Contention Reduction in Core Optical Packet Switches Through Electronic Traffic Smoothing and Scheduling at the Network Edge

Zheng Lu, and David K. Hunter, *Senior Member, IEEE*

***Abstract*—A contention-aware packet-scheduling scheme for slotted optical packet switching (OPS) networks is proposed, which employs edge-traffic shaping to reduce contention, coupled with a modified type of renegotiated service incorporating rate prediction. Queuing and scheduling of traffic is implemented electronically within the edge nodes, shaping user traffic into streams, which have a fixed bit rate only for a short period, which is renegotiated at regular intervals in response to user requirements and network conditions. Via an appropriate protocol, edge nodes gain knowledge of relevant network scheduling and topology information. This is used to schedule user-data packets appropriately, in order to reduce contention. Simulation and analytical results demonstrate that in the core, under typical conditions, packet loss below 10^{-8} may be obtained, with a load of 0.8 and with core optical-packet switch buffers having only 20-slot capacity. The tradeoffs between parameters affecting such results are investigated, demonstrating clearly that much more modest optical core buffers than previously thought necessary can provide acceptable performance. The performance and scalability of these proposals are investigated and discussed, demonstrating their feasibility.**

***Index Terms*—Delay-line buffering, edge smoothing, optical packet switching, rate prediction, renegotiated service, slotted packets.**

I. Introduction

CONTENTION resolution is a fundamental problem in optical-packet switched networks [1], [2], since there is no optical RAM analogous to that used in the electronic domain. Three "dimensions" of contention resolution are commonly proposed.

1) Fiber delay-line (FDL) buffering in the time domain [3]–[5] is often proposed for optical buffering; however, it is limited because such a delay line can only offer a fixed-time delay. Therefore, designing buffered optical-packet switches employing this technique is not straightforward. Many proposals exist in which FDLs are used to resolve contention [3], [4], and attempts have even been made to design routers with very large optical buffers [5], although these often require a considerable amount of bulky fiber.
2) Deflection routing in the space domain [6]–[8] is topology dependent and may cause packet misordering upon arrival at the destination.
3) Wavelength conversion in the wavelength domain [9] is expensive to implement, and tunable-wavelength converters (TWCs) are at an early stage of development.

Combining several of the above techniques to resolve contention has also been studied [10]–[12]. The need for them can be drastically reduced through traffic shaping and smoothing at the network edge, which regulates traffic as it flows into the network. Optical-packet transmission at the edge is usually triggered either when an optical container is filled with sufficient data or (to prevent long delays) by a timeout.

Other work on slotted OPS networks ensures fairness through a capacity-allocation algorithm and addresses contention problems by means of both core-switch buffering and deflection routing [13]. Elsewhere, optical timeslot interchangers have been proposed to switch in the time domain, achieving high-statistical multiplexing gain while eliminating any requirement for wavelength converters [14].

This paper reports on the use of electronics to queue and schedule traffic within the edge nodes, shaping user traffic into streams that have a fixed bit rate only for a short period, which is renegotiated at regular intervals in response to user requirements and network conditions. Fixed-length packets are preallocated onto slots, which are carried on transmission links. Edge nodes acquire knowledge of relevant network-scheduling information and use this to schedule user-data packets appropriately, in order to reduce core contention. Optical nodes are electronically controlled, and all these features combine to realize a flexible and effective architecture. Hence, this research focuses on

1) shifting the computational burden from the network's core to its edge, where powerful electronic processing is available;
2) reducing the need for buffering in the network core;
3) reducing the size and complexity of the core-switch structure.

All the slots between a particular source-destination pair are treated as a byte stream [15], much like the payload of SDH/SONET Virtual Containers. Hence, one slot may contain several IP datagrams or several slots may contain one IP datagram, depending on the lengths of the datagrams relative to the slots.

It is assumed throughout this paper that packets entering each node are synchronized to facilitate correct switch

Manuscript received June 14, 2006; revised August 22, 2006. Z. Lu's Ph.D. studentship was supported by Intel Labs, Cambridge, U.K., under the DTI Smoothed Optical ATD Packet Switching (SOAPS) project.

The authors are with the Department of Electronic Systems Engineering, University of Essex, CO4 3SQ Colchester, U.K. (e-mail: zlu@essex.ac.uk; dkhunter@essex.ac.uk).

Digital Object Identifier 10.1109/JLT.2006.884543

operation. While contention resolution, the subject of this paper, is an important aspect of optical-packet switching, so is packet synchronization, which has also been extensively researched. Synchronization is generally achieved by implementing a variable optical delay line on each node input.

In this paper, a contention-aware packet-scheduling and traffic-shaping scheme is proposed, which reduces the need for core-contention resolution. Section II introduces rate-prediction-based edge-traffic shaping and describes a contention-aware packet-scheduling scheme. Section III evaluates this scheme through analysis and simulation. Section IV concludes this paper.

II. Network Architecture

A. Edge-Traffic Smoothing Based on Rate Prediction

In this paper, traffic shaping smoothes traffic at the edge of the network by means of rate prediction and a modified version of renegotiated service, where each stream's bandwidth is updated at regular intervals (which are not synchronized between different streams) to take account of both user requirements and network conditions. The concept of a "stream" implies circuit switching, and in a real implementation, a stream would be set up in either direction between each pair of edge nodes that needed to communicate. Hence, one stream may well correspond to many simultaneous transmission control protocol (TCP) or real-time transport protocol (RTP) sessions between users. Renegotiated service was originally proposed for transmission of online video [16] with an autoregressive model to predict future bandwidth. Here, this linear predictor is retained, but modifications are made to address the renegotiation failure and the traffic problems specific to wide area networks.

A heuristic algorithm for modified renegotiated service is described in this section in order to accommodate traffic other than video. Unlike the original renegotiated service [16], [17], the modified scheme considers network conditions, as well as user requirements, and is not traffic specific, hence, providing a generic wide-area network solution.

In previous renegotiated-service schemes that only considered user requirements, if a stream requires more bandwidth, which is not available, it suffers a temporary service disruption due to renegotiation failure. Strategies for dealing with this have been proposed [16], such as, even if renegotiation fails, allowing the source to keep whatever bandwidth it already has. Alternatively, the switch controller can reject an incoming request during admission control, even if there is currently a bandwidth available, if future renegotiation failure seems likely. It has also been suggested that the user–network interface could instruct the user or application to reduce its rate.

