Optical Packet Buffers for Backbone Internet Routers

Neda Beheshti, *Graduate Student Member, IEEE*, Emily Burmeister, *Member, IEEE*, Yashar Ganjali, *Member, IEEE*, John E. Bowers, *Fellow, IEEE*, Daniel J. Blumenthal, *Fellow, IEEE*, and Nick McKeown, *Fellow, IEEE*

***Abstract*—If optical routers are to become reality, we will need several new optical technologies, one of which is to build sufficiently large optical buffers. Building optical buffers for routers is daunting: Today's electronic routers often hold millions of packets, which is well beyond the capabilities of optical technology. In this paper, we argue that two new results offer a solution. First, we show that the size of buffers in backbone routers can be made very small—just about 20 packets per linecard—at the expense of a small loss in throughput. Second, we show that integrated delay line optical buffers can store a few dozen packets on a photonic chip. With the combination of these two results, we conclude that future Internet routers could use optical buffers.**

***Index Terms*—Buffer size, integrated optical memory, packet switching, TCP.**

I. Introduction

OVER the years, there has been much debate about whether it is possible—or sensible—to build all-optical datapaths for routers. On one hand, optics promises much higher capacities and potentially much lower power. On the other hand, most of the functions of a router are still beyond optical processing, including header parsing, address lookup, contention resolution and arbitration, and large optical buffers.

Alternative architectural approaches have been proposed to ease the task of building optical routers. For example, label swapping simplifies header processing and address lookup [1]–[3], and some implementations transmit headers slower than the data so they can be processed electronically [4], [5]. Valiant load balancing (VLB) has been proposed to avoid packet-by-packet switching at routers and eliminates the need for arbitration [6].

Manuscript received October 24, 2008; revised July 02, 2009 and February 05, 2010; accepted March 04, 2010; approved by IEEE/ACM TRANSACTIONS ON NETWORKING Editor J. Yates. Date of publication May 24, 2010; date of current version October 15, 2010. This work was supported by DARPA/MTO DOD-N Award W911NF-04-0001/KK4118 (LASOR PROJECT).

N. Beheshti was with the Computer Systems Laboratory, Stanford University, Stanford, CA 94305 USA. She is now with Ericsson Research Lab, San Jose, CA 95134 USA (nbehesht@stanfordalumni.org).

E. Burmeister is with Ciena Corporation, Linthicum, MD 21090 USA (e-mail: emily.burmeister@gmail.com).

Y. Ganjali is with the Department of Computer Science, University of Toronto, Toronto, ON M5S 2E4, Canada (e-mail: yganjali@cs.toronto.edu).

J. Bowers and D. J. Blumenthal are with the Department of Electrical and Computer Engineering, University of California, Santa Barbara, CA 93106-9560 USA (e-mail: bowers@ece.ucsb.edu; danb@ece.ucsb.edu).

N. McKeown is with the Computer Systems Laboratory, Stanford University, Stanford, CA 94305 USA (e-mail: nickm@stanford.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TNET.2010.2048924

In this paper, we consider just one function of an optical router—optical packet buffering—and ask the question: *Is it possible to build optical buffers for an Internet router?*

Conventional wisdom says that it is not. Electronic Internet backbone routers today maintain *millions* of packet buffers in first-come–first-served queues. None of the many proposed schemes to build optical buffers comes close to replacing the huge buffers in an electronic router.

The basic premise of this paper is that because of two recent innovations, we are now much closer to being able to build optical buffers for a backbone router. First, as we show in Section III, there is growing evidence that backbone networks can be built from routers with very small buffers, perhaps only a few dozen packet buffers on each line in each router, if we are willing to sacrifice a small amount of throughput. Second, as we show in Section IV, it is now possible to build optical packet buffers that are capable of holding a few dozen packets in an integrated optoelectronic chip. We describe both innovations, show how they can be applied to build packet buffers for optical routers, and explain some of the shortcomings yet to be overcome.

II. Why do Routers Have Buffers?

There are three main reasons that routers have buffers.

1) *Congestion*: Congestion occurs when packets for a switch output arrive faster than the speed of the outgoing line. For example, packets might arrive continuously at two different inputs, all destined to the same output. If a switch output is constantly overloaded, its buffer will eventually overflow, no matter how large it is; it simply cannot transmit the packets as fast as they arrive. Short-term congestion is common due to the statistical arrival time of packets. Long-term congestion is usually controlled by an external mechanism, such as the end-to-end congestion avoidance mechanisms of TCP, the XON/XOFF mechanisms of Ethernet, or by the end-host application. In practice, we have to decide how big to make the congestion buffers. The decision is based on the congestion control mechanism—if it responds quickly to reduce congestion, then the buffers can be small; else, they have to be large. The congestion buffers are the largest buffers in a router, and so will be our main focus in this paper. A typical Internet router today holds millions of packet buffers for congestion.
2) *Internal Contention*: Even when the external links are not congested, most packet switches can experience internal contention because of imperfections in their datapaths and arbitration mechanisms. The amount of contention, and therefore the number of buffers needed, is determined by

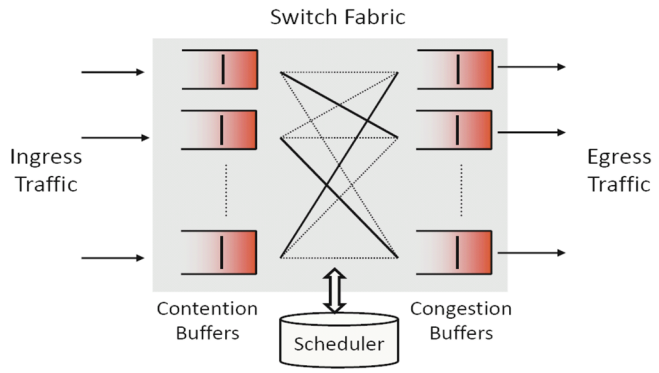


Fig. 1. Buffering in a CIOQ router. Input buffers store packets when there is internal contention. Output buffers store packets when output links are congested.

the switch architecture. For example, an output-queued switch has no internal contention and needs no contention buffers. At the other extreme, an input-queued switch can have lots of internal contention, as typified in the seminal paper of Karol [7] that shows contention can limit the throughput of an input-queued switch to just 58% of its maximum. Between the two extremes, it is possible to build input-queued switches with 100% throughput [8], [9]. These switches need large internal buffers (theoretically, of infinite depth) to hold packets during times of contention. Some architectures can precisely emulate output queueing [10], [13] through careful arbitration and a combination of input and output queues (CIOQ). These switches still need contention queues (at their inputs) to hold packets while the arbitration algorithm decides when to deliver each to its output queue. Most switches today use CIOQ or multiple stages of CIOQ. As we will see in the next section, CIOQ switches typically need very small contention buffers. Fig. 1 shows the generic architecture of a CIOQ switch.

3) *Staging*: Packet switches also have staging buffers for *pipelining* and *synchronization*. Most designs have hundreds of pipeline stages, each with a small fixed-delay buffer to hold a fixed amount of data. Most designs also have multiple clock domains, with packets crossing several domains between input and output; each transition requires a small fixed-size FIFO. In this paper, we will not be considering staging buffers. Their sheer number means they cannot be ignored, but because they are of fixed size and delay, they can be implemented in various ways using small optical delay lines.

III. How Big Should the Buffers Be?

The historical answer to this question is the well-known rule of thumb: Buffers should be at least as large as the delay-bandwidth product of the network to achieve full utilization, i.e., $B = RTT \times C$, where RTT is the average round-trip time of flows, and C is the data-rate of the bottleneck link. According to this rule, 1-Gb buffers are required for a 10-Gb/s link, with an average two-way delay of 100 ms. To follow the rule, this number has to grow linearly as the link speed increases.

Recently, Appenzeller *et al.* [11] showed that with N concurrent flows on the link, the buffer size can be scaled down to $B = RTT \times C/\sqrt{N}$, without compromising the throughput. This means a significant reduction in the buffer size of backbone routers because backbone links often carry tens of thousands of flows. With 10 000 flows on a link, the buffer size can be reduced by 99% without any change in performance (i.e., a 1-Gb buffer becomes 10 Mb). This result has been found to hold very broadly in real networks [11], [15].

