Photon Netw Commun (2011) 22:172–179
DOI 10.1007/s11107-011-0317-z

Node pacing for small optical RAM-buffered packet-switching networks

**Onur Alparslan · Shin'ichi Arakawa ·
Masayuki Murata**

Received: 3 August 2009 / Accepted: 17 June 2011 / Published online: 1 July 2011

Abstract One of the difficulties with optical packet switched (OPS) networks is buffering optical packets in the network. The only available solution that can currently be used for buffering in the optical domain is using long fiber lines called fiber delay lines (FDLs), which have severe limitations. Moreover, 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 very small buffered OPS networks. We therefore propose a new node-based pacing algorithm for decreasing burstiness. We show that by applying some simple pacing at the edge or core backbone nodes, the performance of very small optical RAM buffered core OPS networks with variable-length packets can be notably increased.

Keywords Small buffer · OPS · Optical RAM · TCP · Pacing

1 Introduction

Contention between packets in electronic packet-switched (EPS) networks is resolved by storing the contended packets in an electronic random access memory (RAM). Electronic RAM allows packets to be sent out with $O(1)$ reading operation when the output port is free. However, employing electronic RAM for buffering in optical networks is a challenging task. Recent advances in optical networks like dense wavelength division multiplexing (DWDM), which combines and transmits multiple signals simultaneously at different wavelengths on the same fiber, have allowed us to achieve ultra-high data-transmission rates in optical networks. While optical transmission rates are becoming faster increasingly, electronic components in routers have become a bottleneck. For example, an electronic approach to O/E/O conversion in order to use electronic RAM is not a feasible solution because of electronic processing limitations. The operating speed of the electronic components and electromagnetic interference limit the packet bit rate to about 10 Gbps [1]. One possible solution is to use all-optical O/E/O conversion. The basic operation of such an optoelectronic RAM has experimentally been confirmed for 40 Gbps 16-bit optical packets [1]. Takahashi et al. [1] used parallelized all-optical convertors at the input and output of an electronic RAM that could achieve fast O/E/O conversion. However, electronic RAM becomes the new bottleneck. A very high level of RAM parallelization would be necessary to reach the ultra-high-speed of optical networks. Moreover, large buffer-size requirements would be challenging for router manufacturers, who must use large, slow, on-chip DRAMs [2]. According to a historical rule-of-thumb [3], which is widely used in routers, the buffer size of each output link of a router should be $B = RTT \times BW$, where RTT is the average round trip time of flows and BW is the bandwidth of the output link, in order to achieve high utilization with TCP flows. This buffer requirement increases linearly with link speed, which leads to greater buffer consumption and more board space. Network operators and router manufacturers currently follow a guideline of 250 ms of buffer size per link. According to this guideline, a 40 Gbps link requires 10 Gbits of buffer size. Moreover, DWDM is capable of ultra-high data rates in excess of 1 Petabit per fiber, which would require 250 Tbits of ultra-high-speed buffer per fiber. It is clear that it is very difficult to satisfy such large buffer size requirements in high speed optical networks.

O. Alparslan (✉) · S. Arakawa · M. Murata
Graduate School of Information Science and Technology,
Osaka University, 1-5 Yamadaoka, Suita, Osaka 565-0871, Japan
e-mail: a-onur@ist.osaka-u.ac.jp

All-optical buffering looks like a promising solution to solving these problems as it does not require any O/E/O conversion. Furthermore, storing packets in the optical domain may need less power and board space than electronic buffering. Currently, the only available solution to all-optical buffering is using fiber delay lines (FDLs). Contended packets are switched to FDLs so that they can be delayed. However, FDLs have severe limitations. First, FDLs require very long fiber lines that cause signal attenuation inside the routers. As there can only be a limited number of FDLs in a router due to space considerations, they can only provide a small amount of buffering. Second, FDLs only provide a fixed amount of delay. As FDL buffering is possible with today's technology, many researchers have investigated FDL buffers to resolve contentions in optical networks. However, all-optical RAM is still being researched (e.g., NICT project [4]) and this may become available in the near future. Optical RAM solves the problems with FDLs such as lack of real $O(1)$ reading operation, signal attenuation, and bulkiness. Furthermore, optical RAM may have lower power and space requirements than FDLs and electronic RAM. For example, Shinya et al. [5] demonstrated a photonic crystal-based all-optical bit memory operating at very low power. However, optical RAM is not expected to have a large capacity soon. For example, architecture in [5] had a memory time of 150 ns, which was 0.06% that of the current guideline of 250 ms buffer per link. Therefore, it is necessary to decrease the buffer requirements of OPS networks to make use of optical RAM.

