Comparison of Packet Switch Architectures and Pacing Algorithms for Very Small Optical RAM

Onur Alparslan, Shin'ichi Arakawa, Masayuki Murata
Graduate School of Information Science and Technology
Osaka University
1-5 Yamadaoka, Suita, Osaka 565-0871, Japan
$\{$*a-onur,arakawa,murata*$\}$*@ist.osaka-u.ac.jp*

***Abstract*—One of the difficulties with optical packet switched (OPS) networks is buffering optical packets in the network. The research on optical RAM presently being done is not expected to achieve a large capacity soon. However, the burstiness of Internet traffic causes high packet drop rates and low utilization in small buffered OPS networks. In this article, we investigate and compare optical-buffered switch architectures and pacing algorithms for minimizing the buffer requirements of OPS switches. We simulate two mesh topologies (NSFNET and Abilene) for goodput and packet drop rate comparisons and optimization of XCP parameters. We show that XCP-based pacing algorithm with a shared buffered switch architecture yields high TCP goodput and low packet drop rate in a core OPS network when very small optical RAM buffers are used.**

***Keywords*-small buffer, OPS, optical RAM, optical switch**

I. Introduction

A well-known problem in realizing optical packet switched (OPS) networks is buffering. Recent advances in optical networks such as dense wavelength division multiplexing (DWDM) have allowed us to achieve ultra-high data-transmission rates in optical networks. This ultra-high speed of optical networks has made it necessary to do some basic operations like buffering and switching in the optical domain instead of the electronic domain due to high costs and limitations with electronic buffers. However, the lack of high-capacity optical RAM makes it difficult to buffer enough optical packets in OPS networks [1]. According to a rule-of-thumb [2], the buffer size of a link must be $B = RTT \times BW$, where RTT is the average round trip time of flows and BW is the bandwidth of the output link, to achieve high utilization with TCP flows. However, as this requires a huge buffer size in optical routers due to the ultra-high speed of optical links, this buffer size is unfeasible.

The only available solution that can currently be used for buffering in the optical domain is using fiber delay lines (FDLs), where contended packets are switched to long FDLs so that they can be delayed. However, FDLs pose severe limitations such as signal attenuation, and bulkiness. Most FDL architectures lack the real $O(1)$ reading operation of RAM as it may not be possible to access a packet circulating an FDL until the packet departs the fiber and arrives back to the switch, which causes extra delays depending on the FDL length. Moreover, all-optical RAMs, which can solve the problems with FDLs, are still being researched (e.g., NICT project [3]) and this may become available in the near future. Furthermore, optical RAMs are expected to have a lower rates of power consumption, which is a serious problem with electronic RAMs. However, optical RAMs are not expected to attain large capacities. Therefore, it is necessary to decrease the buffer requirements of OPS networks to make use of optical RAMs.

Appenzeller et al. [4] recently demonstrated that when there are many TCP flows sharing the same link, a buffer sized at $B = \frac{RTT \times BW}{\sqrt{n}}$, where n is the number of TCP flows passing through the link, is sufficient to achieve high utilization. However, there should be many flows on a link to significantly decrease the buffer requirements of ultra-high-speed optical networks. Enachescu et al. [5] proposed that $O(\log W)$ buffers are sufficient where W is the maximum congestion window size of flows when packets are sufficiently paced by replacing TCP senders with paced TCP [6] or by using slow access links. TCP pacing is defined as transmitting ACK (data) packets according to special criteria, instead of immediately transmitting packets when data (ACK) packets arrive [6]. Paced TCP is usually implemented by evenly spreading out the transmission of a window of data packets over a round-trip time. However, the $O(\log W)$ buffer size depends on the maximum congestion window size of TCP flows, which may change. Moreover, using slow access links, which is an extreme way of applying node pacing, is not ideal when there are applications that require large amounts of bandwidth on the network. Furthermore, replacing TCP senders of computers with paced versions can be difficult. Furthermore, this proposal was based on the assumption that most IP traffic is from TCP flows. Theagarajan et al. [7] demonstrated that even small quantities of bursty real-time UDP traffic can increase the buffer requirements of well-behaved TCP traffic on the same link. Therefore, it may be better to design a general architecture for an OPS network that can achieve high utilization in a small buffered OPS network independent of the number of TCP or UDP flows, and that does not require a strict limit to be placed on the speed of access links, and that does not require the sender or receiver TCP and UDP agents of computers using the network to be replaced.

We recently proposed [8] an all-optical OPS network architecture that can achieve high utilization and a low packet drop ratio by using small buffering. We took into consideration an OPS domain where packets entered and exited the OPS domain through edge nodes. We proposed using an Explicit Congestion Control Protocol (XCP)-based [9] intra-domain congestion control protocol to achieve high utilization and a low packet-drop ratio with small buffers. XCP is a new congestion control algorithm using a control-theory framework, which was specifically designed for high-bandwidth and large-delay networks. XCP was first proposed by Katabi et al. [9] as a window-based algorithm for reliably controlling congestion and transmission. We selected the XCP framework because it allows the utilization level of each wavelength to be individually controlled. Moreover, there is no need to modify the TCP and UDP agents of computers or limit the speed of access links to decrease burstiness.

Another difficulty in attaining OPS networks is the switching fabric, which is usually one of the biggest factors determining overall router cost. Many switching fabric architectures like MEMS, optomechanical, electrooptic, thermooptic, and liquid-crystal based switches have been proposed for optical switching [10]. However, the number of switching elements in the fabric increases together with the overall cost, and crosstalk and insertion losses as the number of ports for the switch increases. In our previous papers, we evaluated the performance of our proposed architecture with UDP and TCP-based traffic and output buffering [8][11][12]. In Alparslan et al. [1], we proposed and investigated different optical-buffered switch architectures to further minimize the size of the optical switching fabric of core nodes while achieving higher goodput with small optical RAM buffers. We evaluated the optical RAM requirements of our proposed architecture on a mesh NSFNET topology with TCP traffic. We also compared the performance of our architecture with paced TCP, which is the solution generally proposed for small buffered networks. Our simulations revealed that the average goodput of standard TCP flows in our proposed architecture could even surpass the goodput of paced TCP when buffers were small.

This article discusses the extension of our work and presents our results in Alparslan et al. [1] verified by simulating an Abilene network, which is a larger topology with a higher nodal degree than NSFNET. We optimized XCP parameters on both NSFNET and Abilene topologies to demonstrate what effect XCP parameters have on overall performance. Moreover, we introduce one more metric, which is the overall packet-drop rate in a small buffered core network, that enables better comparison of switch architectures and algorithms.

The rest of the paper is organized as follows. Section II describes the XCP algorithm and switch architectures. Section III describes the simulation methodology and presents the simulation results. Finally, we conclude the paper and describe future work that we intend to do in Section IV.