However, even at 10 Mb, a packet buffer is too large to be implemented optically. Therefore, in this paper we argue that, with two caveats, we can reduce the buffer size even further, to as small as 20 packets. The first caveat is that we must be willing to sacrifice about 15% of the link capacity (e.g., a 100-Gb/s link will operate like an 85-Gb/s link). In the very fastest networks, this might be an acceptable tradeoff to be able to use an all-optical datapath. The second caveat is that we must take steps to ensure the arriving traffic is not too bursty. This turns out to be easier than one might expect: We have found that in a typical backbone network, the multiplexed traffic is sufficiently smooth for our results to hold.

Replacing million-packet buffers by 20-packet buffers in a router linecard implies huge savings in power consumption, board space, and cost and eliminates delay jitter. Most importantly here, this result is very well suited to what can be built by the current optical technology, as we will explain in Section IV.

A. How Big Should the Congestion Buffers Be?

To understand how large to make the congestion buffers, it helps to study output-queued routers, in which packets are immediately transferred to the output ports as soon as they arrive. Each output port has one FIFO queue that is shared by all the flows going through that port. The size of the buffer depends on the arrival traffic: If traffic is light or nonbursty, buffers can be very small; if big bursts arrive, we need much bigger buffers.

In what follows, we explore how large to make the congestion buffers in three scenarios in turn:

1) when a link carries just one TCP flow. This turns out to be the worst case, and leads to the rule of thumb $B = RTT \times C$;
2) when a link carries many TCP flows, allowing us to reduce the buffer size to $B = RTT \times C/\sqrt{N}$;
3) finally, when traffic comes from slower access networks, or when the source paces the packets it sends. In this case, we can reduce the buffer size to about 20 packets. We will refer to this rule as the *tiny buffers* rule.

1) When a Link Carries Just One TCP Flow: To understand why we need $RTT \times C$ buffers with just one TCP flow, we need to understand the dynamics of TCP. The dynamics of a TCP flow are governed by the window size (the number of outstanding unacknowledged packets). A long-lived flow spends most of its time in the additive-increase and multiplicative-decrease (AIMD) congestion-avoidance mode, during which the window size increases additively upon receiving an ACK packet and is halved when a packet or ACK is lost.

The buffer in a router's output port should be big enough to keep the outgoing link busy during times of congestion, so as to

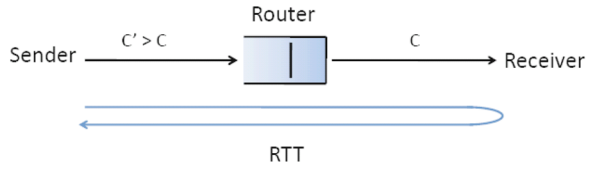


Fig. 2. Single-bottleneck topology. The sender's access link is faster than the receiver's bottleneck link, causing packet accumulation in the router.

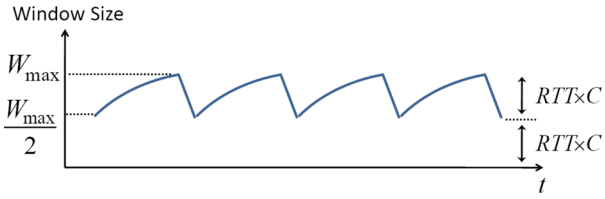


Fig. 3. Window size dynamics of a TCP flow through a bottleneck link. To achieve 100% utilization, the buffer size should be large enough to store $RTT \times C$ packets.

maximize the throughput of the network. If the buffer ever goes empty, the link goes idle and we waste the link capacity.

On the other hand, TCP's sawtooth congestion control algorithm is designed to fill any buffer and deliberately causes occasional loss to provide feedback to the sender. No matter how big we make the buffers at a bottleneck link, TCP will occasionally overflow the buffer.

Consider the simple topology in Fig. 2, where a single TCP source sends data packets to a receiver through a router. The sender's access link is faster than the receiver's bottleneck link of capacity C packets per second, causing packets to be queued at the router. Assume the buffer size at the output link of the router is B. The sender transmits a packet each time it receives an ACK and gradually increases the number of outstanding packets (the window size), which causes the buffer to gradually fill. Eventually a packet is dropped, and the sender does not receive an ACK. It halves the window size and pauses until the number of outstanding packets has fallen to $W_{\max}/2$ (where $W_{\max}$ is the peak window size). Fig. 3 shows the window size dynamics.

The key to sizing the buffer is to make sure that while the sender pauses, the router buffer does not go empty and force the bottleneck link to go idle.

The source pauses until it receives $W_{\max}/2$ ACK packets, which arrive in the next $W_{\max}/2C$ seconds (remember that C is the bottleneck bandwidth). During the pause, $W_{\max}/2$ packets leave the buffer; for the bottleneck link to stay busy, the buffer needs to hold at least $W_{\max}/2$ packets when the pause starts. Now, we just need to determine $W_{\max}$.

At the instant the pause is over, the source can send $W_{\max}/2$ consecutive packets as $W_{\max}/2$ ACKs arrive. It then pauses until it receives an ACK one RTT later (the first ACK arrives after exactly one RTT because the buffer is empty). In other words, it sends $W_{\max}/2$ packets in one RTT, which must be just enough to keep the bottleneck link busy; i.e., $((W_{\max}/2)/RTT) = C$, which means $B = RTT \times C$, the rule of thumb for one TCP flow.

2) When Many TCP Flows Share a Link: If a *small* number of flows share a link, the aggregate window size (the sum of the individual window sizes) tends to follow the same TCP sawtooth, and B is the same as for one flow.

If *many* flows share a link, small variations in RTT and processing time desynchronize the flows [18]–[20], and the aggregate window size becomes smoother with more flows. This is studied in detail in [11], where it is shown that with N long-lived TCP flows, variations in the aggregate window size scales down by a factor $\sqrt{N}$. As with one flow, variations in the aggregate window size dictates the buffer size needed to maintain full utilization of the bottleneck link. Hence, $B = RTT \times C/\sqrt{N}$.

3) When Traffic Comes From Slow Access Networks: In backbone networks, another interesting effect takes place. In addition to the aggregate TCP AIMD sawtooth becoming smoother, each individual flow also becomes smoother. This is because a backbone network interconnects many slower networks. When packets from slower networks are multiplexed together onto a fast backbone, the bursts are spread out and smoothed. We will see that the smoothing substantially reduces the required buffer size.

To get a feel for how smoothing could help reduce the buffer size, imagine for a moment that the traffic was so smooth that it became Poisson. The drop rate would have an upper bound of ρ^{B}, where ρ is the link utilization and B is the buffer size. At 80% load and with a 20-packet buffer, the drop rate would be about 1%, independent of RTT and C. At the other extreme, compare this to the buffer size needed for 100% utilization with a single TCP flow, when RTT is 200 ms and C is 10 Gb/s; $\mathrm{B} = 2$ Gb, or about a million average-sized packets.

Traffic in backbone networks cannot be modeled as a collection of independent Poisson flows. A TCP flow can send a whole window of packets at the start of each RTT, creating significant bursts. However, there are two ways the bursts can be broken. We can explicitly break them by using Paced TCP [17], in which packets are spread uniformly over the round-trip time. The rate and behavior of each flow is almost indistinguishable from regular TCP, but as we will see shortly, the amount of required buffering drops significantly.

Even if we do not modify the TCP source, the burst is naturally broken if the core links are much faster than the access links, as they typically are. As the packets from one flow enter the core, they are spread out, with gaps or packets from other flows being multiplexed between them.

To see how breaking the bursts reduces the required buffer size, we start by analyzing TCP traffic with smooth packet injection. Sources follow the AIMD dynamics, but rather than sending out packets in bursts, they spread traffic over an RTT.