TCP is well-known to exhibit bursty behavior [6]. The bursty nature of TCP is the main problem limiting the decrease in buffer requirements, because this results in high packet drop rate in very small buffered networks. A general solution to solving this problem is to apply pacing, which delays packets according to a special criteria that decreases short-term burstiness and hence smooth network traffic. We proposed [7] a node pacing algorithm that could pace traffic at edge or core nodes by using buffer occupancy information like that in RC traffic shaper [8]. Our algorithm solved the problems with the RC traffic shaper by using a piecewise-linear output rate control function and making use of average input-traffic rate information calculated inside the node. Furthermore, our algorithm was designed for variable length packet-based traffic unlike the RC traffic shaper and IFSA [9] that were designed for fixed cell ATM traffic. We briefly explained [7] some basics of our proposed architecture there. Here we present a detailed description of our algorithm and extensive simulation results on two meshed topologies. We explain how our pacing algorithm can considerably increase the achievable utilization of very small optical RAM-buffered optical core links or namely the throughput of TCP flows using these links.

The rest of the paper is organized as follows. Section 2 introduces related work in the literature. Section 3 describes the details on the proposed algorithm and the switch architecture. Section 4 describes the simulation methodology and presents the simulation results. Finally, we conclude the paper in Sect. 5.

2 Related work

This section introduces and discusses some related work on decreasing buffer requirements by using pacing, which can be applied to packets at the network layer (node pacing) or transport layer (TCP pacing).

2.1 TCP pacing

TCP pacing is defined as transmitting ACK (data) packets according to a special criteria, instead of immediately transmitting when data (ACK) packets arrive [10]. TCP pacing was initially proposed as a solution to ACK-compression [10]. Kulik et al. [11], proposed using paced TCP to solve the problem with queuing bottlenecks by preventing bursty behavior by TCP. Enachescu et al. [12] proposed that $O(\log W)$ buffers are sufficient where W is the maximum congestion window size of flows when packets are sufficiently paced by modifying TCP senders to use paced TCP or by using slow access links. This is known as the Stanford tiny-buffer model. 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 bandwidth on the network. Using paced TCP for these applications by replacing TCP senders with paced versions can be difficult. 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 placing a strict limit on the speed of access links, and that does not require replacing the sender or receiver TCP and UDP agents of computers using the network.

2.2 Node pacing

The main authors of the Stanford tiny-buffer model recently commented that when their initial conditions (using slow access links or modifying TCP) [12] did not hold, traffic shapers could be used at the edge to space out packets entering the network in order to apply the Stanford tiny-buffer model [13]. We agree with them and believe that traffic shaping by node pacing, can be a much more practical solution than changing TCP or limiting the link capacity. Applying node pacing inside the core and/or edge nodes at the network layer can decrease traffic burstiness and hence reduce buffer requirements. However, the problem with node pacing is that unlike TCP pacing, where pacing can be simply applied by spacing transmitted packets across the RTT calculated by TCP, there are no general guidelines in the literature on how to apply node pacing. As intermediate nodes do not normally know the RTT of flows that pass through their links, it is difficult to determine the amount of delay that should be applied to packets to mimic TCP pacing. One possible solution to a closed domain OPS network is to obtain RTT or available transmission rate information inside the domain by exchanging probe packets between the edge nodes of the network. For example, we recently proposed [14] using an explicit congestion control protocol (XCP) based architecture for pacing at the edge nodes. The edge nodes calculate the pacing rate by exchanging probe packets with other edge nodes. We found [15] that our architecture could achieve high utilization and a low packet drop ratio with TCP flows in very small optical RAM-buffered OPS networks, which were even better than those with TCP pacing.

Our edge pacing architecture [14,15] adds some complexity to an OPS network as all OPS nodes inside the domain should be XCP capable and edge nodes should exchange XCP probe packets. A more elegant solution would be to apply pacing with only locally available information without requiring any assistance or probe packets from other nodes in the network. For example, Sivaraman et al. [16] proposed such a delay-based real-time edge pacing algorithm that adaptively chose packet spacing according to the input traffic class to satisfy bounded delay requirements. However, this only applied pacing at the edge nodes that could make use of large electronic buffers to apply pacing before the packets were converted into the optical domain. However core pacing is much more challenging, because the core buffer capacity may not be sufficient to apply both pacing and provide enough buffering to simultaneously deal with contending packets as the core's optical buffer is very small compared to that of edge routers. Using a portion of the buffer for pacing further decreases the capacity of the buffer to resolve contention, and vice-versa. Some traffic shapers were proposed for core nodes in the ATM era, but most of these required information on the traffic rate as an input parameter. To the best of our knowledge, the first core shaper that adapted itself to incoming traffic was proposed by Tung et al. [8] who used an RC traffic shaper for ATM networks that smoothed the traffic by adjusting the output rate based on the buffer occupancy that depended on the incoming traffic rate. There is a direct proportion with the output rate and buffer occupancy. The problem with the proposed algorithm is that it requires a large buffer to prevent cell loss. Furthermore, the peak cell input rate must be known. To solve these problems, Zhu et al. [9] proposed an Interval Filter Shaping Algorithm (IFSA) that smoothed the cell inter-arrival time with a low latency according to cell arrival times. IFSA made use of a low pass filter and a special scheduler to smooth the traffic. However, as both the RC traffic shaper and IFSA were designed for fixed sized cell-based ATM traffic, they may be difficult to apply to optical packet switching with variable-length packets.

