Enabling a Bufferless Core Network Using Edge-to-Edge Packet-Level FEC

Arun Vishwanath[†], Vijay Sivaraman[†], Marina Thottan[‡] and Constantine Dovrolis[§]
[†]School of EE&T, University of New South Wales, Sydney, Australia
Emails: {arunv@ee.unsw.edu.au, vijay@unsw.edu.au}
[‡]Networking Research Lab, Bell-Labs Alcatel Lucent, USA, Email: {marinat@alcatel-lucent.com}
[§]College of Computing, Georgia Institute of Technology, USA, Email: {dovrolis@cc.gatech.edu}

Abstract— **Internet traffic is expected to grow phenomenally over the next five to ten years, and to cope with such large traffic volumes, core networks are expected to scale to capacities of terabits-per-second and beyond. Increasing the role of optics for switching and transmission inside the core network seems to be the most promising way forward to accomplish this capacity scaling. Unfortunately, unlike electronic memory, it remains a formidable challenge to build even a few packets of integrated all-optical buffers. In the context of envisioning a bufferless (or near-zero buffer) core network, our contributions are threefold: First, we propose a novel edge-to-edge based packet-level forward error correction (FEC) framework as a means of combating high core losses, and investigate via analysis and simulation the appropriate FEC strength for a single core link. Second, we consider a realistic multi-hop network and develop an optimisation framework that adjusts the FEC strength on a per-flow basis to ensure fairness between single- and multi-hop flows. Third, we study the efficacy of FEC for various system parameters such as relative mixes of short-lived and long-lived TCP flows, and average offered link loads. Our study is the first to show that packet-level FEC, when tuned properly, can be very effective in mitigating high core losses, thus opening the doors to a bufferless core network in the future.**

I. Introduction

The Internet has witnessed tremendous growth over the past two decades, both in terms of user traffic and core link capacities. Current Internet traffic is already in the exabytes, and projections show global IP traffic will reach zettabytes in the next five years [1]. This has led to a significant increase in the power/energy requirements of the associated networking infrastructure. Despite the fact that the majority of the power consumed is in home area networks and consumer devices, it is clear that the density of power consumption is highest at core routers that are being scaled to switch terabits-per-second of bandwidth to support the increased traffic demand.

As router line card rates continue to increase, we are approaching the limits of semiconductor (SRAM and DRAM) technology in terms of switching speeds, power savings and heat dissipation [2], [3]. Packet buffers used extensively in today's high-speed line cards are arguably an integral part of every router (or switch) as they absorb transient bursts of traffic, and thus play a fundamental role in keeping packet loss to a minimum. On the downside, they introduce delay and jitter, and are largely responsible for the high cost, power consumption and heat dissipation in core routers. This observation has forced high capacity router/switch designers, and network providers to consider leveraging the use of optics for switching and transmission in core routers [4]. For example, recent work has demonstrated working prototypes of all-optical packet switched routers: IRIS (Integrated Router Interconnected Spectrally) [5] and LASOR [6]. These routers employ integrated optical buffers [7], [8]. However, incorporating even a few packets of buffering using an all-optical on-chip memory is a formidable challenge due to the inherent complexity associated with maintaining the quality of the optical signal and the physical size limitations of the chip [9]. At the moment, our IRIS router is capable of buffering 100 nanosec worth of data. At 40 Gbps line rate, this translates to 500 Bytes, which is not sufficient to buffer a typical 1500 Byte Internet packet.

Given these challenges and limitations associated with all-optical buffering, in this paper we investigate if we can enable a high-speed wide-area bufferless (or near-zero buffer) core optical network capable of delivering acceptable end-to-end performance. The notion of a bufferless core network is not outlandish: prior studies on router buffer sizing (surveyed in our recent article [10]) indicate that very small router buffers of the order of 10-20 packets suffice for good end-to-end TCP performance, and current work in [11] shows similar results for near-zero-buffer networks (though they assume wavelength-conversion capability). We wish to emphasise that this paper does not attempt to derive the minimum possible number of buffers to keep loss rate below an acceptable threshold, as have been proposed in the past. Instead, the novelty of this work is that we allow losses to occur in the core but we recover from them using packet-level FEC.

The concern in a bufferless network is that packet losses can be unacceptably high. Several techniques can be applied to deal with these losses. Wavelength conversion and hybrid optical/electronic switching have been proposed [11] for reducing loss; however, these are expensive and can contribute significantly to the cost of the optical switch. Traffic shaping/pacing at the electronic edges [12] can make traffic entering the core smoother, potentially reducing loss; this is effective for losses arising from bursty traffic, but does not address random contention loss. Though coordinated scheduling of packets to prevent contentions in the core can be envisaged, as in [13], such an approach is too complicated to be implemented in practice for an arbitrary wide-area network. In this paper we focus on packet-level FEC coding by the electronic edge routers as a method for recovering from packet loss in the bufferless core. Our reasons for choosing FEC are discussed next.

A. Motivation for choosing FEC

An interesting question that we ask in the context of a bufferless core network is: do we expect that the losses in the core will be bursty? Losses in packet networks are known to be bursty [14], but that is in the case of drop-tail queues and also when the lossy links are the flows' bottlenecks. In practice, the capacity of the core links is orders of magnitude higher than the access/edge links [15], thus ensuring that the bandwidth bottleneck for a flow is either at the access/edge link, and not at any core link. Thus, losses will occur in the core, not because a single flow can saturate the core link with a burst of back-to-back packets, but because we can have the simultaneous arrival of two or more packets from different edge/core links in a given time-slot. Thus, loss at core links is due to *contention*, not congestion.

It is well-known that FEC works best when losses are random, and hence our choice of using FEC is primarily inspired by this fact that packet losses in a bufferless core will occur randomly (due to contention) and not in a bursty manner (due to congestion).

Other reasons for choosing FEC are that it is a well established technique, cost-effective, and can be easily implemented in hardware. FEC can introduce some bandwidth overhead, but this is a small price to pay for building scalable and power efficient bufferless core nodes, and also because ISPs typically operate their core networks at relatively low loads ($\approx$ 20-30%) [16]. In addition, packet-level FEC has, to the best of our knowledge, not been studied before in the context of a bufferless network. Finally, it also complements other techniques outlined above to minimise loss, namely that it is possible to combine FEC with traffic shaping/pacing etc.

B. Our contributions

In the context of recovering from lost packets in a bufferless core network, our contributions are threefold.

1) First, we propose a novel edge-to-edge based packet-level FEC framework as a means of combating high core loss rates, and investigate via analysis and simulation the appropriate FEC strength for a single bufferless link. The analysis is corroborated against simulation and shows that significantly high end-to-end TCP goodput can be obtained over a bufferless core link.