Assume that N long-lived TCP flows share a bottleneck link. Flow i has a time-varying window size $W_i(t)$ and follows TCP's AIMD dynamics. If the source receives an ACK at time t, it will increase the window size by $1/W_i(t)$, and if the flow detects a packet loss, it will decrease the congestion window by a factor of two. In any time interval $(t, t']$ when the congestion window size is fixed, the source will send packets as a Poisson process at rate $W_i(t)/RTT$. Under this assumption, buffering $O(\log W_{\max})$ packets is sufficient to obtain close to peak throughput. This result is stated more precisely in the following theorem and is proved in [14].

Theorem 1: To achieve an effective utilization of θ, a buffer of size

$$B \geq \log_{1/\rho} \frac{W_{\max}^2}{2(1-\theta)}$$

suffices, if the network is overprovisioned by a factor of $1/\rho$, where ρ is less than or equal to 1.

This result assumes that the network is overprovisioned. In other words, it assumes that the maximum traffic rate—with all TCP sources simultaneously transmitting at their maximum rate—is $1/\rho$ times smaller than the bottleneck-link bandwidth. Although this result has not been extended to the underprovisioned case, the simulation results of Section III-C indicate that overprovisioning is not a requirement. Here, θ is the desired effective utilization of the shared link. It represents the fraction we aim to achieve out of the maximum possible utilization ρ (i.e., a fraction $\theta\rho$ of the full link rate).

Theorem 1 suggests that TCP traffic with $W_{\max} = 83$ packets[1] and $\rho = 75\%$ needs a buffer size of 37 packets to achieve an effective utilization of 90%.

According to Theorem 1, if the offered load is constant, then the buffer size needs to increase only logarithmically as the maximum window size increases. In a TCP connection, $W_{\max}$ is the maximum amount of data the transmitter can send over one RTT. This amount is limited by the source transmission rate, even if the operating system does not explicitly limit $W_{\max}$: At a source rate of C_T, at most $C_T \times RTT$ packets can be sent over a round-trip time. If this amount increases from 100 to 10 000 packets, then the buffer size only needs to be doubled.

In [14], Theorem 1 is extended to show that if access links run at least $\log W_{\max}$ times slower than the bottleneck link, approximately the same buffer size is enough. In our example above, $\log W_{\max}$ was less than 7, whereas in practice access links are often two orders of magnitude slower than backbone links (for example, a 10-Mb/s DSL link multiplexed eventually onto a 10-Gb/s backbone link). Under these conditions, the packet loss probability is comparable to Poisson traffic with the same buffer size.

To compare the required buffer size in the above three scenarios, we illustrate them through the simulation of a 10-Gb/s bottleneck link with 800 long-lived TCP flows sharing the link (Fig. 4). The average RTT is 100 ms. We measure the link utilization as we vary the buffer size from only one packet to $RTT \times C = 125\,000$ packets. As the graph shows, utilization remains almost unchanged (and above 99%) with buffer sizes larger than $RTT \times C/\sqrt{N} \approx 4419$ packets. When access links run 100 times slower than the bottleneck link, i.e., at 100 Mb/s, we can set the buffer size to only 10 packets and achieve close to 80% utilization.

B. How Big Should the Contention Buffers Be?

Now, we turn our attention to the size of the contention buffers. Contention is caused by the switch architecture. If we were building an output-queued switch, we would not need any contention buffers. Unfortunately, building an optical output-queued switch is hard because (as with electronic switches) it is hard to build a buffer that can accept packets from all inputs simultaneously.

[1]A 10-Mb/s flow of 1500-byte packets filling a path with an RTT of 100 ms.

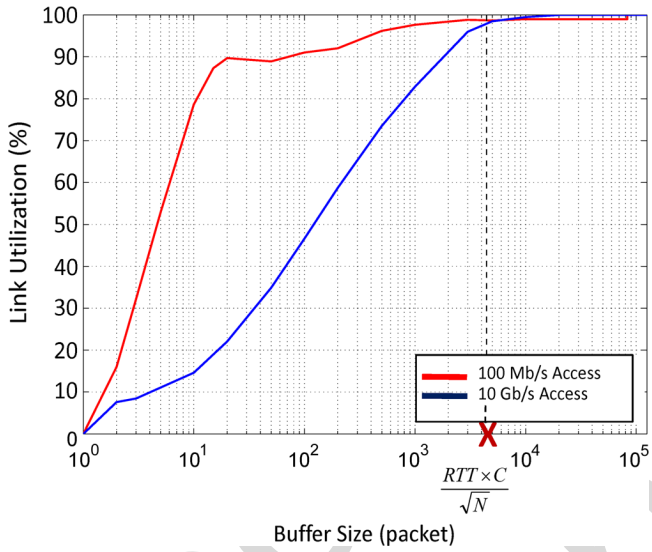


Fig. 4. Link utilization versus buffer size. With 800 flows on the link, close to 100% utilization is achieved if the buffer size is $RTT \times C/\sqrt{N}$. If flows come from slower access links, a tiny buffer size of 10 packets suffices for 80% utilization.

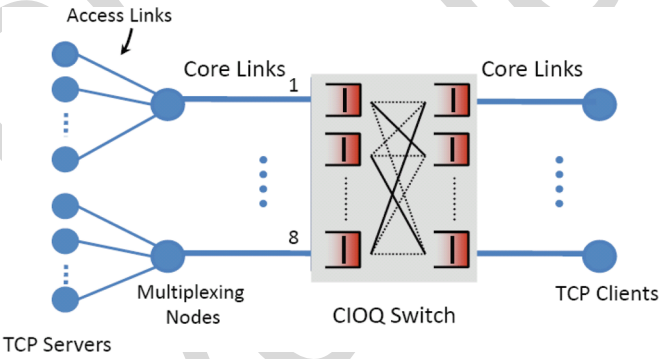


Fig. 5. Simulated network topology.

The size of contention buffers in a CIOQ switch depends on the internal speedup of the switch (i.e., how fast the switch fabric runs compared to the link rate). Larger speedups reduce the average number of packets waiting at the input side since packets are removed faster from input buffers.

In Appendix A, we show that when speedup is at least 2, the occupancy of contention buffers on any port is less than twice the size of congestion buffers. In other words, buffer size $O(\log W_{\max})$ at input ports is enough to achieve the same performance as with an output-queued switch. Our analysis assumes that a stable matching algorithm [13] configures the switch fabric. However, simulation results of Section III-C show that even with more practical algorithms, very small input buffers result in high utilization.

Note that the *tiny buffers* rule does not guarantee that packets are not dropped; TCP requires some packet drops in order to function well. Our results show that with these tiny buffers, TCP will perform well and the throughput will be high, though not 100%.

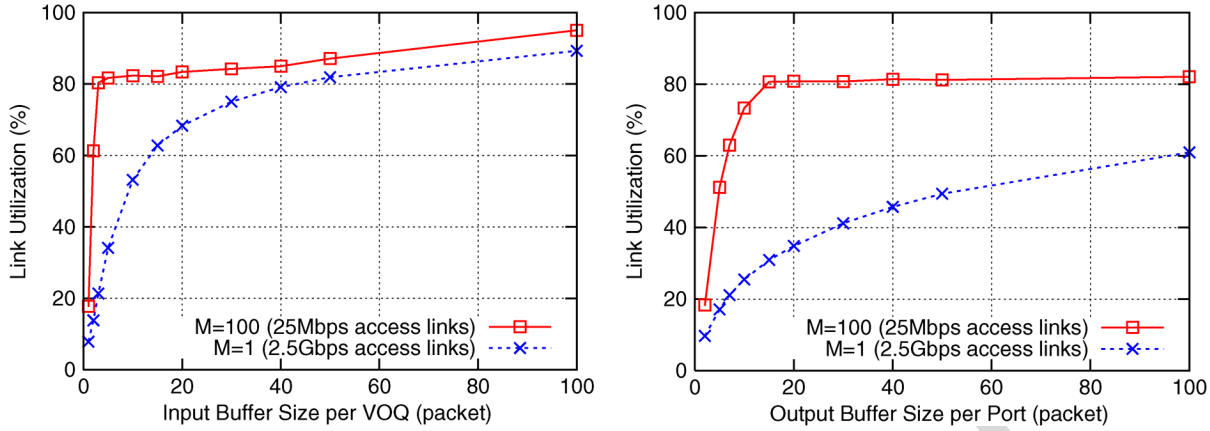


Fig. 6. Link utilization versus input and output buffer sizes. Left: Speedup is 1, and all the queueing takes place at the input. Right: Speedup is 8, and all the queueing takes place at the output. With 25-Mb/s access links, five-packet VOQs and 15-packet output buffers make the utilization above 80%.