3 Node pacing architecture

This section describes our proposed node pacing architecture in detail.

3.1 Network architecture

Figure 1 shows the proposed network architecture for node pacing. Nodes in the OPS domain are divided into three groups of

- Edge nodes (E): Nodes where IP traffic enters or exits the OPS domain,
- Middle core nodes (M): Core nodes that are connected to edge nodes, and
- Center core nodes (C): Core nodes that are not connected to edge nodes.

Figure 1 shows the type of node pacing applied to bidirectional links. Traffic inside the OPS domain can be paced at three different stages, viz., at the edge nodes, middle core nodes, or center core nodes. It is possible to apply node pacing at a single stage or a combination of these.

The effect and capacity of node pacing heavily depends on the placement of the node in the topology. Edge nodes can use large amounts of electronic RAM buffering for pacing before converting incoming traffic to the optical domain. However, as there is no electronic RAM inside the core routers, so they only use small optical buffers for pacing. Core routers are divided into two groups for middle core and center core nodes, because they usually differ greatly in their average nodal degrees. For example, in the Abilene topology, which we simulated and discussed in Sect. 4, average nodal degree of middle core nodes is much higher than that of center core nodes due to large number of edge nodes connected to the middle core nodes. Having a higher nodal degree usually increases the packet contention rate. As it may not be necessary to differentiate between middle and center core nodes, in some networks both of them can be simply regarded as core nodes.

3.2 Traffic shaper

The RC traffic shaper [8] calculates the cell output rate according to the direct proportion