2) Second, we consider a realistic multi-hop bufferless network (the NSFNet) and show that multi-hop TCP flows can have significantly worse performance than single-hop flows. To address this unfairness, we develop an optimisation framework that allows determination of FEC strength on a per-flow basis to achieve max-min fairness. We also propose a workable heuristic that achieves good fairness by choosing the FEC strength based solely on hop-length for given average link load in the network.

3) Finally, we study the efficacy of FEC for various system parameters such as relative mixes of short- and long-lived TCP flows, and average link loads. Our study is the first to show that packet-level FEC, when tuned properly, can be very effective in overcoming high core losses, thus mitigating a major obstacle for the realisation of bufferless core networks in the future.

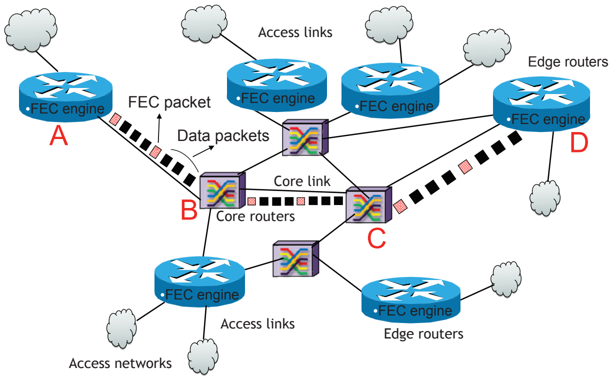

Fig. 1. Topology to illustrate the edge-to-edge FEC framework

The rest of the paper is organised as follows. In Section II, we describe our edge-to-edge packet-level FEC framework. In Section III, we analyse a bufferless core link, and study its performance via simulations using the dumbbell topology and extend it to the NSFNet network. In Section IV, we derive our max-min optimisation framework along with the heuristic algorithm and analyse its performance on the NSFNet. In Section V, we discuss the performance of the FEC framework using short- and long-lived TCP flows, and for different average link loads. We conclude the paper in Section VI and point to directions for future work.

II. Edge-to-Edge Packet-level FEC framework

Fig. 1 shows a small segment of a typical ISP network comprising of electronic edge routers and optical core routers. The distinguishing feature between the core and edge routers is that the core router links are bufferless (near-zero buffer) while the electronic router links have large buffers. The FEC framework presented in this paper is implemented at the electronic edge routers across the aggregate packet flow between an edge router (ingress) to another edge router (egress), and not for flows between any two end-hosts. Hence our implementation is an *edge-to-edge* based FEC implementation. As an example, all traffic that enters the core network from an ingress router say in New York city and exits at an egress router say in Los Angeles is viewed as an edge-to-edge flow, and it is protected by the FEC redundancy. It is important to note that the FEC scheme is scalable since if an ISP network has N edge routers, then each electronic edge router will compute FEC packets for just $N-1$ edge-to-edge flows. Further, the proposed FEC framework has the advantage of being completely transparent to the end-hosts with control purely with the ISP.

The ingress edge routers receive traffic from applications running at various end-hosts on the access network via access links (DSL, cable modem, etc.), classifies the traffic on an edge-to-edge basis, and computes the FEC packets per egress edge router. The FEC framework discussed in this work uses the well-known and simple XOR scheme. The strength of FEC is the number of data packets over which the XOR operation is performed, henceforth referred to as block-size. The scheme has a unique property, namely that if in addition to a block of k data packets the ingress router also transmits

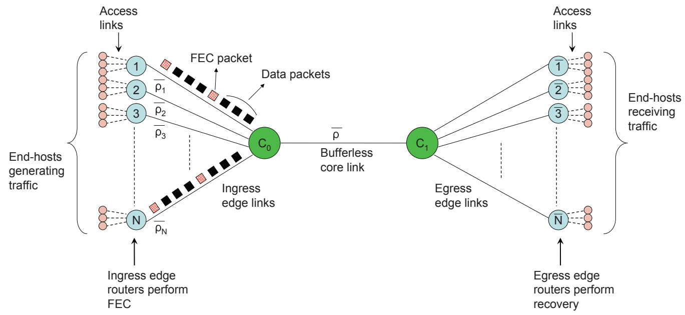

Fig. 2. Example dumbbell topology with a single core link

the XOR of the k data packets (thereby transmitting $k + 1$ packets for every k packets), the corresponding egress router will be able to recover from loss provided there is only one missing data packet.

For ease of illustration, the figure shows traffic flowing from edge router A to edge router D along the path A-B-C-D. Assuming the block-size is three, A keeps a running XOR of the data packets destined to D. After every third packet, the FEC packet comprising the XOR value is transmitted and the XOR is cleared. D also maintains a running XOR of the packets it receives from A. If it detects exactly one lost data packet in the window of $k + 1$ packets, the running XOR is XOR'd with the FEC packet to recover the lost data packet, which is then forwarded on. In the case of zero (or > 1) loss, recovery is not possible and the running XOR is cleared.

To deal with variable-size packets, we assume that the size of each FEC packet equals the MTU size in the optical core. For XOR purposes, smaller packets are treated as being padded with zeros. The egress router must have sufficient information to recover a missing data packet correctly. Therefore, the FEC packet constructed by the ingress router takes into account the header information and the payload of each data packet. This ensures that the reconstructed packet will be identical to the original (missing) data packet.

The edge routers do not introduce much overhead in computing the FEC packet (since the XOR computation can be performed in real-time), nor do they require significant additional memory for the XOR process. The extra memory required is for storing one FEC packet per edge router in the network. This is however not a concern because FEC is performed at the edge routers that have large electronic memory. Insertion of one FEC packet for every k data packets increases the bandwidth requirement by a fraction $1/k$, which may be acceptable in a typical optical core that has abundant bandwidth but limited buffering.

The recovery operation can introduce a delay that in the worse case is the time to receive all k subsequent packets in a window of $k + 1$ packets (assuming the first data packet is lost). The delay will be very small in practice because we have eliminated the large buffers that exist in today's core routers (millions of packets), and we consider edge-to-edge flows with possibly thousands/millions of packets per second.

III. Role of FEC on Single Link Goodput and loss

We implemented the above edge-to-edge FEC framework in *ns-2* (version 2.33), and apply it to the single core-link dumbbell topology shown in Fig. 2. Ten edge links feed traffic into the bufferless core link at router C_0, with each edge link in turn fed by three access links. The 30 end-hosts each have 5 TCP (Reno) agents, and the network therefore simulates 150 long-lived TCP flows. Similarly the TCP flows are sinked by the 30 end-hosts on the right. The propagation delays on the access and edge links are uniformly distributed between $[1, 5]$ ms and $[5, 15]$ ms respectively, while the core link C_0-C_1 has delay 30 ms. In line with the prevalent situation in today's networks, we ensure that the TCP flows are bottlenecked at access links, rather than the core which typically has much higher capacity. Our access link speeds are uniformly distributed in $[3, 5]$ Mbps, all edge links operate at 40 Mbps, and the core link at 400 Mbps. For these link speeds, it can be seen that the access link is the bottleneck since each flow's fair-share of the bandwidth on the access links varies between 0.6-1 Mbps, while on the edges and the core it is 2.67 Mbps. The start time of the TCP flows is uniformly distributed in the interval $[0, 10]$ sec and the simulation is run for 35 sec. Data in the interval $[20, 35]$ sec is used in all our computations so as to capture the steady-state behaviour of the network.