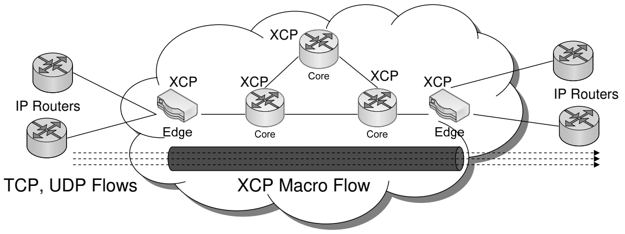


Figure 1. XCP pacing

II. Architecture

This section describes the XCP algorithm and switch architectures.

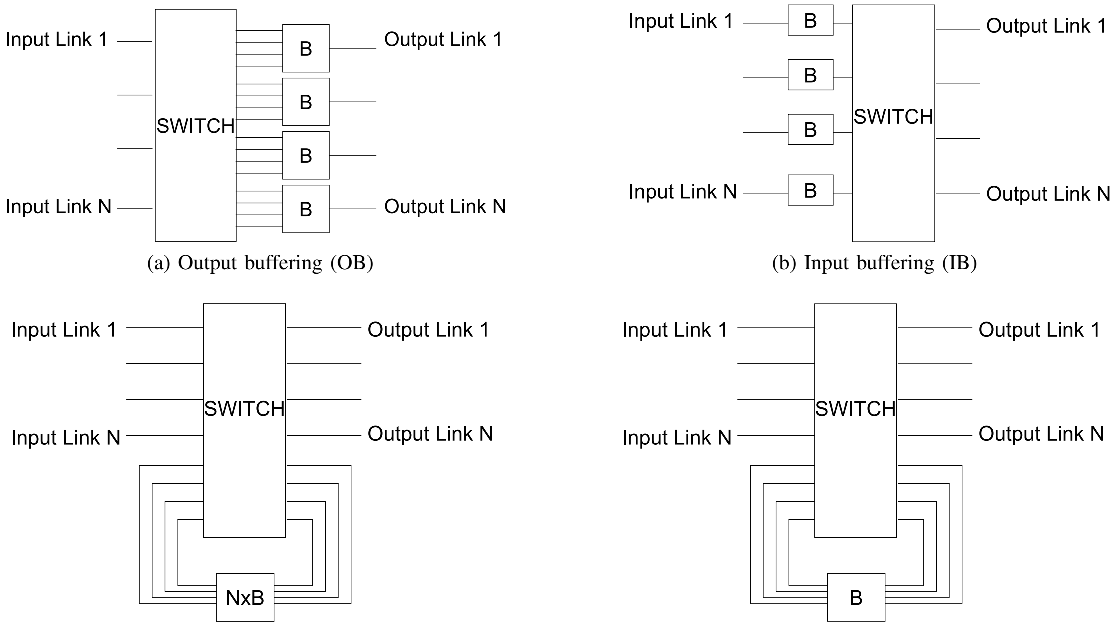
A. Optical Rate-based Paced XCP

XCP is a new congestion control algorithm that has been specifically designed for high-bandwidth and large-delay networks. XCP makes use of explicit feedback received from the network. Core routers are not required to maintain per-flow state information. Each XCP core router updates its control decisions calculated with an Efficiency Controller (EC) and a Fairness Controller (FC) when timeout of a per-link control-decision timer occurs.

EC controls input aggregate traffic to maximize link utilization. A required increase or decrease in aggregate traffic for each output port is calculated by using the equation, $\Phi = \alpha \cdot S - \beta \cdot Q/d$, where Φ is the total amount of required change in input traffic, α and β correspond to spare bandwidth-control and queue-control parameters, and d is the control-decision interval. S is the spare bandwidth that is the difference between the link capacity and input traffic in the last control interval. Q is the persistent queue size.

After EC has calculated the aggregate feedback Φ, FC fairly distributes this feedback to flows according to AIMD-based control. However, convergence to fairness may take a long time when Φ is small. Bandwidth shuffling, which redistributes a small amount of traffic among flows, is used to solve this problem. The amount of shuffled traffic is calculated by $h = max(0, \gamma \cdot u - |\Phi|)$, where γ is the shuffling parameter and u is the rate of aggregate input traffic in the last control interval.

In Alparslan et al. [8], we proposed optical rate-based paced XCP, which is a modified version of XCP adapted to work as an intra-domain traffic shaping and congestion control protocol in an OPS network domain. In our architecture, when there is traffic between an edge-source destination node pair, a rate-based XCP macroflow is created as shown in Figure 1, and the incoming TCP and UDP packets of



(a) Output buffering (OB)

(b) Input buffering (IB)

(c) Shared buffering (SB)

(d) Worst-case shared buffering (WCSB)

Figure 2. Switch architectures

this edge pair are assigned an XCP macroflow similar to TeXCP [13]. The edge nodes of the OPS network apply leaky-bucket pacing to the macroflows by using the rate information provided by XCP to minimize burstiness.

In our optical rate-based paced XCP, XCP feedback is carried in separate probe packets that XCP sender agents only send once in every control period. As there is no feedback information carried in the header of data packets, there is no need to calculate per-packet feedback in core routers unlike in the original XCP [9]. We separated the control channel and data channels. Probe packets are carried on a separate single control wavelength that is sufficiently slow to only carry probe packets. The low transmission rate to control wavelength allows electronic conversion to be applied to update the probe feedback and buffer the probe packets in an electronic RAM in case of a contention.

When a probe packet of macroflow i arrives at a core router, the XCP agent responsible for controlling the wavelength of i calculates positive feedback p_i and negative feedback n_i for macroflow i. Positive feedback is calculated as

$$p_i = \frac{h + max(0, \Phi)}{N} \tag{1}$$

and negative feedback is calculated as

$$n_i = \frac{u_i \cdot (h + max(0, -\Phi))}{u}, \tag{2}$$

where N is the number of macroflows on this wavelength, u_i is the traffic rate of flow i estimated and sent by the XCP sender in the probe packet, and h is the shuffled bandwidth. N can be estimated by counting the number of probe packets received during the last control interval. Another possible method is using the number of LSPs if GMPLS is available [13]. The control interval is the maximum RTT in the network. The control interval can be selected to be a bit longer than the maximum RTT to compensate for the processing and buffering delays in control packets. Feedback, which is the required change in the flow rate, is calculated as $feedback = p_i - n_i$. If this feedback is smaller than that in the probe packet, the core router replaces the feedback in the probe packet with its own feedback. Otherwise, the core router does not change the feedback in the probe packet.