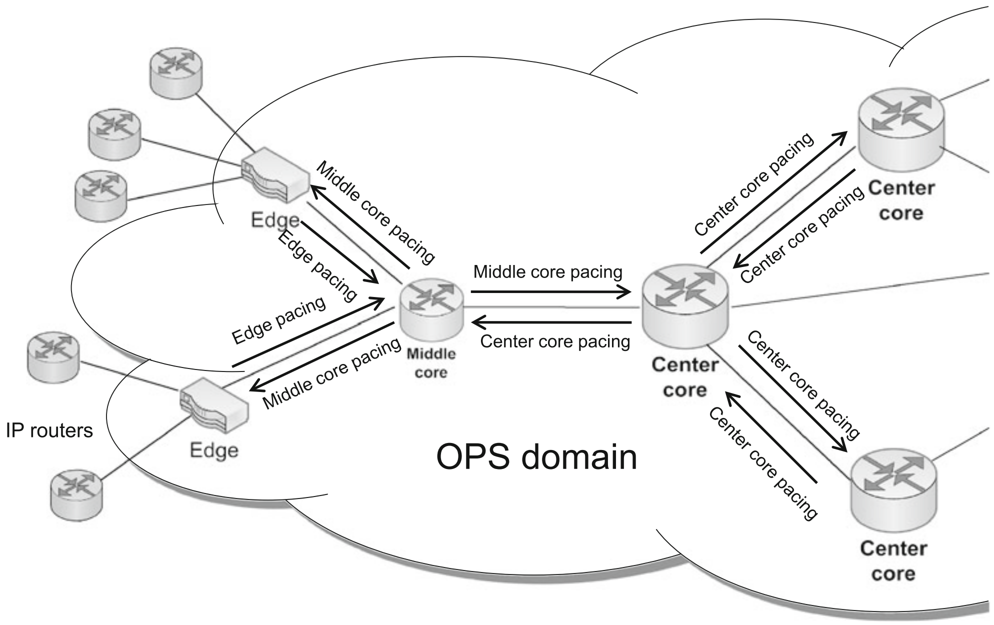


Fig. 1 Network architecture

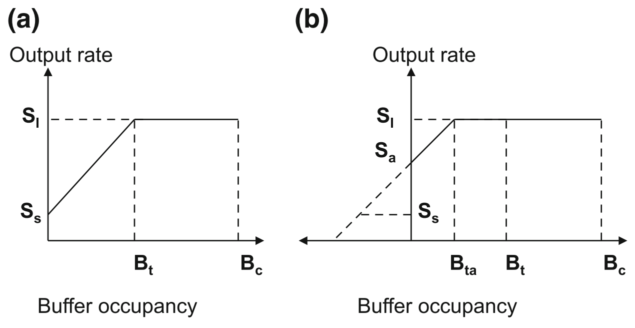


Fig. 2 **a** Transfer function and **b** adaptive transfer function

$$\lambda_o(t) = \frac{B_c(t)}{B_c}\lambda_{i\,\max}, \tag{1}$$

where $\lambda_o(t)$ is the output traffic rate, $B_c(t)$ is the buffer occupancy, B_c is the buffer capacity, and $\lambda_{i\,\max}$ is the maximum input traffic rate. Because algorithm only sets the output traffic rate to equal input traffic rate when the buffer is full, the buffer may easily overflow and cause packets to drop when utilization is high.

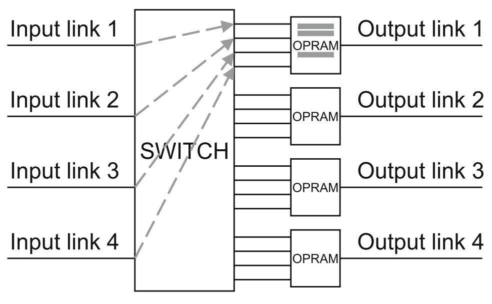
We propose using a piecewise linear transfer function instead of direct proportion. Figure 2a shows the transfer function where B_t is the buffer threshold, B_c is the buffer capacity, S_s is the initial link speed and S_l is the link capacity. The output rate in the RC traffic shaper only reaches the input rate, when the buffer is full. However, in our algorithm the output rate reaches the input rate after a buffer threshold is reached. This provides a safety margin for decreasing buffer overflows where link utilization is high as there is still some free space in the buffer. However, this may not be sufficient for decreasing the average buffer occupancy. When this fixed transfer function is used, an average output traffic rate higher than S_s still requires a non-zero average buffer occupancy that increases packet-drop probability, especially in very small buffer networks. For example, an average output traffic rate that is equal to the link capacity requires an average buffer occupancy of B_t leaving only an average buffer capacity of $B_c - B_t$ for solving contentions. We make use of both the buffer occupancy and the average input traffic rate for calculating the output traffic (pacing) rate to solve this problem. We adaptively shift the x axis (buffer occupancy) of the transfer function according to the average arrival rate, so that pacing requires less buffer space. Figure 2b shows the adaptive transfer function, where S_a is the short-term average input traffic arrival rate that can periodically be received from the router's traffic statistics software. B_{ta} is the new buffer threshold after the transfer function is shifted according to S_a. If S_a is smaller than S_s, S_a is taken as S_s.B_{ta} at time t can be simply calculated by

$$B_{\text{ta}}(t) = B_t \frac{S_l - S_a(t)}{S_l}. \tag{2}$$

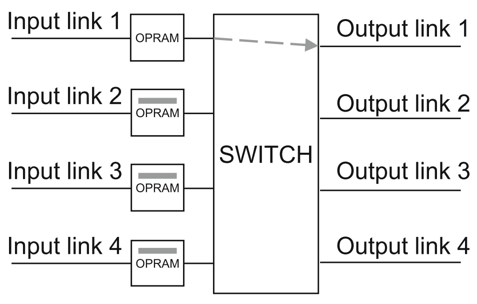
As this adaptive transfer function allows a high output traffic rate even when the buffer occupancy is zero, the average buffer occupancy can be decreased.

3.3 Switch and scheduler architectures

We simulated and compared the output buffered and input buffered core-node architectures as seen in Fig. 3a and b. As the output buffered switch architecture has full speed-up,



(a) Output buffering



(b) Input buffering

Fig. 3 Switch architectures

there is no blocking. The input buffered switch architecture has a speed-up of 1, so it has a smaller switching fabric and cost when compared with the output buffered switch architecture. The examples in Fig. 3a and b show when all the input links of a 4×4 switch have an optical packet destined for output link 1. In output buffering, the switch speed-up must be 4 to prevent blocking. However, in input buffering, we can forward one of the packets and store the rest in the input buffers. Therefore, input buffering requires a smaller switching fabric than output buffering. However, a well-known problem with input buffering is head-of-line blocking, which limits the maximum utilization [17]. We applied virtual output queuing (VOQ) [18] scheduling to minimize this problem.