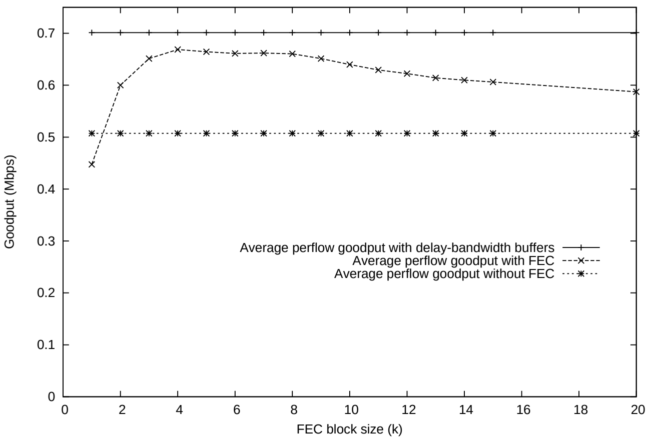

Fig. 3. Average perflow goodput for 150 TCP flows on dumbbell topology

We measure the average per-flow TCP goodput for each setting of the FEC block-size k in simulation. We use goodput as a metric since it has been argued to be the most important measure for end-users [17], who want their transactions to complete as fast as possible. It should be mentioned that *ns-2* does not permit setting buffer size to zero as it simulates store-and-forward rather than cut-through switches. Thus, our simulations use a buffer size of 1 KB (the closest we can come to zero buffer) to accommodate a single TCP packet (TCP packets in our simulation are of size 1 KB) that is stored and forwarded by the switch.

Fig. 3 shows the per-flow TCP goodput as a function of block-size k. For comparison, it also depicts, via horizontal lines, the average goodput without FEC (the bottom line) and the average goodput if the core link were to have sufficient (delay-bandwidth) buffering of around 12.5 MB (top line). Large buffers yield a per-flow goodput of 0.7 Mbps, while eliminating buffers reduces this goodput to 0.5 Mbps, a sacrifice in goodput of nearly 30%. Employing edge-to-edge FEC over the bufferless link can improve per-flow goodput substantially, peaking at nearly 0.68 Mbps when the FEC block-size k is in the range of 3-6, and bringing the per-flow TCP goodput for the bufferless link to within 3% of a fully-buffered link. This small sacrifice in goodput is a worthy price to pay for eliminating buffering at router C_0.

Another interesting aspect to note from Fig. 3 is that TCP goodput initially increases with FEC block-size k, reaches a peak, and then falls as k increases. Qualitatively, this is because stronger FEC (i.e., smaller block-size k) in general improves the ability to recover from loss, but is also a contributor to loss since it increases the load on the link by introducing redundant packets. In the next subsection, we capture this effect via a simple analytical model to determine the optimal setting of FEC block-size that minimises loss on a bufferless link.

A. Analysis

We develop a simple analytical model to quantitatively understand the impact of FEC strength on edge-to-edge loss, and to identify the block-size settings that achieve low loss and consequently larger goodput. Our analysis makes several simplifying assumptions:

1) The end-hosts that generate traffic are independent of each other. Consequently, traffic coming into the core links from the various edge links are also independent of one another. This is a reasonable assumption, even for TCP traffic when the number of flows is large enough [2]. Moreover, we assume that the contribution to the load on the core link from each of the edge links is similar.

2) We assume a time-slotted cut-through link, with loss happening if and only if two or more packets arrive to the link for transmission in the same slot (since the core is bufferless).

3) We do not model feedback, i.e., TCP's adjustment of rate in response to loss. Instead, we will assume that the steady-state load on the link is known aforehand.

Denote by ρ_i the original load (i.e., load without FEC) on each of the edge links (i, C_0), $i \in \{1, N\}$ (see Fig. 2). The offered load at the core link C_0-C_1 is then $\rho = \sum_{i=1}^{N} \rho_i$. Now, if each edge link performs FEC using block-size k, then the new load $\overline{\rho_i}$ on each of these edge links is

$$\overline{\rho_i} = \left(\frac{k+1}{k}\right)\rho_i \tag{1}$$

since FEC inserts one additional packet for every k data packets. Correspondingly, the offered load $\overline{\rho}$ post-FEC at the core link is

$$\overline{\rho} = \sum_{i=1}^{N} \overline{\rho_i} = \sum_{i=1}^{N} \left(\frac{k+1}{k}\right)\rho_i = \left(\frac{k+1}{k}\right)\rho \tag{2}$$

Assuming that each edge link contributes equally to the load $\overline{\rho}$ on the core link, the probability that in a given time-slot a packet arrives from any chosen edge link is $\overline{\rho}/N$ where N denotes the number of edge links. The loss probability $\mathbb{L}_c$ at the core link in a chosen slot is then the probability that packets arrive from two or more edge links:

$$\mathbb{L}_c = \sum_{i=2}^{N} \binom{N}{i} \left(\frac{\overline{\rho}}{N}\right)^{i} \left(1 - \frac{\overline{\rho}}{N}\right)^{N-i} \tag{3}$$

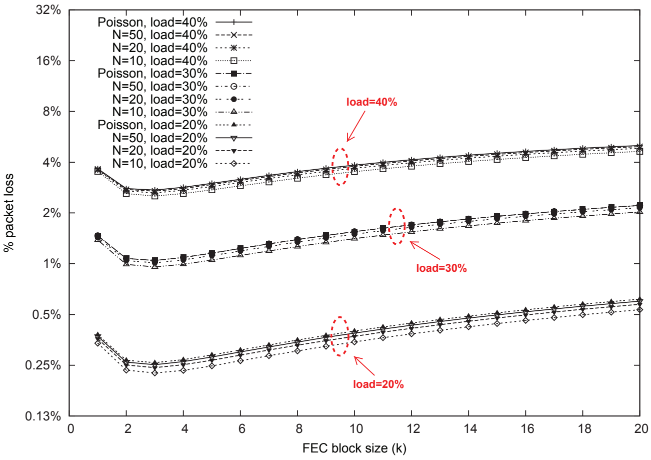

Fig. 4. % edge-to-edge packet loss for different loads and fan-in

It is worthwhile mentioning that for large N, the binomial distribution above converges to the Poisson distribution with the same mean. The number of input links (N) interfacing with a core router is called fan-in, and is fairly large in practice. For example, the mean number of working ports in a core network with 60 nodes and uniform full mesh of demands is about 354 [18]. Therefore, Eq. (3) can be approximated by

$$\mathbb{L}_c = 1 - e^{-\overline{\rho}}(1 + \overline{\rho}) \tag{4}$$

Knowing the probability of packet loss in the core, we can now estimate the edge-to-edge packet loss probability $\mathbb{L}_e$ by computing the expected number of irrecoverably lost packets in a window of $k+1$ packets (comprising k data packets and one FEC packet) as follows:

$$\begin{aligned}\mathbb{L}_e = \mathbb{L}_c \sum_{j=1}^{k} \binom{k}{j} (\mathbb{L}_c)^j (1 - \mathbb{L}_c)^{k-j} \frac{j}{k} + \\ (1 - \mathbb{L}_c) \sum_{j=2}^{k} \binom{k}{j} (\mathbb{L}_c)^j (1 - \mathbb{L}_c)^{k-j} \frac{j}{k}\end{aligned} \tag{5}$$