C. Simulation Results

To validate the results of Section III-A and B, we perform simulations using the ns-2 simulator [12]. We have enhanced ns-2 to include an accurate CIOQ router model.

Fig. 5 shows the topology of the simulated network. Flows are generated at separate source nodes (TCP servers) using TCP Reno,[2] go through individual access links, and are multiplexed onto faster backbone (core) links before reaching the input ports of the switch. Large buffers are used at the multiplexing nodes to prevent drops at these nodes. Core links run at 2.5 Gb/s, and the propagation delay between each server–client pair is uniformly picked from the interval 75–125 ms (with an average of 100 ms). All data packets are 1000 bytes.

The simulated switch is a CIOQ switch, which maintains virtual output queues (VOQs) at the input to eliminate head-of-line (HOL) blocking. In each scheduling cycle, a scheduling algorithm configures the switch and matches input and output ports. Based on this configuration, either zero or one packet is removed from each input port and is sent to the destination output port. All input and output buffers use the FIFO queueing policy.

We define M to be the multiplexing factor, which is the ratio of the backbone-link speed to access-link speed. Today, a typical user is connected to the network via a 10-Mb/s DSL link, and backbone links often run at 40 Gb/s; i.e., $M = 4000$. In our experiments, we conservatively pick M to be 100.

We relax the overprovisioning assumption of the previous sections and make the offered load on every output link 100%. In other words, we set the number of flows (N) sharing output links and the maximum TCP window size $(W_{\max})$ such that the maximum aggregate traffic rate on each output link is equal to the link capacity:

$$\frac{N \times W_{\max}}{RTT} = C$$

With an average RTT of 100 ms and $W_{\max} = 64$ kB, we need about 490 flows on each core link to fill the link.

[2]We only consider long-lived TCP flows since the link utilization is mainly determined by the behavior of these flows.

Baseline: To begin with, we choose a baseline setting, where the switch is an 8×8 switch, and the load is distributed uniformly among output ports, i.e., all output ports are equally likely to be the destination port of a given flow. In this baseline setting, the switch configuration is controlled by the Maximum Weight Matching (MWM) algorithm. MWM is known to deliver 100% utilization for admissible traffic distributions [8], [10], but the algorithm is too complex to be implemented in real routers.

Fig. 6 shows the average link utilization versus input and output buffer sizes in the baseline setting. To see the effect of these buffer sizes independently, we first set the switch speedup to 1, which makes the switch function as an input-queued switch. With a speedup of 1, there is no queueing at output ports because the switch fabric runs no faster than the output links. Next, we set the switch speedup equal to the switch size (8) to eliminate input queueing. With a speedup of 8, the switch functions as an output-queued switch and needs buffering only at the output side. In both input-queued and output-queued scenarios, we run the simulations twice: first with $M = 1$, i.e., access links run at 2.5 Gb/s, and then with $M = 100$, i.e., access links run at 25 Mb/s.

Fig. 6 shows the huge benefit of a larger M. Because the network naturally spaces out packets of each flow, much smaller buffer size is required for high utilization. The plots show that when access links run at 25 Mb/s, buffering five packets in each VOQ and 15 packets at each output port suffices for 80% utilization. These numbers increase to 40 and more than 400 (not shown on this plot), respectively, when access links run as fast as core links.

With speedups between 1 and 8, we can combine the results shown in Fig. 6: For each pair of input and output buffer sizes, utilization is not lower than the minimum utilization shown on these two graphs at the given input (left) and output (right) buffer sizes. This is because if speedup is greater than 1, packets are removed faster from the input queue, and the required buffer size goes down. If speedup is smaller than 8, packets reach the output queue later, and hence the backlog is smaller.

Therefore, with any speedup, we can achieve more than 80% utilization with five-packet VOQs and 15-packet output queues in the baseline setting. Remember that this result is with 100%

offered load on the output links. This suggests that Theorem 1 is conservative in its overprovisioning assumption.

Changing the simulation settings from what we considered in the baseline setting could affect the utilization. However, our analysis and simulations with different settings (e.g., traffic load, TCP flavor, switch parameters, and network topology) resulted in similar buffer size requirements as in the baseline setting—i.e., a few tens of packets [21], [22]. Appendix B discusses the effect of switch parameters on the required buffer size in more detail.

D. Link Utilization Metric

In this paper, our metric for buffer sizing is *link utilization*. This metric is operator-centric; if a congested link can keep operating at 100% utilization, then it makes efficient use of the operator's congested resource. This is not necessarily ideal for an individual end-user since the metric does not guarantee a short flow completion time (i.e., a quick download). However, if the buffer size is reduced, then the round-trip time will also be reduced, which could lead to higher per-flow throughput for TCP flows. The effect of tiny buffers on user-centric performance metrics, such as flow completion time, has been discussed in [15] and [16].

The *tiny buffers* rule assumes that we are willing to sacrifice some throughput and, for example, operate the network at 80%–90% utilization. This might sound wasteful at first glance. However, we should note that in an optical network, capacity is abundant and the buffer size is the bottleneck. In a 40-Gb/s backbone link, we can expect to lose about 20% of the throughput. In other words, the 40-Gb/s link will operate like a 32-Gb/s link.

Our results suggest that the required buffer size is independent of the absolute bandwidth of the bottleneck link. Fig. 7 shows how link utilization stays unchanged when we increase the bottleneck link bandwidth but keep $M = 100$ by increasing the access link bandwidth proportionally (the dotted curve). The buffer size is constant at 20 packets per port. The solid curve in this figure shows utilization of the bottleneck link when the access bandwidth is fixed at 25 Mb/s. In this case, increasing the backbone link bandwidth creates more spacing between packets and reduces burst size; hence, the utilization improves.

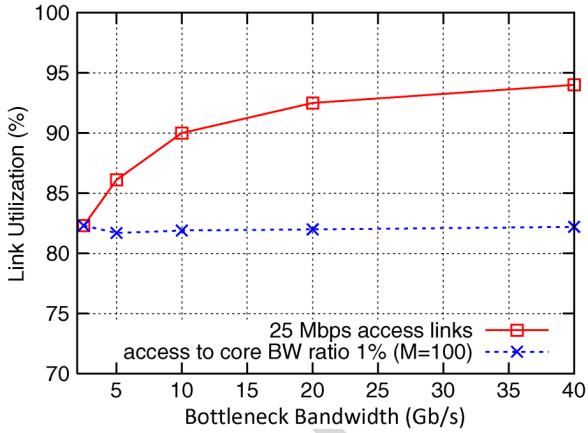


Fig. 7. Link utilization versus bottleneck-link bandwidth. With a fixed access-to-core bandwidth ratio (1%) and a fixed buffer size (20 packets), increasing the bottleneck-link bandwidth does not change the utilization.

IV. How Can Optical Data Be Stored?

After decades of research in optical buffering devices, the first integrated optical random access memory element has recently been demonstrated [25]. Together, this breakthrough and the buffer-sizing results presented in the previous sections demonstrate the feasibility of building optical buffers. This section will show that a physical buffer can be built that will meet all of the necessary requirements for an optical router. We will focus here on integrated recirculating buffers since they have been demonstrated to be a viable approach for high-speed buffering of hundreds of packets and are scalable to thousands of integrated buffers.