The proposed algorithm requires output link buffer occupancy information that is normally zero for an input-buffered switch as there is no buffer at the output links. Therefore, in input-buffering, the effective buffer occupancy value of an output link is calculated by the sum of the virtual output queues destined for this output link.

4 Evaluation

This section discusses our evaluation of the new node pacing architecture and its comparison with the TCP pacing solution.

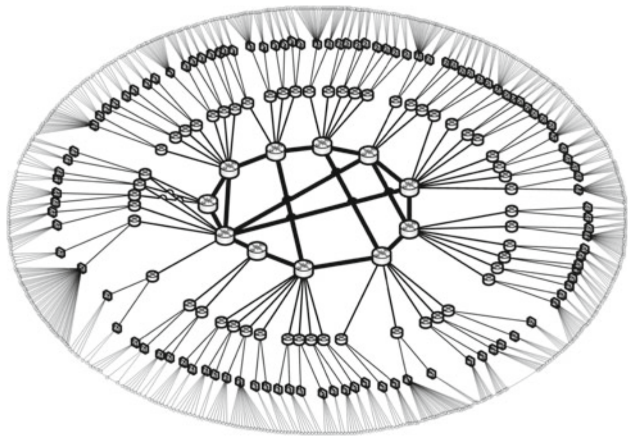


Fig. 4 Abilene topology

4.1 Simulation settings

The proposed network architecture and algorithms were implemented over *ns* version 2.32 [19]. The Abilene-inspired topology in Fig. 4 from Li et al. [20] was used in the simulations. Abilene is the Internet backbone network for higher education and a part of the Internet 2 initiative [20]. The topology has a total number of 869 nodes in total that are divided into three groups of 65 center core nodes (C), 106 middle core nodes (M), and 698 edge nodes (E) as described in Sect. 3.1.

Non-paced TCP Reno and Paced TCP Reno were used as the traffic sources. A total of 4581 TCP flows started randomly and sent traffic between randomly selected edge node pairs. The total simulation time is 20 s. The IP datagram for TCP data (ACK) packets was 1,500 Bytes (40 Bytes), so TCP data (ACK) segment was 1,480 Bytes (20 Bytes). There was a single data wavelength on links. The propagation delay of the edge and core links corresponded to 0.1 and 1 ms. All the links had a 1 Gbps capacity. The core node links were simulated with an optical RAM input buffer (B_c) of 50 Kbits, 100 Kbits, 200 Kbits, and 2 Mbits. The electronic RAM buffer of the edge nodes was 100 Mbits (100 ms). B_t was selected as half of B_c. S_s was selected as 0.1 Gbps for core and and 0.01 Gbps for edge nodes. We simulated a wide range of B_c and S_s parameters and found that choosing an S_s value much lower than the link capacity and a B_c value around half the buffer capacity generally yielded good results. When designing new networks, B_c and S_s parameters can be optimized through simulation. Different combinations of edge node pacing (E), middle core node pacing (M), center core node pacing (C) and no node pacing were simulated with standard TCP and paced TCP flows.

4.2 Abilene topology results

First, we simulated the proposed architecture on the Abilene topology and input buffered core nodes. Figure 5 plots the average utilization of backbone links between 10 center core