The first term on the right in Eq. (5) captures the case when the FEC packet is lost along with j data packets, in which all j data packets are irrecoverable, while the second term captures the case when the FEC packet arrives and $j \geq 2$ data packets are lost, in which case the j packet losses are irrecoverable. Eq. (5) can be simplified yielding

$$\mathbb{L}_e = \mathbb{L}_c \left[1 - (1 - \mathbb{L}_c)^k\right] \tag{6}$$

Eq. (6) states that a data packet is irrecoverably lost only if it is lost in the core (with probability $\mathbb{L}_c$) **and** not all other k packets in the window (this includes the FEC packet) are successfully received (otherwise the lost data packet can be reconstructed).

Eq. (6), in conjunction with Eq. (3) (or Eq. (4)) and Eq. (2), can be used to directly estimate edge-to-edge loss $\mathbb{L}_e$ as a function of FEC block-size k. In Fig. 4 we plot on log-scale the edge-to-edge packet loss probability as a function of the block-size k for different values of load ρ $(20\%, 30\%, 40\%)$ and fan-in N. We first observe that for a given load, loss is not very sensitive to the fan-in N at the core link, and further that the Poisson limit seems to be a good approximation that frees us from having to consider the fan-in parameter N explicitly in the model. The most important observation to emerge from this plot is that for a given load, the loss decreases with block-size k, reaches a minimum, and then starts increasing as the block-size gets larger. This provides some explanation as to why the simulation plot in Fig. 3 shows TCP goodput to first increase and then fall with block-size k, as TCP throughput is inversely related to the square root of end-to-end packet loss [19]. The figure also gives us some estimate of the strength of FEC required ($k = 3$ in this case) for minimising loss: the recovery benefit of stronger FEC (i.e., lower k) is outweighed by the overhead it introduces in terms of load, while weaker FEC (i.e., larger k) does not sufficiently recover lost data packets. In the next subsection we explore if similar observations extend to a more complex multi-hop topology.

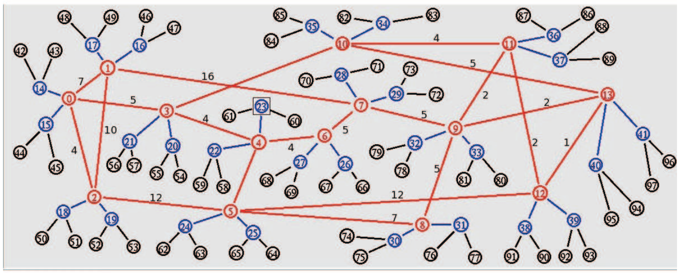

Fig. 5. Example NSFNet topology with 2 access links per edge node and 2 edge links per core node from *ns-2* network animator

B. FEC performance in a multi-hop network

Having seen the benefits offered by FEC for a single link, we now evaluate its performance on a more general wide-area network topology. To this end, we choose the NSFNet topology shown in Fig. 5 as our representative core network, which is made up of core routers (numbered 0 to 13) and the bufferless optical links interconnecting them. The numbers along the core links indicate the propagation delay in milliseconds. For the sake of clarity, the figure shows only two edge routers connected to every core node and each edge router receives traffic from only two end-hosts (via access links). However, in all our simulations, we consider larger number of edge/access links, as described next.

We consider ten edge links feeding traffic into every core router, and each edge router in turn is fed by five access links. All core links operate at 1 Gbps, all edge links at 100 Mbps, and the access link rates are uniformly distributed between $[7, 10]$ Mbps, to reflect a typical home user. These numbers ensure that the core is not the bottleneck for any TCP flow. The destination end-hosts are chosen randomly such that every flow traverses at least one hop on the core network; in all there are 3480 TCP flows in the network comprising of 784 one-hop flows, 1376 two-hop flows and 1320 three-hop flows. We assume all flows to be long-lived (Section V describes results when both short-lived and long-lived TCP flows coexist). Data in the interval $[20, 35]$ sec is used for the computations in order to capture the network's steady state behaviour.

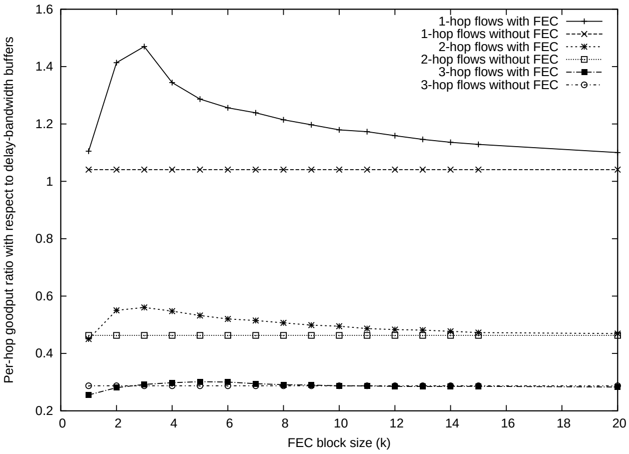

Fig. 6. Ratio (average goodput with FEC to average goodput with delay-bandwidth buffers) for 1-, 2-, 3-hop TCP flows on NSFNet topology

Fig. 6 plots the ratio (average goodput with FEC to the corresponding average goodput with delay-bandwidth buffers) for 1-, 2- and 3-hop TCP flows as a function of the block-size (the maximum number of hops along the shortest path between any two core nodes on the NSFNet is three). It makes sense to use the goodput obtained with delay-bandwidth buffers in the core as the benchmark because core routers today have large buffers [20], and the performance witnessed by ISPs is typically under such large buffering. We note from the simulation results that with delay-bandwidth buffers, the load on the core links varies between 7% to 38% with the average load being ≈ 24%. These numbers are realistic and fall in the regime in which most ISPs operate their networks today. The figure also indicates, via horizontal lines, the corresponding goodput ratios in the non-FEC case.