A. Optical Buffering Approaches

Storage of optical data is accomplished by delaying the optical signal—either by increasing the length of the signal's path or by decreasing the speed of the light. In both cases, the delay must be dynamically controlled to offer variable storage times, i.e., to have a choice in when to read the data. Delay paths provide variable storage time by traversing a variable number of short delay lines—either several concatenated delays (feedforward configuration) or by looping repeatedly through one delay (feedback configuration). Buffers that store optical data through slowing the speed of light do so by controlling resonances either in the material itself or in the physical structure of the waveguide. Integrated recirculating (feedback delay line) buffers have been shown to be the most promising solution by evaluating the requirements that optical memory must meet to provide a viable solution for optical routers [26].

B. Buffering Requirements

Optical memory elements will not immediately surpass electrical memory in all aspects, but must certainly meet several requirements in order to be a reasonable replacement and to meet network performance metrics. First, buffers must be bit-rate-scalable to 40 Gb/s and higher to be considered for future networks. Second, acceptable network loads dictate that packets should be at least 40 bytes, and guard bands no more than a few percent of the packet length. Third, the size, weight, and power of the optical buffer should be at least comparable to electronic memory. Finally, it is critical that the number, complexity, and monetary cost of components included in a given buffer architecture be kept to a minimum to result in a competitive router that is practical to implement. In addition, transparency to packet length and dynamically variable storage times should also be considered as they can lead to better performance. In addition to the above requirements, we focus on architectures that can be integrated on a chip. Integration affords a smaller footprint, lower power requirements, and lower cost.

C. Integrated Delay Line Buffer Structure

The base memory element shown in Fig. 8 can be built using two photonic chips and cascaded to form a practical optical buffer for many packets. The element is flexible in that it may be used as a recirculating (feedback) buffer for a small footprint

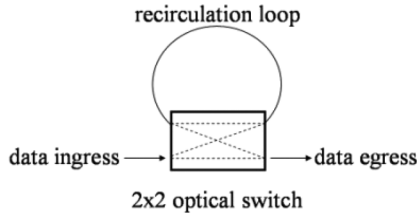


Fig. 8. Schematic of a feedback buffer. A 2×2 switch is combined with a waveguide loop to provide variable delay for an optical signal.

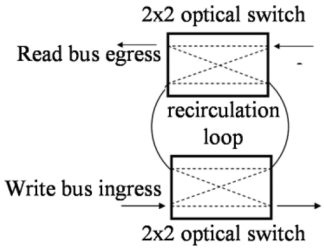


Fig. 9. Physical implementation of speedup and simultaneous read/write.

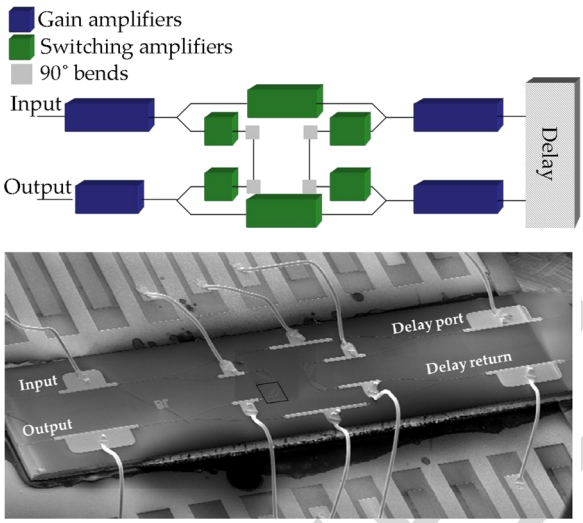


Fig. 10. Schematic and SEM of a fabricated SOA gate matrix switch wire-bonded to an aluminum nitride submount.

and low component count, or concatenated to form a feed-forward buffer for arbitrary packet lengths. Feed-forward configurations require N loops to store a packet for N packet durations, while feedback loops can store packets for many recirculations, presently around 10, but ultimately 100 packet lengths or more, as discussed below.

These buffer elements can also enable easy implementation of simultaneous read/write as well as speedup. These are both definite advantages for the CIOQ architecture. The design extension to enable a speedup of 2 and simultaneous read/write is shown in Fig. 9.

D. Device Design and Results

The integrated buffer is a simple structure, relying on only one passive element—the delay line—and one active element—a 2×2 switch. There are many 2×2 optical switch structures. We have focused on a semiconductor optical amplifier (SOA) gate matrix switch (Fig. 10) because it has large ($\sim$40 dB) extinction ratios, which is important for long storage times. The SOA gate matrix operation is that of a broadcast-and-select architecture. Inside the switch, the signal is directed toward both output ports and passes through three to four amplifiers on each route. The amplifiers at the edges by the ports are used solely for gain, but the center amplifiers can be turned off to absorb the portion of the signal traveling through that path. Thus, the signal for the desired output port is amplified, while the signal at the alternative output port is completely eliminated. The optical amplifiers use an InGaAsP offset quantum well structure. This particular switch design exhibits high extinction (>40 dB), low crosstalk (< -40 dB), and fast switching times (1-ns rise time 20%–80%) [27]. These extinction and crosstalk values guarantee that interference will not limit buffer performance. As previously mentioned, the ability to switch within several nanoseconds along with packet lengths of at least 40 B allows for increased throughput.

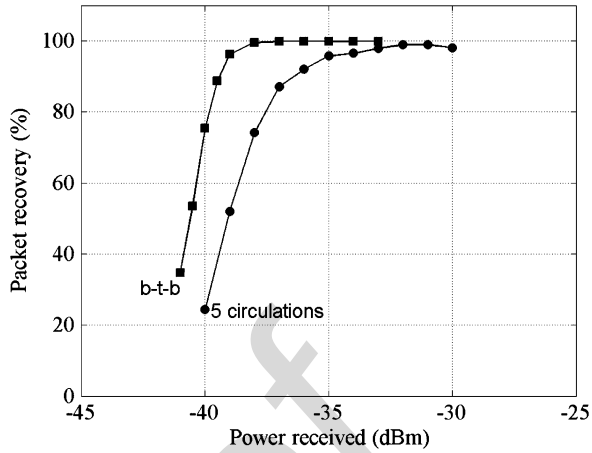


Fig. 11. Packet recovery measurements showing 98% packet recovery for up to five circulations, or 64 ns of storage.

The switch is coupled to a low-loss waveguide delay line to form the integrated buffer element. Silica optical delay lines have low loss, on the order of 0.01 dB/cm. They can be spiral-wound for small size, of order 1 cm^2 in area for a length of 2 m, which is sufficient for 40-byte packets at 40 Gb/s or more. Furthermore, these delay lines can be interleaved such that 16 such delay lines can be integrated into this size.

The integrated optical buffer described achieved 64 ns of packet storage, or five circulations, with 98% packet recovery. Fig. 11 shows the packet recovery measurements, illustrating that although slightly more optical signal power was needed to achieve the same performance, the buffer prototype was successful. Buffering between two packet streams was also demonstrated with both a fiber version [28] and the integrated version described [25].

E. Future Work

This initial demonstration of optical buffering indicates what is possible, but is primitive compared to what should become available over the next few years. This work used silica waveguides butt-coupled to InP gate matrix arrays. Park *et al.* have demonstrated a similar structure using silicon waveguides

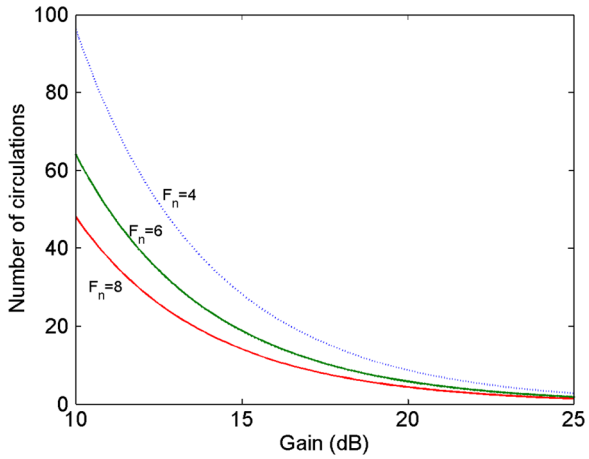


Fig. 12. Maximum number of circulations possible as a function of the loop gain needed in order to maintain an OSNR of 20 dB. Curves are shown for a range of common amplifier noise figures.