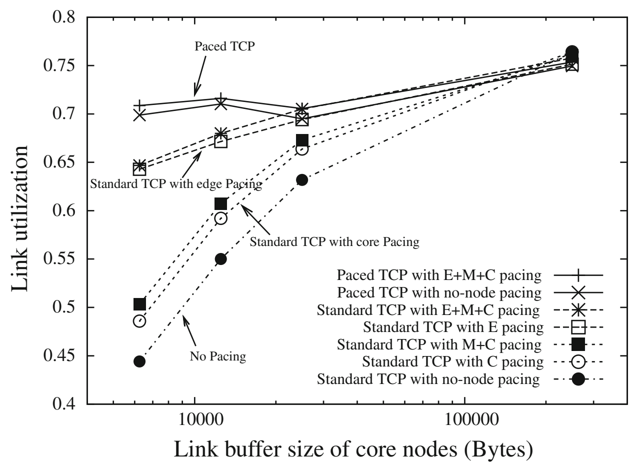


Fig. 5 Input buffered Abilene topology simulation results

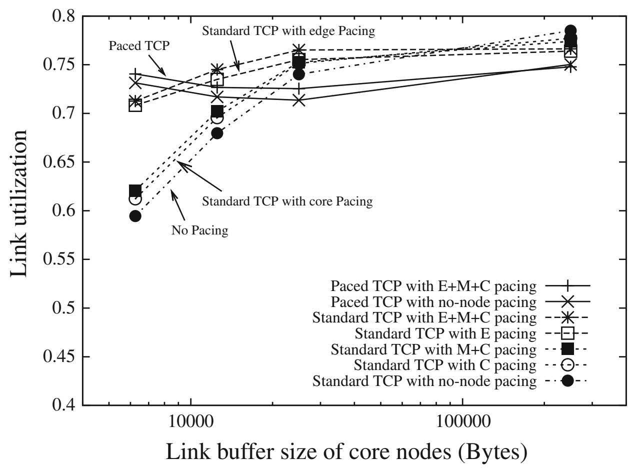


Fig. 6 Output buffered Abilene topology simulation results

nodes at the very center of the network. The *x*-axis plots the core link buffer size on a log scale and the *y*-axis plots the average link utilization in a range from [0.4–0.8] on a linear scale where 0.8 means 80% utilization.

When we checked the simulation results with the core buffer of 50 Kbits in Fig. 5, we found that the simulation results for paced TCP traffic with and without node pacing represented by solid lines were very close to each other. The reason for this is that paced TCP traffic is already sufficiently smooth, so extra node pacing does not bring about much improvement when paced TCP is used. The dashed lines plot the utilization when non-paced TCP is used with edge node pacing. Again, we can see that edge node pacing smooths the traffic sufficiently, so additional center and middle core pacing do not make much of difference. Unlike TCP pacing, edge node pacing paces the traffic without any information on RTT of flows, so it cannot space the packets optimally. Therefore, its utilization is a bit lower than TCP pacing. The dotted lines represent center core pacing, and center core + middle core pacing methods where the latter has slightly higher utilization. They have a lower utilization than edge pacing, because there are fewer core nodes than edge nodes and core nodes have a much smaller buffer (1/2000 that of the edge nodes), which is not enough to fully pace the traffic. The dot-dashed plot has the simulation results for no pacing, which has the lowest utilization as expected. The figure indicates that even when node pacing is only applied to core nodes with very small buffers, it is possible to achieve a considerably greater increase in throughput than when no pacing is applied. When we increase the core buffer to 100 Kbits, we can see that the utilization gap decreases. However, core node pacing still has a considerable gain over no pacing. When we increase the core buffer to 2 Mbits, we can see that all simulations yield very similar results, which means the core buffer is sufficiently large to solve the burstiness problem, so pacing does not make much of a difference.

Figure 6 plots the simulation results on the Abilene topology with output buffered core nodes. When we compare these with the input buffering results, we can see that the standard TCP simulations have higher utilization with output buffering, because output buffering can handle bursty traffic better due to its full speed-up. However, when the traffic is very smooth as in paced TCP traffic, there is not a huge difference in utilization between input and output buffering architectures.

Simulations with a core buffer of 50 Kbits in Fig. 6 reveal that utilization of our edge pacing with non-paced TCP is very close to that of paced TCP. However, center core pacing, and center core + middle core pacing methods do not improve utilization much over standard TCP with no node pacing. The reason for this is that due to the full speed-up of output buffering, many input links can send contending packets to an output buffer concurrently and quickly fill up the output buffer that causes packet losses. However, when there is a contention in the input buffering architecture, contending packets are stored in multiple input buffers. Therefore, the effective buffering capacity increases. As an example, let us assume that we have a switch with a nodal degree of six. The buffer capacity is two packets per link and all buffers are empty. Also consider a packet contention case where five input ports have a single packet simultaneously destined for a single output port. If output buffering is employed, all input ports forward their packets at the same time due to full speed-up and lack of buffering at the input. As a result, one packet is sent to the output link, two other contending packets are stored at the output buffer, and the last two packets are dropped due to a lack of buffer space. If input buffering is used, on the other hand, each contending packet is stored at a different input buffer and sent to the output port as the output becomes idle. Therefore, the effective buffering capacity for contending packets becomes higher than output buffering and no packet losses occur. However, it is necessary for traffic to arrive smoothly for this operation. Input buffering suffers from contending bursty traffic due to low speed-up, as a burst of packets arriving from an input port can easily fill up the input buffer until all contending packets are served and forwarded to output ports. As a result, paced TCP has a high utilization in both output and input buffering due to its efficient pacing and smooth traffic, but non-paced and core paced traffic have lower utilization with input buffering than with output buffering.

When we check the simulation results with core buffer sizes of 200 Kbits and 2 Mbits in Fig. 6, we can see that all the simulations with paced TCP yield lower utilization than the standard TCP simulations. The decreased performance is due to synchronized dropped packets with Paced TCP that cause underutilization. It is well-known that paced TCP suffers from synchronization with large buffers [21]. Therefore, utilization with paced TCP decreases as we increase the size of buffer. As node pacing does not seem to be as greatly affected by synchronization, our edge pacing architecture yields even higher utilization than paced TCP.