A fundamental point that we can infer from the figure is that the bufferless core network (with and without FEC) is very unfair towards multi-hop flows when compared to single-hop flows. On average, 1-hop flows with FEC (at $k = 3$) achieve nearly 1.5 times the goodput (1.05 times in the non-FEC case) when compared to what they achieve when all core routers have delay-bandwidth buffers. This means that 1-hop flows perform better in a bufferless network than in a fully-buffered network! The ratio reduces to 0.56 for 2-hop flows and further to just 0.3 for 3-hop flows, indicating heavy skewing in favour of short-hop flows.

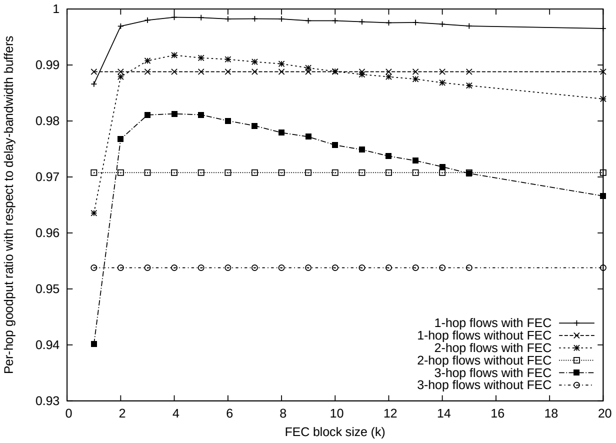

Fig. 7. Ratio (average goodput with FEC to average goodput with delay-bandwidth buffers) for 1-, 2-, 3-hop Poisson flows on NSFNet topology

To see if this large degree of unfairness is predominantly due to the closed-loop nature of TCP or if the same phenomenon can be observed in the case of open-loop UDP, we simulate Poisson traffic with a mean rate of 1 Mbps using the same simulation setting as before. Comparing Fig. 7 that plots the goodput ratios of 1-, 2- and 3-hop Poisson flows with Fig. 6, we note that, although FEC offers some benefit, 1-, 2- and 3-hop Poisson flows by themselves (i.e., without FEC) achieve near-optimum (optimum is 1) goodput ratio of 0.98, 0.97 and 0.95 respectively. These numbers are in stark contrast to what TCP flows are able to obtain.

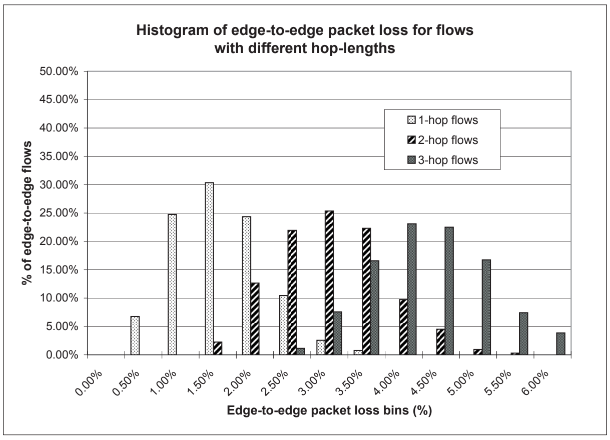

Fig. 8. Histogram of edge-to-edge packet loss for TCP flows with different hop lengths (and no FEC) on the NSFNet

To explain why multi-hop TCP flows perform so poorly, we plot in Fig. 8 the histogram of edge-to-edge packet loss (for the non-FEC case) for flows with different hop-lengths. We can observe that while over 95% of 1-hop flows experience loss only in the range 0.5-3%, it increases to 1.5-4.5% for 2-hop flows, and further to 2.5-6% for 3-hop flows. To appreciate the impact these numbers have on the edge-to-edge performance, if we assume, for example, that the loss rate doubles from 1% to 2%, then the throughput of open-loop UDP traffic reduces by roughly 1%, whereas for closed-loop TCP traffic, it reduces by nearly 30%, since the average throughput of a TCP flow in the congestion avoidance mode is inversely proportional to the square root of packet loss. TCP goodput, however, can be much lower, as seen by Fig. 6. The relatively higher loss rates for 2- and 3-hop flows result in their fair-share of the bandwidth being unfairly utilised by 1-hop flows (since TCP is inherently greedy and is designed to exploit as much of the bandwidth as available), leading to unfairness. These results motivate us to devise a scheme that provides fairness to both single- and multi-hop flows, which will be the focus of the next section.

IV. FEC for a Bufferless Network and Fairness

We observed from the results in the previous section that in a bufferless network, multi-hop TCP flows can experience significantly lower end-to-end goodput than single-hop flows, leading to unfairness. In this section, we address this deficiency by developing a framework that ensures fairness to both single- and multi-hop flows.

A. Analysis

We denote $\lambda_{i,j}$ to be the offered load to the core network by the edge-to-edge flow between ingress router i and egress router j, henceforth represented as (i,j), under the assumption that they are not using FEC. Let $\overline{\lambda_{i,j}}$ be the new load to the core network when the flow employs FEC using $k_{i,j}$ as its block-size. Consequently,

$$\overline{\lambda_{i,j}} = \left(\frac{k_{i,j}+1}{k_{i,j}}\right)\lambda_{i,j} \tag{7}$$

Under the assumption that packet arrivals at a core link (u,v) in every time-slot is Poisson with mean $\overline{\lambda_{i,j}}$, we can compute the packet loss probability $L_c^{u,v}$ at the link as the probability of two or more simultaneous packet arrivals from all flows that traverse (u,v). Thus,

$$\begin{aligned} L_c^{u,v} &= 1 - \left(e^{-\sum_{(u,v)\in r(i,j)} \overline{\lambda_{i,j}}}\right) - \\ &\qquad \left(\sum_{(u,v)\in r(i,j)} \overline{\lambda_{i,j}} \times e^{-\sum_{(u,v)\in r(i,j)} \overline{\lambda_{i,j}}}\right) \\ &= 1 - e^{-\sum_{(u,v)\in r(i,j)} \overline{\lambda_{i,j}}} \left(1 + \sum_{(u,v)\in r(i,j)} \overline{\lambda_{i,j}}\right) \end{aligned} \tag{8}$$

where $r(i,j)$ is the routing path of edge-to-edge flow (i,j). In general, a flow can traverse multiple hops on the core network before reaching the egress edge router. If loss rates on core links are sufficiently small (say 10^{-2} or lower), it is reasonable to assume that edge-to-edge losses are independent and additive over the links the flow traverses. Therefore, denoting $\mathbb{L}_c^{i,j}$ to be the aggregate core path loss probability for the flow (i,j),

$$\mathbb{L}_c^{i,j} = \sum_{(u,v)\in r(i,j)} L_c^{u,v} \tag{9}$$

We can now compute $\mathbb{L}_e^{i,j}$, the edge-to-edge packet loss probability for flow (i,j) by substituting in Eq. (6) the core path loss probability for the flow derived from Eq. (9). Thus,

$$\mathbb{L}_e^{i,j} = \mathbb{L}_c^{i,j}\left[1 - \left(1 - \mathbb{L}_c^{i,j}\right)^{k_{i,j}}\right] \tag{10}$$

Simulation results in the previous section show that bufferless core networks can be unfair towards multi-hop TCP flows. Thus, to achieve fairness, multi-hop flows need more aggressive FEC than single-hop flows. We now capture this notion of fairness by formulating an optimisation problem as follows.