When applying existing renegotiated-service schemes to TCP traffic, only user requirements are considered, and the network becomes progressively more congested, as attempts are made to satisfy source-bandwidth requests triggered by TCP's windowing mechanism. The algorithm proposed here reduces the frequency of negotiation failure by also considering network conditions and quenching sources at the edge when less bandwidth is available, preventing too much traffic from entering the network under critical network conditions. Renegotiated service is connection based and, here, the scheduling and topology information it requires is distributed throughout the network. Protocols that distribute such information are well known, for example Open Shortest Path First (OSPF), although here, it is only necessary for nodes to be informed of transmission-rate changes in other nodes on a "need-to-know" basis. Based upon this information and upon the bandwidth requested by the user application, each source periodically adjusts its sending rate, on a per-connection basis. Although TCP traffic is the motivation behind this work, its effectiveness is demonstrated here via the use of traffic generators.

To facilitate estimation of the service rate to be requested for the next negotiation interval, time is split up into small "units." In the simulations reported later, the size of a unit lies between 5 and 100 ms, depending on traffic type. T is the interval between renegotiation intervals—the renegotiation process reflects longer term change in the network. S_0 is the current service rate or the rate that traffic leaves the buffer. U_i is the measured arrival rate for traffic entering the buffer during unit i, while R_i is the smoothed arrival rate, defined via (1) below. S_f is the service rate to be requested at the next negotiation, obtained by taking the mean of R_i over all units in the last negotiation interval. ν is a factor which decides to what extent previous experience and to what extent the current measured rate should influence the new service rate. This heuristic algorithm is described as follows:

$$R_{i+1} = \nu R_i + (1-\nu)(U_i + B/\tau) \tag{1}$$

with

$$\nu = \begin{cases} \min\left\{\frac{R_i - S_0}{S_{\text{avail}}}, 1\right\}, & \text{if } S_{\text{avail}} < S_{\text{thresh}} \text{ and } R_i > S_0 \\ & \text{critical state} \\ \alpha, & \text{otherwise.} \end{cases}$$

If $|S_f - S_0| > \Delta S$, the algorithm initiates renegotiation by requesting service rate S_f, where ΔS is the permitted variation, within which renegotiation is not necessary. If the available bandwidth S_{avail} is less than this, the request is rejected and the service rate becomes S_{avail}. Otherwise, the service rate remains at S_0. In (1), the first part depends on network conditions, while the second part (and indeed the value of ν itself) depends on user requirements. B/τ represents the extra capacity required to empty the buffer in the next unit, where B is the buffer occupancy, and τ is the duration of a unit. S_{avail} is the available network bandwidth, and S_{thresh} is the threshold rate below which "critical state" may occur. Here, critical state denotes scarce network resources, where high-rate requests frequently exhaust the network prior to it becoming actually congested. S_{avail} may, hence, be regarded as an approximate and simplified indication of available resources.

1) When $S_{\text{avail}} > S_{\text{thresh}}$, before critical state is reached, $\nu = \alpha$, and (1) can be rewritten as $R_{i+1} = \alpha R_i + (1-\alpha)(U_i + B/\tau)$, where the factor α determines to what extent the algorithm responds to user requirements. When α is set to 0.5, it denotes no bias to either previous experience or to current measured rate.

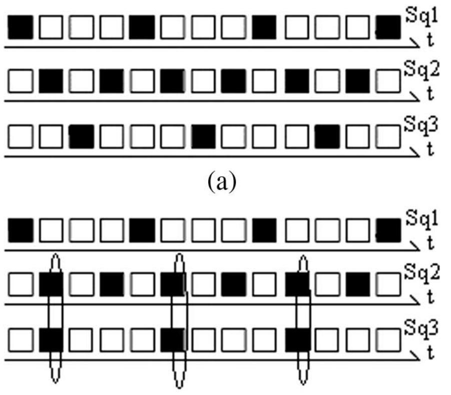


Fig. 1. (a) Packet sequences without contention. (b) Packet sequences with contention between Sq2 and Sq3.

2) When $S_{\text{avail}} < S_{\text{thresh}}$ and $R_i > S_0$, critical state has been reached, so (1) now considers network conditions. In the expression $(R_i - S_0)/S_{\text{avail}}$, the numerator reflects user requirements and the denominator reflects network conditions. As the available bandwidth in the network decreases, the new renegotiated rate is less influenced by user requirements.

B. Contention-Aware Packet Scheduling

Renegotiated service produces, within each renegotiation interval of duration T, a stream of evenly spaced packets. Contention-aware packet scheduling ensures that each packet is transmitted over the link in question on the first free timeslot, based upon knowledge of relevant network-scheduling information from elsewhere in the network, in order to reduce core-packet contention. If there were no contention, all streams, having been shaped at the edge, would have regular packet-sequence patterns consistent with their reservation bandwidth [Fig. 1(a)].

In Fig. 1, Sq1, Sq2, and Sq3 refer to slot sequences with rates R_1, R_2, and R_3, each representing a stream. They are multiplexed onto a link within the core of capacity R_L, where $R_L \geq R_1 + R_2 + R_3$. If there is no contention, the packets in each sequence occupy different slots from those in any other sequence [Fig. 1(a)]. However, if contention occurs [Fig. 1(b)], individual packets must be scheduled to avoid contention on this particular link in the core, where scheduling could take place at either edge or core switches. At the edge, electronic buffers schedule packets, however, in core switches, FDL buffering would be necessary. If contention is predicted, individual packets should be rescheduled only as necessary, but at the edge before contention occurs, in order to reduce core FDL-buffering requirements. Therefore, it is necessary to predict when contention will occur in the core.

The scheduling of packets for each stream over the set of links it traverses only depends on packet positions of higher rate streams. Thus, higher rate streams are granted higher priority by this scheduling algorithm, since they have the smallest range of slot positions to which a packet can be scheduled. In the absence of other streams, each stream i consists of packets at

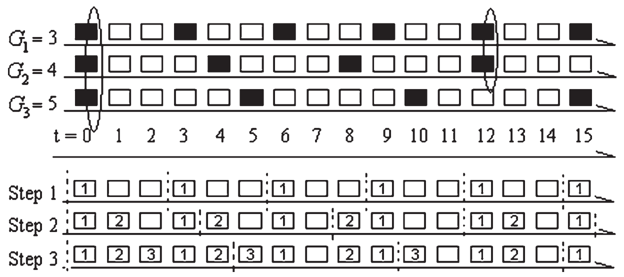


Fig. 2. Illustration of contention reduction, showing packet scheduling.

regular intervals of G_i slots. When each packet is generated by traffic smoothing as described in the previous section, it is scheduled to be transmitted over the link on either the present slot or one of the following $G_i - 1$ slots. The first of these slots is chosen, which is not occupied on any link along its path, by a packet from a higher rate stream. If all these slots are already allocated to higher rate streams, the packet is transmitted immediately. The highest rate stream always transmits packets immediately when they are generated, since no other streams have priority for packet scheduling.