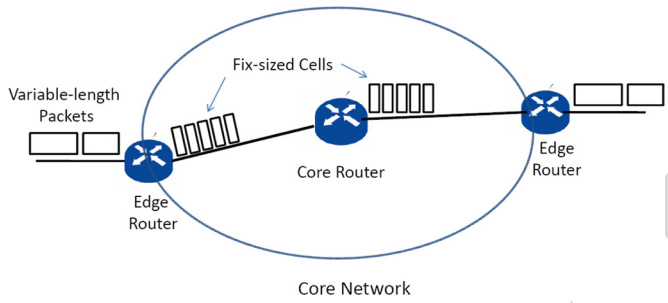


Fig. 13. Core network with fixed-size cells. Edge routers break variable-length packets into small fixed-size cells and reassemble them back into original packets as they depart the core network.

with integrated SOAs [29]. This approach is fully integrated and eliminates the coupling loss between the chips, albeit at the expense of higher propagation loss.

Overall loss is the primary limitation of this optical buffer approach as it reduces the signal-to-noise ratio (amplification is accompanied by amplified spontaneous noise). Through simple design changes, the loss can be easily and drastically reduced. With this improvement and the optimization of the amplifiers, hundreds of circulations will be possible, as shown in Fig. 12. In addition, delay lines incorporating 3R (reamplification, retiming, and reshaping) regeneration should become available. Operation at higher bit rates with faster switch times is also not fundamentally limited. With further advances in integration, hundreds of buffers on one chip should be possible in the next few years.

V. Buffers for Fixed-Size Small Packets

The integrated delay loop structure we introduced in Section IV is capable of buffering fixed-size 40-byte optical packets. On the other hand, in the baseline setting of Section III-C, we assumed that data packets were 1000 bytes, and showed that 15-packet buffers made the utilization above 80%. Here, we want to know whether we can set the buffer size as small with short packets and yet achieve the same performance as what we achieve with long packets.

Fig. 13 shows a network architecture where packets are segmented by edge routers as they enter the core network. Optical buffers of Section IV are designed for fixed-size cells with length equal to the recirculation delay of the memory loops. To implement these buffers in the core network, edge routers must be able to break variable-length packets into fixed-size small cells. Egress edge routers reassemble these fixed-sized cells back into the original packets as they depart the core network. This architecture eliminates the variable-length-packet problem, but can we apply the buffer sizing results to routers regardless of the cell size?

Simulation results [21] show that if packet segmentation happens in slow access networks—i.e., before packets are multiplexed on fast core links—then packet length does not have a significant impact on the required buffer size. For example, at 100% load and 15-packet buffer size, link utilization decreases from 80% to about 71% when the packet size goes from 1000 to 100 bytes. This difference in utilization becomes smaller when the load on the bottleneck link becomes smaller [21].

VI. Conclusion

Optical buffering in Internet routers is not a myth any longer. Integrated optical memory loops have been fabricated and tested. On the other hand, theory, simulations, lab experiments, and experiments in operational networks suggest that under some conditions, a core network will run fine with stringing a tiny number (about 20) of these memory loops together on routers' linecards. This result is well suited to what we can build currently to achieve an acceptable signal-to-noise ratio level—not to mention the critical reduction it brings in the cost, footprint, and energy consumption of routers' buffers.

Our buffering approach is capable of storing optical packets at 40-Gb/s bandwidth with measured performance comparable to electrical memory devices. Currently, these optical recirculating memory devices can store 40-byte packets for up to 10 packet recirculation time. The maximum storage time is expected to increase to a few hundred packets in the near future by applying new methods of loss reduction.

Achieving high throughput with tiny buffers in backbone routers is conditional on one main assumption: that the traffic of individual flows does not appear very bursty on core links. This condition is satisfied if core links run faster than access links. The difference in bandwidth must be large enough to eliminate short-term traffic bursts of individual flows in the core. However, if that is not the case, then paced TCP should be implemented to space traffic generated at the source.

Optical memory loops are designed for buffering fixed-size packets. The storage-time resolution of the loop is limited by its recirculation delay. Thus, it works best if all packets are of length equivalent to this delay time. To handle variable-length packets, edge routers must be able to segment packets into small fixed-size cells before they enter the core and to reassemble the cells into original packets as they depart the core.

The 10%–20% throughput loss that 20-packet buffers result in, in addition to the overhead introduced by segmenting packets, will limit utilization on core links. However, link bandwidth is the abundant resource in optical core networks. The core is usually the most overprovisioned part of the network, so

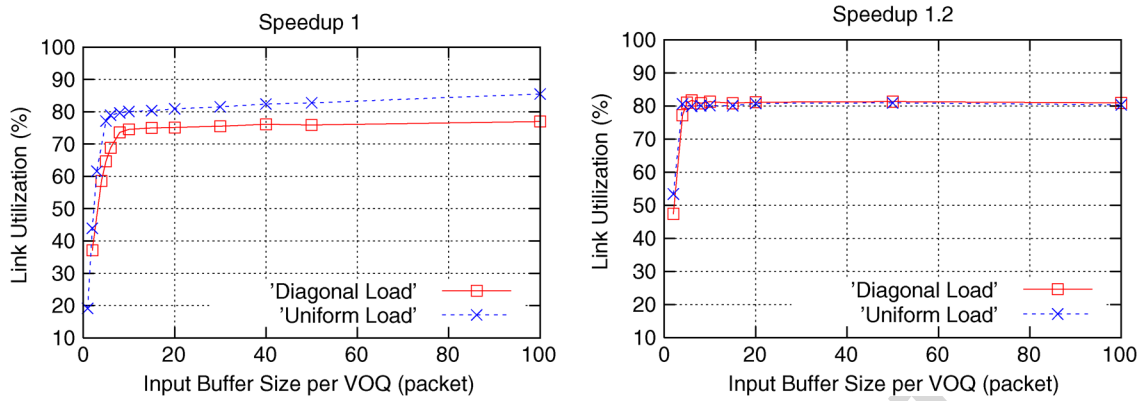


Fig. 14. Link utilization versus buffer size with iSLIP. Left: speedup $= 1$. Right: speedup $= 1.2$.

losing a small fraction of the link bandwidth would still make the network work fine.

Appendix A

We consider a CIOQ router and show that with a speedup of at least 2 and output buffers of size B, the occupancy of input buffers can be made smaller than $2B$.

Definition: Consider two routers R and S, and assume that the same input traffic is fed to both routers. Router R is said to exactly emulate router S if it has exactly the same drop sequence and the same departure sequence as router S.

If input and output buffer sizes are unlimited, a CIOQ router (with a speedup of at least 2 and a stable marriage scheduling algorithm) can exactly emulate an OQ router [13]. In other words, despite the contention at the input side, the CIOQ router does not keep packets longer than the OQ router. Now, assume that S is an OQ router and R is a CIOQ router, both with output buffers of size B. Consider the scenario where router R drops an arriving packet exactly when router S does so (i.e., when the total number of packets destined for a given output port exceeds B). We show that the occupancy of the input buffers in router R is limited according to the following theorem.

Theorem 2: If router R exactly emulates router S, then at any time t, $Q(i,t) \leq 2B$, where B is the size of output buffers in both routers and $Q(i,t)$ is the buffer occupancy of router R at input port i.

Proof: Assume the contrary. There must be a time t_0 and an input port i_0 such that $Q(i_0,t_0) > 2B$. With speedup of 2, at most two packets are removed from port i_0 at any time slot. Therefore, there is a packet in router R that cannot be sent out of the router in B time slots. This contradicts the exact emulation assumption, since any packet in the OQ router is sent out in at most B time slots. ■

Appendix B

In this appendix, we will see how the switch parameters (switch size, scheduling algorithm, and load distribution) affect link utilization and the required buffer size. Network topology and traffic characteristics are the same as in the baseline setting.

Switch Scheduling Algorithm and Load Distribution: In the baseline setting of Section III-C, we assumed that the switch was scheduled by the MWM algorithm and that the load distribution was uniform.