Inputs:

- Offered load $\lambda_{i,j}$ by every edge-to-edge flow (i,j).
- $r(i,j)$ the routing path of the flow (i,j).

Objective function:

$$\min_{k_{i,j}}\left(\max_{i,j}\ \mathbb{L}_e^{i,j}\right) \tag{11}$$

Subject to: $k_{i,j} \in \{1,2,3,\ldots\}$

Output: The set $\{k_{i,j}\}$, which denotes the optimum FEC block-size for every edge-to-edge flow.

The optimisation objective in Eq. (11) is a min-max objective expressed in terms of the edge-to-edge loss rate (conversely, it can be viewed as a max-min objective in terms of edge-to-edge goodput). It seems reasonable to consider the above objective since it ensures that the network bandwidth is assigned to the various (single- and multi-hop) flows in a fair manner, thus preventing the 1-hop flows from exploiting the available bandwidth and penalising the multi-hop flows (assuming that an ISP is equally concerned about multi-hop flows as single-hop flows).

It can be noted that the formulation has a non-linear objective function since the block-size $k_{i,j}$ that we are interested in is in the exponent of Eq. (10). In addition, we also have the constraint that each $k_{i,j}$ must be an integer. Although an optimal solution set exists for the formulation, the above two constraints render the problem intractable to large size networks. Indeed, the problem is NP-Hard since integer linear programming is itself NP-Hard. Thus in what follows, we propose a sub-optimal but practical heuristic that treats flows differently based on their hop-length.

B. Hop-length based simple heuristic

In general, the best block-size to use for an edge-to-edge flow depends on the load on each of the links it goes through, which in turn depends on the FEC strength of the other flows traversing those links. To simplify these interdependencies and reduce the solution space, we consider a heuristic scheme in which flows of identical hop-length h use identical block-size k_h. For this assumption to be reasonable, two flows of similar hop-length should see similar link loads along their paths. One way this could hold is when all links in the network are roughly equally loaded, and further that the relative contribution to load by flows of different hop-lengths is similar across links. If loads vary significantly across links, one could generate a worst case bound in which all links are as heavily loaded as the maximum loaded link in the network. In this section, for purely illustrative purposes, we will assume that all links have the same average load λ and flows of different hop-lengths contribute equally to the load on each link. Specifically, for the NSFNet topology considered, flows are of 1-, 2- or 3-hops, and have offered load $\{\lambda_1, \lambda_2, \lambda_3\}$ given by

$$\lambda_1 = \lambda_2 = \lambda_3 = \frac{\lambda}{3} \tag{12}$$

Denoting $\overline{\lambda_c}$ to be the load on any core link c post FEC, then considering the 1-, 2- and 3-hop flows, $\overline{\lambda_c}$ is given by

$$\overline{\lambda_c} = \left(\frac{k_1+1}{k_1}\right)\lambda_1 + \left(\frac{k_2+1}{k_2}\right)\lambda_2 + \left(\frac{k_3+1}{k_3}\right)\lambda_3 \tag{13}$$

Employing the Poisson assumption, the probability of loss $\mathbb{L}_c$ at a core link c can be expressed as

$$\mathbb{L}_c = 1 - e^{\overline{\lambda_c}}\left(1 + \overline{\lambda_c}\right) \tag{14}$$

Since we are assuming that all core links have the same average load, and hence the same loss rate, the core path loss probability $\mathbb{L}_{c,h}$ for a h-hop flow can be expressed as

$$\mathbb{L}_{c,h} = \mathbb{L}_c h \tag{15}$$

Now, from Eq. (6) the edge-to-edge packet loss probability $\mathbb{L}_{e,h}$ for a h-flow is

$$\mathbb{L}_{e,h} = \mathbb{L}_{c,h}\left[1 - (1 - \mathbb{L}_{c,h})^{k_h}\right] \tag{16}$$

We are now interested in finding the sub-optimal set $\{k_1, k_2, k_3\}$ that minimises the maximum edge-to-edge loss $\mathbb{L}_{e,h}$. Assuming that each k_h can take a value between 1 and 100 (since larger block-sizes are generally not beneficial, as we observed earlier), it is easy to use a simple brute-force approach to determine this set, and using MATLAB, we are able to obtain the result in just a couple of seconds since we only have a million combinations to choose from. For the same simulation setting described in Section III-B (the average NSFNet link load λ for a bufferless network being $\approx 11\%$), the resulting solution is k_1 = 19, k_2 = 4 and k_3 = 2. These results clearly suggest that we need to have different block-sizes for flows with different hop-lengths, and indeed multi-hop flows require a much more aggressive FEC scheme than single-hop flows (since $k_3 < k_1$).

C. Simulation results and fairness on the NSFNet network

Using k_1 = 19, k_2 = 4 and k_3 = 2 (obtained from our heuristic in the previous subsection), we repeated the simulation with identical settings as before (having the same number of TCP flows on the NSFNet topology). Recall that our objective is to minimise the maximum edge-to-edge loss, or equivalently, maximise the minimum edge-to-edge goodput so as to achieve fairness across all flows. We employ the widely used Jain's fairness index [21] as an indicator of the heuristic's performance. It is extremely difficult to determine analytically what the optimum average goodput will be for the various 1-, 2- and 3-hop TCP flows from an overall network perspective, since analysing a single TCP flow is by itself very notorious, and we consider several thousand TCP flows with heterogeneous RTTs flowing through the network. Therefore as our benchmark, we again choose the goodput numbers obtained when all core links have delay-bandwidth buffers. If $\lambda_1^{'}$, $\lambda_2^{'}$ and $\lambda_3^{'}$ are the goodputs of the 1-, 2- and 3-hop flows with large buffers, then the Jain's fairness index FI can be expressed as

$$FI = \frac{\left(\sum_{i=1}^{3} \frac{\lambda_i}{\lambda_i^{'}}\right)^2}{3\left[\sum_{i=1}^{3}\left(\frac{\lambda_i}{\lambda_i^{'}}\right)^2\right]} \tag{17}$$

The fairness index is a real number between 0 and 1, with a higher value indicating better fairness. Fig. 9 shows the fairness index for four pertinent cases - no FEC, FEC with k = 3 for all flows, FEC with k_1 = 19, k_2 = 4 and k_3 = 2, and with delay-bandwidth core buffers.