B. Switch Architectures

In this paper, we compare the performance of output buffering (OB), input buffering (IB), shared buffering (SB) and worst-case shared buffering (WCSB), as shown in Figure 2. The internal speedup is 1 in all switches, which means the line rates are equal in both inside and outside the switch. Switch size is shown as $I \times O$, where I and O are the number of input and output ports, respectively. Output buffering has a large switch size of $N \times N^2$ to prevent internal blocking where N is the nodal degree as seen in Figure 2a. It has buffer size B at each output link. As input buffering has a switch size of only $N \times N$, as seen in Figure 2b, it has the smallest switches. However, a well-known problem with input buffering is head-of-line blocking, which limits the utilization that can be accomplished. We applied virtual output queuing (VOQ) scheduling, which is the

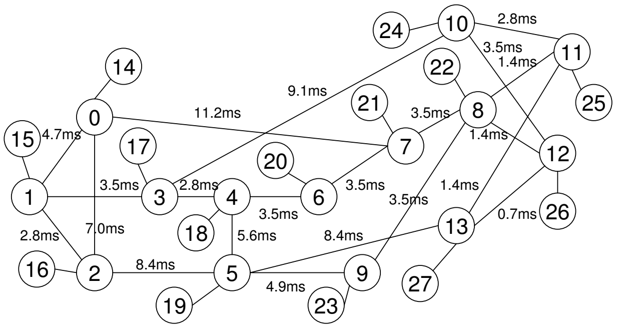


Figure 3. NSFNET

generally proposed solution in the literature, by dividing each input buffer into N sub-queues to minimize the head-of-line blocking problem [14]. Input buffering has buffer size B at each input link. Shared buffering and worst-case shared buffering have a switch size of $2N \times 2N$. As shared buffering has a single buffer with a size of $N \cdot B$, this is in direct proportion to its capacity and the nodal degree. Shared buffering has the same total buffer size per node as input and output buffering to enable fairer comparisons with them at the same B value in the simulations. As worst-case shared buffering has a single buffer with a size of B independent of the nodal degree, it has a buffer capacity of only a single link in input or output buffering.

III. Evaluation

This section describes the simulation methodology and presents the simulation results. We investigated and compared optical-buffered switch architectures and pacing algorithms for minimizing the buffer requirements of OPS switches. We first simulated a wide range of XCP parameters to find the optimum XCP parameters for two mesh topologies. In the second step, we simulated both topologies with four different switch architectures to compare their performance under standard TCP traffic, XCP-paced standard TCP traffic, and paced TCP traffic.

A. Simulation Settings

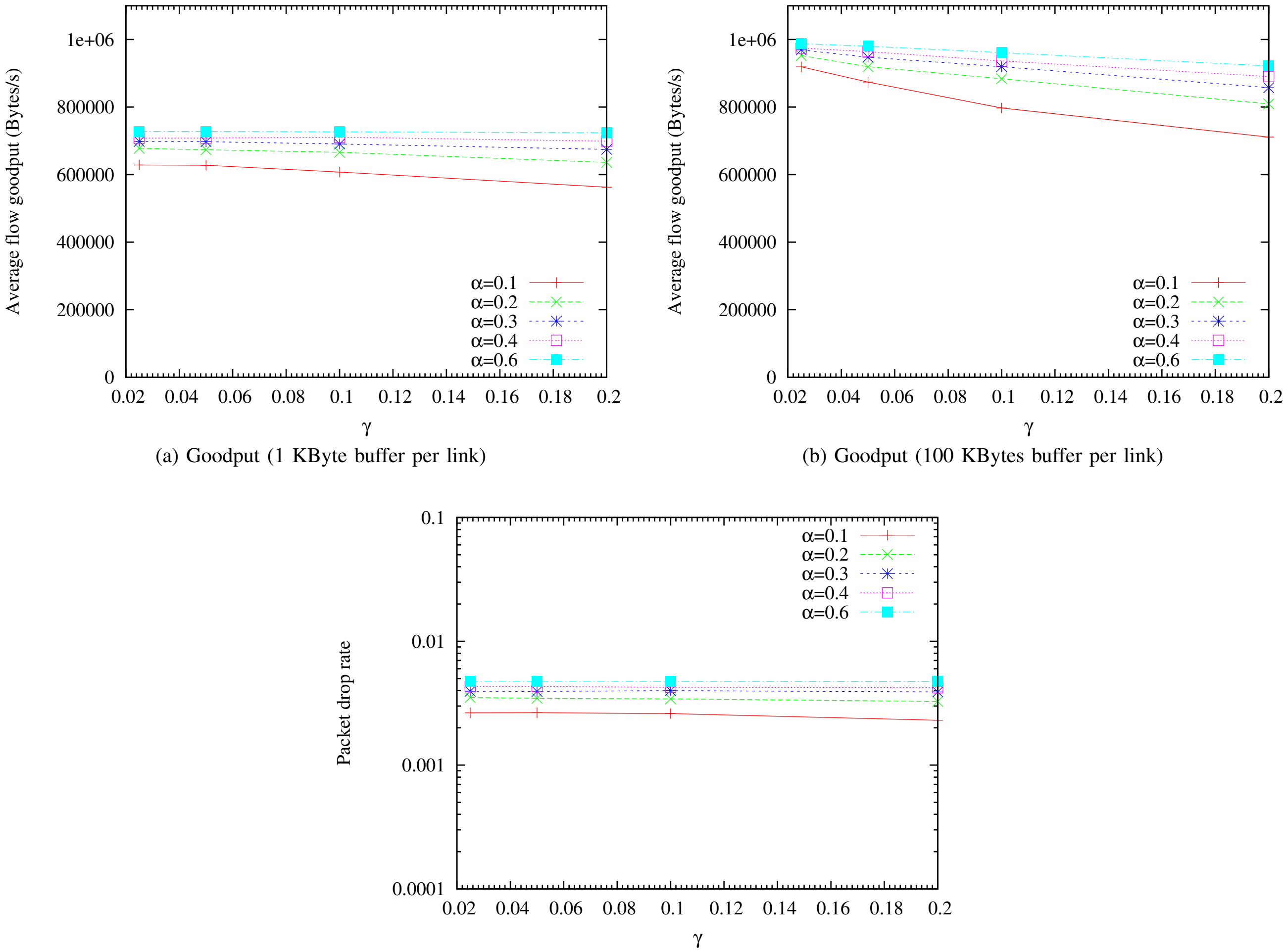
The proposed network architecture and buffering models were implemented over the *ns* simulator version 2.32 [15]. The simulator used cut-through packet switching and buffering for data wavelengths. There was a single store-and-forward switching slow control wavelength dedicated to probe packets. The edge nodes had 0.25 s of electronic buffering, which is commonly used on the Internet. However, core routers only used small optical RAM for buffering optical packets on data wavelengths. The contention of probe packets on the control-wavelength was resolved by an electronic RAM as O/E/O conversion was not a problem in the control-wavelength due to its low speed. As the IP datagram for TCP data (ACK) packets was 1500 Bytes (40 Bytes), the TCP data (ACK) segment was 1480 Bytes (20 Bytes). Maximum segment size (MSS) was 1500 Bytes. The size limit of the congestion window was 20 packets. The target utilization (TU) parameter of XCP was set to 100% at both core and edge links to maximize link utilization.