On each link that a stream traverses, its source edge node determines how each higher rate stream is scheduled, starting from the highest rate stream and working in decreasing order of stream rate. It can hence deduce when to transmit by finding a free slot on all links it traverses. For example, consider a single link with three streams 1, 2, and 3 with intervals between packets of $G_1 = 3$, $G_2 = 4$, and $G_3 = 5$, respectively, proportional to the reciprocal of the service rate S discussed previously, hence, $G_i \ \alpha \ 1/S$. Fig. 2 shows the three streams, as well as the steps in scheduling stream 3's packets at their source edge node, say, node X. In Step 1, node X works out the positions of packets from stream 1. Time is divided up into frames of $G_1 = 3$ timeslots; each frame containing the potential choice of slots for each newly generated packet from stream 1. Since no other streams have allocated slots, a packet from stream 1 is scheduled in the first slot of each frame. In Step 2, node X now considers the effect of stream 2. At $t = 0$ and $t = 12$, the first timeslot in each frame of length four is already occupied by a packet from stream 1; hence, the packet from stream 2 must be placed in the following slot in each case. Finally, now that node X has worked out how streams 1 and 2 are scheduled, it schedules its own packets from stream 3. In particular, slots at $t = 0$ and $t = 1$ are already occupied by packets from streams 1 and 2, respectively; hence, the packet from stream 3 must be transmitted over the link at $t = 2$. Further streams may exist with lower rates than stream 3; however, they do not influence stream 3's packet scheduling.

Scheduling of the packets in a stream may be influenced not only by higher rate streams traversing the same links, but also by higher rate streams elsewhere. These may share links with other streams, which in turn share the same link as the original stream, changing its scheduling. For example, in Fig. 2, stream 2 may have its packets rescheduled on some other link that does not carry stream 3, ultimately yielding packet loss due to incorrect rescheduling with the algorithm presented above. In general, a stream may influence the scheduling of another through one or more intermediate streams, exhibiting a "higher order dependence." The above example only considers a "first-order dependence," where there are no such intermediate streams. To reschedule each packet correctly would require each edge node to have knowledge of practically all streams in many cases and would require complex and time-consuming calculations.

Here, scheduling is only based on first-order dependencies and higher order dependencies are ignored in the interests of simplicity. Hence, it is possible that a better packet-scheduling algorithm could be implemented, although simulations demonstrate that the simple scheme works well. Also, only permitting transmission to take place over a restricted set of rates, such as those related by powers of two, is a topic for further study, which may improve performance and simplify computation. In the present scheme, assume that a stream is to be shifted to comply with N higher rate streams. In the worst case, the slots occupied by them are staggered over successive timeslots, so that the number of steps required to determine the correct shifting is $1 + 2 + \ldots + N \approx O(N^2)$.

Edge nodes are informed, as quickly as possible, of rate changes elsewhere subject to the appropriate propagation delay, and in this event, recomputation of packet schedules takes place immediately.

Large networks may be partitioned into areas to make packet-scheduling scale, with each area implementing packet scheduling independently. Hence, packets crossing more than one area may well be scheduled independently in each. Therefore, core nodes at the border between areas must carry out packet scheduling and will, hence, require more buffering. The other core nodes have small buffers due to the use of packet scheduling, as described above. Splitting a network into areas is a common way of ensuring scalability, with the Internet routing protocol OSPF being a well-known example.

C. Example

In order to illustrate the principles described above and show how packet scheduling is calculated when a new stream is set up or when another stream changes its rate, the network of Fig. 3 is used. Here, five unidirectional streams already exist in the network and originate from source nodes B, D, and E, and a new stream (stream 6) is currently being established from nodes A to G. This information is contained in a database in each node, which is built up via a suitable signaling protocol that also distributes topology information. This section describes how the packet-scheduling algorithm examines the path from A to G in this database representation and computes the correct slot for each packet in stream 6. There is no potential contention with any other streams before node B, but on link $C \rightarrow F$ for example, streams 1 and 2 may contend for the same outlet slots with stream 6. Notation is introduced below to denote the relevant information held in each edge node's database. Stream-rate changes are reported as quickly as possible to each source, which is affected and is placed in its database for use by the packet-scheduling algorithm.

Let X and Y be nodes, and let Z be a traffic stream. $I_{X \rightarrow Y}$ relates to streams leaving node X and passing over link $X \rightarrow Y$,

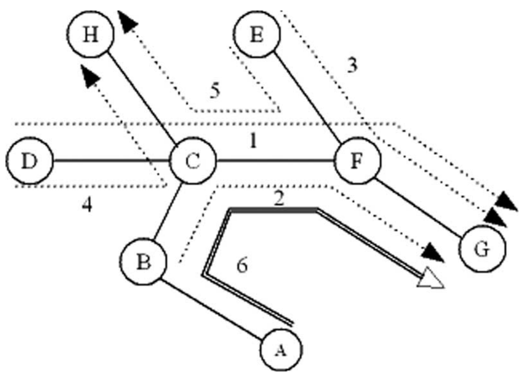


Fig. 3. Scheduling packets in stream 6.

which is used to calculate packet scheduling throughout the network. It consists of the set of flow bandwidths B_Z contending for the same outlet link and the offset O_Z between the current slot position and next estimated arriving slot from other streams. Therefore, at node B, $I_{B \rightarrow C} = \{(B_2, B_6), (O_2, O_6)\}$, and at node C, $I_{C \rightarrow F} = \{(B_1, B_2, B_6), (O_1, O_2, O_6)\}$. When stream 6 arrives at node F, it potentially contends with streams 1, 2, and 3 for link $F \rightarrow G$. Then, at node F, $I_{F \rightarrow G} = \{(B_1, B_2, B_3, B_6), (O_1, O_2, O_3, O_6)\}$.

Source-node A requires $I_{B \rightarrow C}$, $I_{C \rightarrow F}$, and $I_{F \rightarrow G}$ from three intermediate nodes to determine where contention may occur with stream 6. This information is signaled back to source-node A by nodes B, C, and F. Other source-nodes B, D, and E acquire corresponding information. The number of slots between consecutive packet arrivals, denoted by G_Z, where $G_Z \,\alpha\, 1/B_Z$, is written as G_1 through G_6, in this example. This is related to the stream rate.

