similar traffic model [13], with our filtering method developed in [14] that can generate very long sample paths of fractional Gaussian noise. We set the Hurst parameter $H = 0.85$, and the average link loads are as in the previous case. Figure 10 plots the burstiness $\beta(s)$ versus s on log scale for the input traffic, and the slope of $-(1 - H) = -0.15$ validates the Hurst parameter setting of 0.85 (the different slope at lower time-scales is explained by the discretization of the ideal fluid model required to generate packets). The figure also shows burstiness of the ouput produced by our conditioner with delay budget d of 1μsec, 10μsec, 100μsec, 1msec, and 10msec. Once again our conditioner is very effective in reducing burstiness up to time-scales corresponding with the conditioner delay budget. Figure 11 shows how this translates into loss performance at the OPS core node, as the FDL buffer capacity and edge conditioner delay budget are varied. Note first that in the absence of conditioning (smoothing delay budget of $10^0 = 1$ slot), the losses fall off sub-exponentially (following a power-law) as the FDL buffer capacity increases. This is typical for LDR traffic, and signifies that the incremental cost of FDLs required to reduce the loss by a desired amount gets progressively higher. Now observe that increasing the conditioner delay budget reduces losses, though the fall is slower than for Poisson traffic. In the absence of conditioning, loss rates of around 10^{-4} in this example require FDL buffer capacity of 80μsec at the core, whereas with conditioners that add only 1msec of delay, the required FDL buffer capacity is no more than 16μsec, which translates to a substantial savings in cost. These results indicate that our conditioner is effective for both short- and long-range dependent input traffic.

V. SUMMARY

This paper investigated the use of traffic conditioning at the optical edge to reduce contention losses in core optical packet switched (OPS) networks with very limited buffering. We explored novel conditioning mechanisms derived from known off-line optimum smoothing techniques for stored video traffic. We presented an on-line real-time conditioner that has constant amortised computational cost per packet arrival, and is implementable at the high data rates required by optical edge nodes. Via simulation, we demonstrated the efficacy of our conditioner in reducing burstiness and losses at an OPS core node for both short and long range dependent traffic input, which translates into significant cost savings in the design of OPS networks.

Our future work targets extensions of the above conditioner to include multiple traffic classes with distinct delay requirements. We are also working on quantifying the benefits accruing from the use of our conditioner in more extensive OPS network topologies.

REFERENCES

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

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

[3] 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*, 6(4):397–410, August 1998.

[4] S. Sen, J. Rexford, J. Dey, J. Kurose, and D. Towsley. Online Smoothing of Variable-Bit-Rate Streaming Video. *IEEE Trans. Multimedia*, 2(1):37–48, Mar 2000.

[5] 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*, pages 447–454, New York, NY, Mar 1999.

[6] G. Cao, W. Feng, and M. Singhal. Online Variable-Bit-Rate Video Traffic Smoothing. *Computer Communication*, 26(7):639–651, 2003.

[7] Y. Mansour, B. Patt-Shamir, and O. Lapid. Optimal Smoothing Schedules for Real-Time Streams. In *ACM Symposium on Principles of Distributed Computing*, pages 21–29, Portland, OR, 2000.

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

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

[10] T. McDermott and T. Brewer. Large-Scale IP Router using a High-Speed Optical Switch Element. *J. Optical Networking*, 2(7):229–240, Jul 2003.

[11] S. Yao, S. Dixit, and B. Mukherjee. Advances in photonic packet switching: an overview. *IEEE Comm. Magazine*, 38(2):84–94, Feb 2000.

[12] W. E. Leland, M. S. Taqqu, W. Willinger, and D. V. Wilson. On the self-similar nature of ethernet traffic (extended version). *IEEE/ACM Transactions on Networking*, 2(1):1–15, February 1994.

[13] I. Norros. On the use of Fractional Brownian Motion in the Theory of Connectionless Traffic. *IEEE J. Selected Areas in Comm.*, 13(6):953–962, Aug 1995.

[14] D. Ostry. Fast synthesis of accurate fractional Gaussian noise. *Submitted for publication*, 2004.