4.3 Sparse Abilene topology results

We simulated a network where core pacing could be even more effective, which we discuss in this section. The biggest challenge in core pacing is that core nodes have very small buffers so their traffic-pacing capacity is much less than that with paced TCP or edge pacing. However, when the average number of hops between a source and destination node pair is high, packets can undergo cumulative pacing at many core nodes that can increase the effect of pacing and smooth the traffic. However, the nodal degree of center code nodes is high in the Abilene topology. Having numerous links between the center core nodes decreases the average number of hops and the efficiency of core pacing. To obtain a higher hop count, we created a ring topology at the center by removing the six links between center core nodes at the very center of the topology while retaining the links between adjacent center core nodes. We called this topology a sparse Abilene-inspired topology. We simulated the sparse Abilene-inspired topology with input buffered core nodes. Due to the lower nodal degree, many center core links became overutilized, so we decreased the total number of TCP flows to one fifth. Moreover, we removed the congestion window size limit of TCP flows, so even a single TCP flow could fully utilize a link. Moreover, TCP flows could exhibit burstier behavior as they could achieve larger window sizes.

Figure 7 shows that non-paced architecture with standard TCP has very low utilization due to high burstiness and smaller buffers. However, core pacing, and center core + middle core pacing methods with standard TCP considerably improve utilization by almost 30% over that of standard TCP with no node pacing. Again, edge pacing with standard TCP has higher utilization than core pacing due to higher pacing capacity with large edge buffer. Paced TCP again provides the highest utilization due to its optimal pacing with RTT information. All methods yield similar utilization when the buffers are large enough.

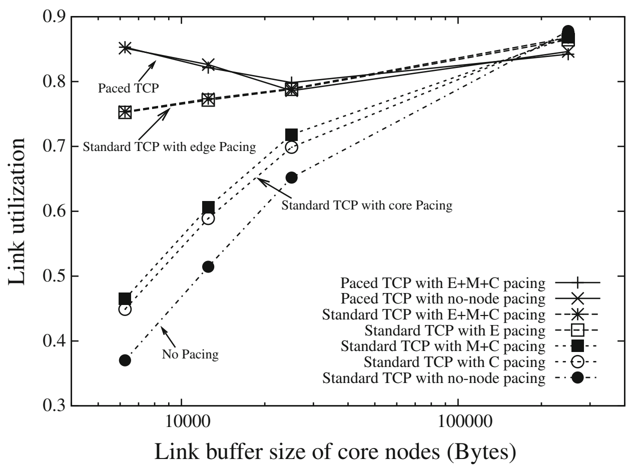


Fig. 7 Input buffered sparse Abilene topology simulation results

5 Conclusions

We presented a new node pacing algorithm for edge and/or core backbone nodes to increase the efficiency of utilization of very small optical RAM-buffered OPS networks with variable-length packets. Our simulation results revealed that our node pacing algorithm could increase the achievable utilization of very small buffered optical core links or namely the throughput of TCP flows using these links. We demonstrated that applying stand-alone edge node pacing could double the utilization of small buffered optical links, which was very close to the performance of paced TCP. Moreover, we showed that core pacing was capable of greatly increasing performance, even though core buffers were very small. We intend to further evaluate the performance of the proposed algorithm with different architectures and traffic types in future work.

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

References

[1] Takahashi, R., Nakahara, T., Takahata, K., Takenouchi, H., Yasui, T., Kondo, N., Suzuki, H.: Photonic random access memory for 40-Gb/s 16-b burst optical packets. IEEE Photonics Technol. Lett. **16**, 1185–1187 (2004)

[2] Appenzeller, G., Sommers, J., McKeown, N.: Sizing router buffers. In: Proceedings of ACM SIGCOMM, pp. 281–292 (2004)

[3] Villamizar, C., Song, C.: High performance TCP in ANSNET. Comput. Commun. Rev. **24**(5), 45–60 (1994)