The packet-scheduling algorithm only takes place at source nodes. First, consider the potential contention at links $C \rightarrow F$, used by streams 1, 2, and 6 with the initiating source-nodes D, B, and A, respectively. If there is contention, not all three streams need reschedule their packets, only those nodes initiating contending streams and not having the highest stream rate must reschedule their stream's packets to avoid potential contention. Assume $B_1 > B_2 > B_6$. If all three streams are assigned the same slot on links $C \rightarrow F$, then nodes B and A (initiating streams 2 and 6, respectively) must reschedule packets. Otherwise, if stream 2 was absent, only node A would have to reschedule packets in its stream. Via signaling, each source node knows the rate of every stream on every intermediate node's outlet link and, hence, can decide whether packets that it originates must be rescheduled.

Assume that propagation delay (in units of slots) is measured as signaling takes place. The node-to-node delay over a route is denoted as D_{route}, e.g., the delay from nodes A to F is denoted by D_{ABCF}. The information required to schedule packets for each stream is communicated to each node, which is listed below for each source node.

1) Source-node A: Stream 6: $I_{B \rightarrow C} \rightarrow I_{C \rightarrow F} \rightarrow I_{F \rightarrow G}$.
2) Source-node B: Stream 2: $I_{C \rightarrow F} \rightarrow I_{F \rightarrow G}$.
3) Source-node D: Stream 1: $I_{C \rightarrow F} \rightarrow I_{F \rightarrow G}$, and Stream 4: $I_{C \rightarrow H}$.
4) Source-node E: Stream 3: $I_{F \rightarrow G}$, and Stream 5: $I_{F \rightarrow C} \rightarrow I_{C \rightarrow H}$.

In the above list, nodes requiring the same information $I_{X \to Y}$ share the same link. The links that may experience contention are $C \to H$, $C \to F$, and $F \to G$. For $C \to H$, streams 4 and 5 influence packet scheduling in source-nodes D and E, and if stream 4 has a higher rate than stream 5, only node E need reschedule packets. A packet leaving source-node E for node H arrives at C after time D_{EFC}, and this, together with D_{DC}, is involved when node E computes the relative positions of the two streams in link $C \to H$. If $G_4 = 3$ and $G_5 = 6$, with $O_4 = O_5 = 0$, then packets in stream 5 may be delayed by one or two slots to avoid contention with stream 4. For $C \to F$ and $F \to G$, the process is similar. Only nodes A, B, D, and E are responsible for packet scheduling, since it occurs at the source of a stream.

III. Analysis and Numerical Results

A. Analytical Model

The worst-case assumption is made that each source changes its rate on every negotiation period. Packet loss due to contention is studied in two different situations.

1) Unstable periods when the negotiated rate has changed, and stream scheduling information is being exchanged between nodes. This period (known as the negotiation delay D), is of the order of the round-trip time between the ingress and egress-edge switches in contrast to the negotiation interval T, which is the time between successive opportunities to change the stream rate.
2) Stable periods during the remainder of each negotiation interval of duration $T - D$.

Unstable periods are denoted by the event U and stable periods by the event S. Contention, denoted by event L, may arise during both stable and unstable periods, so the probability of contention occurring on some slot taken at random, taking both periods into account, is

$$\begin{aligned}\Pr[L] &= \Pr[U]\Pr[L|U] + \Pr[S]\Pr[L|S] \\ &= \frac{D}{T}\Pr[L|U] + \frac{T-D}{T}\Pr[L|S].\end{aligned}$$

Since the aggregated process within the core contains many flows, Bernoulli traffic may reasonably be assumed as an approximation. Following Hluchyj and Karol [18], let Q_m denote the number of packets in an output buffer of a core optical-packet switch at the end of an mth time slot. Assuming an output buffered-node architecture, the buffer occupancy may be modeled by a time-discrete Markov chain with state-transition probabilities P_{ij}

$$\begin{aligned}P_{ij} &= \Pr[Q_m = j | Q_{m-1} = i] \\ &= \begin{cases} a_0 + a_1, & i = 0, j = 0 \\ a_0, & 1 \le i \le b, j = i-1 \\ a_{j-i+1}, & 1 \le j \le b-1, 0 \le i \le j \\ \sum\limits_{m=j-i+1}^{N} a_m, & j = b, 0 \le i \le j \\ 0, & \text{otherwise} \end{cases}\end{aligned}$$

where b is the buffer depth, and N is the number of ports on each node. With finite N, the number of packets arriving at a particular output buffer within a core switch is described by a binomially distributed random-variable A

$$a_k = \Pr[A = k] = \binom{N}{k}\left(\frac{p}{N}\right)^k\left(1 - \frac{p}{N}\right)^{N-k} \quad \text{where } k = 0, 1, \ldots, N. \tag{2}$$

The balance equations of the Markov chain generate the following recursive equations:

$$\begin{aligned} q_0 &= 1 \text{ or some other arbitrary value} \\ q_1 &= \Pr[Q = 1] = \frac{q_0(1 - a_0 - a_1)}{a_0} \\ q_n &= \Pr[Q = n] = \frac{1 - a_1}{a_0} q_{n-1} - \sum_{k=2}^{n} \frac{a_k}{a_0} q_{n-k} \\ &\quad \text{where } 2 \le n \le b. \end{aligned} \tag{3}$$

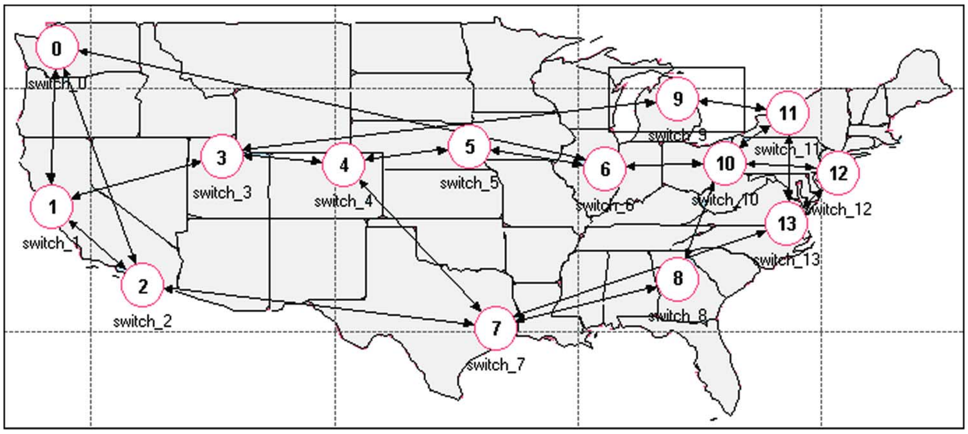
The quantities q_i are then normalized by multiplying them each by the same constant to make them sum to unity, yielding the probabilities that i packets are in an output buffer. The utilization of the output link is divided by the arrival rate p to yield the probability of successful packet delivery, which is $(1 - q_0 a_0)/p$, so the packet-loss probability due to contention is