B. NSFNET Results

Figure 3 plots the simulated NSFNET topology. The nodes numbered from 0 to 13 are the core nodes and the rest are the edge nodes connected to the core nodes. All links (including edge and core links) had a single data wavelength and the same XCP target utilization. We selected a propagation delay for links between the core and edge nodes of 0.1 ms. We selected an XCP control period for core routers and the sending interval of probe packets for edge routers of 50 ms by taking extra processing and queuing delays in the core routers into account. The capacity of the data and XCP control wavelengths were set to 1 Gbps for the former and 100 Mbps for the latter. TCP Reno traffic was applied between the edge nodes of the network. Throughout the simulation, 1586 TCP flows entered the network between randomly selected edge-node pairs according to a Poisson arrival process. The total simulation duration was 40 s. Only the simulation results in the last 5 s were used for the evaluation.

1) Optimization of XCP Parameters: XCP's α, β and γ parameters were chosen as 0.2, 0.056, and 0.1 in Alparslan et al. [1]. In this paper, we first optimized XCP parameters to demonstrate what effect XCP parameters had and to increase overall performance. Figure 4 plots the average goodput of TCP flows under XCP pacing and packet drop rate at the core nodes when shared buffering is used. The x-axis is the γ parameter and each line plots a different α value. Figures 4a and 4c use a small shared buffer with a size of 1 KByte per link, while Figure 4b uses a larger shared buffer with a size of 100 KBytes per link. As no packets were dropped in the core when the shared buffer size per link was 100 KBytes, this has not been plotted. Figures 4a and 4c indicate that when the buffer is small, the average flow goodput and core packet drop rate increase with increasing α values, while they are relatively less sensitive to the γ parameter. Therefore, there is a tradeoff between the drop rate and goodput as we change α. This is an expected result, because XCP behaves more aggressively and utilizes links faster with an increasing α value, which causes more frequent buffer overflows and thus packet drops as a side effect in the core routers. However, XCP may become unstable when α is too large. Katabi et al. [9] proved that the system is stable independent of delay, capacity, or the number of sources when α and β satisfy

$$0 < \alpha < \frac{\pi}{4\sqrt{2}} \tag{3}$$



(a) Goodput (1 KByte buffer per link)

(b) Goodput (100 KBytes buffer per link)

(c) Packet drop rate in the core nodes (1 KByte buffer per link)

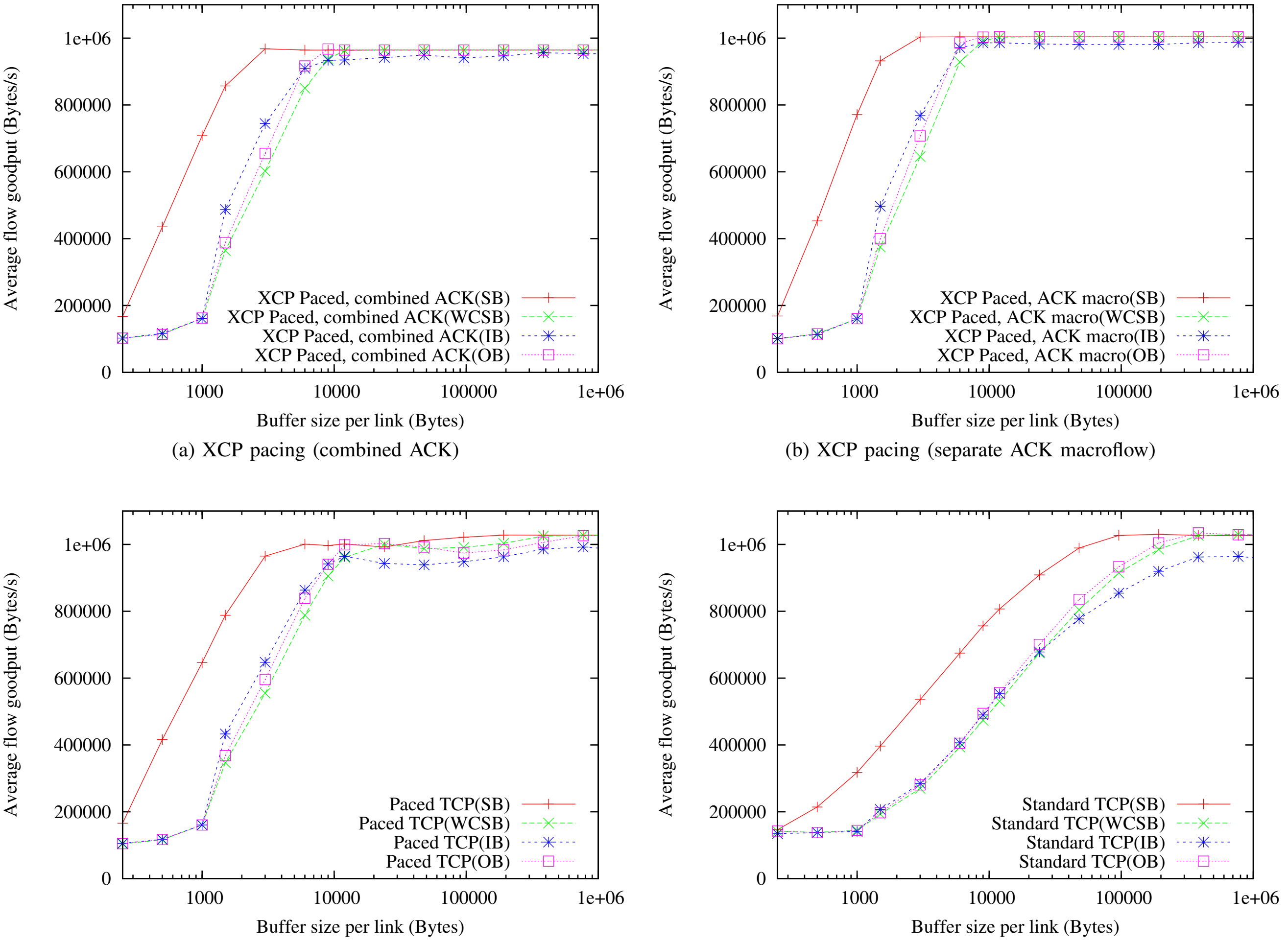
Figure 4. Optimization of parameters in NSFNET

and

$$\beta = \alpha^2\sqrt{2}. \tag{4}$$

Figure 4b shows that when the buffer is larger, goodput becomes more sensitive to the γ parameter. As γ decreases or α increases, the average goodput increases. The reason for this is that XCP encounters a well-known max-min fairness problem under various special conditions. XCP's mechanism to control congestion in a multi-bottleneck environment can cause a flow to receive an arbitrarily small fraction of its max-min allocation, which may cause some bottleneck links to be under-utilized [16]. XCP may end up utilizing only 80% of the bottlenecked bandwidth in the worst case with the default XCP parameters in Katabi et al. [9]. As they give a high goodput in Figure 4, we chose α=0.4 and β=0.226, which are the default values suggested in Katabi et al. [9]. However, we chose γ=0.1, which is half the default value in Katabi et al. [9]. Although choosing a lower γ than the default value decreases the speed of fairness convergence, XCP achieves better max-min fairness and higher worst-case link utilization with higher average goodput as seen in Figure 4b.