Following are the salient observations we can draw from the figure. Firstly, the non-FEC case has a rather low fairness index of 0.78, and although setting the block-size to k = 3 improves the performance of 1- and 2-hop flows quite significantly, it fails to lift the goodput of 3-hop flows, and surprisingly performs poorly on the fairness scale when compared to the non-FEC case (0.70 compared to 0.78).

Network setting	Average goodput (Mbps)			Fairness Index
	1-hop flows	2-hop flows	3-hop flows	
No FEC	1.571	0.667	0.391	0.78
k = 3 for all flows	2.219	0.807	0.397	0.70
k_1 = 19, k_2 = 4, k_3 = 2	1.090	0.760	0.596	0.96
delay - bandwidth buffers	1.509	1.440	1.359	1

Fig. 9. Relative fairness indices

Secondly, tuning the block-size according to our heuristic, corresponding to k_1 = 19, k_2 = 4 and k_3 = 2, results in a good fairness index of 0.96, confirming that the network bandwidth is indeed used by the various 1-, 2- and 3-hop flows efficiently. These results provide valuable insight on how a future bufferless core network can be envisaged using the edge-to-edge FEC scheme.

D. Sensitivity analysis of the block-sizes

We now undertake a sensitivity analysis to ascertain how a slight perturbation to the values of k_1, k_2 and k_3 affects the fairness index. Our heuristic suggests using k_1 = 19, k_2 = 4 and k_3 = 2. We vary each k_h such that k_1 takes on values between 17 and 21, k_2 is in the range $\{3, 4, 5\}$, and k_3 is either 2 or 3. The simulation is repeated for each combination of the respective block-size. We note from the results that the fairness index varies between 0.924 (lower by 3.75%) and 0.962 (higher by 0.21%) when compared to 0.96 that was obtained by the heuristic, suggesting that the block-sizes derived by our heuristic algorithm is stable.

Before concluding this section, we wish to highlight that we performed many simulations by varying the distribution of the number of 1-, 2- and 3-hop TCP flows. The results we obtained follow closely the ones we have reported, omitted for brevity.

V. Discussion

In this section, we first study the efficacy of FEC with realistic mixes of short- and long-lived TCP flows, and subsequently outline when the use of FEC can be beneficial.

A. Mix of short-lived and long-lived TCP flows

We have so far analysed the FEC performance using long-lived TCP flows only. The reader may wonder if the proposed FEC scheme offers acceptable goodputs when many of the TCP flows are short-lived (or equivalently, the number of TCP flows is time-varying). This is an important consideration since measurement based studies at the core of the Internet suggest that a large number of TCP flows (e.g. HTTP requests) are short-lived and carry only a small volume of traffic, while a small number of TCP flows (e.g. FTP) are long-lived and carry a large volume of traffic. To incorporate such realistic TCP traffic we consider the closed-loop flow arrival model described in [22], operating as follows. A given number of users perform successive file transfers to their respective destination nodes. The size of the file to be transferred follows a Pareto distribution with mean 100 KB and shape parameter 1.2. These chosen values are representative of Internet traffic, and comparable with measurement data. After each file transfer, the user transitions into an idle or off state. The duration of the thinking period is exponentially distributed with mean 1 sec. We carry out such a simulation in *ns-2* using the same network setting described in Section III-B. The only difference being that 80% of the TCP flows are short-lived with the rest long-lived.

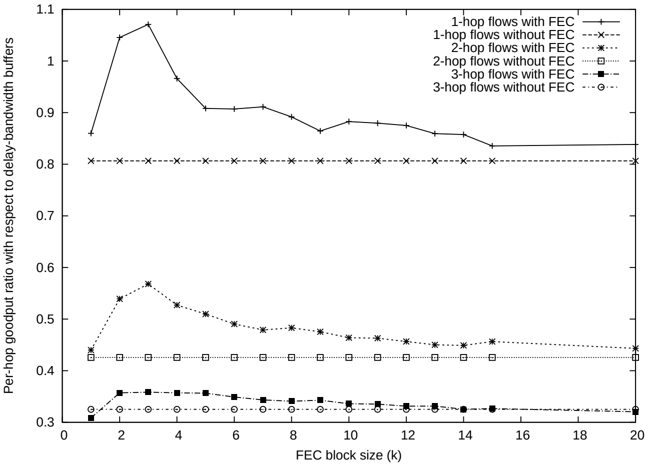

Fig. 10. Ratio (average goodput with FEC to average goodput with delay-bandwidth buffers) for short-/long-lived TCP flows on NSFNet topology

The ratio (goodput with FEC to goodput with delay-bandwidth buffers) for 1-, 2- and 3-hop flows as a function of block-size are shown in Fig. 10. Shown via horizontal lines in the figure are the corresponding ratios for the non-FEC case. We can see that using a block-size of k = 3 across all the flows improves the average per-hop goodput, with 1-, 2- and 3-hop flows being benefited by 32%, 33% and 10% over their non-FEC counterparts. However, the network is not performing fairly because the fairness index for the two scenarios is 0.83 (for k = 3) and 0.86 (no FEC) respectively, although the numbers are higher than the corresponding indices when all TCP flows are long-lived (see Fig. 9). Using our heuristic algorithm, we obtain k_1 = 18, k_2 = 4 and k_3 = 2 as the block sizes for achieving fairness. On repeating the above simulation with these block sizes, the fairness index we obtain is a high 0.98. These results indicate that the proposed FEC framework is not very sensitive to the nature of TCP traffic and performs equally well with combination of short-/long-lived TCP flows.

B. Performance of the fairness heuristic at different loads

The objective of this subsection is to provide some insights for when the proposed FEC fairness heuristic performs best. In Fig. 11, the Y axis on the left (note the log-scale) shows the average edge-to-edge packet loss (of 1-, 2- and 3-hops) as a function of the average offered link load for two scenarios, namely, without FEC and when we use the fairness heuristic. The Y axis on the right shows the overhead (in terms of load) that FEC introduces for a given average offered load. The non-FEC loss rates are obtained from Eq. (15), where the value of $\mathbb{L}_c$ for a given load is derived from Eq. (4). The loss rates with FEC are as a result of the heuristic.