$$1 - \frac{1 - q_0 a_0}{p}. \tag{4}$$

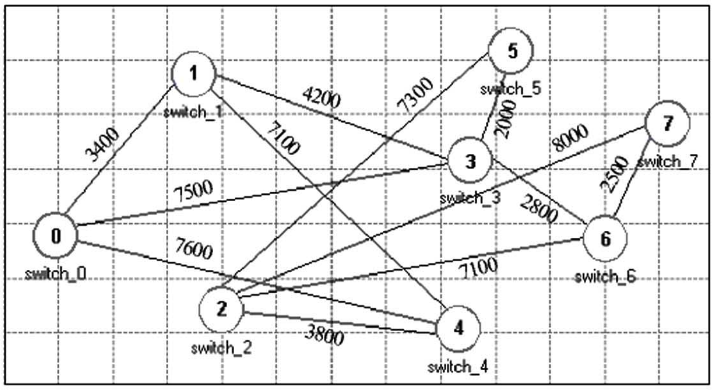
Equation (4) is evaluated by using (2) and (3).

First, consider the unstable period where, in the extreme case, packet scheduling is not effective and the whole core network is approximated through Bernoulli arrivals. Therefore, $\Pr[L|U]$ may be obtained directly from (4), with p replaced by r, the sum of rates for streams contending for the same output link.

Next, consider the stable period. Even if a stream's packet positions exactly follow the pattern assigned to them at the edge, contention may still occur in the core since each edge node usually has incomplete information about scheduling of other streams. Furthermore, a packet may have been rescheduled to a different slot due to delay-line buffering in a previous node. In these cases, contended packets are buffered optically in the core. The contention probability of a given stream may be estimated, defined as the probability that a packet in the stream has been shifted to another slot by FDL buffering, due to contention at the current node or an upstream node. It is denoted by s'_{jk}, where k is the stream entering node j and s_{jk}, which is the contention probability of stream k after leaving node j. The initial contention probability of a stream at an edge node is the probability that, because packet scheduling does not always resolve contention, a packet contends for the same output of the edge node in question with those originating from other edge nodes. If V is the set of all streams going to the same output as stream k on a switch j, the contention probability for that stream is approximately $s_{jk} = \sum_{i \in V} s'_{ji}$. The edge nodes are not aware of this unexpected core contention when carrying out



(a)



(b)

Fig. 4. Network topologies for simulation and analysis. (a) Net1 topology. (b) Net2 topology. (Color version available online at http://ieeexplore.ieee.org.)

packet scheduling, which causes packet loss during the stable period.

In the results reported below, the worst case is assumed where the initial contention probability for a stream leaving an edge node and entering the core is approximated as the sum of the intensity of all streams entering the edge node from elsewhere via the core and also entering that link. The summation introduced above $(s_{jk} = \sum_{i\in V} s'_{ji})$ is then applied repeatedly to calculate the contention probability for each link in the core. r_c is then calculated as the mean of these values. Therefore, $\Pr[L|S]$ may now be obtained through (4) by replacing p with r_c. By combining both components, the overall packet-loss probability is approximated as

$$\Pr[L] = \frac{D}{T}\left[1 - \frac{1 - q_0\left(1 - \frac{r}{N}\right)^N}{r}\right] + \frac{T-D}{T}\left[1 - \frac{1 - q_0\left(1 - \frac{r_c}{N}\right)^N}{r_c}\right].$$

q_0 is obtained via computation of (3).

B. Numerical Results

Two different topologies were simulated using OPNET [19]—the 14-node NSFNET backbone and an 8-node random-network topology (Fig. 4). Each node acts as an edge switch which maps external (edge) traffic into core optical traffic and vice versa, while also acting as a core switch performing switching with reduced buffering. Traffic entering each edge switch is generated by either a Poisson process or a Pareto ON–OFF fractal-point process with a Hurst parameter of 0.8. The latter tests these protocols under bursty conditions, imitating the long-range dependence of Internet traffic [20]–[23]. Besides approximating traffic found in real data networks, it tests the packet-scheduling scheme effectively because it implies frequent renegotiation.

Each slot is 2000 bytes long, almost four times longer than an average IP datagram. Each core switch employs no special mechanisms such as wavelength conversion or deflection routing and is assumed to have a simple output-buffered architecture. The following simulation settings and assumptions are used.

1) The distance between nodes is measured in units of the propagation distance over a slot duration. For NFSNET, this is calculated through the latitude and longitude of nodes in the map of Fig. 4(a), whereas in the random topology, the distance between nodes is represented in Fig. 4(b).
2) Distribution of scheduling information is not modeled explicitly, nor are the protocols to do this described in any detail here. This does not affect the results significantly, since scheduling information is distributed during the renegotiation delay, which forms part of the model.

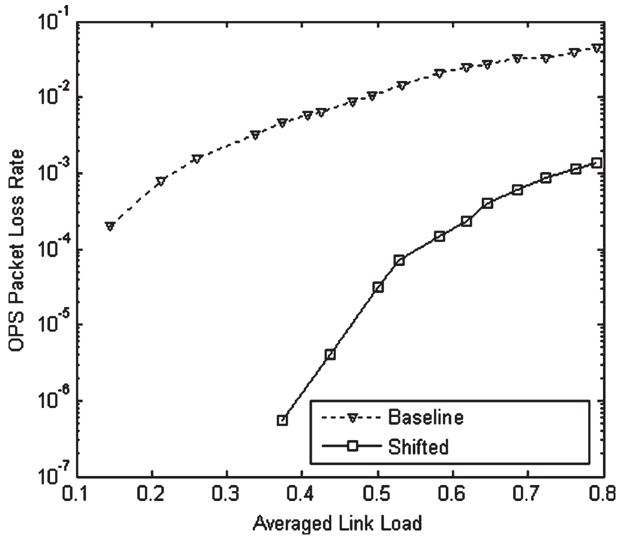


Fig. 5. Loss rate against offered load with five-packet core-buffer depth (simulation).