2) Comparison of Switches and Algorithms: After XCP parameters were optimized, we simulated the NSFNET topology by using the four different switch architectures shown in Figure 2. The switch architecture was a parameter in the simulations, which was applied to all nodes in the network. We compared the performance of the switch architectures under standard TCP, paced TCP, and XCP-paced standard TCP traffic. Figure 5 plots the average goodput of TCP flows on different switch and network architectures based on the optical RAM buffer size per link. In all the figures, the x-axis is the buffer size per link, which is designated as B in Figure 5 on a log scale and the y-axis is the average TCP goodput on a linear scale. Figure 5a plots the TCP goodput when our XCP pacing algorithm was applied to standard TCP Reno traffic. We



(a) XCP pacing (combined ACK)

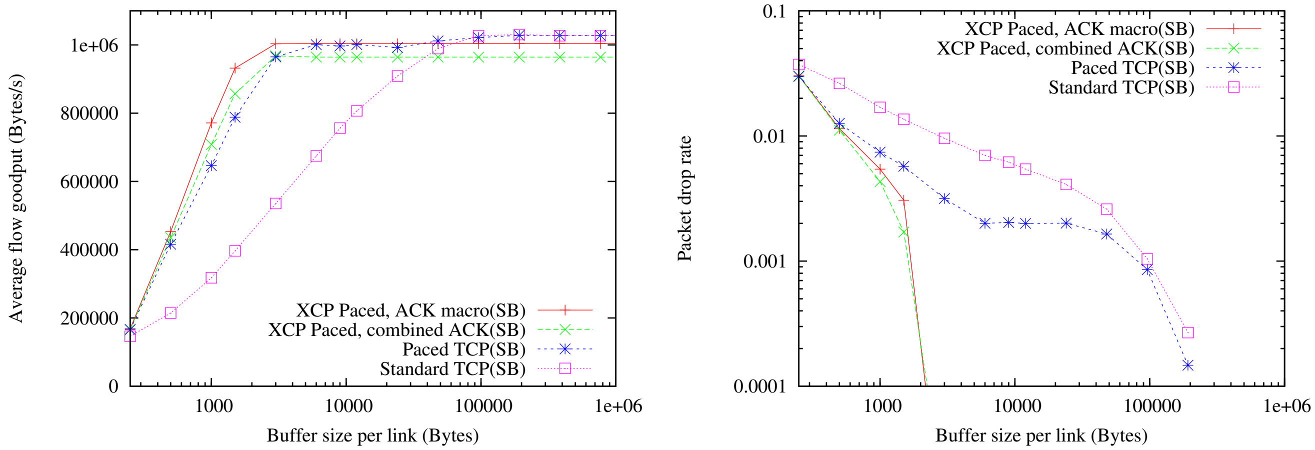
(b) XCP pacing (separate ACK macroflow)

(c) TCP pacing

(d) Standard TCP

Figure 5. Goodput comparison of algorithms and switch architectures in NSFNET

can see that input, output, and worst-case shared buffered switches yield almost the same goodput when the buffer size is less than a 1 MSS or more than around 6 MSS. However, shared buffering yields a much higher goodput than the others even though its per node buffer capacity is the same as that for input and output buffering. If we can use a single shared buffer instead of splitting it into input or output links, it clearly yields much higher goodput as a small buffer capacity is being used with maximum efficiency. When we compare input and output buffering, we can see that when the link buffer is between 1–6 MSS, input buffering yields higher goodput while using the smallest switch in switch architectures. This result was expected, because input buffering can handle packet contentions better than output buffering when the input traffic is sufficiently smooth. For example, let us assume that we have a switch with only single packet capacity output buffers. When there is contention with five packets arriving from five input links that are going to the same output link, if the buffers and links are idle, one packet will be sent to the output link, one packet will be buffered in the output buffer, and the remaining three packets will be dropped as there is no more buffer left. However, if we use an input buffered switch, one packet will be sent to the output link and the other four packets will be buffered at the input ports. As buffered packets can be sent to the output link as the link becomes idle, its tendency to drop packets when there is a contention is lower than that for output buffering. Input buffering greatly benefits from pacing as it smooths the packet arrival from its link, which gives it time to drain its queue. When we check the goodput of worst-case buffering, we can see that it is very close to output buffering even though the whole switch in worst-case buffering has the same buffer capacity of only a single link in output buffering. In other words, worst-case shared buffering yields almost the same goodput as output buffering with a much smaller buffer capacity per node.



(a) Goodput

(b) Core packet drop rate in the core nodes

Figure 6. Comparison of algorithms in NSFNET with shared buffering

The packet level tracing of simulations with our XCP pacing algorithm revealed that there was actually still room for improvement. We saw that the well-known ACK compression problem [17] had caused some utilization inefficiencies, which decreased the average goodput. It is possible in our architecture to solve this problem and increase utilization by simply using separate XCP macroflows for TCP ACK packets [12]. Figure 5b plots the TCP goodput when our optical rate-based paced XCP architecture was used with separate XCP macroflows for TCP ACK packets on the same wavelength. We can see that TCP goodput becomes higher than XCP with the combined ACK architecture in Figure 5a.

Figure 5c plots the TCP goodput when paced TCP Reno is used without XCP control. We can see that its goodput pattern is very similar to that in Figures 5a and 5b when the buffer size per link is less than around 6 MSS. When the buffer size per link was larger than 6 MSS, the simulations yielded some varying results. Figure 5d plots the TCP goodput when standard TCP Reno was used without XCP control. We can see that it has much lower goodput than the simulated paced architectures had. More than 10-fold buffering is necessary to achieve utilization that is as high as that of paced architectures. When the buffer is small, the goodput of input buffering is almost the same as that of output buffering, which indicates that pacing is necessary to surpass the goodput of output buffering.

As shared buffering has the highest goodput for all simulated architectures, we did a general comparison of algorithms with shared buffering. Figure 6 plots the average goodput and core packet drop rate of XCP-paced standard TCP, paced TCP, and standard TCP on a shared buffered switch architecture based on the optical RAM buffer size per link. We can see that when the buffer is small, XCP pacing methods yield higher goodput and lower packet drop rates in the core nodes than paced and standard TCP.