[4] Aoyama, T.: New generation network(NWGN) beyond NGN in Japan. Web page: http://www.akari-project.nict.go.jp/document/INFOCOM2007.pdf (2007)
[5] Shinya, A., Matsuo, S., Yosia Tanabe, T., Kuramochi, E., Sato, T., Kakitsuka, T., Notomi, M.: All-optical on-chip bit memory based on ultra high Q InGaAsP photonic crystal. Opt. Express **16**(23), 19382–19387 (2008)
[6] Jiang, H., Dovrolis, C.: Source-level IP packet bursts: causes and effects. In: Proceeedings of ACM SIGCOMM/Usenix Internet Measurement Conference, pp. 301–306 (2003)
[7] Alparslan, O., Arakawa, S., Murata, M.: Node pacing for optical packet switching. In: Proceeedings of Photonics in Switching (2008)
[8] Tung, T.Y., Chen, Y.J., Chang, J.F.: Design and analysis of RC traffic shaper. IEICE Trans. Commun. **E81-B**, 1–12 (1998)
[9] Zhu, H., Ma, Z., Cao, Z., Wang, Y.: Low latency traffic interval shaping algorithm for traffic access control. Chin. J. Electron. **11**(2), 247–251 (2002)
[10] Zhang, L., Shenker, S., Clark, D.D.: Observations on the dynamics of a congestion control algorithm: the effects of two-way traffic. In: Proceeedings of ACM SIGCOMM, pp. 133–147 (1991)
[11] Kulik, J., Coulter, R., Rockwell, D., Partridge, C.: A simulation study of paced TCP. Technical Report, BBB (1999)
[12] Enachescu, M., Ganjali, Y., Goel, A., McKeown, N., Roughgarden, T.: Part III Routers with very small buffers. ACM SIGCOMM Comput. Commun. Rev. **35**, 83–90 (2005)
[13] Beheshti, N., Ganjali, Y., Goel, A., McKeown, N.: Obtaining high throughput in networks with tiny buffers. In: Proceeedings of IWQoS (2008)
[14] Alparslan, O., Arakawa, S., Murata, M.: Rate-based pacing for small buffered optical packet-switched networks. J. Opt. Netw. **6**(9), 1116–1128 (2007)
[15] Alparslan, O., Arakawa, S., Murata, M.: XCP-based transmission control mechanism for optical packet switched networks with very small optical RAM. Photonic Network Commun. **18**(2), 237–248 (2009)
[16] Sivaraman, V., Elgindy, H., Moreland, D., Ostry, D.: Packet pacing in short buffer optical packet switched networks. In: Proceeedings of IEEE INFOCOM (2006)
[17] Karol, M., Hluchyj, M., Morgan, S.: Input versus output queuing on a space-division packet switch. IEICE Trans. Commun. **35**(12), 1347–1356 (1987)
[18] Anderson, T.E., Owicki, S.S., Saxe, J.B., Thacker, C.P.: High-speed switch scheduling for local area networks. ACM Trans. Comput. Syst. **11**(4), 319–352 (1993)
[19] McCanne, S., Floyd, S.: ns Network simulator. http://www.isi.edu/nsnam/ns/ (2002)
[20] Li, L., Alderson, D., Willinger, W., Doyle, J.: A first-principles approach to understanding the Internet's router-level topology. In: Proceedings of ACM SIGCOMM, pp. 3–14 (2004)
[21] Aggarwal, A., Savage, S., Anderson, T.: Understanding the performance of TCP pacing. In: Proceedings of INFOCOM 2000, pp. 1157–1165 (2000)

Author Biographies

Onur Alparslan received the B.S. and M.S. degrees in Electrical and Electronics Engineering from Bilkent University, Turkey, and Ph.D. degree in Information Science from Osaka University, Japan, in 2002, 2005, and 2008, respectively. He is now a postdoctoral researcher at Graduate School of Information Science and Technology, Osaka University, Japan. His research interests include optical networks, simulation, and network design.

Shin'ichi Arakawa received the M.E. and D.E. degrees in Informatics and Mathematical Science from Osaka University, Japan, in 2000 and 2003, respectively. He is currently a Research Assistant at the Graduate School of Information Science and Technology, Osaka University, Japan. His research work is in the area of photonic networks. He is a member of IEEE and IEICE.

Masayuki Murata received the M.E. and D.E. degrees in Information and Computer Sciences from Osaka University, Japan, in 1984 and 1988, respectively. In April 1984, he joined IBM Japan's Tokyo Research Laboratory, as a Researcher. From September 1987 to January 1989, he was an Assistant Professor with the Computation Center, Osaka University. In February 1989, he moved to the Department of Information and Computer Sciences, Faculty of Engineering Science, Osaka University. In April 1999, he became a Professor of Osaka University, and since April 2004, he has been with the Graduate School of Information Science and Technology, Osaka University. He has contributed more than four hundred and fifty papers to international and domestic journals and conferences. His research interests include computer communication networks and performance modeling and evaluation. He is an IEICE Fellow. He is a member of IEEE, ACM, The Internet Society, IEICE and IPSJ.