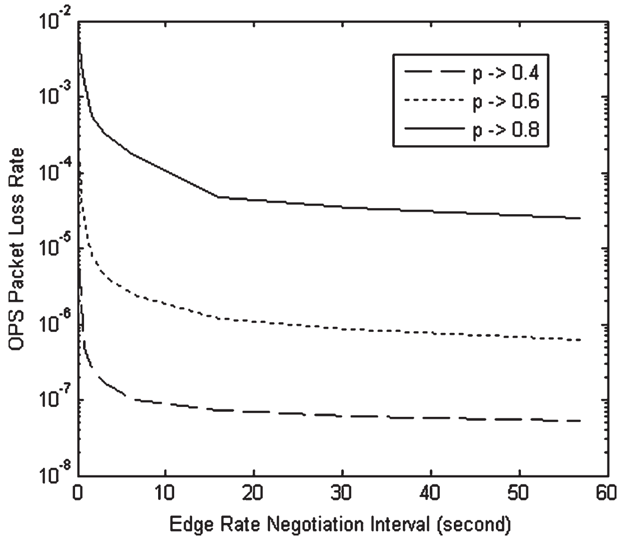


Fig. 6. Loss rate against negotiation interval with five-packet core buffers (simulation), with smoothing and packet scheduling.

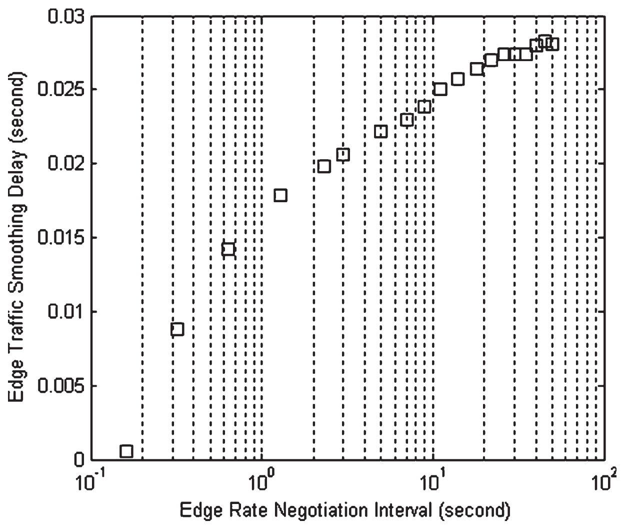


Fig. 7. Edge-smoothing delay against negotiation interval (simulation). Offered load is approximately 0.6.

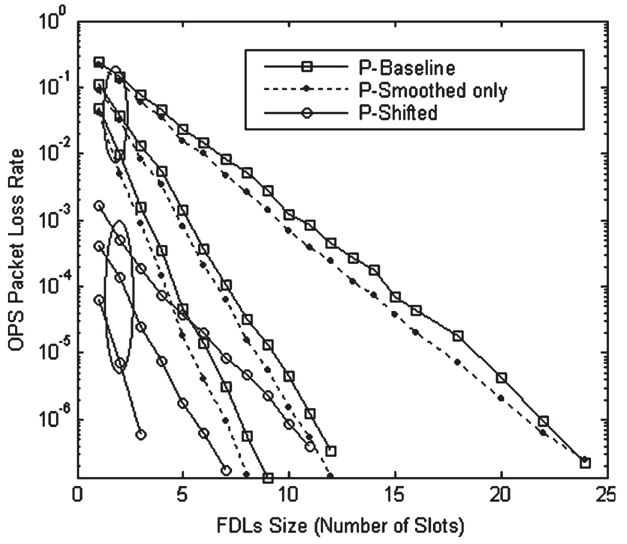


Fig. 8. Simulation of loss rate with Poisson user-traffic (P). Negotiation interval of 20 s. Mean-link loads approximately 0.4, 0.6, and 0.8, from bottom-to-top lines.

3) The offered load is defined as $\sum_{i=1}^{L_N} t_i/CL_N$, where L_N is the number of links in the network, t_i is the traffic on link i, and C is the capacity of a link.
4) Each link carries 10 Gb/s.
5) The slot duration is 1.6 μs.
6) Each core node may be modeled by an output-buffered switch without deflection.
7) Negotiation delay is approximated by the appropriate propagation delay, measured in units of slot duration.

The following notation is used below.

1) Net1 is the network topology of Fig. 4(a).
2) Net2 is the network topology of Fig. 4(b).
3) The "baseline" networking configuration has neither traffic smoothing nor packet scheduling.
4) The "shifted" networking configuration has both edge-traffic smoothing and packet scheduling.
5) The "smoothed-only" networking configuration has traffic smoothing but no packet scheduling. None of these configurations uses wavelength conversion or deflection routing.

In these simulations, routing and capacity allocation of streams were carried out manually so that each link in the core had approximately the same overall load. In a real implementation, a routing algorithm would be employed, this being a field that has been extensively studied for many years.

Figs. 5–7 assume self-similar user traffic, which is smoothed before entering the core. Fig. 5 shows the loss rate for Net1, as mean-link load is varied. Figs. 8 and 9 assume Poisson and self-similar user traffic, respectively, and show packet-loss rates derived via simulation with various combinations of smoothing and packet scheduling. Figs. 10–12 study the effect of negotiation interval on performance and assume Poisson traffic in the core. Packet scheduling improves performance over

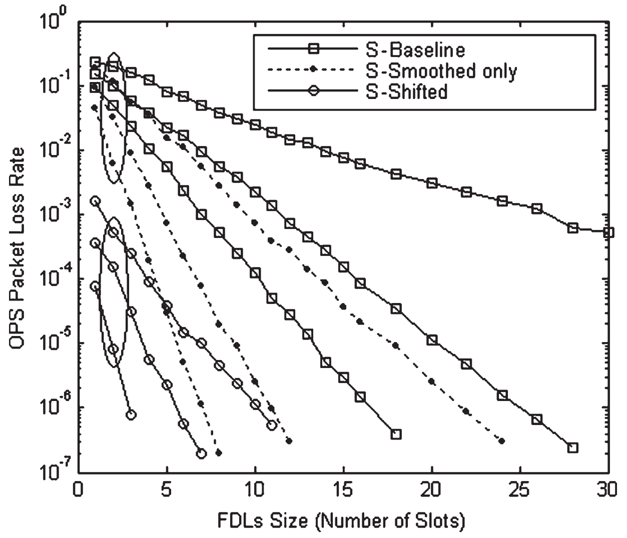


Fig. 9. As for Fig. 8, but with self-similar user-traffic (S).