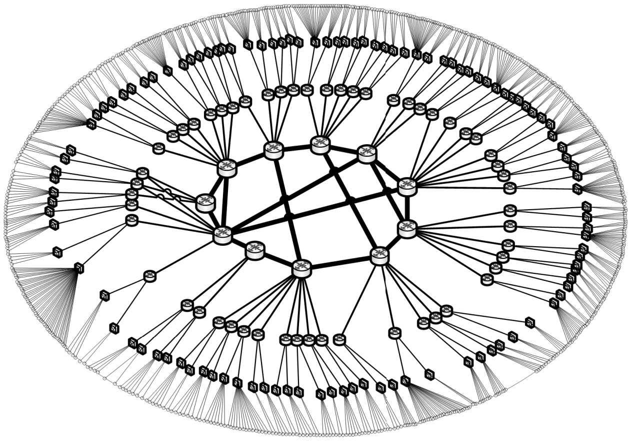
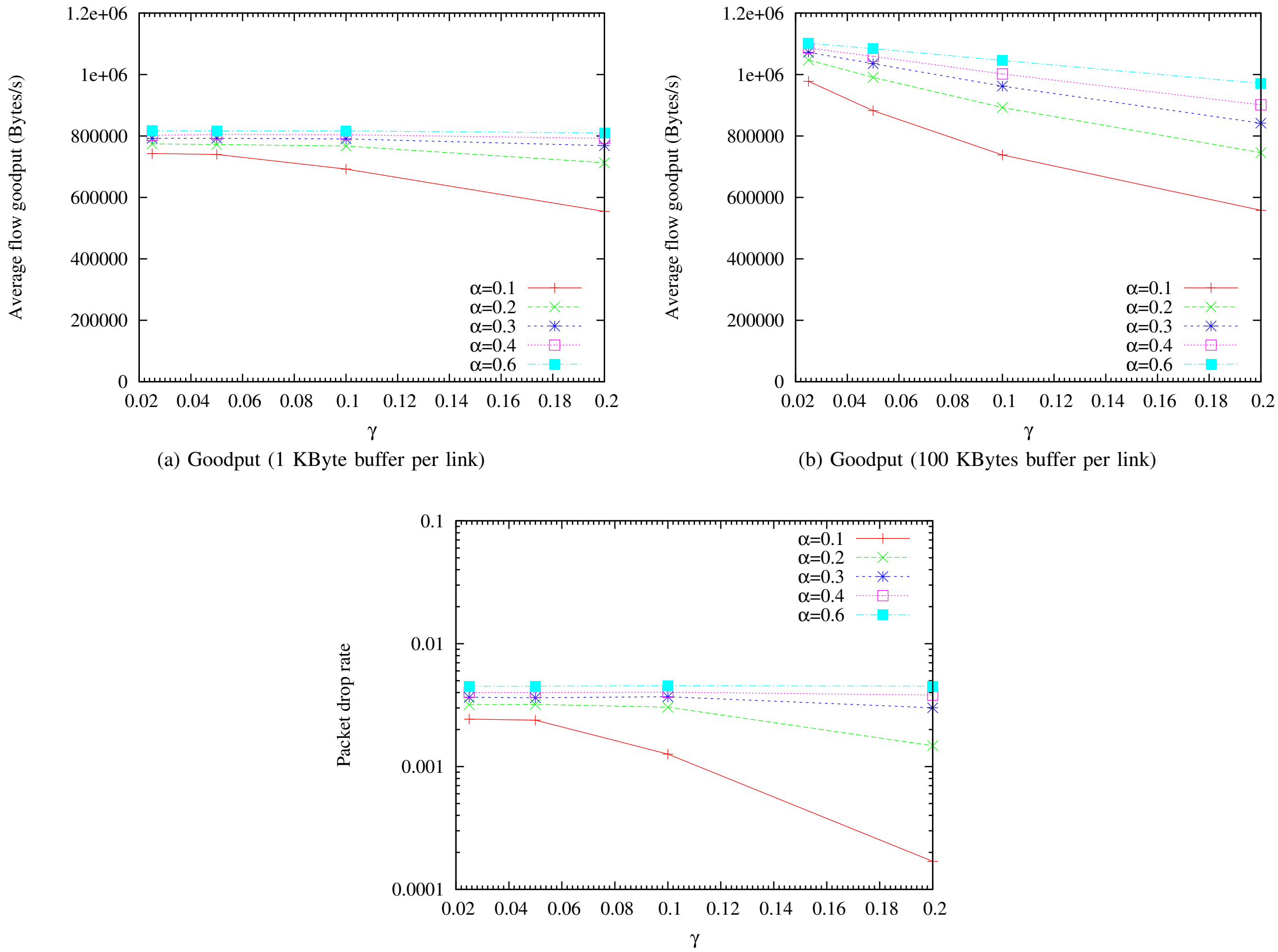


Figure 7. Abilene topology

C. *Abilene Topology Results*

Abilene is an Internet backbone network for higher education and part of the Internet2 initiative. The Abilene-inspired topology in Figure 7 from Li et al. [18] was used in the simulations. The topology has a total of 869 nodes that are divided into two groups of 171 core nodes and 698 edge nodes. A total of 2232 TCP flows started randomly and sent traffic between randomly selected edge node pairs. The total simulation time was 40 s. There was a single data wavelength on the links. The propagation delay of the edge and core links corresponded to 0.1 ms and 1 ms. All the links had a 1 Gbps capacity.

1) Optimization of XCP Parameters: First, we simulated a range of α and γ parameters to optimize the XCP parameters on the Abilene topology. Figure 8 plots the average



(a) Goodput (1 KByte buffer per link)

(b) Goodput (100 KBytes buffer per link)