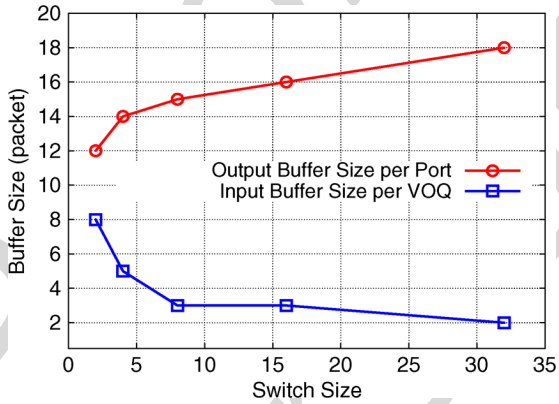


Fig. 15. Minimum required buffer size for 80% utilization versus switch size. The switch has a uniform load distribution, which results in more contention and short-term congestion as the number of ports increases.

Here, we relax these two assumptions and compare the results of the baseline setting to those obtained under the iSLIP scheduling algorithm [24] and nonuniform traffic.

The widely implemented iSLIP scheduling algorithm achieves 100% throughput for uniform traffic. This iterative round-robin-based algorithm is simple to implement in hardware, but the throughput is less than 100% in the presence of nonuniform bursty traffic.

Among various possible nonuniform distributions of load, we choose the *diagonal* load distribution. With a diagonal distribution, 2/3 of the total traffic at a given input port i goes to output port $i+1$, and the remaining 1/3 goes to output $i+2$. Compared to the uniform traffic, this type of traffic is more difficult to schedule because arrivals favor the use of only two matchings out of all possible matchings, and the average backlog in input buffers is larger [23].

Fig. 14 shows the output link utilization versus input buffer size per VOQ. With iSLIP and speedup of 1 (left), there is no queueing at the output side of the switch. When the speedup is 1.2 (right), the switch fabric runs 1.2 times faster than the line rate, which may cause backlog in output buffers. In this case, we have set the output buffer size to only 20 packets per port. That is why, with uniform traffic and large input buffers, increasing the speedup from 1.0 to 1.2 causes some throughput loss.

The results show that with speedup of 1.2 (for all combinations of scheduling algorithm and load distribution) setting the

buffer size to five packets per VOQ and 20 packets per output port raises the utilization to more than 80%. Larger speedups make the impact of the scheduling algorithm even smaller because the switch behaves more like an output-queued switch.

Switch Size: The output link utilization of a switch depends on its size (number of ports). Increasing the number of ports creates more contention among the input ports and adds to the short-term congestion (caused by the statistical arrival time of packets from different input ports) on output links.

Fig. 15 shows the minimum required buffer size for 80% utilization on output links. The simulation setting follows the baseline, except that we vary the switch size from 2 to 32. For all switch sizes, the offered load on the output links is 100%.

In this set of simulations, the switch has a uniform load distribution. If the traffic at a given output port comes from a limited number of input ports, then we do not expect to see any changes when the switch size is varied. With diagonal load distribution, for example, where the traffic on each output link i comes only from ports $i-2$ and $i-1$, the required buffer size for 80% utilization remains constant when we change the number of ports.

We assume that the input ports maintain a separate VOQ for each output port. Therefore, despite the decrease in the VOQ size in Fig. 15, the total buffer size per input port (i.e., the size of the VOQ times the number of output ports) increases.

REFERENCES

[1] D. J. Blumenthal, B. E. Olsson, G. Rossi, T. E. Dimmick, L. Rau, M. Masanovic, O. Lavrova, R. Doshi, O. Jerphagnon, J. E. Bowers, V. Kaman, L. A. Coldren, and J. Barton, "All-optical label swapping networks and technologies," *J. Lightw. Technol.*, vol. 18, no. 12, pp. 2058–2075, Dec. 2000.

[2] A. Carena, M. D. Vaughn, R. Gaudino, M. Shell, and D. J. Blumenthal, "OPERA: An optical packet experimental routing architecture with label swapping capability," *J. Lightw. Technol.*, vol. 16, no. 12, pp. 2135–2145, Dec. 1998.

[3] A. Viswanathan, N. Feldman, Z. Wang, and R. Callon, "Evolution of multiprotocol label switching," *IEEE Commun. Mag.*, vol. 36, no. 5, pp. 165–173, May 1998.

[4] P. Öhlén, B. E. Olsson, and D. J. Blumenthal, "All-optical header erasure and penalty-free rewriting in a fiber-based high-speed wavelength converter," *IEEE Photon. Technol. Lett.*, vol. 12, no. 6, pp. 663–665, Jun. 2000.

[5] E. Olsson, P. Ohlen, L. Rau, G. Rossi, O. Jerphagnon, R. Doshi, D. S. Humphries, D. J. Blumenthal, V. Kaman, and J. E. Bowers, "Wavelength routing of 40 Gbit/s packets with 2.5 Gbit/s header erasure/rewriting using an all-fiber wavelength converter," *Electron. Lett.*, vol. 36, pp. 345–347, 2000.

[6] I. Keslassy, S.-T. Chang, K. Yu, D. Miller, M. Horowitz, O. Solgaard, and N. McKeown, "Scaling Internet routers using optics," in *Proc. ACM SIGCOMM*, Karlsruhe, Germany, Aug. 2003, pp. 189–200.

[7] M. Karol, M. Hluchyj, and S. Morgan, "Input versus output queueing on a space division switch," *IEEE Trans. Commun.*, vol. COMM-35, no. 12, pp. 1347–1356, Dec. 1984.

[8] N. McKeown, V. Anantharan, and J. Walrand, "Achieving 100% throughput in an input-queued switch," in *Proc. IEEE INFOCOM*, Mar. 1996, vol. 1, pp. 296–302.

[9] A. Mekkittikul and N. McKeown, "A practical scheduling algorithm to achieve 100% throughput in input-queued switches," in *Proc. IEEE INFOCOM*, Apr. 1998, vol. 2, pp. 792–799.

[10] J. Dai and B. Prabhakar, "The throughput of data switches with and without speedup," in *Proc. IEEE INFOCOM*, Mar. 2000, vol. 2, pp. 556–564.

[11] G. Appenzeller, I. Keslassy, and N. McKeown, "Sizing router buffers," in *Proc. ACM SIGCOMM*, New York, 2004, pp. 281–292.

[12] "The Network Simulator—ns-2," [Online]. Available: http://www.isi.edu/nsnam/ns/

[13] S. T. Chuang, A. Goel, N. McKeown, and B. Prabhakar, "Matching output queueing with a combined input output queued switch," in *Proc. IEEE INFOCOM*, 1999, pp. 1169–1178.

[14] M. Enachescu, Y. Ganjali, A. Goel, N. McKeown, and T. Roughgarden, "Routers with very small buffers," in *Proc. IEEE INFOCOM*, Barcelona, Spain, April 2006.

[15] N. Beheshti, Y. Ganjali, M. Ghobadi, N. McKeown, and G. Salmon, "Experimental study of router buffer sizing," in *Proc. IMC*, Vouliagmeni, Greece, Oct. 2008, pp. 197–210.

[16] R. Prasad, M. Thottan, and C. Dovrolis, "Router buffer sizing revisited: The role of the input/output capacity ratio," in *Proc. ACM CoNext Conf.*, New York, Dec. 2007, Article no. 15.

[17] A. Aggarwal, S. Savage, and T. Anderson, "Understanding the performance of TCP pacing," in *Proc. IEEE INFOCOM*, Tel-Aviv, Israel, Mar. 2000, pp. 1157–1165.

[18] L. Qiu, Y. Zhang, and S. Keshav, "Understanding the performance of many TCP flows," *Comput. Netw.*, vol. 37, no. 3–4, pp. 277–306, 2001.

[19] G. Iannaccone, M. May, and C. Diot, "Aggregate traffic performance with active queue management and drop from tail," *SIGCOMM Comput. Rev.*, vol. 31, no. 3, pp. 4–13, 2001.