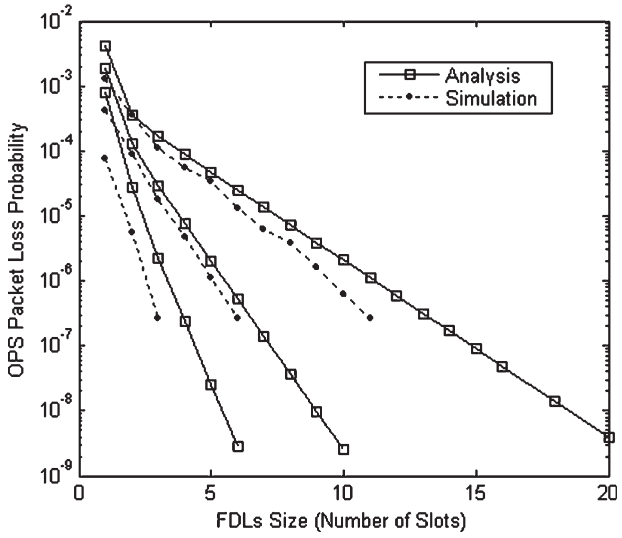


Fig. 11. As for Fig. 10, but with 20-s negotiation intervals.

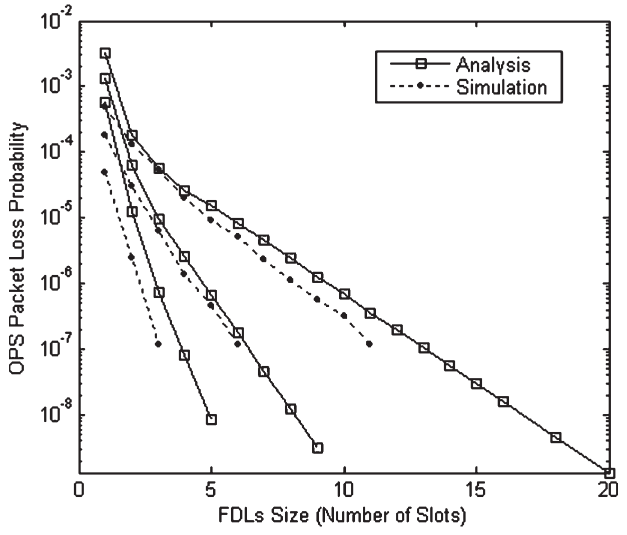


Fig. 10. Loss probability for 60-s negotiation intervals. The mean offered loads are approximately 0.4, 0.6, and 0.8, from bottom-to-top lines.

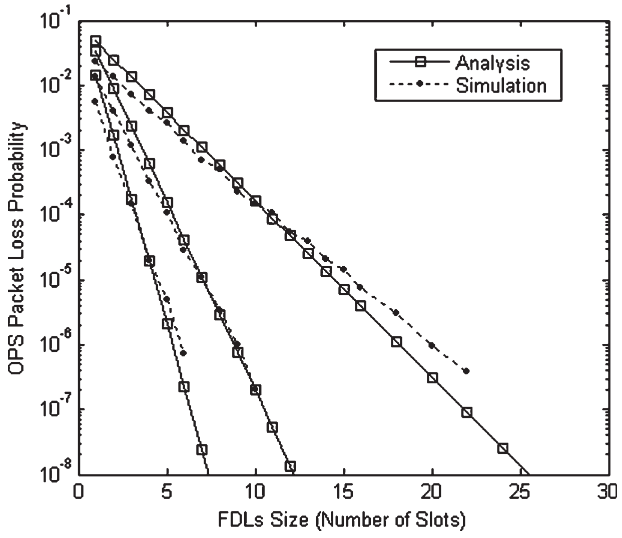


Fig. 12. As for Fig. 10, but with 200-μs negotiation intervals.

the baseline configuration with both topologies. However, in general, the loss may be partly due to congestion, especially as the overall load increases. Therefore, constraint-based routing and effective congestion control are also important influences on packet loss.

Larger networks have greater propagation delay, which may imply some nodes retaining out-of-date scheduling information. With a network of 6000 km diameter, the maximum propagation delay is 30 ms; however, an appropriate negotiation interval lies between 200 ms and several seconds, so there is still sufficient margin to ensure up-to-date records. If necessary, some additional core-switch buffering could be employed to accommodate misscheduled packets arising for this reason.

Distribution of scheduling information always takes place at the beginning of renegotiation intervals and at the startup and closedown of traffic streams. Only streams sharing the same link are affected. Therefore, unlike flooding in OSPF, distribution of scheduling information such as transmission-rate changes due to renegotiation need only take place to edge nodes that are affected. The amount of information required to notify changes in flow rate is small. This consists of the originating node ID, the new renegotiated rate, and the slot offset (specifying the number of empty slots before a packet arrives)—these can be stored in just a few bytes.

Fig. 6 shows that the negotiation interval also affects loss rate. Under fixed load, the loss rate dramatically decreases as the negotiation interval increases, although the curve flattens out for larger negotiation intervals. This is because longer intervals provide more time for slot calculations reducing the frequency of miscalculated slots. However, Fig. 7 shows how smoothing (buffering) delay increases with negotiation interval, since there is more time for the traffic rate to increase during a negotiation interval and, hence, accumulate in the buffer. Therefore, the negotiation interval, which influences the value of B in (1), should be chosen to find an appropriate tradeoff between delay and packet-loss rate.

Figs. 8 and 9 show that with a 20 s negotiation interval, the smoothed scheme has relatively stable performance with either traffic type, and packet scheduling exhibits superior performance. Figs. 10–12 were generated using the Net2 topology of Fig. 4(b). The divergence of analytical results from simulation arises because Bernoulli traffic is assumed within the core in the analytical model in order to make it tractable. Furthermore, the simplifying assumption is made that the loads on all links entering a core node are equal. Nevertheless, the analytical results still highlight the significant components affecting packet loss. Both sets of results yield a pessimistic estimate of packet loss. The loss probabilities with 20 s negotiation intervals are slightly higher than for 60 s; however, the difference is insignificant, because the negotiation delay is much less than the renegotiation interval in both cases. In these circumstances, with packet scheduling functioning efficiently, the system is stable. As the negotiation delay approaches the negotiation interval, instability results and the loss rate increases significantly. This is shown in the results, especially Fig. 12, where a 200 ms negotiation interval is much closer in duration to the negotiation delay. In general, a negotiation interval should be chosen that requires a reasonable amount of edge buffering while operating efficiently with packet scheduling.

IV. Conclusion

In this paper, a contention-aware packet-scheduling scheme coupled with a modified form of renegotiated service with rate prediction has been proposed to reduce contention in core optical-packet switches. Edge nodes use packet scheduling to rearrange possible contended slots before entering the core, thus reducing core optical buffering and moving the computational burden from the core network to the edge, where powerful electronic processing is employed.