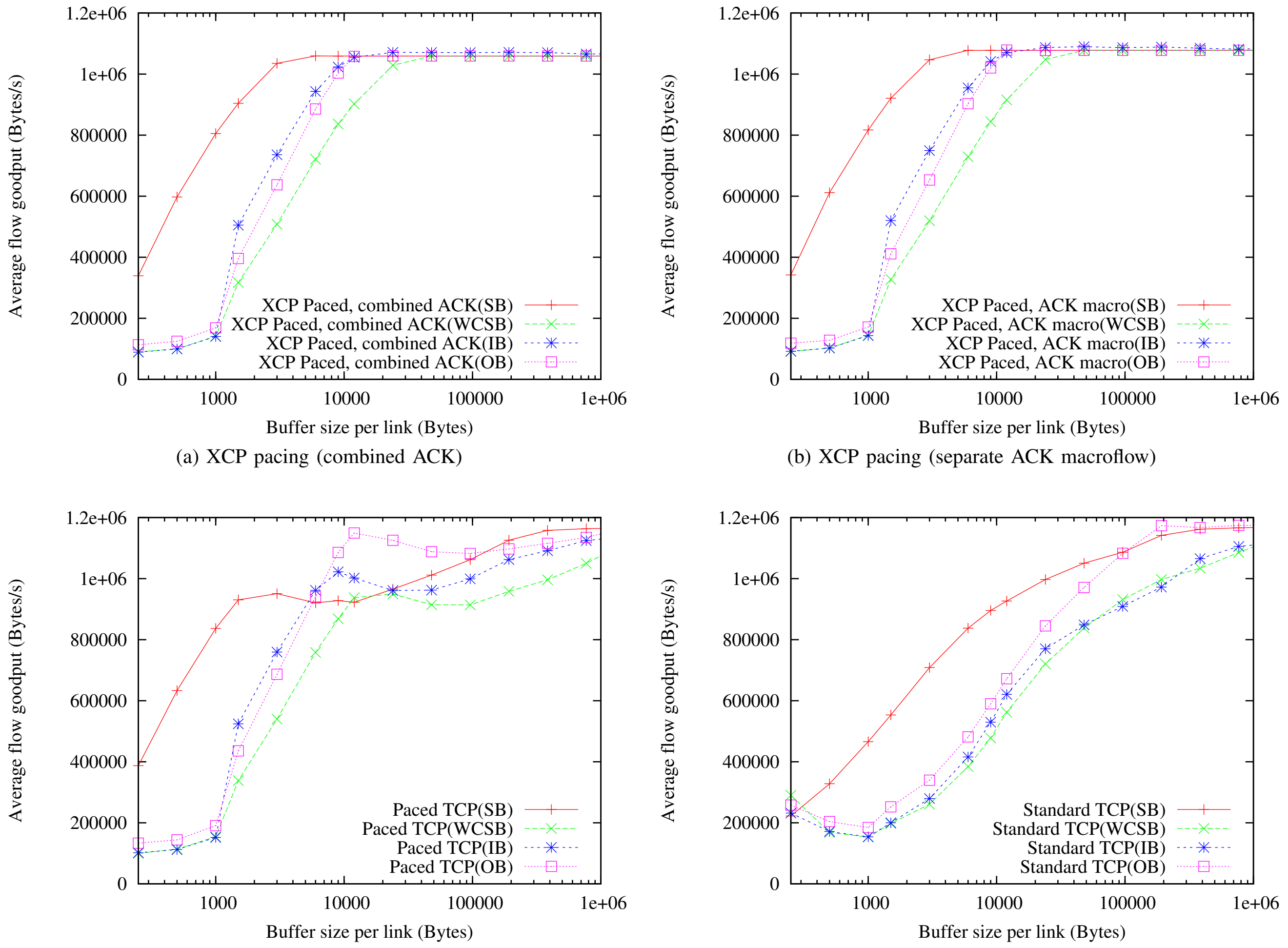
(c) Packet drop rate in the core nodes (1 KByte buffer per link)

Figure 8. Optimization of parameters in Abilene topology

goodput of TCP flows and packet drop rate at the core nodes when shared buffering was used. We can see that both the average goodput and packet drop rate in the core nodes increase with increasing α as occurs in optimizing the NSFNET topology. However, under-utilization due to the max-min fairness problem is more visible in the Abilene topology in Figure 8b. Changing the α or γ parameter yields a wider change in goodput than in NSFNET. Figure 8 shows that the α, β, and γ values selected in Sec. III-B give high goodput, so we chose the same values as those in NSFNET. If the time for fairness convergence is not a concern, a lower γ value can be chosen to further increase goodput.

2) Comparison of Switches and Algorithms: Figure 9 plots the average goodput of TCP flows on different switch and network architectures based on the optical RAM buffer size per link. We can see that the goodput plots of XCP pacing in the Abilene topology in Figures 9a and 9b are similar to those of the NSFNET simulations in Figures 5a and 5b. However, the goodput gap between worst-case shared buffering and output buffering is higher due to the higher nodal degree of the Abilene topology. Figure 9c shows that as we increase the buffer size, the goodput of TCP pacing in the Abilene topology yields even wider fluctuations than those in NSFNET. It seems that output buffering can handle the paced TCP traffic better than other switch architectures and even surpasses the goodput of shared buffering greatly when the buffer is small. Figure 9d shows that output buffering with standard TCP yields a performance boost over input buffering in the Abilene topology, because output buffering can handle bursty TCP traffic in a node with a high nodal degree better.

As a last step, we did a general comparison of algorithms in the Abilene topology with shared buffering. In Figure 10, we can see that when the buffer is very small, XCP pacing methods yield almost the same goodput as TCP pacing. However, when the shared buffer is larger than 1 MSS,



(a) XCP pacing (combined ACK)

(b) XCP pacing (separate ACK macroflow)

(c) TCP pacing

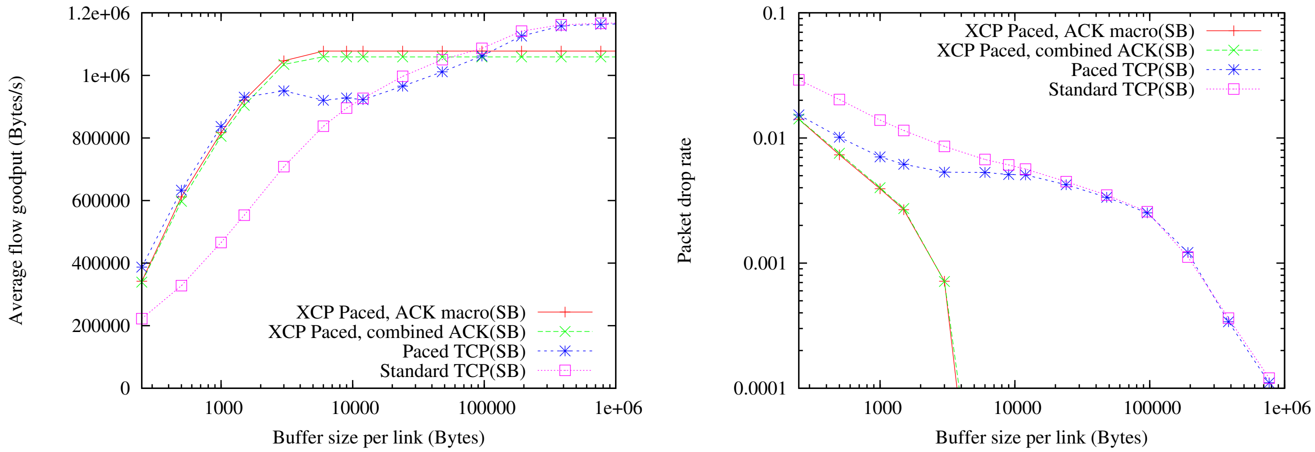
(d) Standard TCP

Figure 9. Goodput comparison of algorithms and switch architectures in Abilene topology

the goodput of TCP pacing stalls and XCP pacing methods yield higher goodput than TCP pacing. XCP pacing methods reach their maximum goodput when around 2–3 MSS per link of shared buffer is used. However, paced TCP and standard TCP require around 30–60 MSS per link of shared buffer to reach the goodput of XCP pacing methods. When the buffer is larger, standard TCP and paced TCP achieve slightly higher utilization than XCP due to the max-min fairness problem with XCP, which causes some bottleneck links to become under-utilized. However, as our aim is to increase performance with small buffers, the difference in goodput with such large buffers is not a concern. Optical RAM is not expected to attain a large capacity, so shared buffering is good. Figure 10b shows that XCP pacing yields a lower packet drop rate in the core nodes than paced or standard TCP just like in the NSFNET simulations. When we compare the two XCP pacing methods in Figure 10, we can see that their goodput and packet drop rates are very close, which indicates that the ACK compression problem in the Abilene topology is much less than that in NSFNET.