[20] C. J. Fraleigh, "Provisioning Internet backbone networks to support latency sensitive applications," Ph.D. dissertation, Department of Electrical Engineering, Stanford University, Stanford, CA, Jun. 2002.

[21] N. Beheshti and N. McKeown, "Routers with tiny buffers: Simulations," Stanford University, Tech. Rep., June. 2008.

[22] N. Beheshti, Y. Ganjali, A. Goel, and N. McKewon, "Obtaining high throughput in networks with tiny buffers," in *Proc. 16th IWQoS*, Enschede, The Netherlands, Jun. 2008, pp. 65–69.

[23] D. Shah, P. Giaccone, and B. Prabhakar, "Efficient randomized algorithms for input-queued switch scheduling," *IEEE Micro*, vol. 22, no. 1, pp. 10–18, Jan.–Feb. 2002.

[24] N. McKeown, "iSLIP: A scheduling algorithm for input-queued switches," *IEEE/ACM Trans. Netw.*, vol. 7, no. 2, pp. 188–201, Apr. 1999.

[25] E. F. Burmeister, J. P. Mack, H. N. Poulsen, M. L. Mašanović, B. Stamenić, D. J. Blumenthal, and J. E. Bowers, "Integrated optical buffer for packet-switched networks," *J. Lightw. Technol.*, 2010, submitted for publication.

[26] E. F. Burmeister, D. J. Blumenthal, and J. E. Bowers, "A comparison of optical buffering technologies," *Opt. Switch. Netw.*, vol. 5, pp. 10–18, Mar. 2008.

[27] E. F. Burmeister and J. E. Bowers, "Integrated gate matrix switch for optical packet buffering," *IEEE Photon. Technol. Lett.*, vol. 18, no. 1, pp. 103–105, Jan. 2006.

[28] J. P. Mack, H. N. Poulsen, E. F. Burmeister, J. E. Bowers, and D. J. Blumenthal, "A 40 Gbps asynchronous optical packet buffer based on an SOA gate matrix for contention resolution," presented at the Opt. Fiber Commun. Conf. 2006, Anaheim, CA, OTuB7.

[29] H. Park, J. P. Mack, D. J. Blumenthal, and J. E. Bowers, "An integrated recirculating buffer," *Opt. Exp.*, vol. 16, no. 15, pp. 11124–11131, Jul. 2008.

Neda Beheshti (S'00) received the B.S. degree from Sharif University of Technology, Tehran, Iran, in 2000; the M.S. degree from Northeastern University, Boston, MA, in 2002; and the Ph.D. degree from Stanford University, Stanford, CA, in 2009, all in electrical engineering.

She joined Ericsson Research Lab, San Jose, CA, in 2009 as a Research Engineer. Her research interests include router and switch architectures, wireless networking, and the architecture of the future Internet.

Dr. Beheshti received the Best Paper Award at the Internet Measurement Conference (IMC) 2008 and the Second Best Demo Award at SIGCOMM 2008 for her work on router buffer sizing.

Emily Burmeister (M'08) received the B.S. degree in engineering physics from the University of Michigan, Ann Arbor, in May 2002, and the Ph.D. degree in electrical engineering from the University of California, Santa Barbara, in May 2008. Her thesis title was "Integrated Optical Buffers for Packet-Switched Networks."

She is currently a Senior Engineer with Ciena in Linthicum, MD.

Yashar Ganjali (S'03–M'07) received the B.Sc. degree in computer engineering from Sharif University of Technology, Tehran, Iran, in 1999; the M.Sc. degree in computer science from the University of Waterloo, Waterloo, ON, Canada, in 2001; and the Ph.D. degree in electrical engineering from Stanford University, Stanford, CA, in 2006.

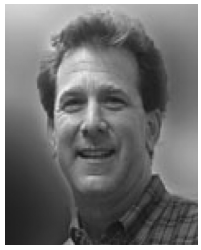
He is a faculty member with the Computer Science Department, University of Toronto, Toronto, ON, Canada. His current research interests include packet switching architectures/algorithms, network protocols and measurement, network management, and online social networks.

Dr. Ganjali has received several awards for his research, including the Best Paper Award at the Internet Measurement Conference 2008, Best Paper Runner-Up at the IEEE INFOCOM 2003, Best Demo Runner-Up at SIGCOMM 2008, Best Demo at the NetFPGA Workshop 2009, the Leaders Opportunity Fund from Canada Foundation for Innovation, and the Cisco Research Award.

John E. Bowers (F'93) received the M.S. and Ph.D. degrees in applied physics from Stanford University, Stanford, CA, in 1978 and 1981, respectively.

He holds the Fred Kavli Chair in Nanotechnology and is the Director of the Institute for Energy Efficiency and a Professor with the Department of Electrical and Computer Engineering, University of California, Santa Barbara (UCSB). He worked for AT&T Bell Laboratories, Holmdel, NJ, and Honeywell, Minneapolis, MN, before joining UCSB. His research interests are in silicon photonic integrated circuits for the next generation of coherent optical systems.

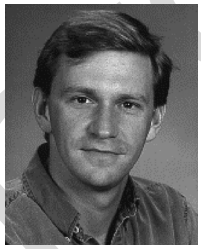
Prof. Bowers is a Member of the National Academy of Engineering, a Fellow of the OSA and the American Physical Society, and a recipient of the OSA Holonyak Prize, the IEEE LEOS William Streifer Award, and the South Coast Business and Technology Entrepreneur of the Year Award. He and his coworkers received the ACE Award for Most Promising Technology for the hybrid silicon laser in 2007.



Daniel J. Blumenthal (S'91–M'93–SM'97–F'03) received the B.S.E.E. degree from the University of Rochester, Rochester, NY, in 1981; the M.S.E.E. degree from Columbia University, New York, NY, in 1988; and the Ph.D. degree from the University of Colorado, Boulder, in 1993.

He is currently a Professor with the Department of Electrical and Computer Engineering, University of California, Santa Barbara (UCSB). He is Director of the LASOR Center at UCSB, a project funded by the DARPA/MTO Data in the Optical Domain Network (DOD-N) program. He currently serves on the Board of Directors for National LambdaRail (NLR) and serves on the Internet2 Architecture Advisory Council. His research interests are in optical communications, photonic packet switching and all-optical networks, all-optical wavelength conversion and regeneration, ultra-fast communications, InP photonic integrated circuits (PICS), and nanophotonic device technologies.

Dr. Blumenthal is a Fellow of the IEEE Photonics and Communications societies and the Optical Society of America (OSA). He is recipient of a 1999 Presidential Early Career Award for Scientists and Engineers (PECASE) from the White House, a 1994 National Science Foundation Young Investigator (NYI) Award, and a 1997 Office of Naval Research Young Investigator Program (YIP) Award.



Nick McKeown (F'05) received the B.E. degree from the University of Leeds, Leeds, U.K., in 1986, and the M.S. and Ph.D. degrees from the University of California, Berkeley, in 1992 and 1995, respectively, all in electrical engineering and computer science.

He is a Professor of Electrical Engineering and Computer Science and Faculty Director of the Clean Slate Program at Stanford University, Stanford, CA. From 1986 to 1989, he worked for Hewlett-Packard Labs, Bristol, England. In 1995, he helped architect Cisco's GSR 12000 router. In 1997, he co-founded Abrizio Inc. (acquired by PMC-Sierra) in Mountain View, CA, where he was CTO. He was co-founder and CEO of Nemo ("Network Memory") in Los Altos, CA, which is now part of Cisco. His research interests include the architecture of the future Internet and tools and platforms for networking teaching and research.

Prof. McKeown is a Fellow of the Royal Academy of Engineering (U.K.) and the Association for Computing Machinery (ACM). He is the STMicroelectronics Faculty Scholar, the Robert Noyce Faculty Fellow, a Fellow of the Powell Foundation and the Alfred P. Sloan Foundation, and a recipient of a CAREER Award from the National Science Foundation. In 2000, he received the IEEE Rice Award for the best paper in communications theory. He was awarded the British Computer Society Lovelace Medal in 2005, and the IEEE Kobayashi Computer and Communications Award in 2009.