The objective of this paper was to minimize FDL buffering and avoid wavelength conversion or deflection routing. Also, core switches are primarily responsible for forwarding optical packets upon their arrival, with less need to resolve contention. Analysis and simulation demonstrate the effectiveness of this proposal in two different network topologies. Large networks may be partitioned into areas, each of which implements packet scheduling independently to address scalability issues.

References

[1] S. Yao, B. Mukherjee, and S. Dixit, "Advances in photonic packet switching: An overview," *IEEE Commun. Mag.*, vol. 38, no. 2, pp. 84–94, Feb. 2000.

[2] D. K. Hunter and I. Andonovic, "Approaches to optical internet packet switching," *IEEE Commun. Mag.*, vol. 38, no. 9, pp. 116–122, Sep. 2000.

[3] I. Chlamtac *et al.*, "CORD: Contention resolution by delay lines," *IEEE J. Sel. Areas Commun.*, vol. 14, no. 5, pp. 1014–1029, Jun. 1996.

[4] D. K. Hunter, M. C. Chia, and I. Andonovic, "Buffering in optical packet switches," *J. Lightw. Technol.*, vol. 16, no. 12, pp. 2081–2094, Dec. 1998.

[5] D. K. Hunter *et al.*, "SLOB: A switch with large optical buffers for packet switching," *J. Lightw. Technol.*, vol. 16, no. 10, pp. 1725–1736, Oct. 1998.

[6] A. S. Acampora and I. A. Shah, "Multi-hop lightwave networks: A comparison of store-and-forward and hot-potato routing," *IEEE Trans. Commun.*, vol. 40, no. 6, pp. 1082–1090, Jun. 1992.

[7] J. P. Jue, "An algorithm for loopless deflection in photonic packet-switched networks," in *Proc. IEEE ICC*, New York, Apr. 2002, vol. 5.

[8] 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.

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

[10] J. J. He, D. Simeonidou, and S. Chaudhry, "Contention resolution in optical packet switching networks under long-range dependent traffic," in *Proc. OFC*, Mar. 2000, vol. 3, pp. 295–297.

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

[12] F. Xue *et al.*, "End-to-end contention resolution schemes for an optical packet switching network with enhanced edge routers," *J. Lightw. Technol.*, vol. 21, no. 11, pp. 2595–2604, Nov. 2003.

[13] H. Zang, J. P. Jue, and B. Mukherjee, "Capacity allocation and contention resolution in a photonic slot routing all-optical WDM mesh network," *J. Lightw. Technol.*, vol. 18, no. 12, pp. 1728–1741, Dec. 2000.

[14] J. Ramamirtham and J. Turner, "Time sliced optical burst switching," in *Proc. IEEE INFOCOM*, San Francisco, CA, Apr. 2003, pp. 2030–2038.

[15] D. K. Hunter and D. R. McAuley, "An IP-over-OPS network," in *Proc. Int. Workshop IP Over WDM*, Pisa, Italy, May 24, 2002.

[16] M. Grossglauser *et al.*, "RCBR: A simple and efficient service for multiple time-scale traffic," *IEEE/ACM Trans. Netw.*, vol. 5, no. 6, pp. 741–755, Dec. 1997.

[17] M. Furini and D. F. Towsley, "Real-time traffic transmission over the internet," *IEEE Trans. Multimedia*, vol. 3, no. 1, pp. 33–40, Mar. 2001.

[18] M. G. Hluchyj and M. J. Karol, "Queueing in high-performance packet switching," *IEEE J. Sel. Areas Commun.*, vol. 6, no. 9, pp. 1587–1597, Dec. 1988.

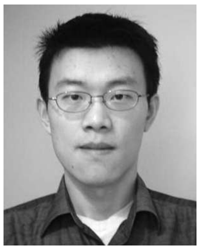
[19] OPNET network modeling and simulation software. [Online]. Available: http://www.opnet.com

[20] V. Paxson and S. Floyd, "Wide area traffic: The failure of poisson modeling," *ACM/IEEE Trans. Netw.*, vol. 3, no. 3, pp. 226–244, Jun. 1995.

[21] A. Erramilli, O. Narayan, and W. Willinger, "Experimental queuing analysis with long-range dependent packet traffic," *IEEE/ACM Trans. Netw.*, vol. 4, no. 2, pp. 209–223, Apr. 1996.

[22] A. Eramilli, W. Willinger, and J. L. Wang, "Modeling and management of self-similar traffic flows in high-speed networks," in *State-of-the-Art in Performance Modeling and Simulation*, K. Bagi, Ed. New York: Gordon and Breach.

[23] W. E. Leland, M. S. Taqqu, W. Willinger, and D. V. Wilson, "On the self-similar nature of ethernet traffic (Extended Version)," *IEEE/ACM Trans. Netw.*, vol. 2, no. 1, pp. 1–15, Feb. 1994.



Zheng Lu received the B.Eng. degree (being awarded the "Excellent Student Prize" every year) from the PLA University of Science and Technology, Nanjing, China and the M.Sc. degree (with distinction) in telecommunications and information systems from the University of Essex, Colchester, U.K., in 2002 and 2003, respectively. Since then, he has been working toward the Ph.D. degree in the Department of Electronic Systems Engineering, University of Essex, where he is researching protocols for optical-packet-switched networks.

David K. Hunter (S'88–M'90–SM'00) received the B.Eng. degree (first-class honors) in electronics and microprocessor engineering from the University of Strathclyde, Glasgow, U.K., where he received the Ph.D. degree for research on optical TDM switch architectures, in 1987 and 1991, respectively.

Thereafter, he remained at the University of Strathclyde to research on optical networking and optical-packet switching, holding an Engineering and Physical Sciences Research Council (EPSRC) Advanced Fellowship from 1995 to 2000. After spending a year as a Senior Researcher in Marconi Labs Cambridge, U.K., he moved to the University of Essex, Colchester, U.K., in August 2002, where his teaching concentrates on TCP/IP, computer networks, and network performance evaluation. He is currently a Reader in the Department of Electronic Systems Engineering in the University of Essex. He has authored or coauthored over 100 publications and has acted as an External Ph.D. Examiner for the Universities of Cambridge, London, and Essex.

Dr. Hunter was an Associate Editor for the IEEE TRANSACTIONS ON COMMUNICATIONS from 1999 to 2003 and for the JOURNAL OF LIGHTWAVE TECHNOLOGY since 2001. He participated in editing a Special Issue of that journal on Optical Networks, which was published in December 2000. He is a professional member of the Association for Computing Machinery (ACM).