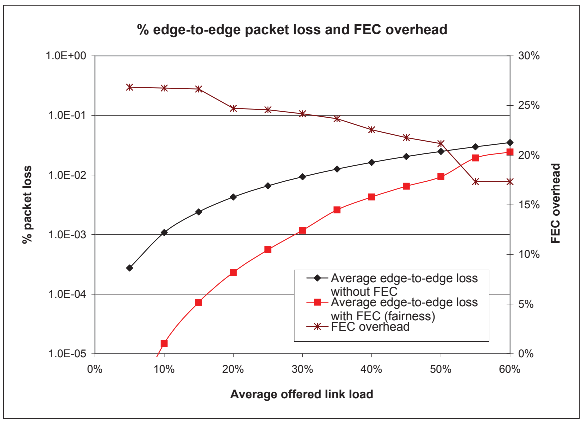

Fig. 11. % edge-to-edge packet loss and FEC overhead for different average offered link loads obtained from analysis

There are essentially two main points that we want to make regarding the figure. First, if the ISP wishes to operate the network such that the edge-to-edge loss is below a certain acceptable threshold (say 10^{-3}), then we note that without FEC, the load can be pushed to at most 10%. On the other hand, employing the FEC heuristic allows the network to be run at 30% load, thus contributing significantly to the ISPs revenue stream. Second, operating the network at high load restrains FEC because there is not much room to introduce additional redundancy (as the total load cannot exceed the available capacity). Thus the benefit offered by the heuristic seems to diminish at higher loads ($> 60\%$). If the network is very lightly loaded ($< 10\%$), then loss rates are significantly low to begin with that there really is no incentive to use FEC. Since ISPs typically operate their networks at 20-30% load [16], the use of FEC in a future bufferless core network seems valuable.

These numbers must only be viewed in light of the proposed heuristic (that does not model TCP feedback and uses static block-sizes) since we could design more efficient TCP aware adaptive FEC schemes that adjust the block-sizes dynamically based on the loss rate that an edge-to-edge flow observes in its path. In general, FEC seems beneficial only under certain load regimes and not across the entire spectrum.

VI. Conclusions and Future Work

As buffering of packets in the optical domain is a complex and expensive operation and we are soon approaching the limitations of electronics, in this paper, we addressed the feasibility of enabling a wide-area bufferless (near-zero buffer) core optical network and made three new contributions. First, we proposed a novel edge-to-edge based packet-level FEC architecture as a means of battling high core losses. Second, we considered a realistic core network (NSFNet), developed an optimisation framework, and proposed a simple heuristic to improve fairness between single- and multi-hop flows. Third, we studied the performance of FEC considering realistic mixes of short- and long-lived TCP flows, and broadly identified the load regimes under which FEC can be beneficial.

This paper is a first step towards understanding the potential of FEC in envisioning a future bufferless core network, and there is much further work that needs to be undertaken. Our analysis assumes that the offered load is known afore-hand; one could deduce this load by modeling the feedback nature of TCP to adjust load based on edge-to-edge loss in the network. Our work in this paper has considered static block-sizes; one could easily extend the FEC scheme to incorporate dynamic adaptation of the FEC strength (block-size) based on on-the-fly measurement of actual losses in the network. This would require explicit network-wide signalling between edge routers so that they can make an informed decision to adjust their FEC strength. Our heuristic for choosing block-size made its decision based solely on flow hop-length, one can design more sophisticated schemes that take additional parameters into account when optimising the FEC strength.

References

[1] Cisco white paper, *Approaching the zettabyte era*, `http://www.cisco.com/en/US/solutions/collateral/ns341/ns525/ns537/ns705/ns827/white_paper_c11-481374.pdf`, Jun 2008.

[2] G. Appenzeller, I. Keslassy and N. McKeown, "Sizing router buffers," *Proc. ACM SIGCOMM*, USA, Aug-Sep 2004.

[3] J. Chabarek et al., "Power awareness in network design and routing," *Proc. IEEE INFOCOM*, USA, Apr 2008.

[4] I. Keslassy, S. Chuang, K. Yu, D. Miller, M. Horowitz, O. Solgaard, and N. McKeown, "Scaling internet routers using optics," *Proc. ACM SIGCOMM*, Germany, Aug 2003.

[5] P. Bernasconi et al., "Architecture of an integrated router interconnected spectrally (IRIS)," *Proc. IEEE High Performance Switching and Routing*, Poland, Jun 2006.

[6] D. J. Blumenthal, "Optical labeled packet switching and the LASOR project," *Proc. 17th Annual Meeting of the IEEE Lasers and Electro-Optics Society (LEOS)*, Nov 2004.

[7] E. F. Burmeister et al., "SOA gate array recirculating buffer for optical packet switching," *Proc. IEEE/OSA OFC*, USA, Feb 2008.

[8] J. D. LeGrange et al., "Demonstration of an integrated buffer for an all-optical packet router," *Proc. IEEE/OSA OFC*, USA, Mar 2009.

[9] J. D. LeGrange et al., "Demonstration of an integrated buffer for an all-optical packet router," *IEEE Photonic Technology Letters*, vol. 21, no. 2, pp. 781, Jun 2009.

[10] A. Vishwanath, V. Sivaraman and M. Thottan, "Perspectives on router buffer sizing: Recent results and open problems," *ACM SIGCOMM CCR Editorial Zone* vol. 39, no. 2, pp. 34-39, Apr 2009.

[11] E. W. M. Wong et al., "Towards a bufferless optical internet," *IEEE/OSA Journal of Lightwave Technology*, 2009 (To appear).

[12] V. Sivaraman et al., "Packet pacing in short buffer optical packet switched networks," *Proc. IEEE INFOCOM*, Spain, Apr 2006.

[13] J. Naor, A. Rosen, and G. Scalosub, "Online time-constrained scheduling in linear networks," *Proc. IEEE INFOCOM*, USA, Mar 2005.

[14] H. Jiang and C. Dovrolis, "Why is the Internet traffic bursty in short time scales?", *Proc. ACM SIGMETRICS*, Canada, Jun 2005.

[15] R. S. Prasad, C. Dovrolis and M. Thottan, "Router buffer sizing revisited: The role of the output/input capacity ratio," *Proc. ACM CoNEXT*, USA, Dec 2007.

[16] A. Odlyzko, "Data networks are mostly empty and for good reason," *IT Pro*, vol. 1, no. 2, pp. 67-69, Mar/Apr 1999.

[17] N. Dukkipati and N. McKeown, "Why flow-completion time is the right metric for congestion control", *ACM SIGCOMM CCR*, vol. 36, no. 1, pp. 59-62, 2006.

[18] S. K. Korotky, "Network global expectation model: A statistical formalism for quickly quantifying network needs and costs", *IEEE/OSA Journal of Lightwave Technology*, vol. 22, no. 3, pp. 703-722, 2004.

[19] M. Mathis, J. Semke, J. Madhavi, and T. Ott, "The macroscopic behavior of the TCP congestion avoidance algorithm," *ACM SIGCOMM CCR*, vol. 27, no. 3, pp. 67-82, 1997.

[20] C. Villamizar and C. Song, "High performance TCP in ANSNet," *ACM SIGCOMM CCR*, vol. 24, no. 5, pp. 45-60, 1994.

[21] R. Jain, D. Chiu and W. Hawe, "A quantitative measure of fairness and discrimination for resource allocation in shared computer system," *DEC technical report* TR-301, 1984.

[22] B. Schroeder, A. Wierman, and M. Harchol-Balter, "Closed versus open: A cautionary tale," *Proc. USENIX NSDI*, USA, May. 2006.