IV. Conclusion and Future Work

We investigated and compared optical-buffered switch architectures and pacing algorithms for minimizing the buffer requirements of OPS switches. By using two mesh topologies, our simulations revealed that even under bursty TCP traffic, using our architecture based on optical rate-based paced XCP, which is a modified version of XCP adapted to work as an intra-domain traffic shaping and congestion control protocol in an OPS network domain, could yield equal or higher TCP goodput and lower packet drop rates in the core nodes than using paced TCP, which is the solution that has generally been proposed in the literature. Moreover, simulations in the Abilene topology revealed that the goodput of paced TCP might exhibit some fluctuating behaviors, which adversely affect its performance even with relatively small buffers.



(a) Goodput

(b) Packet drop rate in the core nodes

Figure 10. Comparison of algorithms in Abilene topology with shared buffering

There are many advanced switch architectures in the literature like combined input output buffered switches, but most of them have have high scheduling complexity, which may become a bottleneck at ultra high speed of optical networks, so we limited our work to simpler architectures with lower scheduling complexity. When we compared the small buffered switch architectures simulated in this work, we could see that shared buffering yielded much higher TCP goodput than input or output buffering as it used small buffer capacity in the node more effectively. When the traffic was paced, input buffering yielded higher TCP goodput than output buffering while using much smaller switches. In NSFNET topology, where the nodal degree of nodes was small, worst-case shared buffering yielded almost the same goodput as output buffering with a much smaller buffer capacity per switch. Therefore, output buffering looks as though it is the worst choice as a switch architecture for small buffered optical networks with a low nodal degree.

Overall, the combination of applying XCP pacing and using shared buffered switches generally yielded the highest performance in terms of goodput and packet drop rate for small buffered OPS networks. Our XCP pacing proposal only operates at the edge/core routers of OPS domains and there are still no optical-RAM-buffered OPS networks deployed on the Internet, so it can be applied to OPS networks when they become commercially available, while paced TCP solution is harder to deploy as this requires replacing the TCP stack of computers on the Internet.

As a future work, we will work on the max-min fairness problem of XCP, which causes some bottleneck links to be under-utilized. Our work has revealed that shared buffering requires much less buffering than input and output buffering, but the required buffer size is not clear. Therefore, we will try to formulate the relationship between the number of wavelengths, traffic type, nodal degree, and the required shared buffer size. We intend to simulate other state-of-the-art congestion control algorithms and pacing methods with different types of traffic to gain a broader understanding of their performance in different switch architectures in future work.

ACKNOWLEDGMENTS

This work was partly supported by the National Institute of Information and Communications Technology (NICT).

REFERENCES

[1] O. Alparslan, S. Arakawa, and M. Murata, “Packet switch architectures for very small optical RAM,” in *Proceedings of The First International Conference on Evolving Internet (INTERNET 2009)*, 2009, pp. 106–112.

[2] C. Villamizar and C. Song, “High performance TCP in ANSNET,” *Computer Communication Review*, vol. 24, no. 5, pp. 45–60, 1994.

[3] T. Aoyama, “New generation network(NWGN) beyond NGN in Japan,” Web page: http://akari-project.nict.go.jp/document/INFOCOM2007.pdf, 2007.

[4] G. Appenzeller, J. Sommers, and N. McKeown, “Sizing router buffers,” in *Proceedings of ACM SIGCOMM*, 2004, pp. 281–292.

[5] M. Enachescu, Y. Ganjali, A. Goel, N. McKeown, and T. Roughgarden, “Part III: Routers with very small buffers,” *ACM SIGCOMM Computer Communication Review*, vol. 35, pp. 83–90, 2005.

[6] L. Zhang, S. Shenker, and D. D. Clark, “Observations on the dynamics of a congestion control algorithm: The effects of two-way traffic,” in *Proceeedings of ACM SIGCOMM*, 1991, pp. 133–147.

[7] G. Theagarajan, S. Ravichandran, and V. Sivaraman, “An experimental study of router buffer sizing for mixed TCP and real-time traffic,” in *Proceedings of IEE ICON*, 2006.

[8] O. Alparslan, S. Arakawa, and M. Murata, "Rate-based pacing for small buffered optical packet-switched networks," *Journal of Optical Networking*, vol. 6, no. 9, pp. 1116–1128, September 2007.

[9] D. Katabi, M. Handley, and C. Rohrs, "Congestion control for high bandwidth-delay product networks," in *Proceedings of ACM SIGCOMM*, 2002, pp. 42–49.

[10] G. Papadimitriou, C. Papazoglou, and A. Pomportsis, "Optical switching: Switch fabrics, techniques, and architectures," *Journal of Lightwave Technology*, vol. 21, no. 2, pp. 384–405, 2003.

[11] O. Alparslan, S. Arakawa, and M. Murata, "Rate-based pacing for optical packet switched networks with very small optical RAM," in *Proceedings of IEEE Broadnets*, September 2007, pp. 300–302.

[12] ——, "XCP-based transmission control mechanism for optical packet switched networks with very small optical RAM," *Photonic Network Communications*, vol. 18, no. 2, pp. 237–243, October 2009.

[13] S. Kandula, D. Katabi, B. Davie, and A. Charny, "Walking the tightrope: Responsive yet stable traffic engineering," in *Proceedings of ACM SIGCOMM*, 2005, pp. 253–264.

[14] N. McKeown and T. E. Anderson, "A quantitative comparison of iterative scheduling algorithms for input-queued switches," *Computer Networks*, vol. 30, no. 24, pp. 2309–2326, 1998.

[15] S. McCanne and S. Floyd, "ns Network Simulator," Web page: http://www.isi.edu/nsnam/ns/, July 2002.

[16] S. Low, L. Andrew, and B. Wydrowski, "Understanding XCP: equilibrium and fairness," in *INFOCOM*, 2005, pp. 1025–1036.

[17] J. C. Mogul, "Observing TCP dynamics in real networks," in *Proceedings of ACM SIGCOMM*, 1992, pp. 305–317.

[18] L. Li, D. Alderson, W. Willinger, and J. Doyle, "A first-principles approach to understanding the Internet's router-level topology," in *Proceeedings of ACM SIGCOMM*, 2004, pp. 3–14.